UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y REDES DE COMUNICACIÓN

TEMA

SISTEMA DOMÓTICO PARA PERSONAS CON DISCAPACIDAD VISUAL USANDO UNA BLUETOOTH MESH

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERÍA EN ELECTRÓNICA Y REDES DE COMUNICACIÓN

AUTOR: Luis Fernando Farinango Terán

DIRECTOR: Ing. Edgar Alberto Maya Olalla ,MSc

Ibarra – Ecuador

2023

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

IDENTIFICACIÓN DE LA OBRA

En base al Art. 144 de la Ley de Educación Superior ,realizo la entrega del presente proyecto a la Universidad Técnica del Norte para su uso en el Repositorio Digital Institucional ,para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CEDULA DE IDENTIDAD :	100407906-5		
APELLIDOS Y NOMBRES :	Farinango Terán Luis Fernando		
DIRECCIÓN :	Panamá 2-65 y Brasil		
EMAIL:	lffarinangot1@utn.edu.ec		
TELÉFONO FIJO:	062 510891	**TELÉFONO MÓVIL:**	099 245 9125

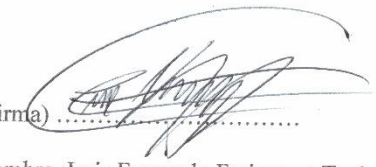
DATOS DE LA OBRA	
TITULO:	"Sistema Domótico para personas con discapacidad visual usando una Bluetooth mesh"
AUTOR (ES) :	Luis Fernando Farinango Terán
FECHA : DD/MM/AA	16/02/2023
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA :	■ **PREGRADO** ☐ **POSTGRADO**
TITULO POR EL QUE OPTA:	Ingeniero en Electrónica y Redes de Comunicación
ASESOR/DIRECTOR :	Ing. Edgar Alberto Maya Olalla ,MsC.

CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros ,por lo tanto ,la obra es original y que es titular de los derechos patrimoniales ,por lo que asume la responsabilidad sobre el contenido de esta y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros

Ibarra ,a los 16 días del mes de Febrero de 2023

EL AUTOR:



(Firma) .....

Nombre :Luis Fernando Farinango Terán

CERTIFICACIÓN

UNIVERSIDAD TÉCNICA DEL NORTE

UNIVERSIDAD ACREDITADA RESOLUCIÓN 002-CONEA-2010-129-DC
Resolución N° 001-073-CEAACES-2013-13

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES *RPC-SO-31-No 573-2016*

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y REDES DE COMUNICACIÓN (No vigente, habilitado para registro de títulos)

CERTIFICACIÓN

ING. EDGAR MAYA M.Sc , DIRECTOR DEL PRESENTE TRABAJO DE TITULACIÓN

CERTIFICA:

Que ,el presente trabajo de titulación denominado : "SISTEMA DOMÓTICO PARA PERSONAS CON DISCAPACIDAD VISUAL USANDO UNA BLUETOOTH MESH", ha sido desarrollado por el Sr. Luis Fernando Farinango Terán bajo mi supervisión.

Es todo en cuanto puedo certificar en honor de la verdad.

Ing. Edgar Maya M.Sc.

Director de Tesis

DEDICATORIA

“La vida es una aventura desafiante o nada en absoluto “

Autor :Helen Keller

Dedico este trabajo a Dios por iluminar mi camino y darme la sabiduría para afrontar todos los retos en el transcurso de mi formación universitaria.

A mis padres Luis y Laura por estar a mi lado en cada paso con su apoyo incondicional y comprensión animándome a seguir adelante.

A mi familia en general por su ayuda, conocimiento y asesoría para cumplir esta meta .

Gracias a todos por su valiosa colaboración para cumplir esta meta .

AGRADECIMIENTO

A la Universidad Técnica del Norte y sus profesionales por los conocimientos impartidos durante mi formación profesional que me permitieron desarrollar nuevas habilidades y mejorar mi pensamiento crítico ,cualidades esenciales para culminar mi formación.

Al Ing. Edgar Maya y al Ing. Carlos Vásquez por su tiempo ,conocimiento y asesoría brindada para el desarrollo y culminación de mi trabajo de tesis .

A mis profesores por contribuir a enriquecer mis conocimientos y desarrollarme profesionalmente .

RESUMEN

En el presente trabajo se realiza el diseño y construcción de un sistema domótico para personas con discapacidad visual usando una Bluetooth mesh con el fin de ayudarlas en el hogar y reducir los accidentes en ambientes internos.

El sistema ha sido diseñado en base a los obstáculos diarios que una persona no vidente debe vivir dentro de su hogar. Funcionará de tal forma que emitirá alertas en base a los datos recogidos de una red de sensores que funcionan bajo el perfil Bluetooth Mesh ,dichos datos serán enviados a una plataforma en la nube, la cual los guardará en una base de datos en tiempo real .Ademas se integra una plataforma web que permitirá observar el estado actual de la red de sensores así como el registro histórico de las alertas emitidas por el sistema. Todos los nodos de la red de sensores serán controlados a través de una aplicación móvil ,la cual ha sido construida usando una librería que permite al perfil Bluetooth Mesh funcionar sobre un Smartphone. La aplicación móvil implementa el reconocimiento de palabras por voz para ayudar a la persona no vidente a solicitar los datos de la red de sensores así como algunas características de accesibilidad .

El presente documento explica a detalle cada uno de los parámetros tomados en cuenta durante el proceso de diseño del sistema en base a la metodología utilizada ,así como la presentación de la arquitectura de red del sistema ademas se presenta las pruebas desarrolladas para la demostración de la utilización de perfil Bluetooth Mesh en la red de sensores construida.

ABSTRACT

In the present work, the design and construction of a home automation system for the visually impaired is carried out using a Bluetooth mesh in order to help them at home and reduce accidents in internal environments.

The system has been designed based on the daily obstacles that an blind person must live within their home. It will work in such a way that it will issue alerts based on data collected from a network of sensors operating under the Bluetooth Mesh profile ,that data will be sent to a cloud platform, which will store it in a real-time database. In addition, a web platform is integrated that will allow observing the current state of the sensor network as well as the historical record of the alerts issued by the system. All nodes in the sensor network will be controlled through a mobile app, which has been built using a library that allows the Bluetooth Mesh profile to work on a smartphone. The mobile app implements speech recognition to help the blind person request data from the sensor network as well as some accessibility features.

This document explains in detail each of the parameters taken into account during the system design process based on the methodology used ,as well as the presentation of the network architecture of the system also presents the tests developed for the demonstration of the use of Bluetooth Mesh profile in the network of sensors built.

Índice

Índice de figuras

Índice de tablas

Índice de Ecuaciones

1. Capítulo I Antecedentes

1.1. Problema

Las personas con discapacidad visual deben aprender a realizar sus actividades de forma diferente ,en muchos casos se contrata a un tercero para ayudar con dichas tareas o se opta por el uso de cursos especiales .Como resultado pueden llevar una vida casi normal la mayor parte del tiempo, algunas herramientas de ayuda como el sistema braille ,impresión 3D, asistentes de voz ,gafas inteligentes entre otras han permitido que una persona con este tipo de discapacidad tenga una mejor calidad de vida. Por otro lado también existen herramientas para permitir la movilidad de forma muy efectiva ,para ambientes externos se usan :perros guías ,bastones, habilidades aprendidas por terceros y sistemas de navegación especiales (Kuriakose et al., 2020).De las herramientas mencionadas ,los bastones y sistemas de navegación también se aplican a ambientes internos .El avance de la tecnología ha permitido que los bastones sean también parte de los sistemas de navegación ,posteriormente ha sido posible añadir dispositivos wearables para que los sistemas sean mucho más robustos, puedan recolectar mucha más información y puedan brindar un mejor servicio, incluso hoy en día es posible rastrear al usuario por medio de GPS(Ramadhan, 2018).Pero el costo de los equipos y mano de obra era equivalente dado que solo eran aplicables para un sector en específico, por lo que la producción en masa no presentaban ganancias económicas ;además los dispositivos solo funcionaban en solo un tipo de ambiente especifico (Bhowmick & Hazarika, 2017). El aparecimiento del Internet de las Cosas logro que todos los avances tecnológicos se combinen y sean aplicables en ambientes internos y externos por igual, reduciendo el costo y permitiendo el acceso de los equipos por igual a cada persona. Además la cantidad de datos recogidos permitió brindar mucha más información en tiempo real sin enfocarse solamente en un factor en específico y ,también faculto la aparición de nuevos escenarios donde los sistemas pueden funcionar(Miori et al., 2019).

Los sistemas inteligentes se tomaron muchos escenarios, uno de estos fue dentro del hogar del usuario. Para el caso de una persona con cualquier tipo de discapacidad, estos sistemas buscan que el consumidor tenga una autonomía dentro del entorno donde se realiza la instalación; brindando una mejor calidad de vida y en algunos casos también realizando el monitoreo de la salud para prevención de enfermedades. Sin embargo la presentación de información debe ser la adecuada para el usuario si se considera la discapacidad del mismo(S. Sharma & Umme Salma, 2021).

Para las personas con discapacidad visual el sentido del tacto se vuelve indispensable ,ya sea para movilidad o detección de objetos; por lo que la probabilidad de accidentes es muy alta en entornos desconocidos .Las soluciones IoT permiten reducir los riesgos de accidentes (Leporini & Buzzi, 2018).Sin embargo en el país no existen sistemas a gran escala de IoT a pesar de que la penetración del internet es muy alta ,encontrándose en valores de 8.14% para el año 2019 (Pacheco, 2021). Por otro lado en el país según el Consejo Nacional para la Igualdad de las Discapacidades ,las personas con discapacidad visual registradas son 54480 de las cuales un porcentaje de 65% son mayores de edad y dentro de este rango el 37.59 % superan los 65 años; perteneciendo a la tercera edad(Consejo Nacional para la igualdad de Discapacidades, 2022). Por lo que la mayor parte del grupo social citado no puede costear un sistema IoT tan complejo en aspectos tecnológicos, debido a que no poseen un salario estable. Inicialmente los dispositivos para este grupo de personas se basaban en cuatro aspectos : movilidad ,navegación ,reconocimiento de objetos e interacción social(Leporini & Buzzi, 2018). En la actualidad, se ha añadido aspectos como: seguridad, control entre otros que dependen de las necesidades del usuario; en definitiva, los sistemas se adecuan al usuario; por lo que los limites son dados de acuerdo a la tecnología usada para la construcción del sistema. Todos los factores mencionados pueden ayudar a las personas con discapacidad a reducir accidentes y aumentar su autonomía dentro del hogar.

Adicionalmente ,tomando los factores antes mencionados ,se debe considerar que los sistemas IoT actuales no están desarrollados exclusivamente para personas que presenten discapacidades, también existen sistemas que permiten gestionar el control de: luces ,interruptores ,sistemas de ventilación ,electrodomésticos, sensores de presencia ,sensores de gas y otros más(Hajjaji et al., 2021).

La arquitectura de sistema propuesto se dividirá en tres capas .Estas capas se basan en el diseño de (Vanishree et al., 2017). La capa de aplicación será la capa más alta y brindara los servicios de IoT directamente al usuario ,la capa de red será la capa media y se encargara de la interconexión de sensores ,finalmente la capa de percepción será la capa de los sensores .El sistema planteado busca proporcionar al usuario algunos parámetros para asegurar seguridad y una mejor calidad de vida dentro del domicilio ,los parámetros son los siguientes : donde se encuentra actualmente la persona ,cual es el estado del cuarto o habitación ,rutas dentro del hogar ,situaciones peligrosas y las causas de ruidos inesperados. Todos estos parámetros se medirán con una red de sensores inalámbricos que brindarán datos en tiempo real, utilizando el protocolo Bluetooth Low Energy interconectados en una topología en malla. La tecnología utilizada es ideal dado su bajo costo de implementación y se encuentra implementada en dispositivos de uso común como, teléfonos móviles, cámaras de y otros accesorios. Los datos serán procesados en plataformas en la nube para posteriormente brindar estadísticas y presentar los datos al usuario mediante una pulsera inteligente. Los datos serán presentados en formato de voz, mediante menús, los comandos se recogerán mediante un micrófono y la ubicación del dispositivo será a elección del usuario. Para la implementación del sistema se realizará capacitaciones para el correcto uso de este.

1.1. Objetivos

1.1.1. Objetivo general:

Diseñar un sistema domótico con la tecnología Bluetooth Mesh para ayudar a personas con discapacidad visual en el hogar y reducir los accidentes en ambientes internos.

1.1.2. Objetivos Específicos

- Analizar las ventajas y desventajas del uso del perfil Mesh en domótica, basándose en las condiciones de vida de las personas con discapacidad visual dentro del hogar.
- Construir un sistema IOT utilizando dispositivos de bajo coste orientados a personas con discapacidad visual de estratos económico bajos y medios.
- Realizar pruebas de funcionamiento del sistema para determinar la óptima ubicación de los sensores dentro del hogar de una persona con discapacidad visual, así como el formato de presentación de la información y alertas al usuario.

1.2. Alcance

En el presente proyecto se realizará el diseño y construcción de un sistema domótico para personas con discapacidad visual usando la tecnología Bluetooth basándose en el perfil Mesh, se apoyará en el modelo en v para desarrollo de proyectos. Se elaborará un análisis de los riesgos que las personas con discapacidad visual tienen dentro del hogar y afectan su autonomía al realizar actividades cotidianas y también comprometen su salud, para ello se llevara a cabo un estudio de la situación actual de las personas con discapacidad visual dentro del hogar y como se desempeñan en su diario vivir. También se incluirá un estudio de las características de la tecnología Bluetooth en el perfil Mesh considerado su aplicación en entornos IoT con la identificación de ventajas y desventajas , de igual forma se establecerá una comparativa de costos de un sistema domótico típico tomando en cuenta el poder adquisitivo de una persona con discapacidad visual. Toda la información recolectada se procesará en base

a la metodología del modelo en v ,tomando en cuenta cada uno de los componentes de la misma(Zumba & Arreaga, 2018).

Con la información de las necesidades de los futuros usuarios del sistema y los beneficios de la tecnología planteada, se define las exigencias mínimas que el proyecto debe cumplir tomando como factores fundamentales a la presentación de información y la no interferencia en la vida cotidiana del usuario. De acuerdo al modelo en V, con cada uno de los resultados de las investigaciones realizadas Se deberá obtener un resultado considerando las necesidades del usuario y la metodología aplicada(Nugroho et al., 2017).

Se diseñará un sistema domótico con módulos de: control de presencia, control de luces, control de aire, detección de inundaciones y detección de incendios.

El módulo de control de objetos y posición tendrá dos submódulos: el submódulo de control de posición y el submódulo de control de estado de objetos. El submódulo de control de posición de objetos se basará en alertas al usuario si un tercero ha entrado a determinadas habitaciones consideradas privadas o habitaciones donde los mubles deben estar estáticos. Mientras el submódulo de control de estado de objetos se encargará de información al usuario de situaciones tales como: planchas conectadas, neveras abiertas entre otras. Para ello cada uno de los objetos que estén dentro de este submódulo deberán equipar un sensor adecuado. El módulo de control de luces será fundamental para el apagado de luces en las habitaciones del hogar, tomando en cuenta que para los usuarios del sistema es complicado detectar este fenómeno. El módulo de control de aire será encargado de controlar fugas de gas, ya sea que estas se traten de cilindros de gas, calefones o incineración de materiales. El módulo de detección de incendios detectara humo y alertara al usuario, funcionara en conjunto con el módulo de control de aire. Finalmente, el módulo de detección de inundaciones detectara presencia de agua en lugares del domicilio donde esta no debería estar presente.

Cada uno de los módulos tendrá asociado una serie de sensores asociados en un hub, cada hub será un nodo de la red Mesh que utilizará la tecnología de comunicación inalámbrica Bluetooth Low energy para la intercomunicación basándose en el estándar Mesh. Para la presentación de información al usuario, se desarrollará una aplicación móvil que podrá ser instalada en un smartphone preferiblemente con el sistema operativo Android, la interacción será a través de alertas de voz .Las alertas para cada uno de los módulos se calibrarán a través de los nodos que estarán asociados a eventos específicos de acuerdo a los sensores presentes en dicho nodo .

Se realizarán pruebas para comprobar el correcto funcionamiento del sistema, así como determinar la correcta ubicación de los dispositivos para asegurar que no se realizó ninguna interferencia en el estilo de vida del usuario.

1.3. Justificación

En el país existen leyes y reglamentos que amparan a las personas con discapacidad, de igual forma buscan la inclusión en cada uno de los sectores de la sociedad. Según Parágrafo II en los artículos 63 al 70 de la Ley Orgánica de Discapacidades, se especifica que “El Estado promocionará el uso de la lengua de señas ecuatoriana, el sistema Braille, las ayudas técnicas y tecnológicas, así como los mecanismos, medios y formatos aumentativos y alternativos de comunicación; garantizando la inclusión y participación de las personas con discapacidad en la vida en común”(CONADIS, 2012). Dentro de los objetivos del Buen Vivir en el plan 2017-2021 ,dentro del artículo 1 se especifica las garantías para que todas las personas tengan una vida digna(Ministerio del Interior, 2017) .

El presente proyecto se enfoca en desarrollar un ambiente seguro y con la menor posibilidad de accidentes dentro del hogar de una persona con discapacidad visual, que permita la gestión de diversos módulos mediante la ayuda de una aplicación instalada dentro de un smartphone. La tecnología de comunicación entre los elementos del sistema es Bluetooth,

específicamente el perfil Mesh. Es una tecnología de red lanzada por la SIG de bluetooth en el año 2017, se basa en la especificación de Bluetooth Low Energy. Consiste en una topología de nodos con comunicación multicast entre ellos (many to many). En un escenario donde se tenga n nodos entonces todos los nodos podrán comunicarse entre sí, incluso si los dos nodos que van a participar de la comunicación no se encuentran en un rango directo. La tecnología plantea su uso para aplicaciones de IOT con un consumo energético más bajo además de funcionar bajo equipos de diferentes marcas, la transmisión de información puede superar fácilmente obstáculos e interferencia gracias al tamaño de la red y adicionalmente permite interoperabilidad con la versión 4.2, dicha versión está distribuida ampliamente en el mercado. Esta estructura de red no es nada nueva pero con el boom de IOT se convirtió en la que más se adecuaba a las nuevas necesidades de infraestructura y conectividad de red(Bluetooth S.I.G., 2017), pero de acuerdo al tipo de tecnología su precio se elevaba debido a la alta cantidad de dispositivos usados para construir la infraestructura de red .También Bluetooth con el estándar mesh permite reducir considerablemente los costos energéticos de los equipos debido a que funciona sobre el estándar del bluetooth low energy.

Con todos los parámetros nombrados, se busca que el sistema a diseñar sea robusto, escalable y a su vez de bajo costo para que la capacidad adquisitiva del usuario no sea un problema.

2. Capítulo II Marco Teórico

2.1. Discapacidad visual

2.1.1. Definición

Desde el nacimiento de una persona, la visión es uno de los sentidos más importantes. Reconocer a sus progenitores es una de las primeras actividades que realiza un bebe. Con el paso de tiempo se vuelve una herramienta indispensable para actividades económicas y sociales. Se define como discapacidad a la característica que tiene una persona y le impide adaptarse a la sociedad, según la Organización Mundial de la Salud, la discapacidad visual es el impedimento de algunas personas para llevar una vida normal y llevar a cabo actividades cotidianas debido a la reducción parcial o total de capacidades visuales. Esta puede ser adquiridas o de nacimiento y no siempre se pueden clasificar como discapacidad.

2.1.2. Clasificación de discapacidad visual

Las condiciones que pueden afectar al sentido de la visión son demasiadas como para clasificarlas si se toma en cuenta la complejidad del órgano, por lo que se puede dividir en las afecciones que producen ceguera y cuáles no. Esta clasificación afecta en cuando a niveles generales, pero según la Organización Mundial existen niveles claramente diferenciables y no por sede un nivel más bajo no significa que no sea un factor que pueda evolucionar en algo grave.

Hay que destacar que uno de los factores de más riesgo para el desarrollo de una discapacidad visual es el envejecimiento, enfermedades como presbicia, cataratas, glaucoma y degeneración macular son factores que inciden en el deterioro de la visión. La presencia de bacterias o virus también es una causa. Los valores para los cuales una persona se considera con discapacidad visual son una agudeza visual menor a 6/18 y un campo de visión menor a 10 grados, dentro de estos valores una persona califica como una persona con baja visión .Mientras que se considera ceguera a un valor de agudeza visual menor a 3/60. Estos valores

son tomados en cuenta para cuando se ha realizado todo tipo de actividad de corrección por parte del personal adecuado, razón por la cual las campañas de prevención son una de las tareas más importantes para evitar el desarrollo de una discapacidad visual(Organización Mundial de la Salud, 2020).

Figura 1. Clasificación de la gravedad de la deficiencia visual basada en la agudeza visual del ojo que ve mejor.

Categoría	Agudeza visual en el ojo que ve mejor	
	Peor que:	Igual o mejor que:
Deficiencia visual leve 	6/12	6/18
Deficiencia visual moderada 	6/18	6/60
Deficiencia visual grave 	6/60	3/60
Ceguera 	3/60	
Deficiencia visual de cerca 	N6 o 0,8 m a 40 cm	

Fuente: (Organización Mundial de la Salud, 2020). Informe mundial sobre la discapacidad. Organización Mundial de La Salud, 388. Recuperado de https://apps.who.int/iris/bitstream/handle/10665/331423/9789240000346-spa.pdf

2.1.3. Registros de la discapacidad visual

La búsqueda de la inclusión para las personas con discapacidad visual ha llevado a tener bases de datos con cifras actualizadas para conocer cuál es la situación general de este grupo de la sociedad.

2.1.3.1. Cifras a nivel internacional

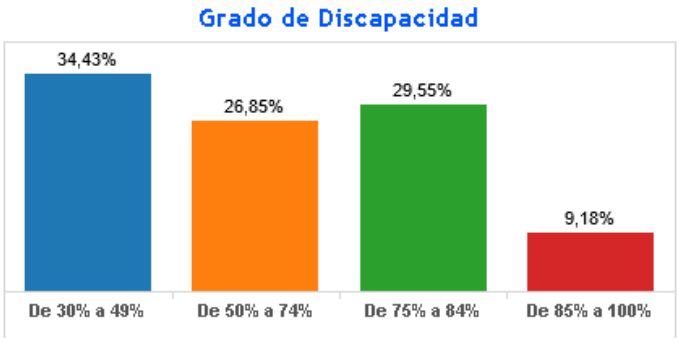
Según la Organización Mundial de la Salud actualmente existen 200 millones de personas con deterioro de la visión cercana o distante. En al menos 1000 millones de esos casos, es decir, en aproximadamente la mitad de este grupo, el deterioro visual podría haberse evitado con un tratamiento o en el peor de los casos no se ha aplicado ningún tipo de tratamiento. Entre las principales causas de ceguera se tiene a los son los errores de refracción no corregidos (uso de lentes) y las cataratas. (Organización Mundial de la Salud, 2020).

2.1.3.2. Cifras a nivel local

En el ámbito local ,la CONADIS es la encargada de la administración de datos de las personas con discapacidad visual .La información es de dominio público ,se actualiza a través del Ministerio de Salud Pública .

Según estas cifras actualmente en el país existen 54662 personas con discapacidad visual registradas. El 34.43 % presenta un nivel de discapacidad del 30 a 40 por ciento ,el 26.85% tiene un nivel discapacidad de entre 50 a 74 por ciento ,el 29.55% presenta un nivel de discapacidad de entre 75 a 84 por ciento y solo el 9.8% tiene una discapacidad visual de entre 85 a 100 por ciento como se observa en la Figura 2.

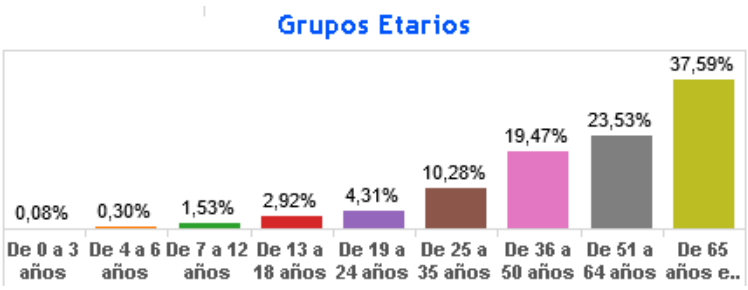
Figura 2.Estadísticas de Discapacidad para las personas registradas en el CONADIS, de acuerdo con el nivel de discapacidad.



Fuente :(Consejo Nacional para la igualdad de Discapacidades, 2022), Estadísticas de Discapacidad – Consejo Nacional para la Igualdad de Discapacidades. Recuperado el 17 de junio de 2021, de https://www.consejodiscapacidades.gob.ec/estadisticas-de-discapacidad/

Considerando la misma fuente de información ,es posible conocer los grupos etarios en donde es más común la presencia de una discapacidad visual .En donde más del 50% corresponde al grupo de 51 años en adelante, como se puede observar en la Figura *3* .

Figura 3.Estadísticas de Discapacidad para las personas registradas en el CONADIS, de acuerdo con el grupo etario.



Fuente :(Consejo Nacional para la igualdad de Discapacidades, 2022), Estadísticas de Discapacidad – Consejo Nacional para la Igualdad de Discapacidades. Recuperado el 17 de junio de 2021, de https://www.consejodiscapacidades.gob.ec/estadisticas-de-discapacidad/

2.1.4. Obstáculos comunes en el diario vivir

Dos de los mayores problemas a resolver de una persona con discapacidad es la autonomía personal y la adaptabilidad a la sociedad .Sin importar la discapacidad presente ,dichos factores aparecen tarde o temprano en la vida de la persona .Aquellos factores están ligados a como la persona ha desarrollado su propio espacio personal y como percibe al mundo exterior .También se encuentra la forma en como la sociedad los percibe y trata ,en algunos casos puntuales . (Organización Nacional de Ciegos Españoles, 2012).

2.1.4.1. Riesgos en ambientes internos y externos

Las personas con discapacidad visual generalmente aprenden a desplazarse en entornos externos con la ayuda de un entrenamiento llamado O&M .Orientation and Mobility por sus siglas en Inglés ,se trata de un entrenamiento del cual se aprende técnicas para detectar curvas en las calles ,intersecciones ,estados de semáforos y conciencia general de los alrededores .Para lo cual se puede usar herramientas especiales como bastones o perros guias .En si se tratan de herramientas muy útiles para aumentar la autonomía de la persona ,pero no lo inhiben de riesgos a los cuales incluso la persona sin discapacidad también está expuesta .

Las personas con discapacidad visual tienen un aprendizaje sobre la escala espacial muy diferentes a las personas sin discapacidad visual, esto es mucho más notable en la percepción del espacio. Por ejemplo, una persona sin discapacidad puede controlar sus movimientos en base a las características del entorno de forma automática y a futuro usara eso como referencia para realizar una acción .Mientras que una persona con discapacidad visual necesita ser conscientes de los elementos próximos a ellos en cada movimiento ,resultando en un mayor uso de atención y esfuerzo que puede generar estrés al momento de movilizarse. Esta sensación de estrés se presenta mayormente en paseos al aire libre y se reduce considerablemente en ambientes internos debido a la experiencia que la persona ya tiene en dichos entornos

,considerando que el individuo ya ha estado en ese lugar antes. Como consecuencia es posible identificar dos factores :la reacción de las personas con discapacidad visual es mucho más lento que una persona sin discapacidad visual ,y la experiencia en los entornos puede reducir el tiempo de reacción por lo que un obstáculo inesperado puede aumentar mucho más el tiempo de reacción(Giudice, 2018).

Según un estudio desarrollado en EEUU acerca de hechos impactantes que les sucedieron a los encuestados ,se menciona que de las personas encuestadas en su artículo, las personas con discapacidad visual tienen un mayor rango de riesgos comparados con la población en general. Existen eventos particulares como :fuego o explosiones ,accidentes muy serios y otros que solo les ocurrieron a los participantes durante una situación muy específica. Se también explica que no existe un tipo de incidente que les sucede más a las personas con discapacidad visual que a las personas sin discapacidad. Una cifra muy notable es que exposición a fuego y explosiones es mucho mayor en las personas que perdieron su vista en algún punto de su vida con respecto a aquellos que nacieron con esa condición (Brunes & Heir, 2021).

Por otro lado en un estudio realizado en Reino Unido se habla de la relación entre el estilo de vida ,factores ambientales en los accidentes de personas con discapacidad visual y como influenciaría la modificación de sus condiciones de vida en la reducción de estas. Se clasifica tres factores :riesgos ,dificultad ,dificultad de usar objetos y ambientes interiores.

- Dentro de los riesgos es posible mencionar : obstáculos en corredores ,obstáculos en cuartos ,piso resbaladizo, extremos de muebles y diferencias de nivel en el suelo.
- Dentro de las dificultades es posible mencionar :Todo tipo de movimientos dentro del hogar especialmente dentro de la distribución de cuartos dentro del hogar .

- Dentro de la dificultad de usar objetos se menciona :uso de domésticos en el hogar y localización de cosas y ventilación.

Los resultados demuestran que la mayor cantidad de accidentes se da debido a la presencia de elementos extraños en el piso como :agua ,limpiadores entro otros. Además se afirma que los accidentes son mucho mayores en personas enfermas aumentado aún más la probabilidad de accidentes (Lee & Yoo, 2015).

En base a los factores presentados es posible validar la mayoría de los módulos del sistema a desarrollar con sus respectivos sensores y elementos.

2.1.4.2. Ámbito Social y Emocional

Desde tiempos antiguos las personas con discapacidad visual eran excluidos de la participación en la sociedad, incluso en los registros de las personas que llegaron a desempañarse normalmente no se les catalogaba con dicha clasificación. Ser considerado "discapacitado "era traducido a dependencia. Además, la tecnología de la época no ayuda a ofrecer algún tipo de rehabilitación o dispositivo que ayude a reducir un poco la dependencia que la sociedad marcaba. Hoy en día ya existen muchas herramientas que ayudan a disminuir estas diferencias, pero aún queda un largo camino por recorrer. Igualmente, de los problemas mencionados, el deterioro de la visión o la pérdida total termina afectando a la economía mundial, según la OMS se calcula que las perdidas por mano de obra relacionada a algún tipo de deficiencia visual llegan a valores de US$ 244 000 millones y US$ 25 400 millones a nivel mundial.

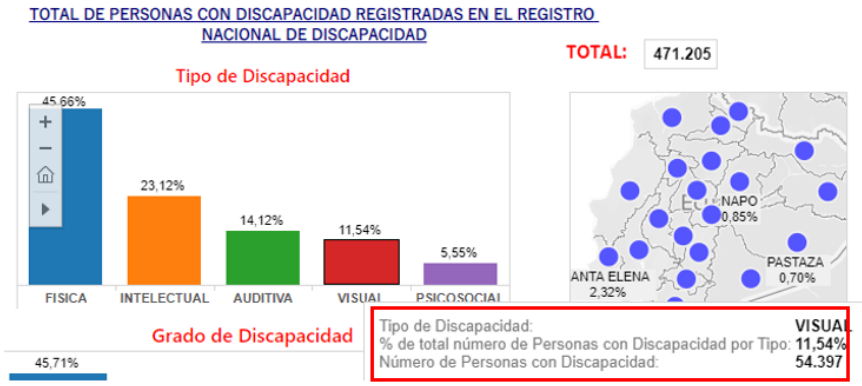
2.1.5. Situación Socioeconómica de las personas con discapacidad visual en el Ecuador

La condición socioeconómica de un individuo toma en cuenta diferentes variables como: nivel de educación ,situación laboral e ingresos percibidos ,es importante analizar estos

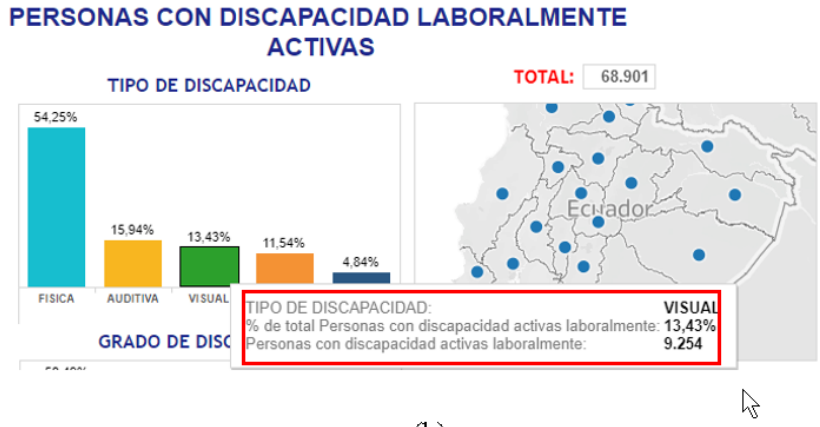
factores en el grupo objetivo de la presente tesis para determinar el acceso a la tecnología y servicios .

En el Ecuador las personas con discapacidad visual son 54.397 según el Consejo de las discapacidades y de ellas 9.254 pertenecen a la Población Económicamente Activa estos se pueden observar en la *Figura 4* .

Figura 4.Número total de personas con discapacidad visual y personas con discapacidad visual laboralmente activas



(a)



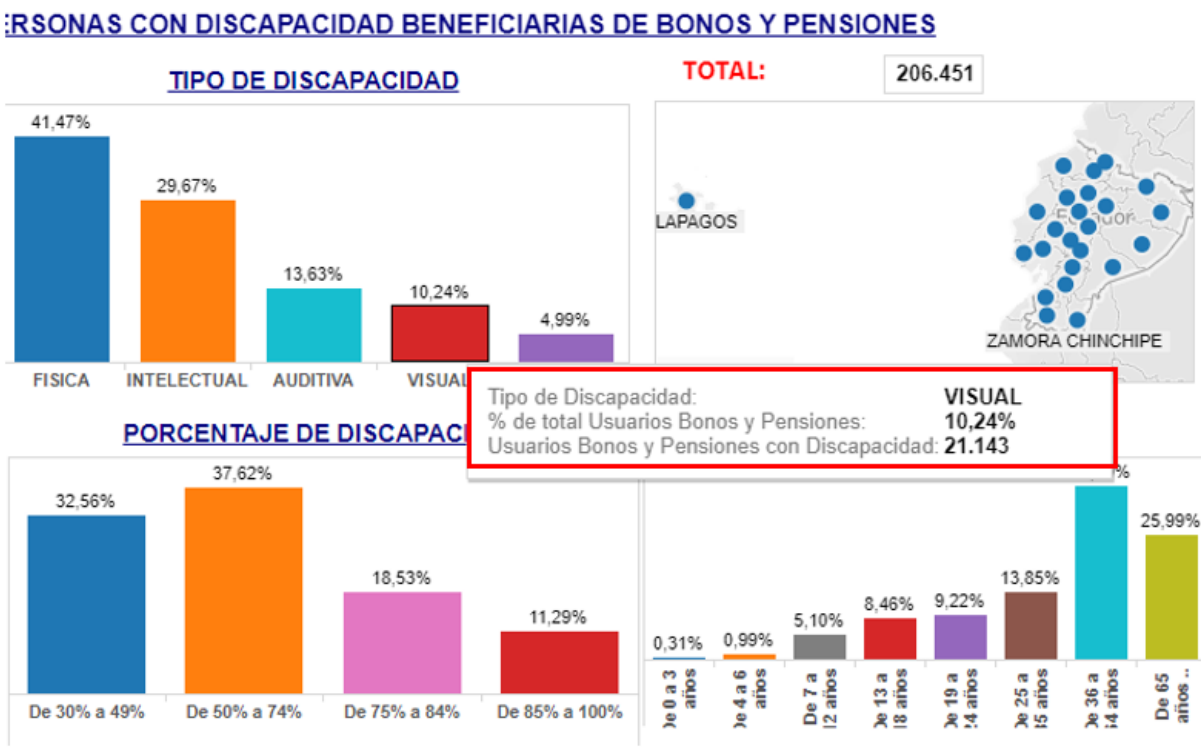
(b)

Fuente :(Consejo Nacional para la igualdad de Discapacidades, 2022), Estadísticas de Discapacidad – Consejo Nacional para la Igualdad de Discapacidades. Recuperado el 17 de junio de 2021, de https://www.consejodiscapacidades.gob.ec/estadisticas-de-discapacidad/

El gobierno en su afán de contribuir y mejor el estilo de vida de dicho grupo creo bonos y ayudas económicas llegando a favorecer a 21.143 personas ,esto se puede observar en la *Figura 5* .Este grupo vulnerable también se beneficia del descuento del 50% en telefonía fija ,

móvil , servicios básicos y devolución de impuestos(Consejo Nacional para la igualdad de Discapacidades, 2022).

Figura 5.Personas con discapacidad visual que acceden a bonos en el Ecuador



Fuente :(Consejo Nacional para la igualdad de Discapacidades, 2022), Estadísticas de Discapacidad – Consejo Nacional para la Igualdad de Discapacidades. Recuperado el 17 de junio de 2021, de https://www.consejodiscapacidades.gob.ec/estadisticas-de-discapacidad/

Dentro del grupo etario de la población objetivo ; 2625 están cursando la educación básica media y bachillerato ,estas cifras se pueden observar en la *Figura 6*.

Figura 6. Número de personas con discapacidad visual que se encuentran cursando educación básica ,media y bachillerato en el Ecuador .

Fuente :(Consejo Nacional para la igualdad de Discapacidades, 2022), Estadísticas de Discapacidad – Consejo Nacional para la Igualdad de Discapacidades. Recuperado el 17 de junio de 2021, de https://www.consejodiscapacidades.gob.ec/estadisticas-de-discapacidad/

El bono Joaquín Gallegos Lara asciende a $240,00 ,las personas con discapacidad visual cumplen con las condiciones para acceder a este beneficio.

En los últimos años se ha considerado a este grupo vulnerable como parte de diversos programas sociales con el objetivo de darles mayor visibilidad frente a la población, sin embargo, en el campo tecnológico debido a la condición de nuestro país no se evidencia un avance significado en el campo salvo el incremento de conexiones a internet y la reducción de costos de este servicio debido a la creación de nuevas empresas que se dedican a proveer del mismo.

Considerando lo antes mencionado se concluye que es importante la reducción de costos en el desarrollo del presente proyecto para que sea accesible a la mayoría de las personas con discapacidad visual debido a su condición socioeconómica que las estadísticas muestran .Uno de los objetivos para la reducción de costos será buscar componentes con una durabilidad relativamente larga y de un valor menor que cumplan con la función sin afectar el rendimiento del sistema .

2.1.6. Legislación aplicada

La legislación aplicada a las personas con discapacidad visual tiene como fin el respeto a los derechos humanos basados en el enfoque social a dichas personas .En el ámbito internacional dentro de la legislación formal es posible mencionar a la Convención sobre los derechos humanos de las personas con discapacidad .

En el artículo 8 se menciona la toma de conciencia incluyendo a los gobiernos nacionales así como a las personas naturales para la sensibilización y respeto a nivel social de las personas con discapacidad. Así como también la lucha contra los estereotipos dirigidos a las personas con discapacidad para promover la toma de conciencia .Todo esto apoyado en campañas de sensibilización y conciencia social para lograr respeto a los derechos de las personas con discapacidad .En el artículo 9 ,se habla de la accesibilidad para lograr la independencia e igualdad de condiciones en la mayoría de los ámbitos de la vida cotidiana ,ya sea en ambientes al aire libre como dentro de su hogar. Todo se logrará con la remoción de obstáculos físicos y en servicios de instituciones públicas como privadas que atenten contra la accesibilidad y fluidez en la vida cotidiana .En los artículos 19 y 20 se aborda :el derecho a vivir de forma independiente y la inclusión en la sociedad ,y la movilidad personal. Considerando la vida digna dentro del entorno social ,con el menor tipo de barreras físicas y sociales .Así como la ayuda de expertos tomando en cuenta sus necesidades para evitar el aislamiento social y asegurar su independencia sin ser obligados a llevar un tipo de vida en particular(Asamblea General de la Organización de las Naciones Unidas, 2008).

También se puede citar al Informe Mundial sobre la discapacidad ,exactamente en las recomendaciones 3 y 4 .Donde se menciona la creación y adecuación de planes de acción a nivel nacional acerca de discapacidad que cada gobierno debería aplicar ,y la inclusión de personas con discapacidad respectivamente(World Health Organization, 2013).

Con respecto a la legislación local hay que considerar los organismos que amparan a las personas con discapacidad para el cumplimiento de las leyes .Los cuales son : CONADIS y MIES,. Las leyes y reglamentos en el país son varios ,se comienza el análisis con la Ley Orgánica de Discapacidades con su respectivo Reglamento expedidos en el año 2012 y 2017 respectivamente.

En la sección Sexta correspondiente a la vivienda ,dentro del artículo 56 se especifica el derecho a una vivienda digna que se adecue a las necesidades con la mayor sencillez de acceso ,de tal forma que sea posible la mayor autonomía posible .En la sección octava correspondiente a las tarifas preferenciales ,extensiones arancelarias y del régimen tributario ;en el artículo 79 ,sección 2 ,se hace mención a que se aplicará un subsidio del 50% al servicio de energía eléctrica hasta un 50% del salario básico unificado(CONADIS, 2012).

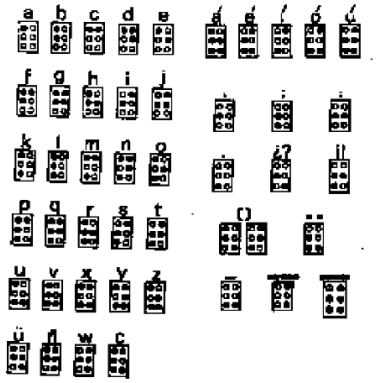
Continuando con la legislación local ,la constitución de la Republica del 2008 tiene una sección dedicada a la equiparación de oportunidades para las personas con discapacidad con el fin de lograr la inclusión social de las mismas .Dentro del artículo 47 se identifica los derechos ,entre ellos el acceso a :atención de profesionales especializadas en los ámbitos necesarios para reducir su aislamiento social, rebajas y subsidios ,condiciones de trabajo dignas ,acceso a vivienda ,educación de calidad y otros más .En el artículo 48 se aborda más a detalles políticas de inclusión (Asamblea Nacional del Ecuador, 2008).

2.1.7. Sistema Braille

Es el sistema de lectura y escritura diseñado y usado para las personas no videntes. Se basa en una celda básica conocida como celdilla, existen un conjunto de puntos que están contenidos dentro de esta celdilla. Los puntos pueden tener seis posiciones diferentes y cada una de las combinaciones que se pueden realizar están relacionadas a las letras ,números y símbolos del alfabeto .Este sistema puede ser usado para lectura como para escritura ,la gran

desventaja que presenta es que se necesita de una gran nivel de habilidad para usarlo así como una gran nivel de practica para que su uso sea fluido(Simón et al., 1995).

Figura 7. Sistema Braille.



Fuente (Simón et al., 1995) ,El sistema Braille: Bases para su enseñanza- aprendizaje. *Comunicación, Lenguaje y Educación*, *7*(4), 91–102. https://doi.org/10.1174/021470395763771891

2.2. Asistentes y sintetizadores de voz

La voz humana ha sido un medio de comunicación desde el aparecimiento del hombre ,con el aparecimiento del teléfono se idearon métodos de transporte de esta a través de varias distancias .Hoy en día es posible utilizar software que imitan la voz de acuerdo a ciertos parámetros como : intensidad ,tono ,velocidad y otros más. Es posible también el reconocimiento de palabras muy sencillas hasta simular conversaciones con asistentes de voz para ejecutar alguna acción es especifico ,en consecuencia, los asistentes y sintetizadores de voz son herramientas muy útiles para las personas con discapacidad visual.

2.2.1. Historia

El primer sintetizador de voz fue construido por un profesor ruso llamado Christian Kratzenstein, podía reproducir las vocales, pero una forma muy rudimentaria .Se trataba de una serie de resonadores con una forma específica para cada vocal. A partir de este punto se realizaron modificación para añadir más vocales y consonantes al modelo inicial .

La aparición de la telefonía y el teléfono mismo ,se dio un volque total al desarrollo de las aplicaciones de sinterización de voz. Para el transporte de esta se debía capturar un determinado rango de frecuencia del espectro audible , por lo que el ancho de banda que se

utilizaba era demasiado. Para solucionar este problema se utilizó un dispositivo llamado vocoder ,el cual se encargaba de dividir la señal original en diez bandas de frecuencia con el fin de reducir el ancho de banda utilizado .En el otro lado de la línea se realizaba el proceso inverso ,se utilizaba una portadora con ruido para el proceso. En la teoría era una idea innovadora pero en la práctica no tuvo éxito dado que la portadora en realidad no contenía información .El dispositivo fue mejorado y se usó en las conversaciones de Roosevelt y Churchill durante la Segunda Guerra Mundial ,en la actualidad es usado en la industria de la música(Katz & Assmann, 2019).

Con la aparición de los microprocesadores ,se mejoró la sinterización de voz a tal punto que hoy en día es posible crear audios con diferentes tonos de voz ,diferentes tipos de voz e incluso es posible formular oraciones en donde se puede escoger el género y velocidad de pronunciación de las palabras .Por otro lado ,los asistentes de voz pueden realizar conversaciones con la ayuda de algunos algoritmos de inteligencia artificial usando comandos de voz .Obviamente se utilizan voces pregrabadas que realizan preguntas en base a los requerimientos del usuario .En conclusión ,los dispositivos utilizan la voz humana como código de comunicación tal como lo haría una persona común.

2.2.2. Influencia de la voz humana en los asistentes y sintetizadores de voz

La voz humana es una de las señales analógicas más difíciles de digitalizar debido al gran acho de banda que utiliza, la principal razón radica en la originalidad de la voz humana .Por ello es necesario el estudio de :las características de la voz humana ,la digitalización de la voz humana

2.2.2.1. Características de la voz humana

La voz de una mujer es diferente a la de un hombre ,diferente a la de un niño e incluso diferente a la de otra mujer .El rango de frecuencia que tiene la voz humana se encuentra entre 4 KHz y 8 KHz, la voz masculina se encuentra en un rango de 0.077 KHz a 0.482 KHz mientras

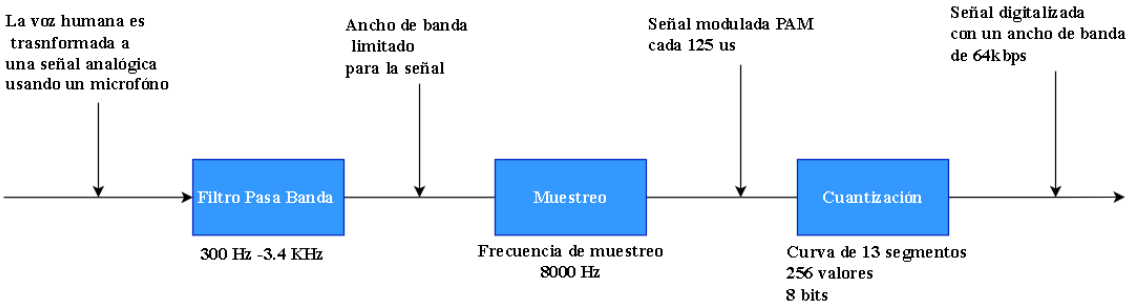
la voz femenina se encuentra entre las frecuencias de 0.0137 KHz a 0.634 KHz. Cada uno de los individuos tendrá asociado diferentes tonos y timbres con los cuales será posible distinguirlos entre ellos. Para digitalizar la voz humana se realiza tres pasos : muestreo ,cuantificación y codificador .En consecuencia durante el proceso de digitalización se pierden una gran cantidad de datos ,por lo que es muy diferente escuchar a una persona en una llamada que escucharla en una conversación frente a frente(Singh, 2019).

2.2.2.2. Digitalización de la voz

Como se mencionó anteriormente ,la digitalización de la voz sigue un proceso muy detallado. Varia en algunos casos ,por ejemplo, en la telefonía tradicional se utiliza algunos valores de filtrado de voz diferentes que los utilizados en la telefonía móvil o telefonía IP. Considerando que se utiliza un smartphone para la toma de voz en este documento ,se trabajara con la digitalización de la voz para telefonía móvil.

El proceso inicia cuando transforma la voz en una señal analógica con ayuda de un micrófono .A continuación se utiliza un filtro pasa banda que filtra el rango de frecuencias entre 300 Hz y 3.4 KHz ,se considera que se va a utilizar un canal con un ancho de banda de 64 kbps E1. En el siguiente paso se muestrea la señal obtenida con 8000 muestras por segundo es decir 8 KHz .La señal muestreada resultante es una señal modulada con amplitud de onda cada 125 us .Finalmente la señal muestreada se cuantifica con valores que van desde 0 a 255 y está lista para ser enviada por el canal de comunicación .Este proceso se resumen en la *Figura 8*.

Figura 8. Proceso de digitalización de una señal analógica de la voz humana.

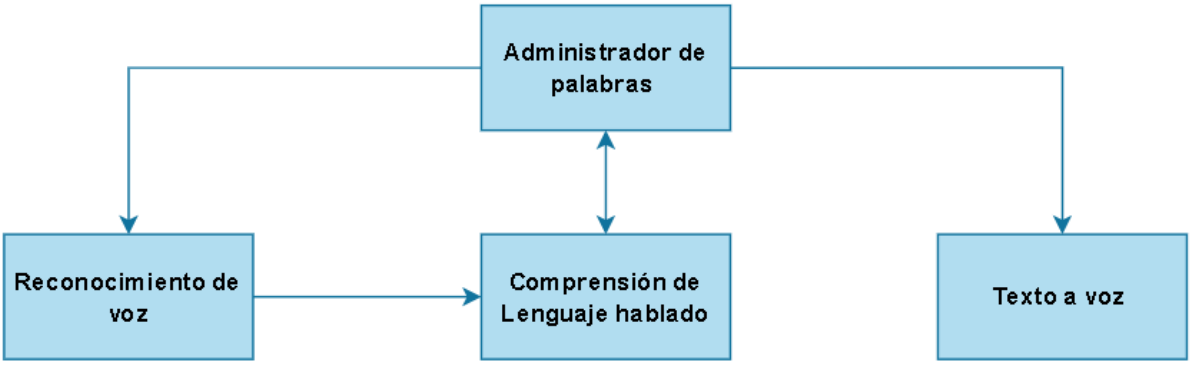


Fuente : Basado en (Sauter, 2017)

2.2.2.3. Comunicación humano-maquina

Para la comunicación humano máquina ,se debe digitalizar la voz para que la máquina o dispositivo puede entender lo que el ser humano dice .La forma típica es usar la función voz a texto ,esta función esta implementada en la mayoría de las computadoras ,smartphones y otros dispositivos. Mediante esta herramienta se traduce la voz de un ser humano a un texto plano que puede ser entendido por el dispositivo electrónico, se puede observar el proceso en la *Figura 9*. Siempre se usa una etapa de filtrado de ruido durante la lectura de datos ,la cual se hace con un micrófono .

Figura 9.Componentes de la función voz a texto



Fuente :Basado en (Yu & Deng, 2014)

Pero existe un factor que se debe identificar al momento del desarrollo de un proyecto que utilizará comandos de voz ,se usara la función de voz a texto o se usara el reconocimiento de voz .Los dos términos son similares pero difieren en que : la función de voz a texto traduce

todas las palabras a texto plano sin importarle la originalidad de la voz del usuario ;mientras que la función de reconocimiento de voz solo reconocerá a un usuario específico ,dicho usuario ha grabado un comando específico con anterioridad. Si se toma en cuenta el proceso de digitalización de la voz con sus respectivos pasos ,el proceso cambia en la etapa de codificación .Para la función de voz a texto se obtiene una cadena de caracteres mientras que la función de reconocimiento de voz se obtiene un archivo en formato AVI que se comparara con otro archivo ya existente del mismo formato(Abdullah et al., 2021).

2.2.3. Sintetizadores y asistentes de voz en la programación de aplicaciones móviles

Las funciones de texto a voz y viceversa están presentes de forma nativa en smartphones ,solo hace falta realizar una configuración adicional en los dispositivos. Tanto en Android como en iOS es posible utilizar comandos de voz, así como la lectura en voz alta de un texto .Las dos compañías brindan una API muy robusta, la cuales pueden ser usadas para el desarrollo de aplicaciones móviles sin tomar en cuenta las limitaciones de hardware a utilizar. Así como también es posible la reproducción de audios personalizados dentro de una determinada aplicación. En las páginas oficiales es posible encontrar información muy detallada acerca de las APIs.

Para el caso de iOS ,se llama Speech .Mediante la librería es posible traducir texto a voz y viceversa ya sea de :una grabación ,un video ,audio proveniente del micrófono y otros más .Según la documentación oficial se recomienda una conexión a internet estable(Apple Inc., 2021).

Para el caso de Android ,se tiene la función android.speech .La librería soporta la traducción de voz texto de igual manera que su contraparte .También soporta la traducción de audio en vico así como de el audio de videos en vivo(Google Developers, 2021).

Los sintetizadores de voz son mucho más adecuados para la creación de audios personalizados para una determinada acción ,pueden ser remplazados o modificados sin mucho esfuerzo, pero para la percepción humana pueden oírse sin emociones .

Por otro lado ,los asistentes de voz presentan estructuras mucho más complicadas, pero también personalizables como por ejemplo el caso de Alexa propiedad de Amazon ,es posible añadir nuevas Skills que permitan realizar acciones fuera del código base . Todo esto se realiza con la ayuda de APIs propias de la compañía y dispositivos de Amazon .Lógicamente se necesita el dispositivo en cuestión y allí es donde se realizan las configuraciones(Amazon, 2021).

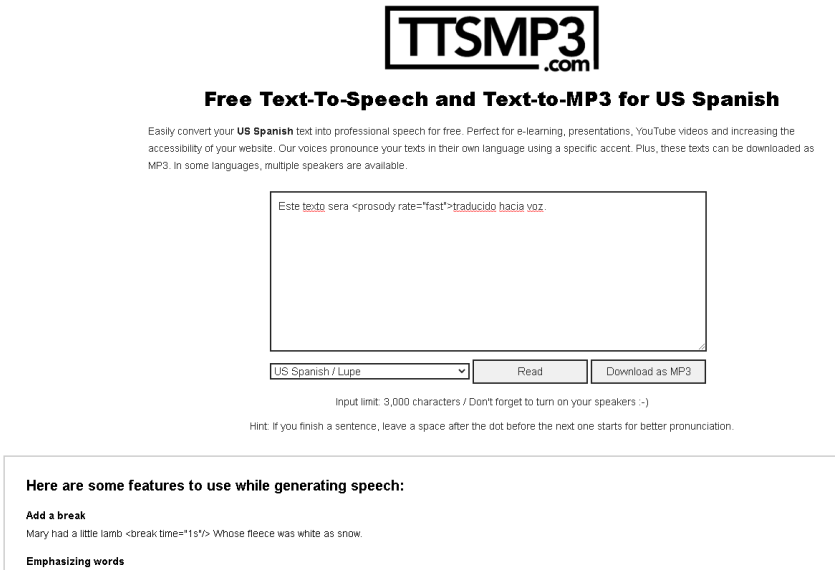
Los asistentes de voz terminan siendo herramientas que funcionan con dispositivos de la misma compañía o que función con la misma tecnología de comunicación ,por lo que no se recomienda utilizarlos como conexión de aplicaciones móviles personalizadas al menos que se tenga un amplio conocimiento en el tema de desarrollo de aplicación móviles.

En conclusión ,en el desarrollo de aplicaciones móviles es necesario el criterio del diseñador del sistema para la elección de sintetizadores de voz o conexiones con asistentes de voz.

2.2.4. Sintetizadores de voz en el mercado

Actualmente es posible utilizar un sintetizador de voz con un solo clic en un buscador. Algunas páginas web brindan el servicio ,en donde se escribe un determinado texto en lenguaje HTML y como resultado se obtiene un archivo de audio que puede ser descargado. Gracias a ello es posible realizar cualquier tipo de audio con cualquier contenido .En la se puede observar el sitio TTSMP3.com en donde se puede realizar audios con el sintetizador de voz .

Figura 10. Página TTSMP3.com para realizar audios personalizados

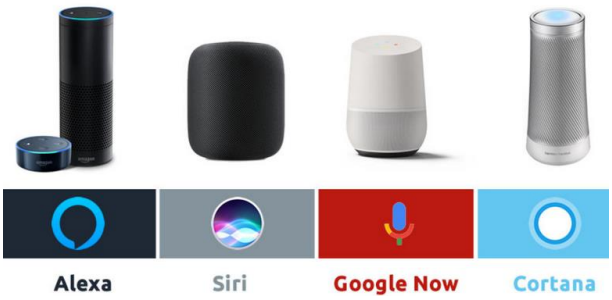


Fuente : Autoría

2.2.5. Asistentes de voz en el mercado

Una de las opciones mayormente implementada en smartphones, wearables y demás dispositivos inteligentes es un asistente virtual. La mayoría de estos puede configurarse para funcionar con la voz del usuario. Google por ejemplo tiene integrado un asistente dentro de la mayoría de los dispositivos Android que puede activarse para ser usado como asistente de voz. El asistente se puede configurar para reconocimiento de voz y otras opciones variadas dependiendo del usuario. Por su parte Amazon tiene a Alexa, nació con los altavoces Smart y es usada en algunas aplicaciones de domótica incluso puede vincularse a aplicaciones externas como YouTube, Netflix entre otras. Actualmente también se puede instalar en todo dispositivo con la versión de Android soportada. Otro asistente es Siri ,se trata de un asistente de la marca Apple tiene funcionalidades similares a los ya mencionados pero en productos de la marca .La mayor parte del mercado de asistentes virtuales esta entre las empresas mencionadas(Ville, 2021).En la se puede observar los asistentes de voz más comunes en el mercado.

Figura 11. Asistentes de voz más comunes en el mercado



Fuente:(Universidad de Salamanca, 2019). Los dispositivos activados por voz en España. https://universoabierto.org/2019/12/11/los-dispositivos-activados-por-voz-en-espana/

2.3. Domótica y su relación con IoT

2.3.1. Domótica

Se trata de un término acuñado en la década de 1990, en donde era usado para la combinación de tecnologías informáticas y telemáticas ,las cuales combinadas ayudaban a realizar tareas dentro del hogar. Hoy en día el termino es usado para referirse a : casas inteligentes ,home Networking ,casas automatizadas ,casas con sistemas embebidos ,casas adaptativas y otras más. En si el termine se refiere a la instalación de tecnologías como :sensores ,actuadores y monitores y cableado que permiten programas ,operar y hacer funcionar una serie de dispositivos y electrodomésticos dentro del hogar. Generalmente los sensores suelen ser muy pequeños por lo que pueden ser instalados en cualquier lugar ,ya sea en hub o en forma individual .Este tipo de sistemas se basan en los siguientes principios :

- Proveer comodidad dentro del entorno local
- Asegurar un nivel de confort personal cuando se usa
- Proveer seguridad a niveles superiores a lugares donde no existe este tipo de sistema
- Ahorro de energía
- Vigilancia y monitoreo a distancia
- Reducción de riesgos asociados a la vida diaria

Ciertamente algunos de los principios no siempre se cumplen ,pero son ciertos para la mayoría de sistemas existentes en la actualidad .(Sale, 2018). Originalmente este tipo de sistemas no tenían una salida al exterior ,pero con el avance de la tecnología y la aparición de IoT ,los sistemas tienen una salida a internet y pueden funcionar a la vez localmente .

2.3.2. IoT

Es una tecnología que permite crear redes de dispositivos informáticos, maquinas digitales y mecánicas, objetos, personas o animales provistas de un identificador único que poseen la habilidad de realizar la transferencia de información a través de la red sin intervención de tipo persona-persona (P2P) o persona-máquina (P2M),es decir el sistema o dispositivo que pueden recolectar información de los elementos conectados así como tomar decisiones en base a dicha información para realizar cierta actividad (C. Sharma & Gondhi, 2018).

2.3.3. Diferencias y similitudes entre IoT y Domótica

El termino casa domótica hace referencia a un conjunto de sistemas o dispositivos que se encuentran organizados con el fin de administrar determinadas actividades dentro del hogar de forma automática ya sea de forma local o externa ,para ello se utiliza funciones para ciertos escenarios las cuales son preprogramadas y pueden ser activadas con un orden o comandos .Por otro lado IoT es un término que abarca todos los aparatos y tecnologías que pueden manejar y recolectar datos del entorno para optimizar y simplificar la vida del usuario .Los sistemas domóticos solo pueden ser instalados por un experto en el área y en la mayoría de los casos interviene de forma directa en los sistemas tradicionales dentro de un hogar ,mientras los sistemas IoT son mucho más flexibles al utilizar sistemas ya presentes sin la necesidad de ser invasivos .Anteriormente ,los dos tipos de sistema debían instalarse de forma separada pero hoy en día el termino usado es Casas domóticas usando IoT. Este tipo de sistemas consiste en una serie de dispositivos como :control de luces ,seguridad ,entretenimiento y otros .Dichos

dispositivos están conectados entre sí en una red común usando una tecnología de comunicación inalámbrica ,generalmente existe un nodo central que asegura la administración de la red así como el acceso a internet (Kumar Pani & Pandey, 2021).

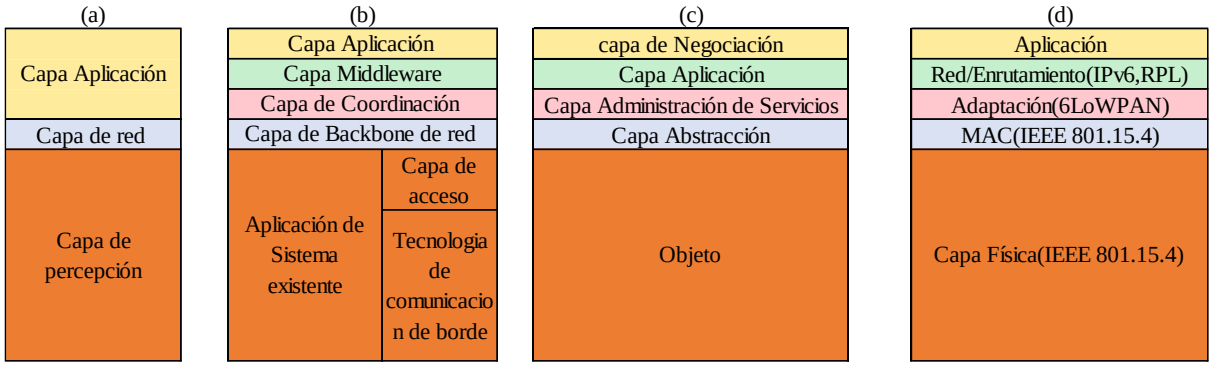
2.3.4. Stack de protocolos en un sistema IoT

IoT basa su funcionamiento en topologías y protocolos ya existentes, pero para la conexión de dispositivos con las plataformas industriales se utilizan protocolos de capa aplicación especiales que sirven dependiendo de las necesidades del usuario. Se lista algunos de los protocolos más típicos utilizados

- OPC UA: Protocolo sucesor de OPC, sirve para realizar la comunicación de PLC directamente a la nube. Se trata de un protocolo cliente-servidor, utilizada encriptación en capa transporte igualmente se firma en los mensajes que se envían. Es utilizado en redes ya existentes.
- HTTP(REST/JSON): Es un protocolo de hiper transferencia de hipertexto usado en muchas aplicaciones ya existentes que se está adaptando a los entornos de IOT, no presenta ningún tipo de seguridad.
- MQTT :Es un protocolo bajo el modelo publicación/suscripción diseñada para el funcionamiento sobre redes remotas .Es un protocolo de capa aplicación ,es usado en aplicaciones en donde no se conoce la infraestructura de la red y a diferencia de http este puede integrar seguridad a nivel de capa transporte.
- AMQP: Es un protocolo que sigue el mismo modelo de los protocolos presentados, es uno de los más robustos al tener un Stack completo. Su uso en la industria se concentra principalmente en el sector económico. específicamente en las transacciones.
- DDS : Es un protocolo enfocado en los equipos de borde de la red ,se trata de un protocolo que no necesita nodos centrales de red .Cada punto se comunica con UDP a nivel de capa transporte (C. Sharma & Gondhi, 2018).

Existen otros protocolos utilizados para IoT en capa aplicación, queda a criterio del administrador de red cual será utilizado para la aplicación o sistema. Se puede observar un ejemplo en la Figura 12 en donde se nota que existen varios modelos, los cuales varían de acuerdo con factores como: tecnología de comunicación inalámbrica, numero de capas y otras más. Por lo que incluso se puede aplicar cambios para construir una arquitectura personalizada.

Figura 12. Stack de protocolos en un dispositivo IoT.



Nota : (a) Modelo de tres capas ,(b) modelo de 5 capas con capa de percepción desglosada ,(c) modelo de 5 capas ,(d) modelo de 5 capas low energy .

Fuente : (Soni et al., 2017), Internet of Things and Wireless Physical Layer Security: A Survey. En Lecture Notes in Networks and Systems (Vol. 5, pp. 115–123). Springer, Singapore. https://doi.org/10.1007/978-981-10-3226-4_11

2.3.5. RTOS (Real Time Operating System)

Se trata de sistemas operativos especializados para realizar tareas en tiempo real por lo que se asegura tiempos de respuesta cortos a eventos, además permiten funcionalidades multitareas, así como soporte a librerías y drivers de un sin número de dispositivos. Todo esto se realiza dentro de un solo CPU, la realización de tareas se basa en la activación de ciertos parámetros que son configurados por un desarrollador Para utilizar un RTOS en proyectos se debe tomar en cuenta los siguientes parámetros:

- Planificación de tareas y sus respectivos eventos para no sobrecargar el sistema
- Cambio de tareas guardando el estado de la tarea actual.
- Disponibilidad del CPU en base a las tareas planificadas.

La utilización de hardware para la planificación de tareas puede reducir los tiempos de cambio de tareas pero sobrecargar al sistema(Oosako et al., 2019).Los sistemas IoT pueden

usar o no los RTOS dependiendo de la arquitectura a utilizar pero lo sistemas embebidos generalmente utilizan siempre un RTOS. Actualmente se puede elegir el RTOS a utilizar de acuerdo al sistema que se quiere construir, tecnología de comunicación, entre otros, es decir el desarrollador elegirá el adecuado de acuerdo a sus necesidades. Entre algunos ejemplos de un RTOS se puede listas: Zephyr ,FreRTOS,SAFERTOS,VxWorks,QNX,eCos,RTLinux,RTX entro otros más.

2.3.6. Servicios en la nube

Los servicios en la nube generalmente dependen del tipo plan o infraestructura que se planea montar en la nube ,por lo que el presupuesto para construir el sistema juega un papel muy fundamental .Por ello con el paso del tiempo han surgido infraestructuras que los proveedores ponen al servicio de los usuarios para brindar soluciones generales o específicas ,todo depende de la necesidad del cliente .Los servicios son variados y como se mencionó anteriormente ,el usuario decidirá que usar o que es más adecuado para el éxito y desarrollo de su proyecto. Existen soluciones de paga como de uso libre ,la diferencia suele radicar en la limitación de componentes en planes gratuitos.

2.3.6.1. laaS (Infraestructura como servicio)

El proveedor del servicio se encarga de la administración total del servicio en la nube, entonces el cliente solicita los servicios a través de la conexión a internet. Es decir que el usuario se encarga de la gestión de software mientras el proveedor se encarga de la gestión de hardware. Cualquier servicio adicional para implementar estará limitado por el usuario

2.3.6.2. PaaS (Plataforma como servicio)

En este caso el proveedor de servicios se encarga de la gestión de hardware y software ,mientras el usuario accede al servicio mediante una plataforma .Es útil para desarrolladores que solo necesitan realizar y desarrollar aplicaciones sin tener conocimiento previo en telecomunicación o desarrollo de aplicaciones con software .La mayor desventaja de este tipo

de servicios es la pobre personalización que ofrecen ,a menos que el usuario desde el inicio coloque los requerimientos adecuados ,caso no muy posible considerando el público objetivo.

2.3.6.3. SaaS (Software como servicio)

En este caso se ofrece una plataforma únicamente de software que se encargara de la gestión de todos los servicios que se van a brindar en la nube. Generalmente las aplicaciones son del tipo web o móviles. El usuario la gestiona a través de una API brindada por el proveedor del servicio. Este tipo de servicio es de los más robustos en el mercado siendo limitado únicamente por la infraestructura física que el cliente quiera montar en el desarrollo de su proyecto (Encargo, 2018).

2.3.6.4. BaaS(Backend como servicio)

Este servicio es de los más recientes en el mercado ,en este caso el proveedor del servicio ofrece la estructura de Backend como servicio .Es decir ofrece todas las funciones internas para que una aplicación funcione, dejando al desarrollador la función de construir la estructura de Front End .En conclusión este tipo de servicio permite usar las funciones de cualquier tipo de servicio ya sean :bases de datos ,servidores y otros .El desarrollador solo deberá diseñar aplicaciones que usen estos servicios para mostrar información (Google Cloud Platform, 2022).Este tipo de servicio en la nube es adecuado para el desarrollo de este proyecto.

2.3.7. Accesibilidad en IoT para personas con discapacidad

La autonomía e independencia son factores esenciales en el día a día de las personas con discapacidad visual ,las actuales soluciones comerciales pueden solucionar sus necesidades con la ayuda de varias tecnologías de comunicación inalámbrica y dispositivos espaciales .Pero se debe realizar un análisis acerca de : ¿Realmente se está logrando una solución o solo se la está imponiendo ?.Para obtener una respuesta se debe analizar tres aspectos esenciales :

- Hardware y software utilizado .-Se debe tomar en cuenta en dónde y cómo funcionará un sistema IoT. Por ejemplo una aplicación móvil podrá controlar el sistema de forma

remota y a su vez será posible presentar resultados mediante una página web. Es posible también controlar el sistema con uno o varios dispositivos de control ,también se debe tomar en cuenta que sensores utilizar para la lectura de datos y donde se van a ser colocados .Incluso el precio de los materiales también es un factor primordial.

- Expectativas del usuario .-El usuario es el actor más importante en el posible desarrollo del sistema ,será el cual elegirá que función le podrá ser útil y cual no. Según (Leporini & Buzzi, 2018) ,las personas con discapacidad buscan las siguientes funciones :Sistema de alarmas con sensores de presencia ,administración de puertas ,administración de jardines ,soporte en la cocina ,sistema de control sencillo ,administración de etiquetas en objetos cotidianos y ropa. Ciertamente las opciones presentadas no son mandatorias pero pueden ser útiles para la modelación de un sistema IoT para una persona con discapacidad visual.
- Sugerencias e indicaciones acerca de posibles funcionalidades que exploten al máximo la funcionalidad del usuario para lograr accesibilidad .Es posible que después de la implementación del sistema el usuario realice sugerencias de diseño de la interfaz de presentación de información para que sea mucho más sencillo de usar para una persona con discapacidad visual ,en base a la interacción con el sistema.(Leporini & Buzzi, 2018). Por ello ,la elección de la tecnología de comunicación inalámbrica del sistema será clave para que el usuario realice el menor esfuerzo posible ,especialmente cuando controle el sistema .

2.4. Redes malladas en IOT

Las redes en malla son redes con la capacidad de organización dinámica y configuración automática propia de tal forma que se establece convergencia entre todos los nodos de la red. A diferencia de las redes en estrella, las redes en malla no necesitan un nodo central. Esta estructura de red no es nada nueva, pero con el boom de IOT se convirtió en la que más se

adecuaba a las necesidades de infraestructura y conectividad. Mientras dentro de las mayores desventajas se tiene que: políticas de seguridad para cada uno de los nodos de la red, no escalabilidad dependiendo de la topología utilizada, nuevos protocolos y equipos para trabajar en redes funcionales de propietarios, detección de nodos caídos y latencia.

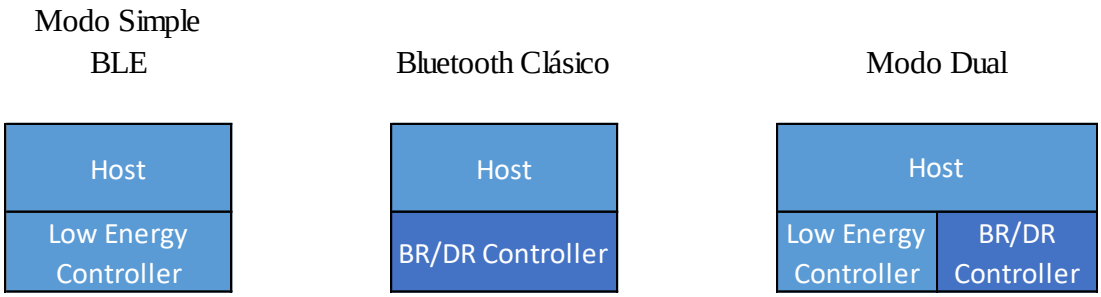
Para la solución de los problemas se puede utilizar ya sea redes de backbone en Mesh o redes hibridas ,para mejorar el rendimiento y efectividad de las redes (Liu et al., 2017).

2.5. Antecedentes de Bluetooth Mesh

2.5.1. Bluetooth Clásico

Hay que recalcar que las trata de la especificación de Bluetooth desde la version 1 a la versión 3, también es llamado Bluetooth BR/EDR. Se transmite datos a través de 79 canales con un espaciado de 1 MHz que operan en la banda ISM de 2.4 GHz, actualmente es usada principalmente para transferencia de datos y streaming de audio en topologías punto a punto. Las modulaciones utilizadas son GFSK, DQPSK con 45 grados y 8DPSK además se utiliza FHSS para la transmisión física a través de radio. La topología típica es piconet basándose en comunicaciones punto a punto donde siempre se define un maestro y un esclavo, así como se utilizan scatternet. Versiones más recientes de Bluetooth se basan en muchos de los conceptos sentados por Bluetooth Clásico ,por lo que es posible que los nuevos dispositivos en el mercado soporten Bluetooth Clásico y Bluetooth Low Energy ,para la intercomunicación entre las dos versiones se utiliza un dispositivo intermedio que actúa como un mediador entre las dos versiones de Bluetooth como se puede observar en la Figura 13 por lo que no es posible la comunicación entre las respectivas versiones .A parir de la version 5.0 se tienen dispositivos duales (Bluetooth S.I.G., 2021).

Figura 13. Estructura lógica simple para: Bluetooth Clásico, Bluetooth Low Energy y dispositivos duales.



Fuente : (Argenox Technologies LLC, 2020), Introduction to Bluetooth Classic. Recuperado el 23 de noviembre de 2021, de https://www.argenox.com/library/bluetooth-classic/introduction-to-bluetooth-classic/

Por lo que las versiones clásica y low energy no son compatibles entre sí, un dispositivo a partir de la version 4.2 puede detectar dispositivos con las dos versiones, pero obviamente se escogerá la version adecuada de acuerdo al rol necesario.

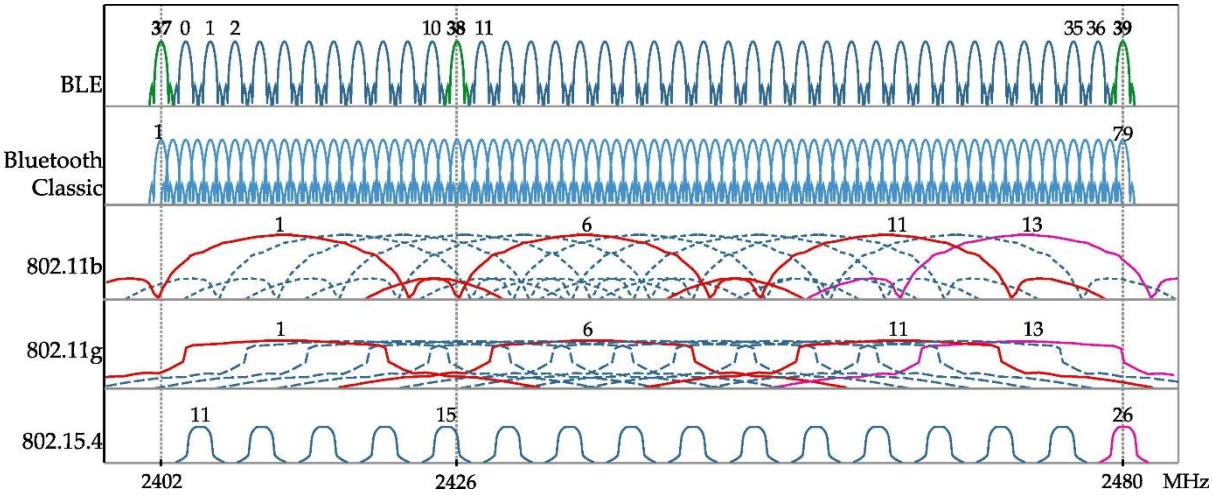
2.5.2. Bluetooth Low Energy

Se trata de la especificación de Bluetooth a partir de la version 4 o superior opera también en la banda ISM de 2.4 GHz, emplea la técnica de salto de frecuencias para reducir la interferencia por agentes externos. Se utiliza 40 canales de comunicación con un espaciamiento de 2 MHz, de los cuales 3 canales son usados para señalización y 37 canales se utilizan para tráfico de datos. Se utiliza solamente modulación GFSK junto con FHSS para transmisión, la estructura de red de punto a punto, así como Broadcast son soportadas, pero se añade la opción de red en Malla (Mesh) se añaden características para detectar posicionamiento de los nodos de la red mediante: presencia, proximidad, dirección y distancia a través de Aprovisionamiento ,RSSI ,Angulo de entrada/ángulo de salida y posición respectivamente. Las versiones de Bluetooth que soportan las características son las versiones 4 o superiores .A diferencia del Bluetooth clásico esta tecnología esta optimizada para un consumo de energía mucho menor que la version llamada clásica por lo que es muy utilizada en aplicaciones IoT en conjunto con otras tecnologías y arquitecturas de red debido a las limitantes de Bluetooth(Bluetooth S.I.G., 2021).

Una de las mejoras al Bluetooth clásico fueran la ubicación de los canales primarios de comunicación utilizados dentro de la banda ISM, si se le compara con Wi-Fi. Los canales

37,38 y 39 se ubican en 2402, 2426 y 2480 MHz respectivamente .Si se realiza una comparativa con los canales típicos que usa Wi-Fi se puede notar que no existe una sobreposición notable de las frecuencias utilizadas .Hay que recalcar que generalmente se usa los canales utilizados por Wi-Fi son los canales 1 y 6 (Cao et al., 2021).

Figura 14.Ubicación de los canales usados en Bluetooth Low energy , Wi-Fi y otras en la banda ISM.



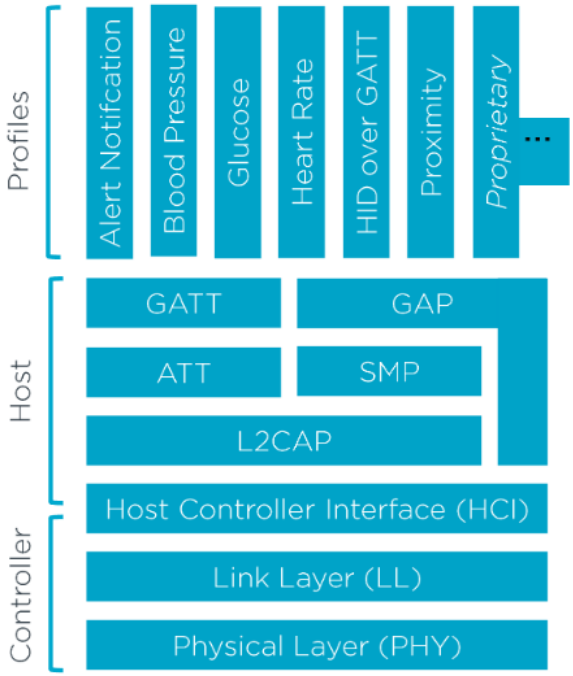
Fuente :(Valenzuela-Pérez et al., 2022)

Fuente : (Cao et al., 2021)Measurement and Analysis of RSS Using Bluetooth Mesh Network for Localization Applications. Network 2021, Vol. 1, Pages 315-334, 1(3), 315–334. https://doi.org/10.3390/NETWORK1030018

2.5.2.1. Stack de protocolos de Bluetooth Low energy

Las conexiones se realizan desde un maestro a uno o varios esclavos, generalmente en una configuración en estrella. El stack de protocolos se distribuye en tres capas : aplicación ,host y controlador .Cada una de estas secciones tiene varias capas (Nordic Semiconductor, 2020).Se puede observar el stack de protocolos de Bluetooth Low Energy en la Figura 15.

Figura 15. Arquitectura de Bluetooth Low Energy.



Fuente : (Nordic Semiconductor, 2020), Introduction to Bluetooth Low Energy. Recuperado el 16 de noviembre de 2021, de https://webinars.nordicsemi.com/introduction-to-bluetooth-low-6

Dentro de la capa de aplicación se tiene los perfiles, esta capa define como los dispositivos pueden comunicarse con otros, así como el descubrimiento de nuevos dispositivos. Cada uno de los perfiles están hechos para una determinada aplicación en específico que puede ser propietaria, así como libre, cada perfil tiene su propia especificación. La capa de host contiene las capas superiores del stack de protocolos con varias subcapas bastante complejas; dentro de estas subcapas se define: protocolo para encontrar modelos y atributos de los dispositivos (ATT), seguridad en los enlaces (SMP), atributos de los perfiles tales como características y servicios (GATT), roles de los perfiles, así como conectar y descubrir elementos de la red (GAP).

Finalmente, la capa de controlador define dos subcapas: la capa física y la capa enlace. La subcapa física define como dos equipos envían bit en un enlace de radio mientras la subcapa de enlace define: estados, direcciones y formato del paquete.

Para finalizar esta sección se presenta una tabla de resumen de las versiones de Bluetooth, se puede observar dicha tabla en la Tabla *1*:

Tabla 1. Comparación entre Bluetooth Clásico y Bluetooth Low Energy .

Parámetros	Bluetooth Clásico	Bluetooth Low Energy
Uso	Intercambio de grandes flujos de datos y audio.	Sensores, control de dispositivos.
Velocidad de trasmisión	1 hasta 3 Mbps sin soporte para ahorro de energía	125 Kbps hasta 2 Mbps
Canales de operación (datos)	79 canales con espaciamiento de 1 MHz	40 canales con espaciamiento de 2 MHz
Canales de operación (scanning)	32 canales	3 canales (canales de señalización)

Fuente : (Bluetooth S.I.G., 2021), Bluetooth Technology Overview | Bluetooth® Technology Website. Recuperado el 16 de noviembre de 2021, de https://www.bluetooth.com/learn-about-bluetooth/tech-overview/

2.6. Estándar de comunicación de red :Perfil Bluetooth mesh

2.6.1. Definición

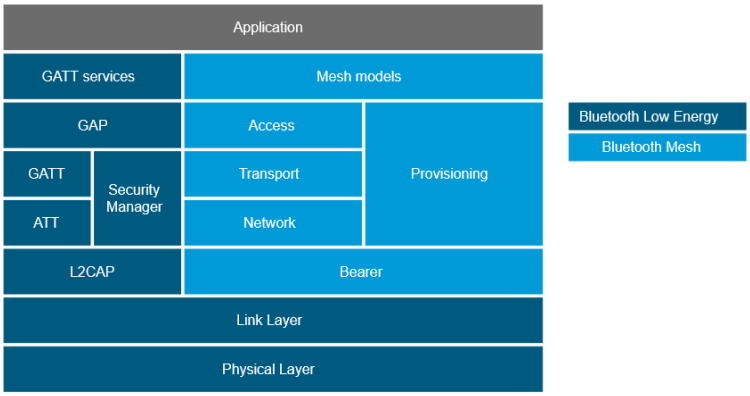
Es una tecnología de red lanzada por la SIG de bluetooth en el año 2017, se basa en la especificación de Bluetooth Low Energy como base ,específicamente las capas más bajas de dicha tecnología. Consiste en una topología de nodos con comunicación multicast entre ellos (many to many). En un escenario donde se tenga n nodos entonces todos los nodos podrán comunicarse entre sí, incluso si los dos nodos que van a participar de la comunicación no se encuentran en un rango directo de comunicación entre so. Este fenómeno se da porque los mensajes realizan saltos multihop a través de la red, cada mensaje enviado llegara a su destino sin importar si un nodo el trayecto caiga, es decir se enviará a través de otro camino que asegure su llegada. La tecnología plantea su uso para aplicaciones de IOT con un consumo energético más bajo asimismo en equipos de diferentes marcas debido a que se trata de un estándar no propietario, la transmisión de información puede superar fácilmente obstáculos e interferencia gracias al tamaño de la red. En cuanto a seguridad la SIG plantea estándares robustos de seguridad de acuerdo a tres ámbitos : seguridad en aplicación ,red y dispositivo (Darroudi et al., 2020).

El estándar de comunicación de red Bluetooth Mesh funciona con las versiones de Bluetooth a partir de la 4.x, pudiendo coexistir con las nuevas versiones del core de Bluetooth debido a que utiliza el bloque de host, capa en donde se realizaron mejoras para versiones a partir de la 5.x.

2.6.2. Stack de protocolos del estándar de comunicación de red Bluetooth Mesh

Basa su pila de protocolos sobre la especificación de Bluetooth Low energy, para el caso de Bluetooth Mesh la capa de host es remplazada por un bloque totalmente nuevo mientras las capas de aplicación y controlador se mantienen sin cambios. Hay que comprender que se trata de una nueva tecnología de red mas no de una nueva tecnología inalámbrica, por lo que algunos conceptos de Bluetooth Low Energy aún son usados. En la se puede observar una comparación del stack de protocolos de Bluetooth Low Energy con Bluetooth mesh.

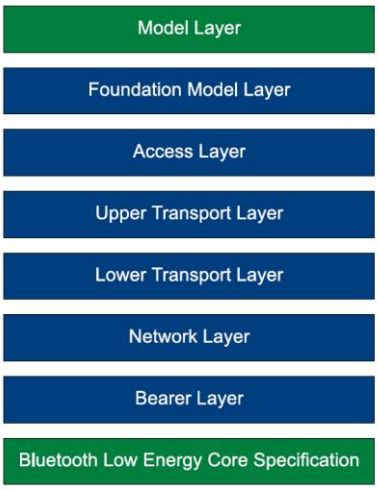
Figura 16. Comparación de la arquitectura de Bluetooth mesh con Bluetooth Low Energy.



Fuente :(Nordic Semiconductor, 2018a), Basic Bluetooth Mesh concepts. Recuperado el 17 de noviembre de 2021, de https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.meshsdk.v1.0.1%2Fmd_doc_introduction_basic_concepts.html

El nuevo bloque puede dividirse aún más en las capas que se pueden observar en la Figura *17* .Hay que recalcar que la nueva capa de host de Bluetooth mesh no es compatible con la capa de host de Bluetooth Low energy .Pero existe una función que permite dentro de la nueva capa que permite activar la función de legacy(Nordic Semiconductor, 2018a) .

Figura 17. Arquitectura del nuevo bloque añadido en bloque host.



Fuente :(Bluetooth S.I.G., 2019a). Mesh Profile 1.0.1 . Recuperado de https://www.bluetooth.com/specifications/specs/mesh-profile-1-0-1/

2.6.3. Capas de la arquitectura de Bluetooth Mesh

2.6.3.1. Model Layer

Capa que define ls modelos que son utilizados para la operación típica en escenarios específicos definidos en el perfil de la norma. Se definen los comportamientos, así como la posterior implementación de estos.

2.6.3.2. Foundation Model Layer

Capa que define los estados, mensajes y modelos requeridos para realizar la configuración y administración de una red Mesh.

2.6.3.3. Access Layer

Capa que define los mecanismos de como las capas superiores usaran las capa Upper Transport Layer, se define el formato de la información, los mecanismos de encriptación y desencriptación también se verifica que los datos de la capa superior sean para la red y aplicación a la que pertenece el nodo para enviar los datos a capas superiores.

2.6.3.4. Upper Transport Layer

Capa que se encarga del cifrado y descifrado así con la autenticación de la información que pasan por la capa de acceso, adicional se define como se realiza el transporte de los mensajes de control entre nodos normales y nodos Friend.

2.6.3.5. Lower Transport Layer

Capa que define como la Upper Transport Layer segmenta y reensambla los mensajes, si el paquete tiene más de 31 bytes se realiza el proceso de segmentación para el envío de mensajes considerados como largos al destino a través de multicast. El proceso contrario se debe realizar en el nodo de destino.

2.6.3.6. Network Layer

Capa que define como se envían los mensajes hacia uno o más nodos y decide cuando se aceptan o rechazan los paquetes. De igual forma define el formato de los PDU de la ca transporte para que puedan ser transportados a través de la bearer Layer.

2.6.3.7. Bearer Layer

Capa que define como se intercambia los mensajes por los varios tipos de nodos de la red ,también permite la conexión con capas superiores con la especificación de Bluetooth Loe energy .Es decir brinda compatibilidad para que la especificación Bluetooth Mesh funciones a través del hardware de Bluetooth .Se define dos Bearers : advertising bearer y GATT Bearer .El advertising bearer utiliza las funciones de scanning y publicity GAP de Bluetooth Low Energy para la transmisión y recepción de PDUs mientras que el GATT Bearer permite que dispositivos que no pueda funcionar con el advertising bearer realiza el intercambio de información con los nodos de la red .Los nodos Proxy poseen los tipos mencionados .

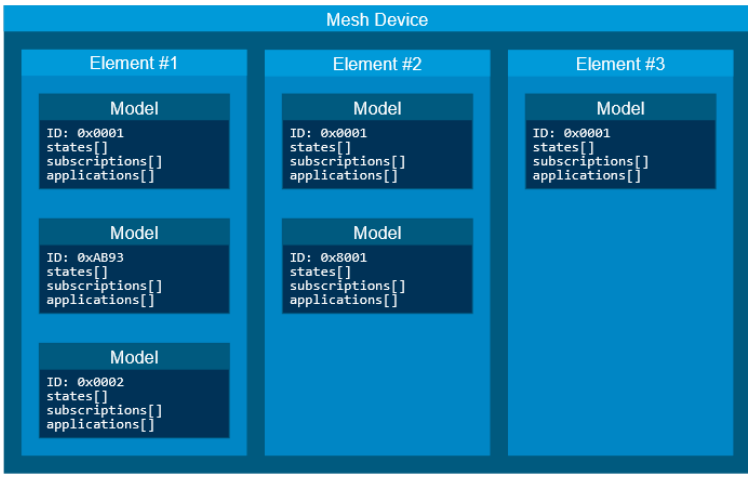
2.6.3.8. Bluetooth Low Energy Layer

Esta capa indica que se debe trabajar bajo un dispositivo que soporta toda la pila de Bluetooth Low Energy con todos lo que esto significa, se utiliza las funciones de escaneo y aprovisionamiento para enviar y recibir mensajes dentro de la red entro otras características propias del stack de bluetooth Low Energy.

2.6.4. Elementos, nodos, modelos y estados en Bluetooth mesh

Un elemento es una entidad que se le puede asignar una dirección dentro de un nodo, cada nodo tiene al menos un elemento y puede existir varios elementos secundarios a la vez dentro de dicho nodo. La estructura y el número de los elementos es fija dentro de un nodo y no cambiara mientras un nodo pertenezca a una red. Si el nodo es reconfigurado se debe realizar obligatoriamente el proceso de aprovisionamiento, cada elemento tendrá una dirección única. Cada uno de los elementos puede poseer uno o varios modelos ,los modelos a su vez definirán estados de acuerdo al tipo de modelo .Dichos estados pueden viajar a través de la red para comprobar el estado de un nodo(Nordic Semiconductor, 2018a), toda esta estructura lógica se define en la capa de acceso .Este esquema esta realizado de tal forma que permite estandarizar la comunicación entre dispositivos sin importar la marca ,se puede observar esta estructura lógica en la Figura 18

Figura 18. Estructura lógica de un nodo dentro de una red Bluetooth mesh.



Fuente :(Nordic Semiconductor, 2018a), Basic Bluetooth Mesh concepts. Recuperado el 17 de noviembre de 2021, de https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.meshsdk.v1.0.1%2Fmd_doc_introduction_basic_concepts.html

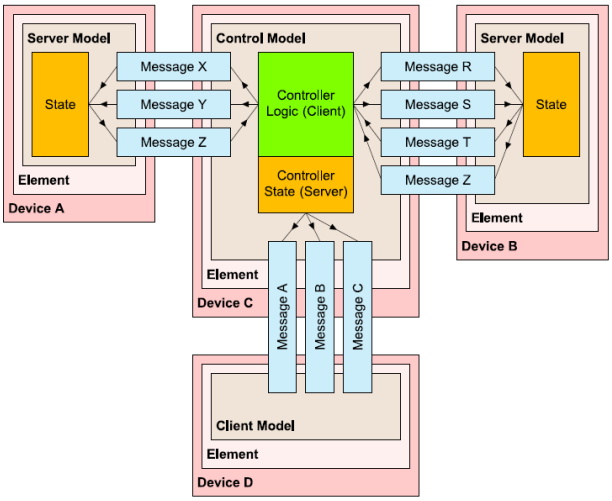
2.6.5. Modelos de comunicación dentro de Bluetooth mesh

Bluetooth Mesh usa los GATT Bearer, que para ciertos casos permiten que los nodos compartan una estructura común de información. Dentro de la pila de Bluetooth estas entidades se llamaban modelos mientras que para Bluetooth Mesh se hacen llamar modelos, un modelo

es un conjunto de estado y mensajes que funciona en comportamiento y servicios específicos que se especifican en el estándar. Cada uno de estos modelos están organizados en elementos de tal forma que cada mensaje es administrado por un elemento y a su vez un elemento será propio de un nodo. Se especifican tres tipos diferentes de modelos, recalcando que un dispositivo puede contener los tres tipos de modelos

- Modelo de servidor: está compuesto de uno o más estados que abarcan uno o más elementos. Se define una serie de mensajes que son obligatorios para transmisión y recepción, así como lo que se debe realizar al finalizar la transmisión y recepción de mensajes
- Modelo de cliente: De igual forma el modelo define mensajes obligatorios y opciones que un cliente usa para solicitar, cambiar o consumir el estado de un servidor, dicho estado del servidor está definido por el modelo anterior dado que el cliente no tiene un estado.
- Modelo de control: este modelo puede contener un modelo de cliente para la comunicación con otro modelo de servidor o un modelo de servidor para la comunicación con otro modelo cliente. Un modelo de control funciona bajo una lógica de control que define un conjunto de reglas y requerimientos para realizar la coordinación de las interacciones entre los modelos a los cuales se conecta el modelo de control.

Figura 19. Modelo de control de comunicación.



Fuente : (Bluetooth S.I.G., 2019a), Bluetooth Mesh Profile Specification, Revision v 1.0.1 Recuperado de https://www.bluetooth.com/specifications/specs/mesh-profile-1-0-1/

Existen modelos que no se define por el SIG y son propios de los fabricantes de los equipos miembros de la SIG. Se define un conjunto de hasta 52 modelos de una red Mesh estándar. Cada uno de estos modelos tendrá su respectiva coincidencia para cada elemento .Según la Figura 20 se tiene cuatro grupos de modelos ,se tiene modelos :genéricos ,para sensores ,para tiempo y escenarios y para iluminación(Bluetooth S.I.G., 2017) .Estos modelos son genéricos por lo que pueden ser aplicados a dispositivos de acuerdo al funcionamiento deseado dentro de la red .Actualmente se pueden usar códigos prefabricados para realizar pruebas de cada uno de los modelos.

Figura 20. Modelos para Bluetooth Mesh

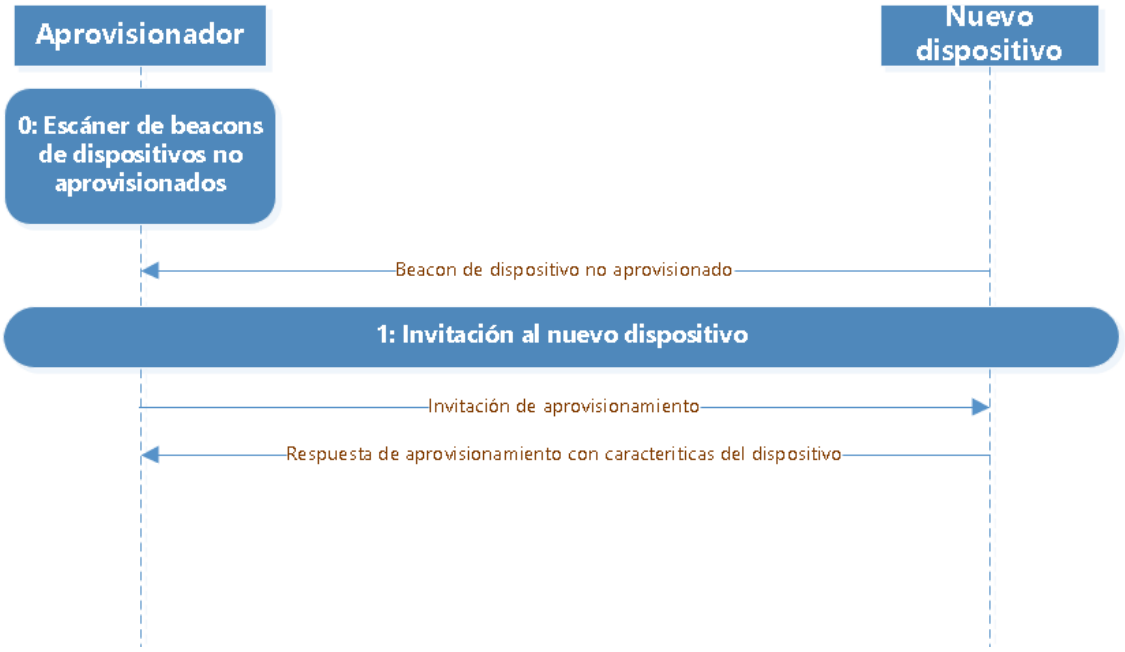
Fuente:(Woolley, 2019). Bluetooth Mesh Models Technical Overview, (March), 1–41. Recuperado de://www.bluetooth.com/wp-content/uploads/2019/04/1903_Mesh-Models-Overview_FINAL.pd

2.6.6. Aprovisionamiento de elementos dentro de una red Bluetooth Mesh.

Se trata del proceso que debe seguir un nodo al ingresar a una red de este tipo, se considera que el nodo no ha ingresado antes y de acuerdo a la configuración de la red este proceso puede ser llevado a cabo por cualquiera de los nodos que ya están dentro de la red. Un nodo puede generar mensajes publish hacia una direccion unicast o un grupo de direcciones, así como una dirección virtual, y los nodos que van a participar dentro de este proceso realizan el proceso de suscripción a estas direcciones. Es decir, se sigue este proceso siempre que se va a realizar la agregación de un nuevo dispositivo a una red Mesh, dicho proceso sienta las bases de la seguridad que usa Bluetooth Mesh, este proceso cuenta con 5 etapas donde el dispositivo aprovisionador logra que un dispositivo esclavo obtenga datos que le van a permitir ser parte de una red Bluetooth Mesh. Los pasos son los siguientes

- Beaconing: esta fase funciona de manera similar que en Bluetooth Low energy, se realiza el mecanismo de advertising. El dispositivo no aprovisionado se anuncia como un beacon cuando recibe el advertising bearer. Pero cuando se utiliza el GATT bearer, se activa un servicio llamado Mesh Provisioning Service, de tal forma que el servicio es detectado en el nodo que realiza el proceso de suscripción(aprovisionador).
- Invitación: el nodo aprovisionador envía un PDU llamado Provisioning invite y el dispositivo responde con el PDU llamado Provisioning capacibilities, dentro del PDU invite se indica el tiempo que será necesario para que el dispositivo no suscrito envíe datos o similar. Mientras el PDU de capacibilities indica el número de elementos del dispositivo. La información proporcionada servirá para que el nodo aprovisionador conozca todas las cualidades y capacidades del nodo que va a entrar en la red. En la Figura 21 se observa un diagrama de secuencia del proceso de Beaconing e Invitacion.

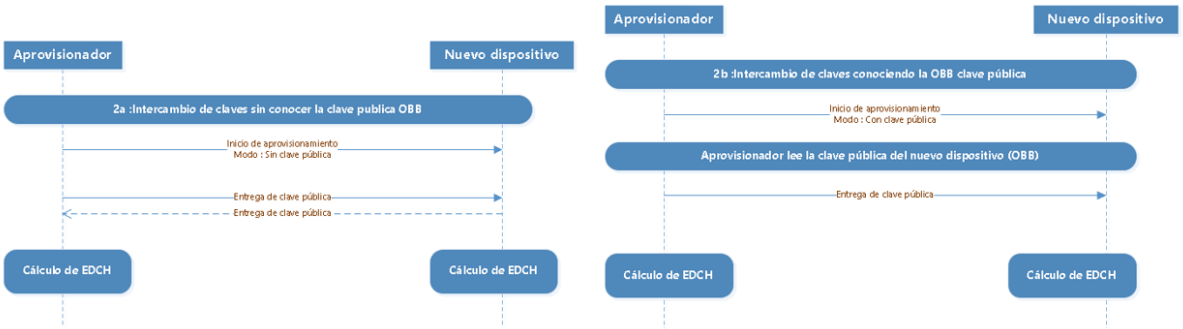
Figura 21. Proceso de invitación del aprovisionamiento a un nuevo nodo.



Fuente : (Bluetooth S.I.G., 2019a), Bluetooth Mesh Profile Specification, Revision v 1.0.1 Recuperado de https://www.bluetooth.com/specifications/specs/mesh-profile-1-0-1/

- Exchange Publics Keys: Bluetooth Mesh utiliza una combinación de cifrados simétricos y asimétricos para permitir que los nodos de ahorro de energía puedan enviarlas sin problemas debido a la falta de potencia. Se utiliza el ECDH para el intercambio de claves mediante el uso de un archivo OBB. Dentro de este proceso existen dos casos, el primero es cuando el aprovisionador no conoce la clave publica (no se tiene información del archivo OBB) y el segundo es cuando dicha clave es conocida.

Figura 22. Proceso de intercambio de claves de un nuevo dispositivo.

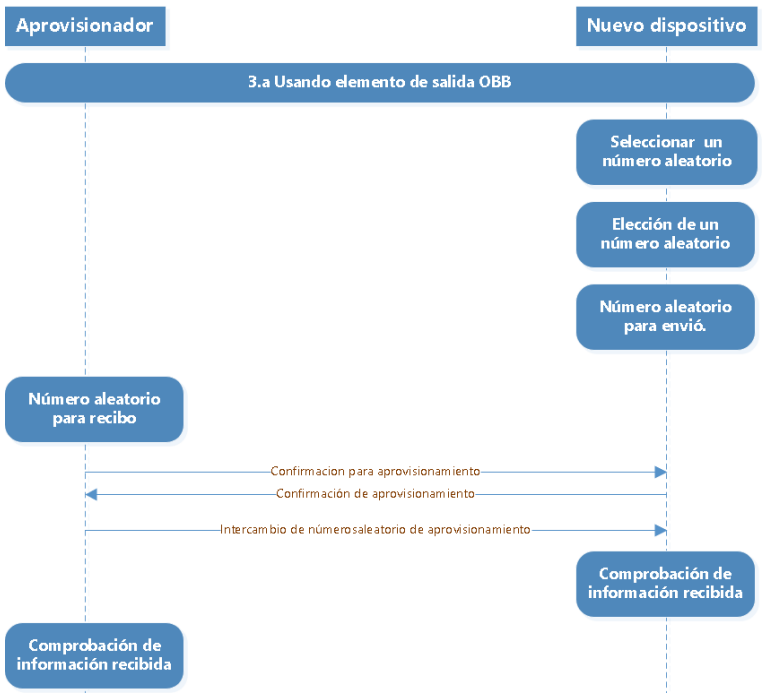


Nota: La imagen de la izquierda muestra el proceso cuando no se conoce la clave publica (2a), la imagen de la derecha muestra el proceso cuando existe una clave publica conocida(2b).

Fuente : (Bluetooth S.I.G., 2019a), Bluetooth Mesh Profile Specification, Revision v 1.0.1 Recuperado de https://www.bluetooth.com/specifications/specs/mesh-profile-1-0-1/

- Authentication: Después de realizar el proceso de intercambio de claves se procede a realizar una autenticación por medio de la elección de un numero aleatorio, una vez registrado este número entonces el nodo aprovisionador confirma y genera un valor. Se puede utilizar este método de forma invertida. Este proceso se basa en el rol que el nodo va a tener dentro de la red, por lo cual se tiene tres casos específicos.Cuando se trata de un elemento que realizara acciones de salida como encender elementos, así como presentar información, se puede observar el proceso en la Figura 23.

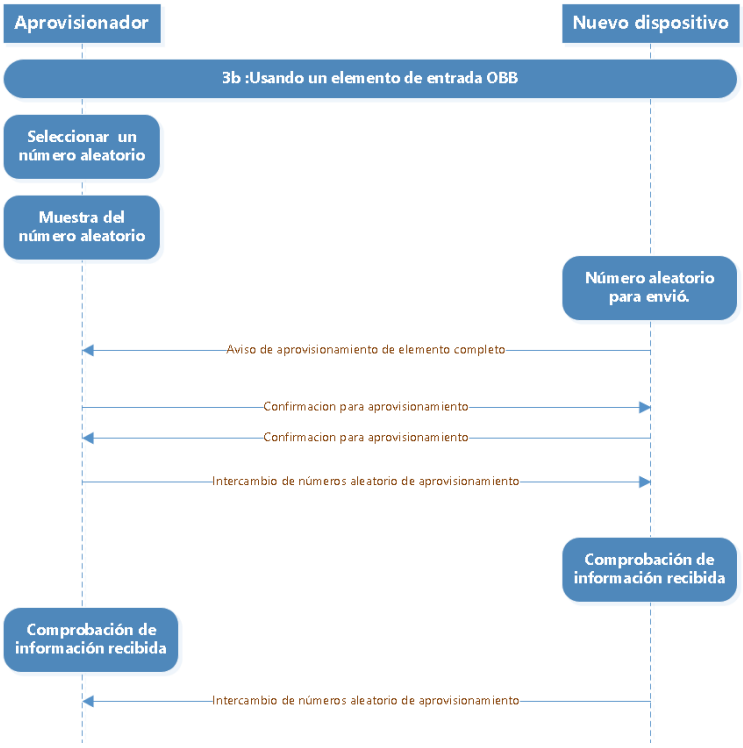
Figura 23. Proceso de invitación de autenticación de un nuevo nodo cuando se trata de un elemento Output.



Fuente : (Bluetooth S.I.G., 2019a), Bluetooth Mesh Profile Specification, Revision v 1.0.1 Recuperado de https://www.bluetooth.com/specifications/specs/mesh-profile-1-0-1/

Cuando se trata de elementos de lectura es decir elementos que ingresen datos, se puede observar el proceso en la Figura 24

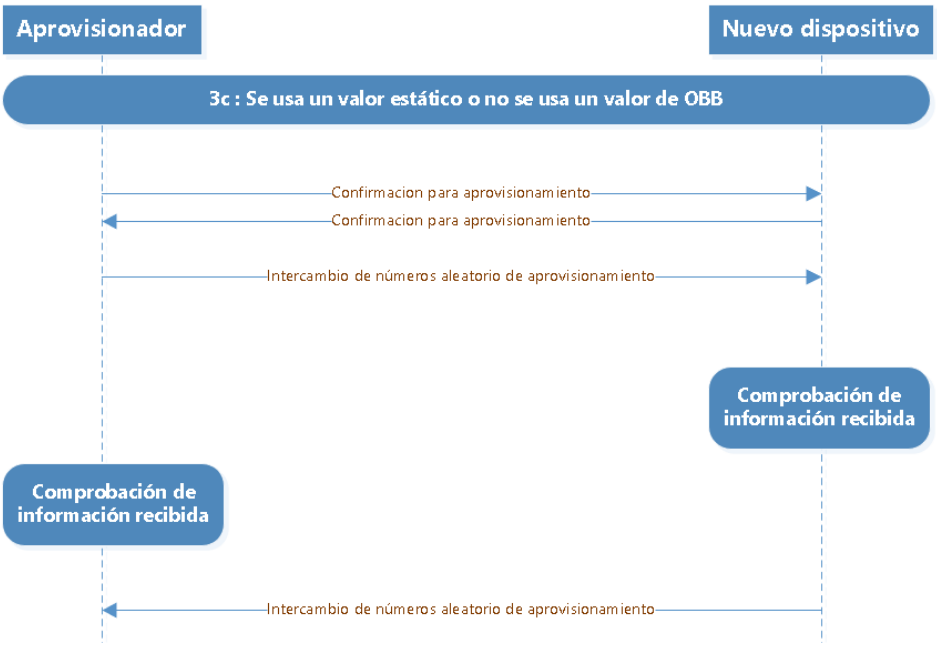
Figura 24. Proceso de invitación de autenticación de un nuevo nodo. cuando se trata de un elemento Input.



Fuente : (Bluetooth S.I.G., 2019a), Bluetooth Mesh Profile Specification, Revision v 1.0.1 Recuperado de https://www.bluetooth.com/specifications/specs/mesh-profile-1-0-1/

Cuando se usan un valor de OBB estático o no se usan un valor, se puede observar el proceso en la Figura 25.

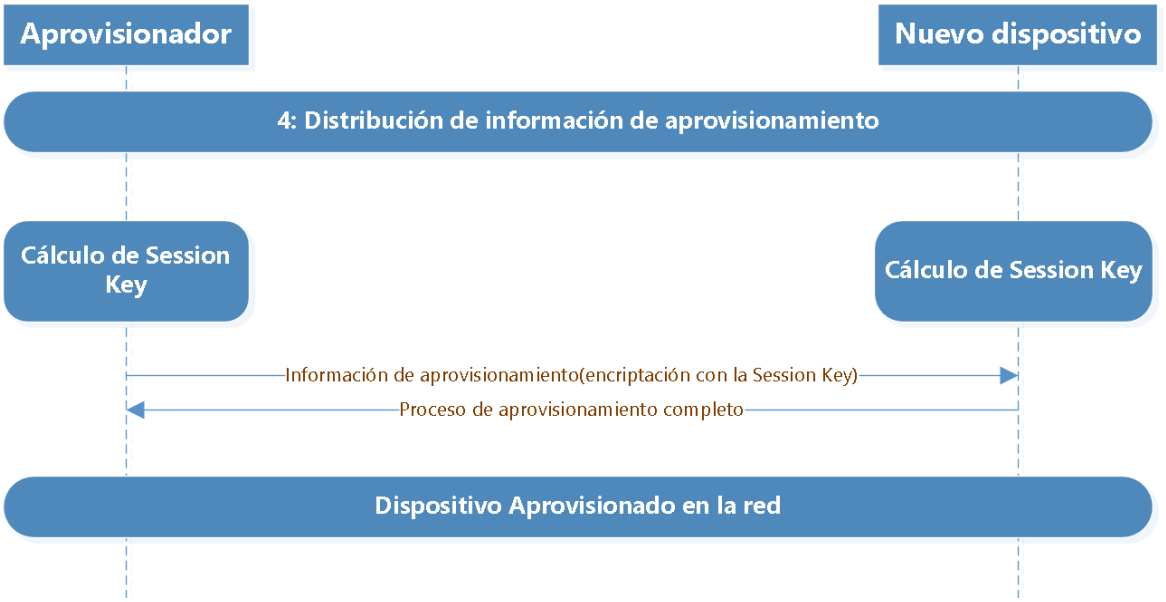
Figura 25. Proceso de invitación de autenticación de un nuevo nodo cuando se usa un valor OBB estático o no se usa un valor.



Fuente : (Bluetooth S.I.G., 2019a), Bluetooth Mesh Profile Specification, Revision v 1.0.1 Recuperado de https://www.bluetooth.com/specifications/specs/mesh-profile-1-0-1/

- Distribución of Provisioning Data: Con el proceso de autenticación terminado entonces se envían datos como: Dev Key (clave de dispositivo), NetKey (clave de red), y direcciones únicas al dispositivo recién aprovisionado. Esta información es enviada por medio de una encriptación AES-CCM. Cuando este proceso ha terminado, entonces el nodo puede considerarse como aprovisionado y ya forma parte de la red Mesh.

Figura 26. Proceso de invitación de distribución de datos de aprovisionamiento un nuevo nodo.

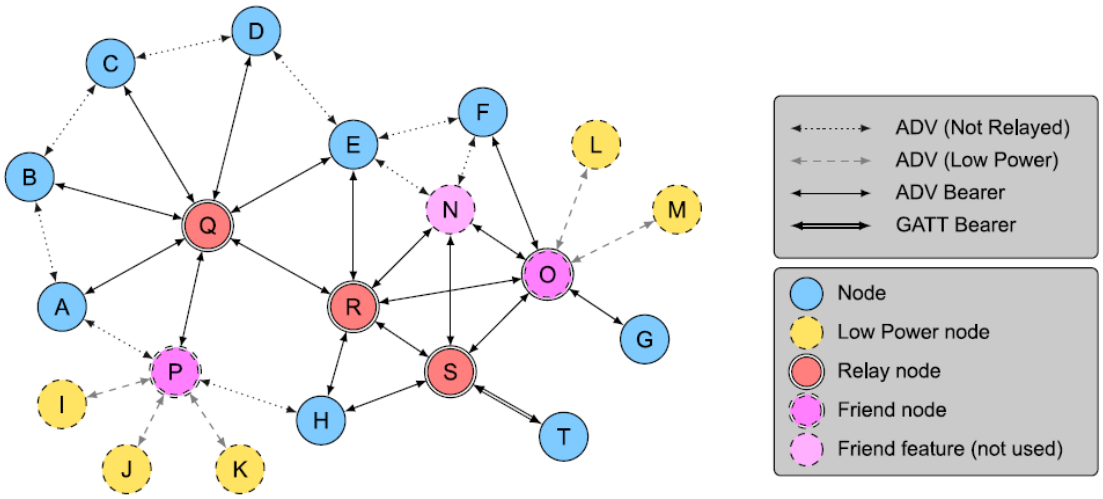


Fuente : (Bluetooth S.I.G., 2019a), Bluetooth Mesh Profile Specification, Revision v 1.0.1 Recuperado de https://www.bluetooth.com/specifications/specs/mesh-profile-1-0-1/

2.6.7. Tipos de nodos en una red Mesh de Bluetooth

Todos los nodos pueden recibir y enviar mensajes, pero cada uno de ellos tendrá un rol dentro de la red que le darán capacidades únicas, pero existen un determinado número de roles dados mediante el estándar. La Figura 27 indica una topología típica de una red bluetooth Mesh.

Figura 27. Ejemplo de una topología Mesh en bluetooth



Fuente : (Bluetooth S.I.G., 2019a), Mesh Profile 1.0.1 . Recuperado de https://www.bluetooth.com/specifications/specs/mesh-profile-1-0-1/

La red en bluetooth Mesh está compuesta de 6 tipos de nodo, a continuación, se listan:

2.6.7.1. Nodo Friend

nodo definido por el estándar específicamente, está diseñado para realizar la inclusión de dispositivos con bajo consumo de energía es decir casi siempre estará asociado a un nodo de bajo consumo por lo que deberá estar calificado para enviar y almacenar mensajes asociados al nodo de bajo consumo de energía.

2.6.7.2. Nodo Relay

Es uno nodo que se encarga de la interconexión de nodos, es decir recibe y transmite mensajes entre nodos de la red bluetooth Mesh. Utiliza el advertising bearer, en si se encarga de extender el rango de funcionamiento de la red haciendo posible que los nodos lejanos se comuniquen mediante la retransmisión de mensajes, además se encarga de realizar el multisalto de mensajes. Existe un número máximo de saltos que se puede realizar dentro de la red que están determinados por el TTL presente en cada uno de los paquetes que recorren la red.

2.6.7.3. Nodo Proxy

Este nodo puede realizar el intercambio de mensajes tanto entre nodos de la red y nodos no provisionados. Es decir, los nodos BLE solamente se podrán intercomunicar con la red Mesh.

2.6.7.4. Nodo Low Energy

Es el nodo que se encuentra en los extremos de la red y generalmente funciona con baterías, se caracteriza por funcionar en baja potencia. Siempre está asociado a un nodo Friend, los mensajes que se envían a un nodo Friend no se quedan guardados de forma local por lo para la comunicación con la red Mesh siempre será necesario la presencia del nodo Friend.

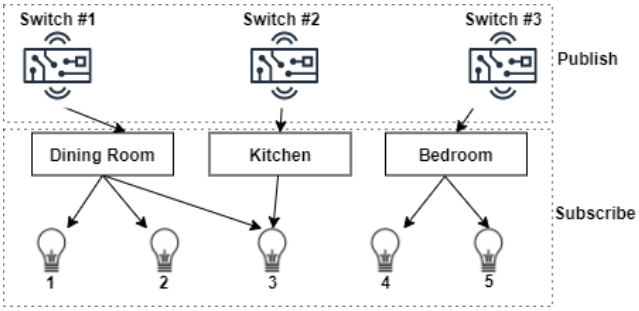
2.6.7.5. Nodo básico

Es uno nodo que no es ningún tipo de los nodos presentados y generalmente está en espera de realizar el proceso de aprovisionamiento a la red.

2.6.8. Proceso de publicación y suscripción

Los nodos de una red en bluetooth mesh se pueden configurar para que se comuniquen a través del proceso de publicación -suscripción. Dicho proceso asegura la coexistencia de dispositivos de diferentes marcas dentro de la red. El proceso consiste en la publicación de mensajes que pueden ser enviados a un dispositivo o a un grupo de dispositivos a los cuales dicho dispositivo esta suscrito Es decir un nodo envía mensajes a solo los nodos dentro de un mismo grupo ,los cuales son reconocidos como un grupo mediante una direccion física o virtual .Este proceso se realiza después del aprovisionamiento y es útil cuando se añade o remueve nodos de la red .En la se puede observar un ejemplo del proceso mediante el uso de switchs que publican mensajes a focos ,los focos están suscritos a los respectivos switchs por lo que reciben mensajes de dichos switchs(MathWorks, 2021).

Figura 28. Ejemplo del proceso de publicación - suscripción

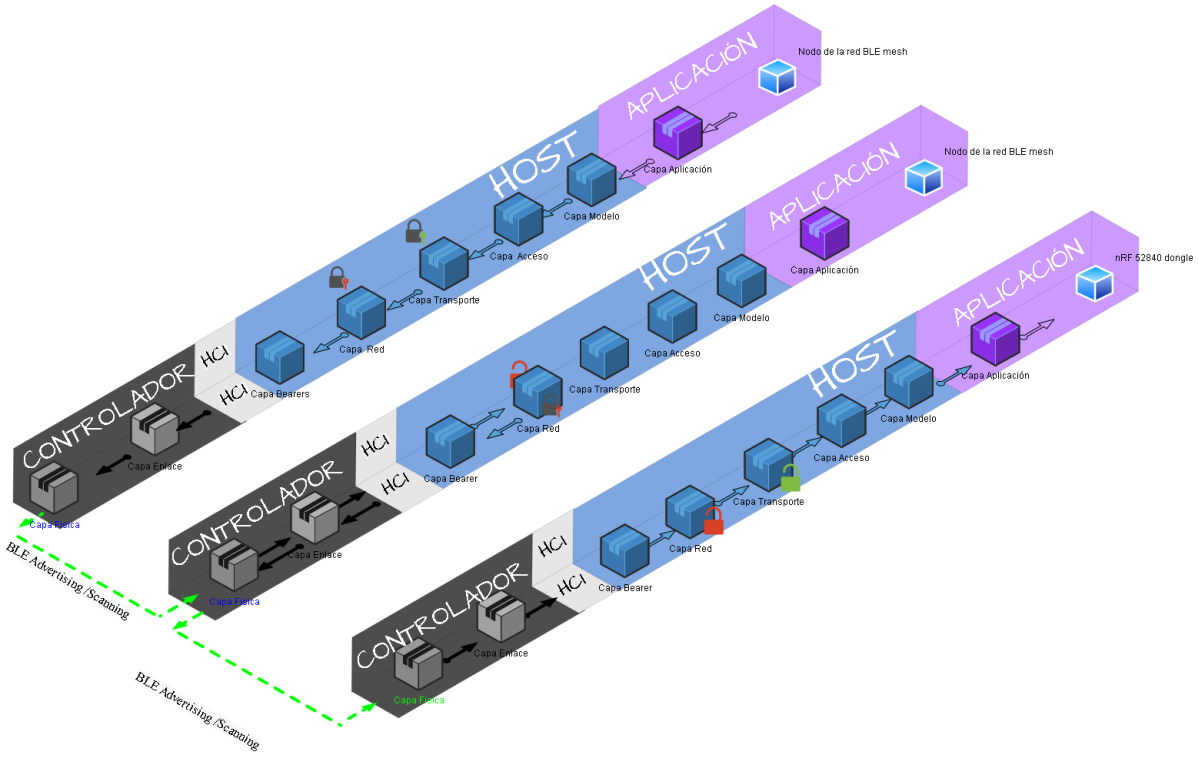


Fuente: (MathWorks, 2021). Bluetooth Mesh Networking - MATLAB & Simulink - MathWorks América Latina. Recuperado el 23 de noviembre de 2021, de https://la.mathworks.com/help/comm/ug/bluetooth-mesh-networking.html

2.6.9. Comunicación entre nodos dentro de una red Bluetooth Mesh.

El proceso de intercambio de información dentro de una red Mesh en Bluetooth con presencia de un nodo Relay se puede resumir en la Figura 29 .Los nodos de origen y destino presentan todas las capas del estándar, mientras que el nodo Relay solo procesa el paquete hasta la capa de red. En la capa de red se analiza la llave de red, dado que para el ejemplo la llave de red es similar entonces retransmite el mensaje hacia el siguiente nodo. En el nodo destino se analiza si la direccion de destino es similar a la direccion unicast o de grupo, si cualquiera de los dos valores coincide entonces se lleva el paquete a capas superiores para su tratamiento.

Figura 29. Envío de mensajes dentro de una red bluetooth Mesh.



Basado en (Seminconductor, 2021) Bluetooth mesh stack architecture — nRF Connect SDK 1.5.99 documentation. Recuperado el 20 de junio de 2021, de https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/ug_bt_mesh_architecture.html

2.6.10. Mensajes y direccionamiento del estándar Mesh

La red tipo Mesh realiza la intercomunicación, se tiene mensajes de control y mensajes de acceso. Los mensajes de control son utilizados para la gestión de los nodos de la red Mesh mientras los mensajes de acceso se utilizan para las aplicaciones y se tiene tres tipos:

- Set Message: Se encargan de modificar el estado de los nodos, dentro de este tipo se tiene dos subtipos; se tiene los mensajes de ACK que se generan ante un cambio de estado y los mensajes de NACK que no son respondidos.
- Get Message: Se encargan de recuperar y solicita el estado del nodo en el momento del Envío del mensaje, el nodo responde con el mensaje con el estado actual del nodo.
- Status Message: Estos mensajes son enviados por los nodos con el estado actual de los mismos.

Para el caso de los mensajes de direccionamiento, estos dependen de los tipos de direcciones existentes en la red Mesh. Es decir, se envían mensajes de direccionamiento a direcciones unicast, direcciones broadcast y direcciones virtuales. Según la topología típica los mensajes de origen siempre tendrán direcciones unicast y el destino de estos mensajes pueden ser de multicast ,unicast y demás ya mencionados con anterioridad (Darroudi et al., 2020).

2.6.11. Seguridad

La transferencia de datos en redes con demasiados dispositivos es siempre un reto, la SIG de bluetooth implementa la separación de conceptos y generalmente se aplica durante el proceso de aprovisionamiento, se utilizan claves para diferentes dispositivos de la red.

- Dev Key: Clave de dispositivo, se asigna a cada uno de los nodos de la red Mesh. Cada nodo es identificado por este valor y es dado por el dispositivo que realiza el proceso de aprovisionamiento por lo cual solo se usa solo durante ese periodo.

- NetKey: calve de red, se asigna a un nodo y puede tener más de una clave. Se encarga de la creación de subredes dentro de la red Mesh principal. Esta clave evita que se dé la retransmisión de mensajes a determinados niveles y se permite que solo se retransmitan mensajes a determinados niveles. Por lo que una determinada clave de red identifica a un nodo dentro de una subred de la red Mesh.
- AppKey: Clave de aplicación, esta clave es generada por el dispositivo aprovisionador. Se da a un conjunto de nodos que tengan una funcionalidad similar entre ellos.

Para la prevención de ataques de hombre en el medio se utiliza el intercambio de claves durante el proceso de aprovisionamiento .Para la prevención de los ataques de Relay se utiliza número de secuencia e índice de vector de inicialización .Estos elementos varían cada vez que se realiza la publicación de un mensaje ,por lo que se realiza la comprobación de este valor cada vez que se recibe un mensaje(Bluetooth S.I.G., 2019a).

2.6.12. Protocolo proxy

Solo los equipos que han implementado de forma nativa el estándar de comunicación de red pueden conectarse nativamente entre ellos, es decir utilizando la capa de Controlador de Bluetooth Low Energy y el Stack de Bluetooth mesh en la capa de Host. Pero para un equipo que no haya adaptado dicho estándar, esta función no estará disponible; como es el caso de smartphone que funciona como aprovisionador de red. Para lograr comunicarse con una red de este tipo se hace uso del protocolo Proxy.

El protocolo establece un cliente y un servidor, el cliente será un dispositivo externo a la red que no tenga la capacidad de funcionar bajo el estándar de comunicación de red. Mientras el cliente será un nodo dentro de la red, dentro del cual se ha habilitado la función proxy.

En el lado del cliente, se permite que el dispositivo use el perfil GATT para encapsular paquetes a través del stack de Bluetooth Low Energy, específicamente a través de valores de

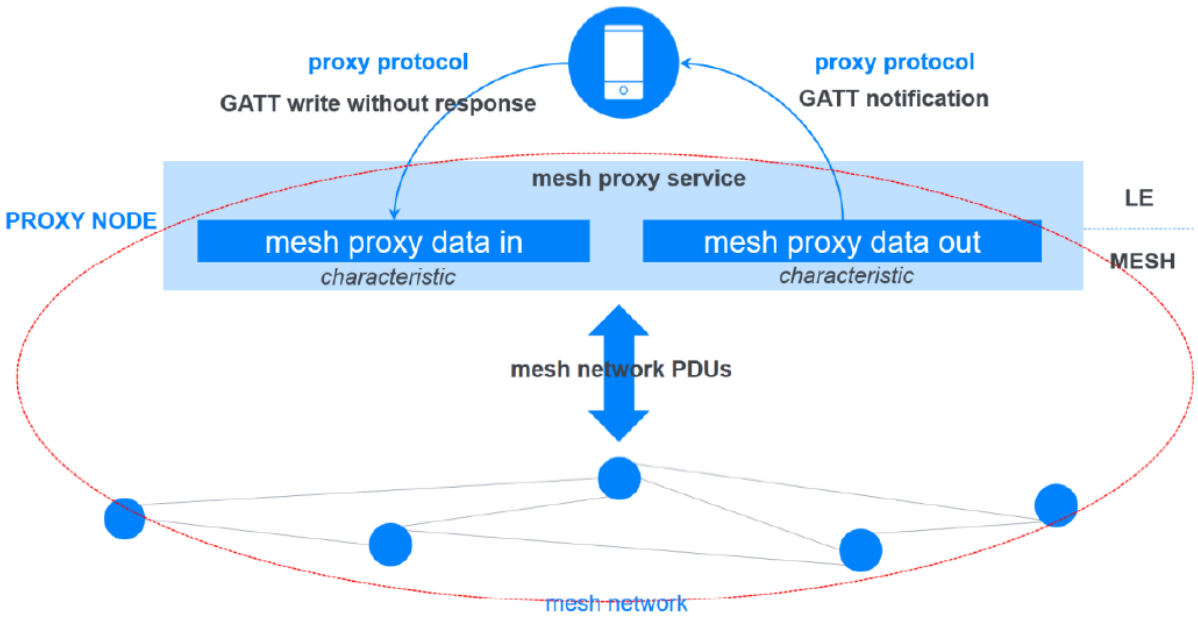
una característica GATT. Para enviar datos a la red, se utiliza una característica GATT de escritura sin respuesta, mientras que para recibir datos se utiliza una notificación GATT.

Por otro lado, en el servidor se implementan un servicio GATT llamado Mesh Proxy Service, el cual define dos características:

- Mesh Proxy Data In: Implementa un buffer para PDUs que inyecta datos a la red Bluetooth mesh desde el cliente.
- Mesh Proxy Data Out : Implementa un buffer para PDUs que obtiene los datos de la red Bluetooth mesh hacia el cliente externo (Bluetooth S.I.G., 2020). .

Este proceso se puede observar a detalle en la Figura 30,notar el límite de la red mesh y la ubicación del nodo proxy.

Figura 30.Funcionamiento del protocolo Proxy en una red Bluetooth mesh

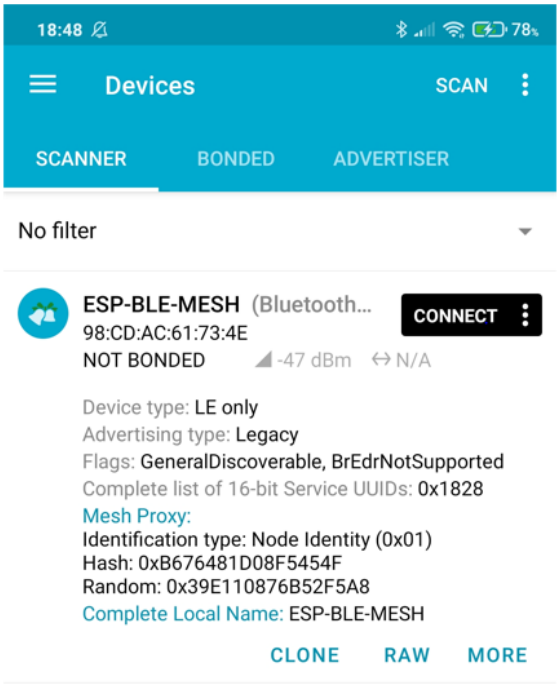


Fuente :(Bluetooth S.I.G., 2020)

Es posible revisar el estado de la función de proxy través de las aplicaciones de aprovisionamiento en un smartphone. Se puede comprobar esta función con la aplicación nRF Connect, esta aplicación permite la lectura de dispositivos bluetooth clásico y Bluetooth Low energy. Los dispositivos que funcionan bajo el estándar Bluetooth mesh no aparecerán en la

aplicación, pero después de activar la opción de proxy se podrá observar los parámetros de un nodo mesh, en la se muestra el nodo en cuestión en la aplicación nRF Connect.

Figura 31. Parámetros de un nodo de la red en la aplicación nRF Connect después de activar la función de proxy.



Fuente: Autoría.

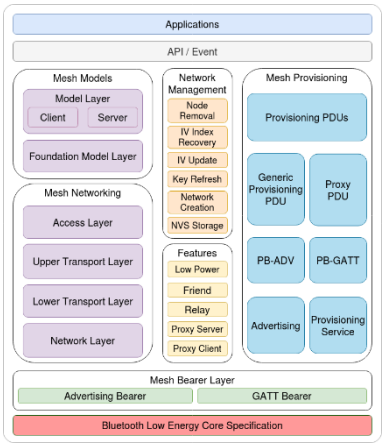
2.7. Compañías que brindan software y hardware que soportan el perfil Bluetooth mesh

En secciones anteriores se mostró las características de un RTOS y como estos han sido fundamentales para la creación de sistemas embebidos que puedan trabajar con IoT. Zephyr es uno de estos sistemas operativos en tiempo real que está enfocado en Bluetooth, especialmente en Bluetooth Low Energy con el apoyo de varios fabricantes, contribuidores y usuarios dado que se trata de un RTOS libre. En la respectiva página oficial se puede encontrar el listado de todas las características soportadas en Zephyr ,entre ellas el estándar mesh(Zephyr Project, 2021).Hay que recalcar que la mayoría de software que se van a listar a continuación implementan el estándar mediante API de las respectivas marcas y en su mayoría usa C para la programación de tareas.

2.7.1. Espressif

Espressif implementa el ESP-BLE-MESH ,un protocolo de código abierto que esta certificado por el Bluetooth SIG que soporta la mayoría de funcionalidades y características presentes en la especificación Bluetooth mesh v 1.0.1, ademas gracias a las características del protocolo es posible la interoperabilidad entre marcas .Basado en el stack de Bluetooth Mesh de Zephyr .La mayoría de los chips con soporte de Bluetooth v4.2 o superior pueden utilizar ESP-BLE-MESH para funcionar bajo el estándar Bluetooth Mesh, en la Figura 32. se puede observar la arquitectura que implementa Espressif para la utilización del estándar Bluetooth mesh en sus equipos(Espressiff Systems, 2021).La documentación oficial brinda ejemplos para comprender cono funciona el protocolo de comunicación de red ESP-BLE-MESH.

Figura 32Arquitectura implementada por Espressif en el protocolo ESP-BLE-MESH



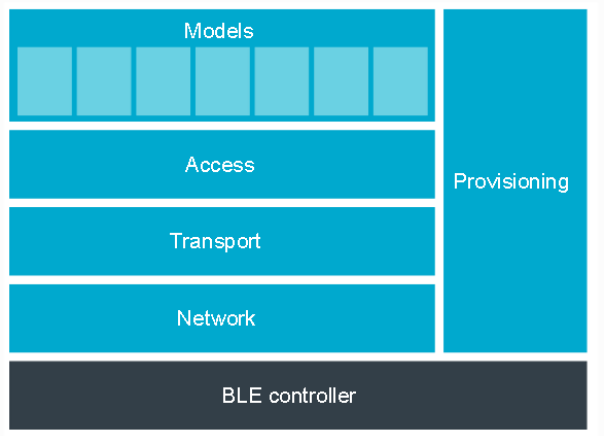
Fuente: (Espressif Systems, 2020b), ESP-BLE-MESH Architecture. Recuperado el 24 de noviembre de 2021, de https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/esp-ble-mesh/ble-mesh-architecture.html

2.7.2. Nordic Semiconductor

Nordic Semiconductor también se basa en el stack de Bluetooth Mesh de Zephyr para implementar Bluetooth mesh, a través de la herramienta nRF Connect SDK se puede utilizar todas las funciones y características del estándar, así como las funciones adicionales. La herramienta nRF Connect es compatible con los SoCs nRF52, nRF53 y nRF91, dichos SoCs cuentan con versiones de Bluetooth 5 o superior y se pueden encontrar en la mayoría de los kits de desarrollo que ofrece la empresa. Además, se utilizan los llamados softdevices para

equipos que usan la herramienta nRF5 SDK, esta herramienta está contenida en nRF Connect SDK. En sí ,un softdevices permite utilizar archivos preconfigurados con las características de Bluetooth Low Energy es decir es utilizado cuando no se utilizara un RTOS(Nordic Semiconductor, 2021b).En si los equipos de Nordic Semiconductor permiten desplegar todas las características de Bluetooth Mesh en sus equipos de acuerdo a la arquitectura oficial del Bluetooth SIG como se puede observar en la Figura 33 donde los modelos se implementan mediante la aplicación nRF Connect SDK mientras lo demás es a través de stack de Bluetooth Mesh de Zephyr , existe tutoriales y ejemplos en la página oficial que explican detalladamente el funcionamiento de Bluetooth mesh en los equipos Nordic Semiconductor.

Figura 33. Arquitectura implementada por Nordic Semiconductor para el funcionamiento de Bluetooth mesh.



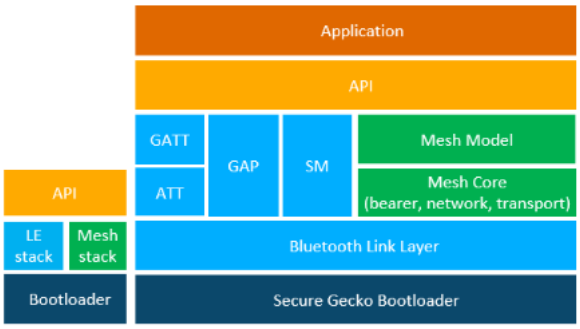
Fuente :(Nordic Semiconductor, 2021a), Bluetooth mesh stack architecture — nRF Connect SDK 1.7.1 documentation. Recuperado el 25 de noviembre de 2021, de https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/ug_bt_mesh_architecture.html

2.7.3. Silicon Labs

Silicons Labs implementa el stack llamado Silicon Lab Bluetooth stack, el cual se basa en APIs que permiten funcionar el estándar Bluetooth en los dispositivos de la marca, en si para usar el estándar se utiliza el Bluetooth mesh SDK que utiliza el Silicon Labs Bluetooth Mesh stack, el cual permite usar todas las características del estándar Bluetooth Mesh. Por lo que se utiliza la misma arquitectura implementada en el estándar original. También se utiliza el llamado Secure Gecko en las placas de la marca que permite obtener actualizaciones de

seguridad de firmware .Debido a esta última característica de la marca ,solo es posible acceder a códigos de ejemplo y herramientas adicionales si se compra el equipo(SiliconLabs, 2021) .En la Figura 34 se puede observar la arquitectura implementada por Silicon Labs para el funcionamiento del estándar Bluetooth mesh ,como se mencionó anteriormente es similar a la arquitectura presentada en el estándar.

Figura 34. Arquitectura implementada por Silicon Labs para el funcionamiento de Bluetooth mesh.

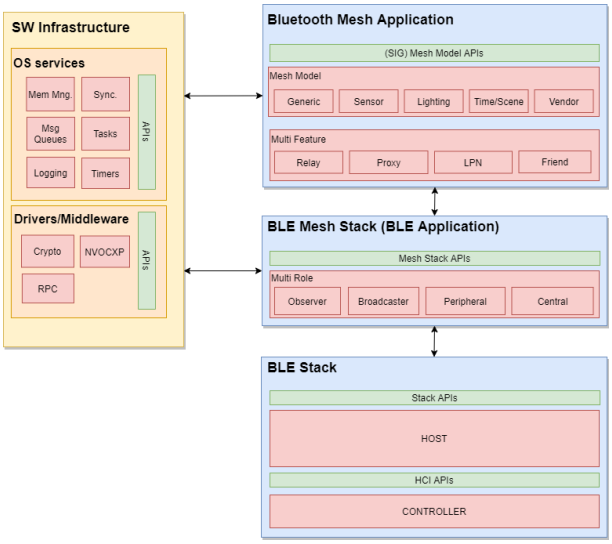


Fuente :(SiliconLabs, 2019), AN1098: Understanding the Silicon Labs Bluetooth® Mesh Lighting Demonstration in SDK v1.x. Recuperado el 25 de noviembre de 2021, de https://www.silabs.com/documents/public/application-notes/an1098-understanding-bluetooth-mesh-lighting-demo.pdf

2.7.4. Texas Instruments

Texas Instruments también se basa en el stack de Bluetooth Mesh de Zephyr así como utiliza APIs para la implementación de las funcionalidades de Bluetooth mesh en sus equipos .Actualmente solo solo brinda soporte para solo ciertos modelos genéricos y algunos modelos de fabricante .La implementación se realiza a través de tres elementos : el stack de protocolos de Bluetooth Low Energy ,el stack de protocolos de Bluetooth Mesh y las aplicaciones de Bluetooth mesh .El stack de Bluetooth mesh se basa en Zephyr mientras el stack de Bluetooth Low Energy es propio de la marca basado en las normas de Bluetooth SIG(Texas Instruments Incorporated, 2020).La version de Bluetooth de los equipos es 5 y superior .En la Figura 35 se observa la arquitectura implementada por Texas Instruments.

Figura 35. Arquitectura implementada por Texas Instruments para Bluetooth Mesh.



Fuente : (Texas Instruments Incorporated, 2020), Overview — SimpleLinkTM CC13x2 / CC26x2 SDK BLE5-Stack User's Guide 2.02.00.00 documentation. Recuperado el 25 de noviembre de 2021, de https://software-dl.ti.com/simplelink/esd/simplelink_cc13x2_26x2_sdk/4.40.00.44/exports/docs/ble5stack/ble_user_guide/html/ble-mesh/overview.html#ti-bluetooth-mesh-software-architecture

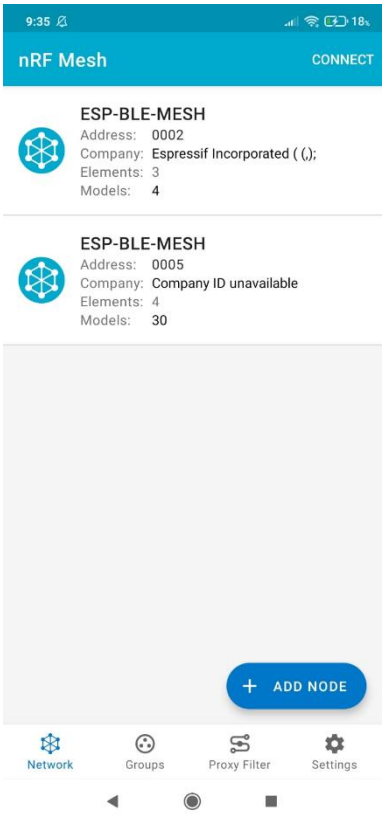
2.8. Proceso de aprovisionamiento utilizando diferentes equipos

Este procedimiento será siempre realizado por el administrador del sistema. La etapa de aprovisionamiento de la red puede realizarse directamente a través de aplicaciones móviles propias de los fabricantes como:

- Nordic Semiconductor con nRF Mesh
- Espressif con ESP-Mesh
- Silicons Labs con Bluetooth Mesh
- Etc.

Estas aplicaciones cuentan con una interfaz especializada para: aprovisionamiento, creación de grupos, detección de dispositivos y otras más de acuerdo con el estándar. O también pueden realizarse a través de equipos propios de la red, es decir cualquiera de los nodos puede tomar el papel de aprovisionador de red si se le programa para este rol. En la se puede observar la interfaz de la aplicación nRF Mesh de Nordic Semiconductor.

Figura 36. Interfaz gráfica de la aplicación nRF Mesh de Nordic Semiconductor.

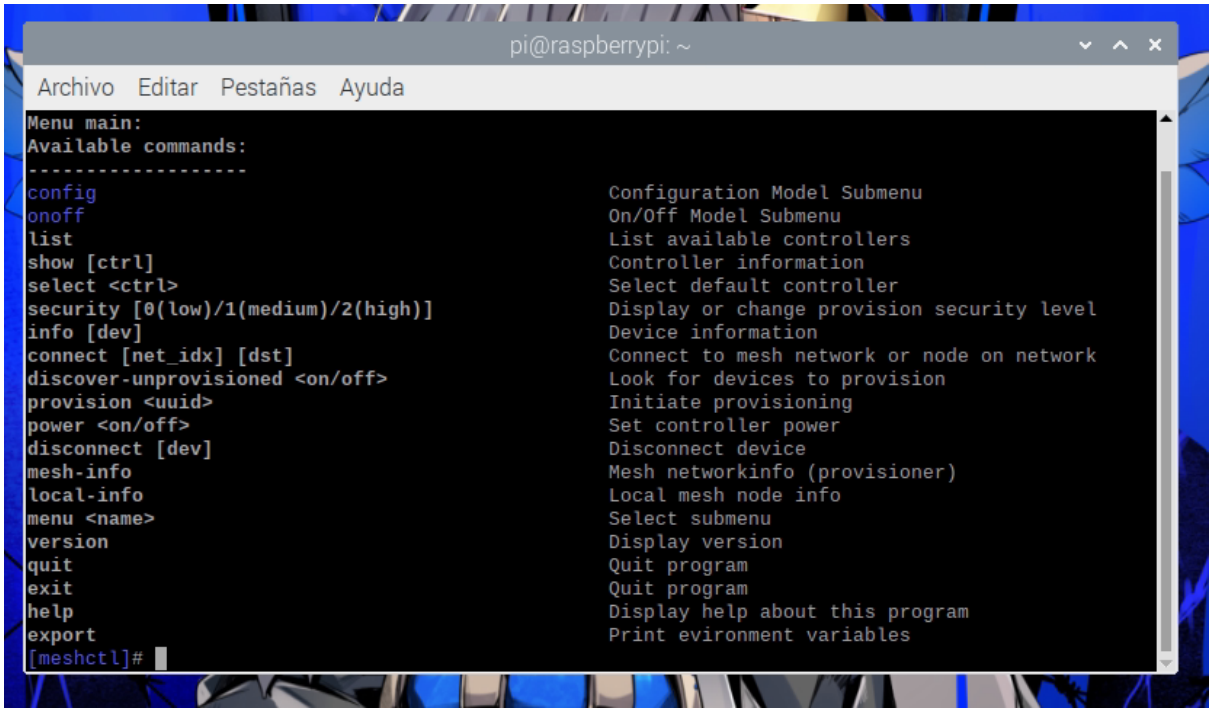


Fuente: Autoría.

También el Raspberry Pi puede usarse para este rol. La Bluetooth SIG brinda soporte para este proceso a través de la herramienta BlueZ a partir de la version 5.50. Las configuraciones previas en el Raspberry Pi se pueden encontrar en la página web oficial ,al terminar dicho tutorial es posible utilizar al Raspberry Pi en las versiones más recientes como un aprovisionador de red Bluetooth Mesh ,las versiones más antiguas soportan la funcionalidad luego de realizar un proceso de actualización de kernel especificado en el tutorial mencionado(Bluetooth S.I.G., 2019c).

Se utiliza el comando meshctl en la carpeta donde se realizó las configuraciones, el menú que aparecerá será el que permite realizar la mayoría de las funciones de un dispositivo aprovisionador. Las opciones que ofrece la herramienta BlueZ se pueden observar en la Figura *37*

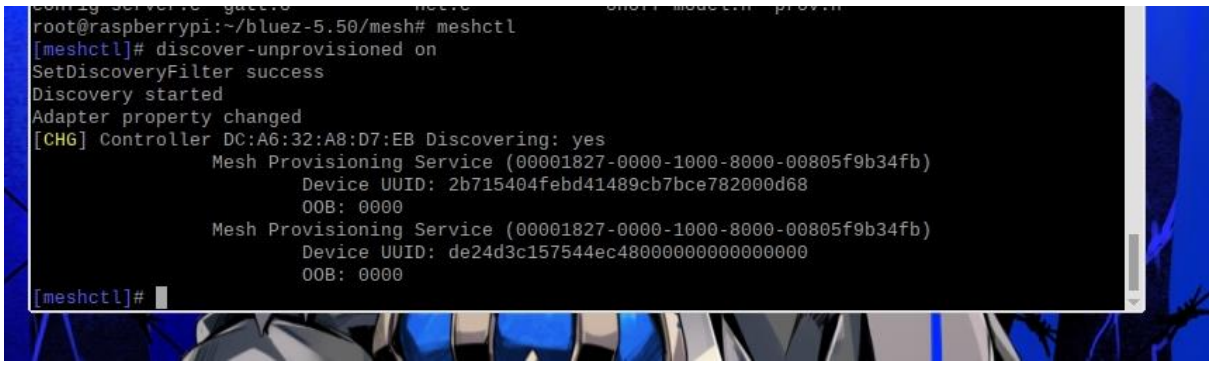
Figura 37. Opciones que ofrece la herramienta BlueZ en Raspberry.



Fuente: Autoría

Para realizar el proceso de aprovisionamiento se utiliza la opción *discover-unprovisioned on*, esta opción permitirá descubrir los dispositivos que funcionen con el estándar Bluetooth Mesh y que no estén aprovisionados en el rango de la antena del Raspberry Pi. Si este proceso se realiza cuando los dispositivos ya están encendidos, los dispositivos aparecerán de acuerdo a la Figura *38*

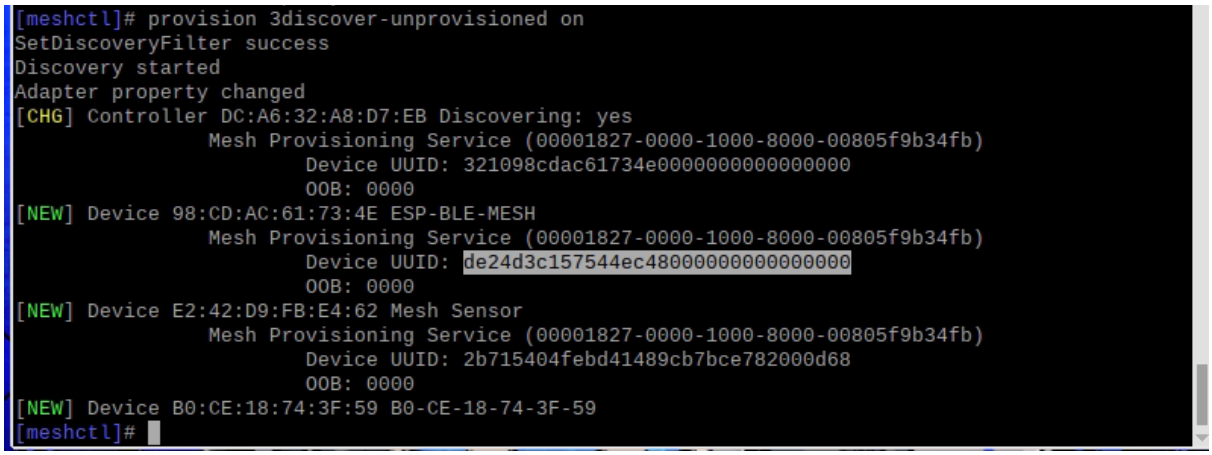
Figura 38. Dispositivos que se muestran luego de usar el comando para descubrir dispositivos no aprovisionados.



Fuente: Autoría

Si se realiza este proceso y un dispositivo exterior es encendido mientras se realiza el descubrimiento de nuevos dispositivos se muestra la etiqueta “NEW” junto a la respectiva información del equipo, como se observa en la Figura *39*

Figura 39. Dispositivos nuevos que se muestran luego de usar el comando para descubrir dispositivos no aprovisionados



Fuente: Autoría

El formato de la información mostrada de los nuevos dispositivos no aprovisionados consta de : la direccion física del dispositivo ,identificador del servicio GATT del dispositivo ,OBB que utiliza el dispositivo y un identificador UUID .Para realizar el aprovisionamiento del dispositivo será utiliza el valor de UUID ,según BlueZ se utiliza el comando *provision* seguido del número de UUID ,si el número es correcto se solicitara algunos datos como valor de NetKey y otros .Al finalizar el proceso se le asigna un ID de red al nuevo dispositivo ,este identificador será muy útil para controlar al nuevo dispositivo desde el aprovisionador (la Raspberry Pi en este caso específico).Una de las mayores desventajas de usar al Raspberry pi como aprovisionador es su pobre alcance con la antena que posee .La antena esta sobre la placa por lo que no es posible aumentar su potencia y no se ofrece documentación oficial para añadir una antena mucho más potente

En comparativa, es más factible utilizar un smartphone como aprovisionador de red debido a una antena más poderosa de Bluetooth y un mayor soporte de librerías para implementar una red utilizando este estándar de comunicación de red.

2.9. Librerías y software que permiten utilizar el perfil Bluetooth mesh

Los datos de los sensores de una red Bluetooth mesh se deben extraer de alguna forma hacia un dispositivo externo de la red, ya sea para un futuro análisis o la presentación de alertas

al usuario en el uso de sistemas .Dentro del mercado existen algunos librerías y equipos que permiten desarrollar una aplicación externa usando hardware o software especifico ,a continuación se analizara algunos y las versiones del estándar que soportan.

2.9.1. BlueZ

Se trata del controlador de protocolo Bluetooth que utiliza Linux por defecto, soporta las funcionalidades de Bluetooth mesh desde la version 5.47. Existen guias oficiales donde se instala los paquetes necesarios en un Raspberry Pi .Soporta el aprovisionamiento por medio de PB GATT usando el comando meshctl y PB ADV usando el comando mesh-cfgclient.Gracias a que se trata de una herramienta de código abierto existe documentación pero para el caso específico de Bluetooth mesh no existe guias acerca del control de sensores ,las guias existentes solo se basan en control de luces(Bluetooth S.I.G., 2020) .Aun así son muy útiles para entender cómo funciona el estándar de comunicación de red cuando se monta en un Raspberry Pi. Ademas se exige un alto conocimiento en el sistema operativo Linux para poder manejarlo ,también existen limitaciones de los controladores Bluetooth físicos de los equipos donde se piensa montar un futuro sistema ,y no ha recibido muchas actualizaciones en su core ni documentación .En conclusión es un software que debería utilizarse para fines didácticos únicamente .

2.9.2. STBLEMesh

Se trata de una librería para desarrollo de aplicación de la marca STMicroelectronics ,la cual está hecha para funcionar sobre los productos de dicha marca. Es posible realizar funciones de aprovisionamiento ,controlar luces y sensores en una red bajo el estándar de comunicación Bluetooth mesh. Presenta un stack de protocolos adaptado para el funcionamiento en dispositivos Android e iOS.(STMicroelectronics, 2021)Es posible encontrar el código base y librería de la aplicación pero la última actualización de la misma ha sido en

2019 por lo que solo soporta la version del perfil mesh y los modelos mesh en la version 1.0. En conclusión se trata de un software deprecado que se espera se actualice en un futuro.

2.9.3. Python bluetooth mesh

Se trata de un software para desarrollo de aplicaciones basado en Python que utiliza el estándar de comunicación de red Bluetooth mesh .Este software es propiedad de Silvair, soporta la version del perfil siendo la version 1.0.1 pero en el caso de modelos solo se aceptan los modelos genéricos para luces . Existen ejemplos muy explicativos con bombillas de la marca escritos 100% en Python ,está realizado para su uso en Linux(SilvairGit, 2022) .No existe documentación aparte de explicaciones superficiales de los modelos mencionados ,por lo que se espera que en el futuro se añade soporte para más modelos y documentación más extensa.

2.9.4. Android nRF Mesh Library

La librería Android nRF Mesh Library, es desarrollada por la compañía Nordic Semiconductor soporta la mayoría de las funciones de:

- Perfil Mesh en la version 1.0.1
- Modelos Mesh en la version 1.0.1
- Propiedades de Dispositivos mesh en la version 2

La librería funciona en dispositivos Android con una version 4.3 y superior ,puede ser usada bajo la licencia BSD 3- Clause. De acuerdo a las funciones mencionadas ,es posible aplicar el estándar Bluetooth mesh completamente (Mobile Applications Team Nordic Semiconductor ASA., 2022).Se trata de la librería más robusta dentro del mercado ,soporta las funciones de aprovisionamiento, control de sensores ,creación de llaves y otras más que se especifica en la norma .Esta escrito en Java y la documentación en foros y en la página de GitHub oficial es muy variada ,esta será la librería utilizada para el desarrollo del proyecto específicamente en la version 3.19.

2.10. Android

Es un sistema operativo basado en el kernel de Linux, funciona en dispositivos móviles como smartphones ,tabletas y otros más .Inicialmente se basaba en Java pero actualmente se está utilizando Kotlin como lenguaje de programación. Presenta una robusta arquitectura que permite la administración de los dispositivos en los que funciona .El código fuente original Android Open Source Project es de uso libre baja bajo la licencia Apache 2.0, algunas herramientas adicionales son de propietarios y en algunos casos se han construido sistemas operativos basados en el código fuente como MIUI(Meike & Schiefer, 2022).

2.10.1. Breve historia

Fue desarrollado Google en el año 2003 para el funcionamiento de cámaras digitales ,en el año 2005 Google adquirió la compañía Android Incorporation. En aquel tiempo el sistema operativo estaba en desarrollo pero con el lanzamiento del IPhone en 2007 ,se aceleró el desarrollo del sistema operativo .Se realizo una alianza llamada Open Handset Alliance .La alianza cuenta con 84 miembros ,se encarga de crear estándares de código abierto para todo tipo de dispositivos móviles .Fue creada para competir con los desarrolladores de sistemas operativos móviles como :iOS, Windows Phone y otros que con el paso de los años han desaparecido

La version beta de Android fue lanzada en noviembre de ese mismo año. Meses más tarde se lanzó el T-Mobile G1 con la version 1.0 de Android por la (Haase, 2021).Actualmente la última version de Android es la 12 llamada Snow Cone.

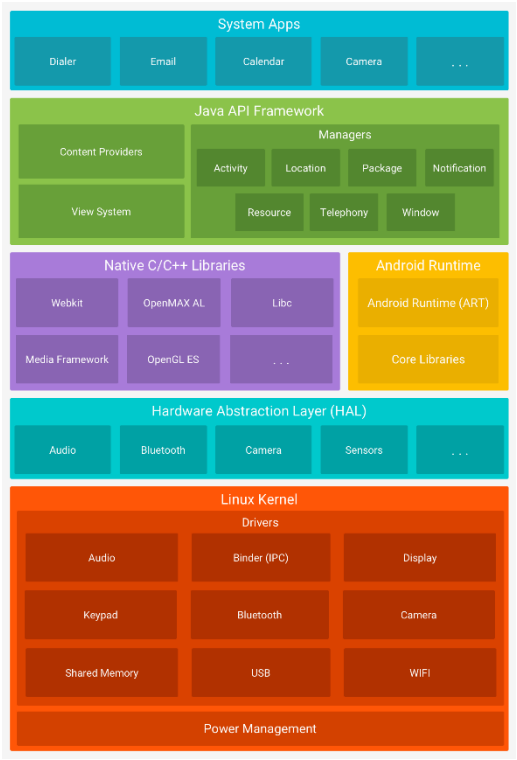
2.10.2. Arquitectura

Al basarse en el kernel de Linux ,Android utiliza una arquitectura de software basado en capas .Dentro del cual se tiene 6 componentes claramente diferenciados .

- Kernel de Linux :Se trata de la base del sistema operativo ,se encarga del control de drivers del dispositivo, así como el consumo de energía pero principalmente maneja la administración de proceso y subprocesos internos de memoria en el nivel más bajo.
- (HAL)Capa de Abstracción de Hardware: Se encarga del control lógico de interfaces para la interacción de las interfaces físicas del dispositivo con el usuario .Se realiza la intercomunicación el API de Java/Kotlin usando librerías que controlan cada acción .
- Android Runtime : Android maneja un tiempo de ejecución y tiempo de vida de actividades y acciones ,para ello se utiliza instancias del tiempo de ejecución de Android .Las tareas se realizan utilizando la interacción con la capa HAL ,tratando de ocupar la menor cantidad de memoria de bajo nivel posible .También se controla el uso de librerías para el lenguaje de programación de alto nivel utilizado.
- Librerías C/C++ nativas :Se trata de librerías nativas que permiten el funcionamiento de Android escritas en C/C++,en resumen permiten controlar las funcionalidades ,componentes y servicios nativos mediante una API en el lenguaje de programación de alto nivel.
- Java/Kotlin API Framework :Se trata de la API del lenguaje de programación del alto nivel que permite que los desarrolladores puedan crear y administrar aplicaciones móviles
- Apps de sistema : Son aplicaciones mandatorias en dispositivos móviles (Google Developers, 2019).

En la *Figura 40* se puede observar cómo se ubican los componentes así como las actividades que controlan en la arquitectura de Android .

Figura 40.Arquitectura de Android.

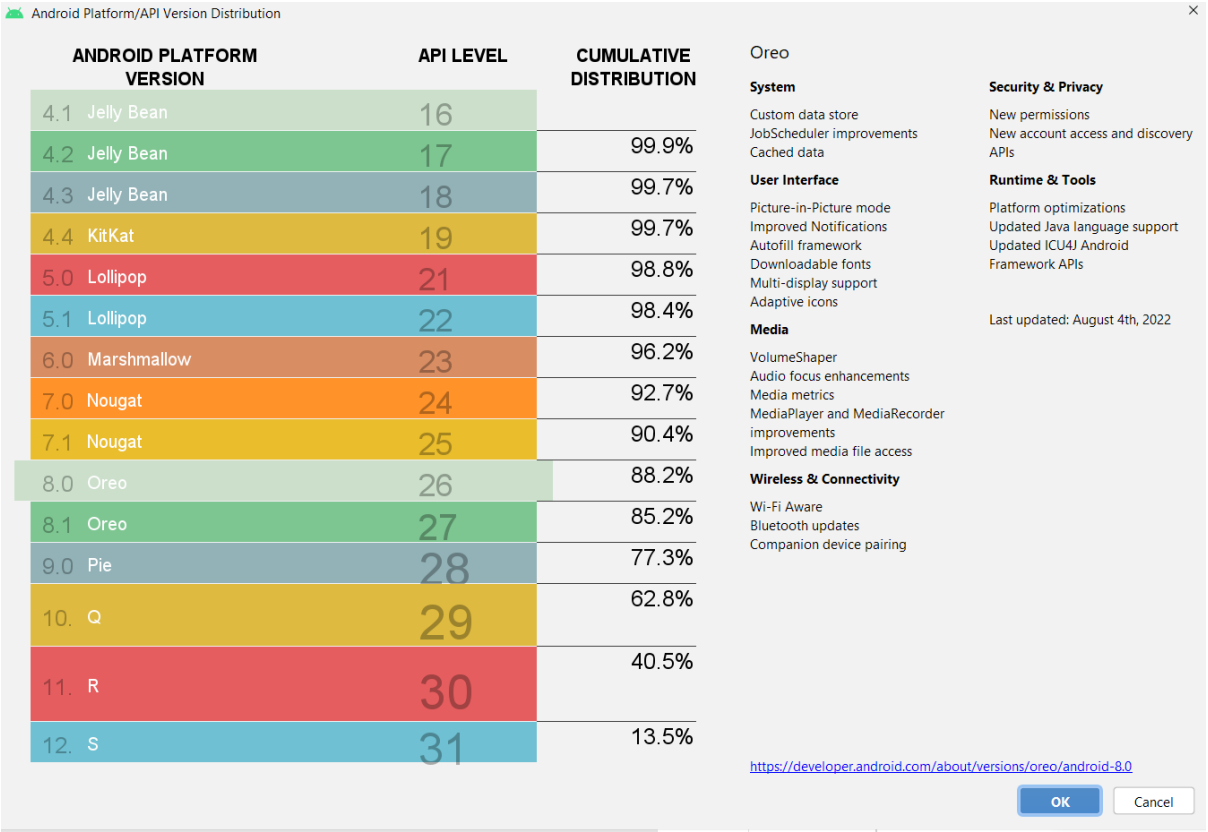


Fuente: (Google Developers, 2019)

2.10.3. Versiones de Android en el mercado

Como se mencionó anteriormente la última version de Android es la version Snow Cone ,siendo esta la que funciona en la mayoría de los teléfonos nuevos adquiridos de fábrica. Android Studio brinda información acerca de distribución acumulativa ,esta información permite conocer que porcentaje de mercado se podría utilizar si se usa una determinada version de Android .En la *Figura 41* se puede observar los valores ,por ejemplo para el caso de la versión 8.0 al menos el 88.2 de los dispositivos podrán utilizar una determinada aplicación a desarrollar.

Figura 41.Distribución Acumulativa de las versiones de Android

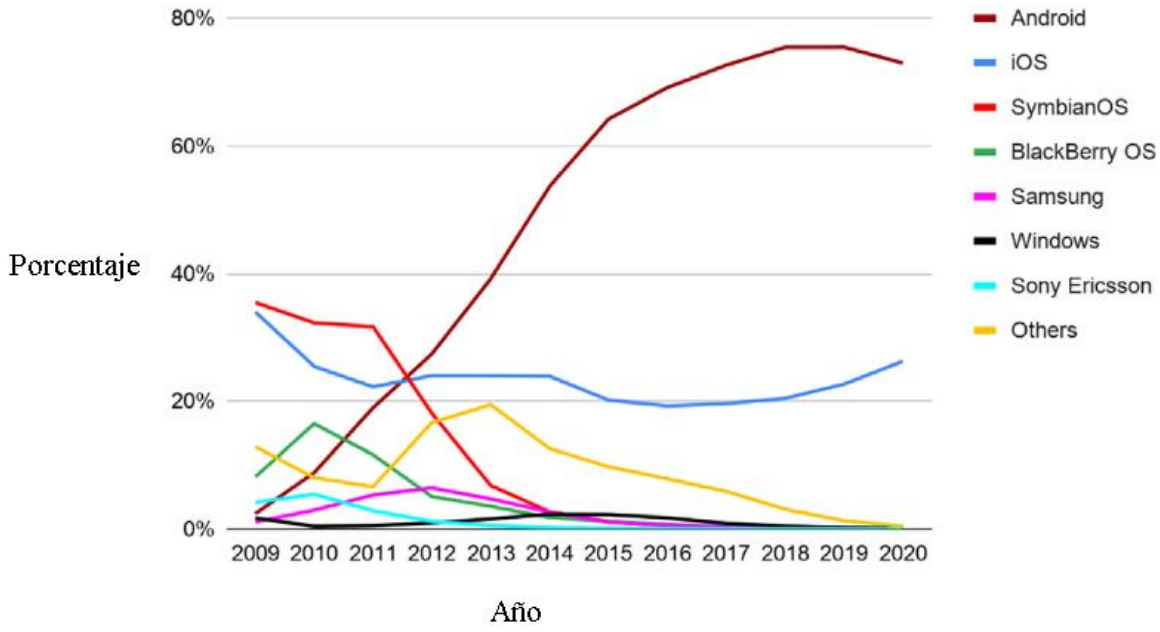


Fuente: (Google Developers, 2021)

2.10.4. Distribución de Android en el mercado global

Los smartphones se han vuelto una necesidad en la vida cotidian ,al año 2021 se registra al menos 3.8 billones de usuarios de los mismo .Dentro del mercado en la actualidad existen dos sistemas operativos que lideran el mercado : Android e iOS,los dos presentan sus respectivas ventajas y desventajas.Pero Android lidera el mercado global y se proyecta al crecer en los años venideros (Garg & Baliyan, 2021). En la Figura 42 se puede observar la couta del mercado de algunos sistemas operativos de dispositvos moviles en los años 2009-2020.

Figura 42.Cuota del mercado de sistema operativos móviles.



Fuente: (Garg & Baliyan, 2021).

2.11. Metodologías para el desarrollo de proyectos :Modelo en V

Es la metodología escogida para el desarrollo del presente proyecto ,se trata de una variación del modelo en cascada. Básicamente se divide el proceso de desarrollo en tres secciones: fase de requerimientos, fase de modelado y análisis y fase de desarrollo. Las tareas de testeo se realizan al mismo tiempo que las etapas de desarrollo permitiendo que el resultado final sea altamente confiable ,las variaciones del modelo implementan fases adicionales entre las tres secciones mencionadas .La version utilizada para el desarrollo es la version : modelo en v version extendida para modelado de sistemas informáticos(Graessler et al., 2018).

Según la conferencia “System of Systems Engineeringand Family of Systems Engineering from a Standards, V-Model, Dual V-Model(Clark, 2008), and DoD Perspective”,las etapas del modelo en v son las siguientes ;considerando que en cada uno de los niveles se aplica una etapa de retroalimentación para pruebas que interconecta el extremo derecho e izquierdo del modelo.

- Proceso 1: Identificación de usuarios y stakeholders

Se establece un listado de los actores que van a participar de forma directa o indirecta dentro del proceso de construcción del prototipo. Se diferencia los roles de usuarios y stakeholders de acuerdo a su nivel de participación.

- Proceso 2: Requerimientos de hardware y software del sistema y subsistemas.

Dentro de este proceso se establece los requerimientos a los cuales estará sujeto el sistema y deben cumplir de forma obligatoria, se realiza esta fase mediante comparativas de los objetos que se planea utilizar ya sea que se trate de hardware o software.

- Proceso 3: Diseño de la arquitectura del sistema y subsistemas.

En base a la información obtenida en los procesos anteriores, se diseña el hardware para cumplir con los requerimientos de los usuarios, así como las normativas actuales. Además, se realiza planos detallados del sistema.

- Proceso 4: Construcción y codificación del código del sistema y subsistemas

Se establece la estructura del código de programación para cada uno de los módulos y submódulos del sistema con las respectivas pruebas de validación tomando en cuenta la información de procesos anteriores.

- Proceso 5: Construcción del sistema y subsistemas

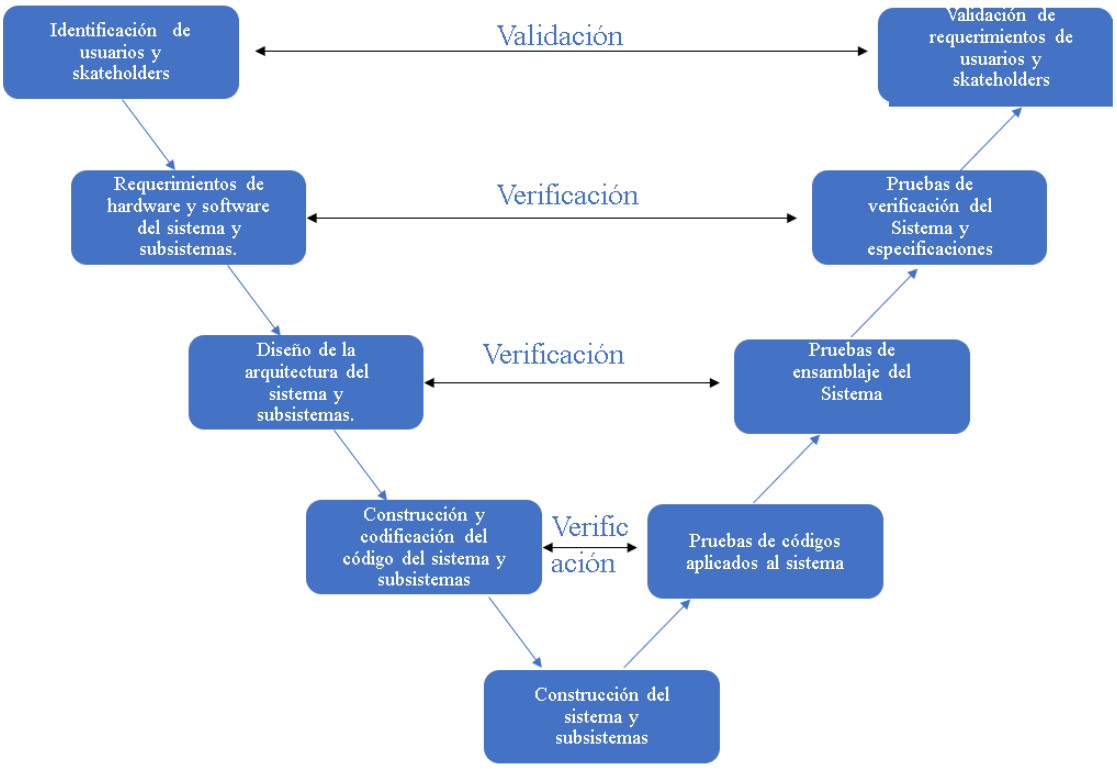
En esta etapa se realiza la construcción del prototipo o sistema con base al hardware y software escogido. Se aplica los códigos de programación a los dispositivos correspondientes.

- Proceso 6: Análisis y pruebas de los procesos.

Esta etapa corresponde a la zona derecha del modelo en v, se realiza pruebas de funcionamiento en base a cada uno de los procesos ya mencionados para finalmente implementar en prototipo tomando en cuenta los resultados de las pruebas de funcionamiento.

En la siguiente figura se puede observar cada uno de los procesos del modelo en v usado como referencia.

Figura 43. Procesos del modelo en V



Nota: basado en “System of Systems Engineering and Family of Systems Engineering from a Standards, V-Model, Dual V-Model, and DoD Perspective” (Clark, 2008),

Esta es la metodología que más se adapta al desarrollo del proyecto ,por lo cual esta será usada en el mismo.

3. Capítulo III Desarrollo

En esta sección se realizará el diseño del sistema en base a la metodología escogida ,se tomará en cuenta los requerimientos necesarios para construir el sistema, considerando los módulos y submódulos del sistema .También se analizará los elementos a utilizar, así como la interacción del futuro sistema domótico con los usuarios ,ya sea directos e indirectos.

3.1. Análisis de la situación actual

Se abordará la situación actual desde tres enfoques en común :los usuarios ,la tecnología de comunicación inalámbrica o estándar de comunicación de red y el sistema operativo en donde se mostrará la información al usuario .Para lo cual se utilizará métodos como entrevistas ,observación directa y experiencia en proyectos previos durante el ciclo de estudio para el desarrollo de este proyecto .

3.1.1. Situación actual de los usuarios

Para establecer la situación actual de los usuarios se ha realizado una entrevista a un cliente potencial del sistema a diseñar ,las respuestas se encuentran en el Anexo 2. El entrevistado es el Sr. Gonzalo Farinango Jijón de 68 años de edad ,el cual presenta una discapacidad visual del en el carné del CONADIS del cien por ciento. La entrevista se divide en dos secciones :la primera busca obtener las dificultades de las personas no videntes en el hogar y como buscan posibles soluciones a las mismas .La segunda sección trata acerca de la interacción previa con un sistema IoT .A continuación se analizará las respuestas de la primera sección .

La respuesta a la primera pregunta indica que el entrevistado encuentra difícil el manejo de herramientas tecnológicas dado que algunos equipos no tienen características de accesibilidad para personas no videntes .Así mismo existe problemas para manejar los aparatos eléctricos dentro del hogar ,en la cocina también existe problemas con el manejo de la misma así como sus utensilios .Otro problema que el entrevistado manifiesta es saber la hora actual ,pero menciona que gracias a relojes parlantes ha solucionado en parte este problema .

Con la respuesta de la segunda pregunta ,se sabe que una persona no vidente realiza una inspección manual de primera mano al existir un desperfecto eléctrico en un aparato, pero por precaución se llama a un profesional en el ámbito o en su defecto a un familiar-vecinos que conozca del tema.

La respuesta del entrevistado en la tercera pregunta permite saber que él vive con familiares **,**específicamente con su esposa y un familiar más siendo tres personas dentro de su domicilio. La respuesta a la cuarta pregunta por parte del entrevistado indica que las personas no videntes tienen un código para realizar ciertas actividades que las personas con las que comparte vivienda no conocen ,debido a ello existen situaciones como :las puertas y cajones se quedan abiertos en ciertas habitaciones ,las puertas de hornos así como llaves de quemadores también se quedan abiertas y funcionando sin ningún aviso previo .Todo ello desencadena en golpes y situaciones de peligro que están fuera de control según el entrevistado.

. En la quinta pregunta ,el entrevistado indica que :no ha sufrido lesiones graves pero si accidentes caseros como golpes ,resbalones debido a que los familiares con los que convive colocan obstáculos en su ruta de movilidad dentro del domicilio o están realizando alguna actividad .

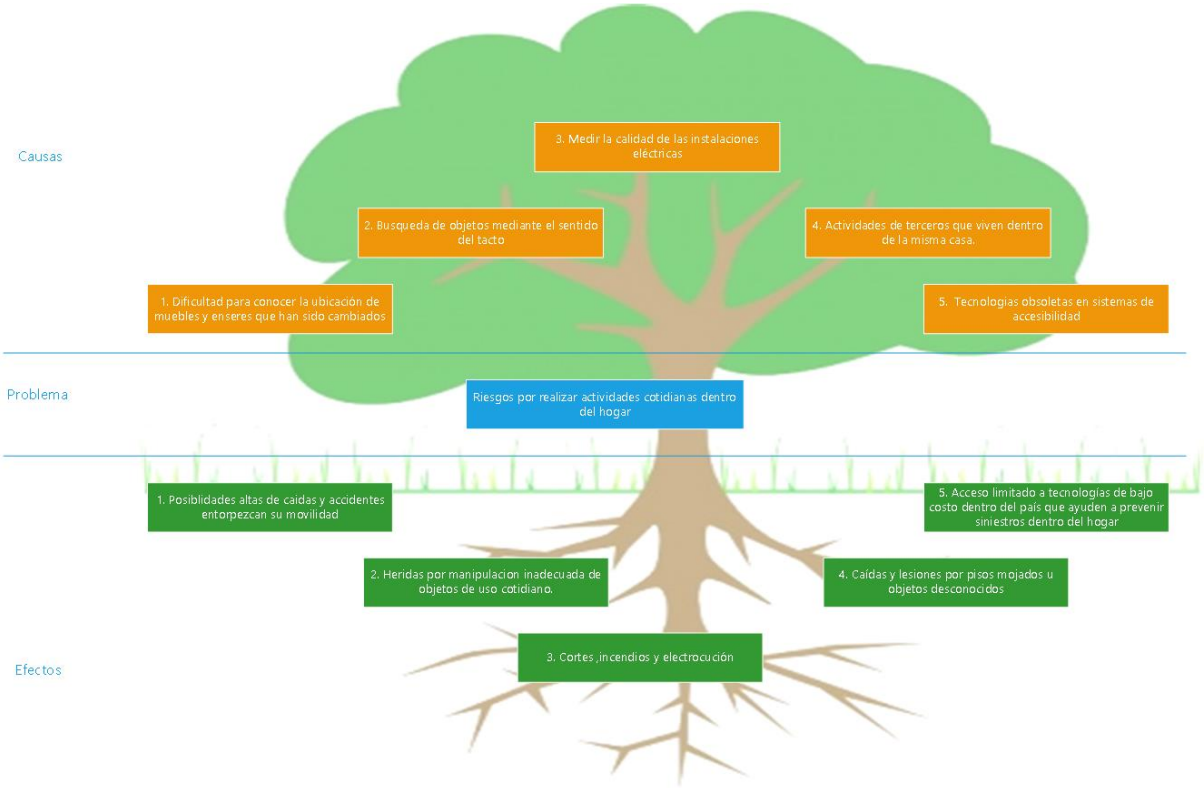
En la segunda sección se obtuvieron las respuestas mostradas a continuación .El entrevistado comenta en la sexta pregunta que :posee acceso un plan de internet dentro de su domicilio, el cual utiliza para navegar mediante su celular y computadora .La respuesta de la séptima pregunta es que el entrevistado no posee ningún tipo de herramienta tecnológica para movilizarse dentro del hogar que ayude a su movilidad y seguridad

.Finalmente en la octava pregunta ,el entrevistado indica que estaría de acuerdo en adquirir un sistema que ayude a reducir los riesgos en su entorno dentro del hogar ,siempre y cuando el costo del sistema no sea elevado .

Tomando como antecedente el análisis a la entrevista realizada , el método de observación directa y los estudios presentados en secciones anteriores ,es posible establecer la situación actual de la problemática presentada, se establece un problema central: Riesgos por realizar actividades cotidianas dentro del hogar. La mayor parte de personas deben enfrentarse a estos riesgos siempre, pero para las personas con discapacidad visual se tiene un nivel de peligrosidad más alto, especialmente por las actividades que realizan sus familiares dentro de su hogar.

En la siguiente figura se presenta un análisis realizado usando la técnica de árbol de problemas, tomando en cuenta las referencias presentadas.

Figura 44. Análisis de la situación actual utilizando el esquema de árbol de problemas.



Fuente: Autoría.

3.1.2. Perfil Mesh

El estándar de comunicación de red solamente funciona con la tecnología de comunicación inalámbrica Bluetooth a partir de las versiones 4.2 y superiores ,pero la version 5 no está soportada de forma nativa .Esto se debe a que en la version 5 y superiores de Bluetooth

se introdujeron mejoras en el bloque de controlador ,este bloque es usado completamente por el perfil Mesh, es decir no se puede utilizar el core de Bluetooth en conjunto con el perfil Mesh.

Los sensores utilizados podrán cambiarse en un futuro ,solo se deberá realizar pruebas de calibración y en caso de añadir más sensores ,se deberá realizar un cambio en la aplicación móvil que tendrán un nivel de dificultad de acuerdo a la habilidad de programación del desarrollador del sistema.

3.1.3. Versiones de Android en el mercado

Como se presentó en segmentos anteriores ,la version 8.0 y superiores funcionan en más del 80 % de los dispositivos mercado global ,por lo que se utilizara esta version como mínima para el funcionamiento de la aplicación móvil a desarrollar.

3.2. Análisis de los requerimientos del sistema

Dentro de esta fase se define los parámetros del proyecto basado en el estándar ISO/IEC/IEEE 29148:2018, dicho estándar especifica los procesos requeridos para la implementación de actividades que tengan como fin la búsqueda de requerimientos para sistemas, software, productos y servicios de ingeniería a través del ciclo de vida de este. Dentro de este estándar se explica la información necesaria para la aplicación de los requerimientos y procesos de dichos requerimientos especificados en el estándar ISO/IEC/IEE 15288 y ISO/IEC/IEE 12007 Ademas se especifica toda la información, formatos, entre otros que se debe incluir en documentos de ingeniería cuando se realiza análisis de requerimientos para un sistemas, software, productos y servicios. Esta información es aplicable :sin importar la metodología utilizada ,para cualquier persona que necesite utilizar un análisis de requerimientos en proyectos de ingeniería ,para personas que planeen regirse el estándar ISO/IEC/IEE 15288 y ISO/IEC/IEE 12007(ISO/IEC/IEEE, 2018).

El estándar especifica los requerimientos con los respectivos ítems que se explican y analizan a continuación:

3.2.1. Requerimientos de stakeholders

Antes de comenzar a listar los requerimientos en cuestión, se debe identificar a cada uno de los stakeholders involucrados en el desarrollo de este proyecto. En la Tabla 2 se lista a los actores considerados como stakeholders con el rol dentro del proyecto.

Tabla 2. Listado de Stakeholders involucrados en el proyecto.

No	Denominación	Rol
1	Personas con discapacidad visual	Usuarios directos del proyecto
2	Personas que conviven con las personas con discapacidad visual	Usuarios indirectos del proyecto
3	Msc. Edgar Maya	Director de Trabajo de Grado
4	Ms. Carlos Vásquez	Asesor del Trabajo de grado
5	Luis Farinango	Estudiante a cargo del proyecto de Grado
5	Universidad Técnica del Norte	Entidad de respaldo

Fuente: Autoría

Hay que recordar que el estándar ISO/IEC/IEEE 29148 especifica siglas para cada uno de los requerimientos por analizar en el desarrollo de proyectos. Las siglas utilizadas se definen en la

Tabla 3. Siglas y significado de los requerimientos a analizar

Siglas	Significado
StRS	Requerimientos para stakeholders
SyRS	Requerimientos para el sistema
SRS	Requerimientos para arquitectura

Fuente: Autoría

Con la lista de stakeholders y de acuerdo con la situación actual de los usuarios del sistema entonces es posible obtener los requerimientos de los stakeholders. Estos requerimientos definen como va a ser utilizado el sistema en determinados escenarios bajo la perspectiva de los involucrados en el proyecto. Se definen en base a los usuarios y al funcionamiento operacional del sistema. Los respectivos requerimientos se presentan en la

***Tabla 4*.**

Tabla 4. Requerimientos de stakeholders.

StRS					
Requerimientos de Stakeholders					
No	**Requerimiento**	**Prioridad**			**Relación**
		Alta	**Media**	**Baja**	
Requerimientos de Usuarios					
StRS1	Las alertas generadas por el sistema serán comunicadas al usuario por medio de sintetizadores de voz	X			
StRS2	Implementación de una aplicación móvil para el control centralizado del sistema				
StRS3	Función de navegación con voz dentro de la aplicación		X		
StRS4	La aplicación debe tener un indicativo perceptible para las personas con discapacidad visual cuando se navegue entre pantallas	X			
StRS5	Los costos de diseño y construcción serán bajos.	X			
StRS6	La ubicación y tamaño de los sensores no afectará la ejecución de actividades cotidianas de los usuarios dentro del domicilio.		X		
StRS7	El sistema debe permitir el acceso a usuarios indirectos y al administrador del sistema en forma continua e ininterrumpida	X			
Requerimientos Operacionales					
StRS8	El nodo central del sistema debe ser un smartphone con características de hardware para emitir alertas de voz		X		

StRS9	Todos los componentes de la red de sensores del sistema deben contar con soporte de la tecnología Bluetooth en su version 4.x o superior.	X		
StRS10	El nodo central del sistema debe tener conexión a internet estable		X	
StRS11	El smartphone que se va a usar en el sistema debe contar con sistema operativo Android 8.0 superior.			X
StRS12	La aplicación móvil que controlara el sistema debe desarrollarse usando un IDE que brinde documentación sólida y libre acceso	X		
StRS13	El sistema debe tener alimentación eléctrica para cada uno de los nodos usando una fuente externa o una batería		X	
StRS14	La aplicación móvil tardara máximo de 1 minuto en emitir las notificaciones de alerta		X	

Fuente: autoría

Los requerimientos mencionados han sido escogidos en mayor parte con el uso de una entrevista realizada a un cliente potencial del sistema ,también se ha añadido factores

3.2.2. Requerimientos del sistema

Los requerimientos del sistema permiten identificar las condiciones técnicas dentro de las cuales el proyecto debe funcionar, así como la interacción sistema-usuario. En si los requerimientos que se van a listar a continuación describen que debería hacer el sistema, especialmente dentro del contexto donde va a funcionar y tomando en cuenta el óptimo funcionamiento de este. Los requerimientos se pueden observar en la .

Tabla *5* .

Tabla 5. Requerimientos del sistema

SyRS					
Requerimientos del sistema					
No	**Requerimiento**	**Prioridad**			**Relación**
		Alta	**Media**	**Baja**	
Requerimientos de uso					
SyRS1	El usuario recibirá notificaciones de audio mediante el smartphone usando archivos mp3 alojados dentro de la aplicación móvil	X			StRS1
SyRS2	Recabar datos de: aire, estado de conexiones eléctricas, niveles de agua y posición de personas para mostrarlos en una aplicación móvil.	X			
SyRS3	La aplicación móvil tendrá funciones de accesibilidad que permitan a personas con discapacidad visual hacer uso de las funciones.	X			
SyRS4	El sistema presentará un resumen de estadísticas y eventos al desarrollador para realizar mejoras a futuro.			X	
SyRS5	El sistema debe permitir la coexistencia con otras tecnologías de comunicación inalámbricas para realizar mejoras a futuro.		X		
Requerimientos de rendimiento					
SyRS6	La aplicación móvil tendrá un retardo máximo de 1 minuto al activar cualquiera de sus funciones de alerta.		X		
SyRS7	Los sensores deben tener el tiempo de respuesta corto y con la menor cantidad de errores	X			
SyRS8	La interfaz web de presentación de datos tendrá bajo consumo de datos y corto tiempo de respuesta en el proceso de actualización de datos		X		
SyRS9	La comunicación entre los sensores y el nodo central tendrá un retardo máximo de 1 minuto	X			
SyRS10	La aplicación móvil tendrá un tiempo de respuesta corto en el proceso de interacción con el usuario		X		

SyRS11	Las alertas de voz tendrán una duración corta y un mensaje claro en la medida de lo posible	X			
	Requerimientos de interfaces				
SyRS12	Las placas de desarrollo utilizadas en el sistema deben tener una entrada USB o SWD para que puedan ser configuradas mediante una conexión serial a un ordenador.			X	
SyRS13	Las placas de desarrollo utilizadas deben contar con antenas integradas para la conexión inalámbrica a través de Bluetooth.	X			
SyRS14	Un nodo del sistema debe funcionar con alimentación eléctrica ya sea usando cables USB o baterías		X		
SyRS15	El nodo central de la red debe contar con una interfaz inalámbrica para conexión Bluetooth y una interfaz inalámbrica para conexión a Internet de forma simultanea	X			
SyRS16	El sistema tendrá una interfaz de presentación de información para usuarios directos e indirectos		X		
SyRS17	La base de datos guardará las notificaciones del sistema en registros organizados y legibles			X	
	Requerimientos de modos /Estados				
SyRS18	El sistema consumirá los datos que la aplicación el envié cuando se encuentre activo	X			
	Requerimientos físicos				
SyRS19	El tamaño de los módulos del del sistema no debe ser mayor a 15 cm.		X		
SyRS20	Los sensores no se conectarán de forma permanente a la placa de desarrollo para un posterior remplazo de componentes			X	**SyRS19**
SyRS21	Los módulos del sistema deben instalarse en una caja empotrada en la pared o similar.		X		**SyRS19**
SyRS22	Los nodos de la red deben ser portátiles para realizar la reubicación de estos en caso de ser necesarios			X	**SyRS19**

Fuente: autoría.

3.2.3. Requerimientos de la arquitectura

Los requerimientos de arquitectura identifican a los componentes que se van a utilizar y las características que deben tener para cumplir con cada uno de los objetivos del proyecto. Los componentes abarcan tanto a software como hardware con el cual el proyecto funcionara de forma óptima. En la

Tabla 6 se muestran cada uno de los requerimientos de arquitectura a tomar en cuenta.

Tabla 6. Requerimientos de arquitectura

SRS					
Requerimientos de Arquitectura					
No	**Requerimiento**	**Prioridad**			**Relación**
		Alta	**Media**	**Baja**	
Requerimientos Lógicos					
SRS1	El sistema debe ser capaz de enviar y recibir datos desde y hasta al cliente en un esquema de acuerdo con el Perfil Mesh	X			
SRS2	El sistema debe funcionar en conjunto a una aplicación móvil compatible con el Perfil Mesh	X			
Requerimientos de diseño					
SRS3	Las placas y equipos usadas en el sistema deben contar con características de software y hardware para la construcción de una red Bluetooth Mesh sin la necesidad de equipos adicionales.	X			**SRS1**
SRS4	El nodo central emitirá alertas de audio por cada sensor presente en el sistema.	X			
SRS5	El sistema podrá controlar la iluminación de la vivienda sin interferir con las instalaciones eléctricas actuales.		X		
SRS6	Cada módulo del sistema tendrá asociado una serie de sensores dentro de en un hub.		X		
SRS7	El proceso de cambio de sensores de un módulo puede ser realizado con el módulo en funcionamiento			X	
SRS8	Los módulos deben contar con un identificador en código Braille con la palabra sensor o similar.	X			
Requerimientos de Hardware					
SRS9	Las placas de desarrollo para la construcción de módulos	X			**StRS9**

	usados deben contar con conectividad Bluetooth version 4.2 o superior.			
SRS10	Placa de desarrollo debe contar con entradas analógicas y digitales para conexión de sensores.	X		
SRS11	Sensores compatibles con las placas de desarrollo utilizadas.	X		
SRS12	Los sensores de aire deben detectar principalmente GLP.	X		
SRS13	El sensor de agua debe medir nivel de agua y realizar la detección sin ser introducido totalmente al medio o en su defecto el circuito de detección debe estar alejado y ser impermeable.	X		
SRS14	El sensor de incendios debe estar a una distancia segura del flagelo.		X	
SRS15	El sensor de corriente debe obtener los valores de forma no invasiva en el ambiente de medición		X	
SRS16	El sensor de corriente podrá desacoplarse de forma fácil de un lugar de medición para localizarlo en otro			X
SRS17	El sensor de detección de personas debe tener un alcance de al menos 6 metros		X	
	Requerimientos de software			
SRS18	La placa de desarrollo debe contar con soporte para librerías de los sensores a utilizar.		X	
SRS19	El lenguaje de programación deberá ser basado en código abierto y sin costo.		X	
SRS20	La aplicación móvil debe permitir accesibilidad para personas con discapacidad visual.	X		

SRS21	Las placas de desarrollo utilizadas deben contar con un entorno de programación para el desarrollo de aplicaciones y de uso libre		X		
SRS22	El lenguaje de programación usado para el desarrollo de la aplicación móvil debe ser intuitivo e integrar funciones nativas sin la necesidad de complementos adicionales		X		
SRS23	El SDK para el desarrollo de la aplicación móvil debe tener librerías o plugins que permitan administrar conexiones inalámbricas usando Bluetooth Low Energy.	X			
	Requerimientos eléctricos				
SRS24	Los dispositivos del sistema deben funcionar con una alimentación eléctrica de entre 3.3 v y 5 v D.C a excepción de las luminarias .			X	**SRS25**
SRS25	El sistema de alimentación eléctrica de los módulos del sistema debe ofrecer 600 mA de corriente mínima, a excepción de las luminarias		X		**SRS24**

Fuente: autoría

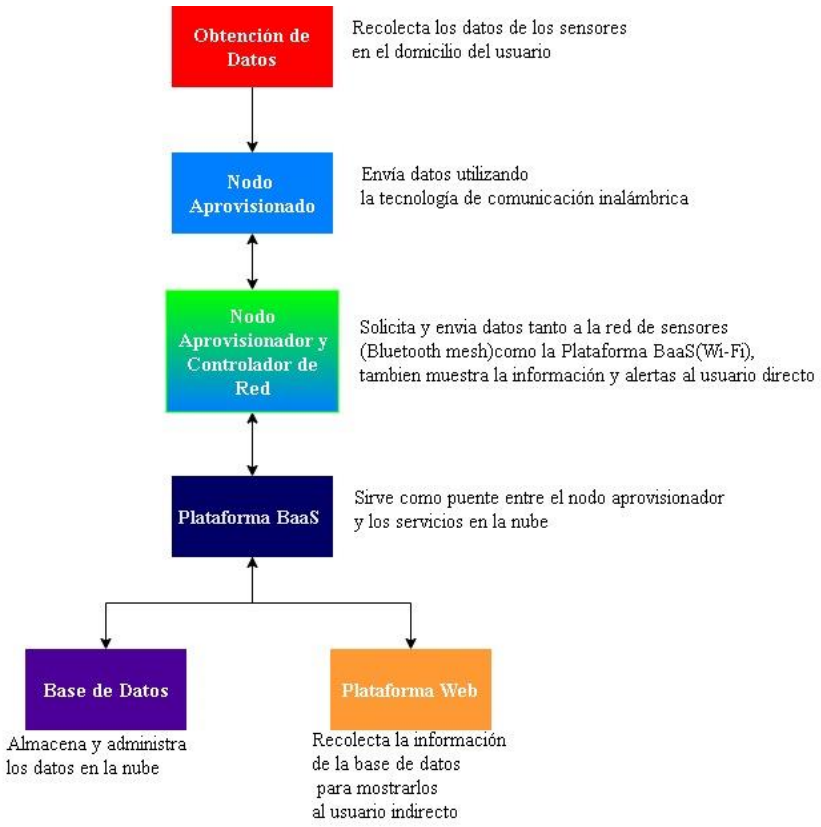
3.3. Diseño del sistema

Con cada uno de los requerimientos del sistema definidos se procede a la fase de diseño cumpliendo con las respectivas normas y estándares. Dentro de esta fase se define cada uno de los módulos del sistema y la relación que tendrán entre cada uno para lograr el óptimo funcionamiento del sistema. Así como también se escogerá los equipos y programas con los que se va a trabajar en base a los requerimientos ya definidos previamente. De igual forma se diseñará los códigos y algoritmos de programación óptimos para cada módulo del sistema.

3.3.1. Diagrama de bloques

El diagrama de bloques muestra el proceso de funcionamiento del sistema. Cada una de las fases juega un papel muy importante para el correcto funcionamiento del sistema .El sistema funciona de acuerdo con la Figura 45.

Figura 45. Funcionamiento del sistema representado mediante un diagrama de bloques.



Fuente: Autoría.

El sistema para lectura de datos con sensores usando una bluetooth mesh consta de 5 fases claramente diferenciadas :

El primer bloque se encarga de la obtención de datos en el ambiente de funcionamiento del sistema ,dichos datos son recabados mediante sensores conectados a una placa de desarrollo .La placa de desarrollo se encargará de reducir al mínimo los errores de medición de los sensores ,así como de la alimentación eléctrica de los mismos. Se elije el pin al cual se conectará los sensores para envío de datos y alimentación eléctrica, así como el funcionamiento del ADC utilizando un voltaje de referencia adecuado. Además, los nodos necesitan alimentación eléctrica también ,la cual se logra utilizando un cargador USB conectado a la placa de desarrollo. Estos datos serán solicitados por el usuario directo cuando el sistema entre en funcionamiento.

El segundo bloque establece como se controlará a los nodos de la red ,es decir definirá las características con las cuales un nodo se conectará a la red para el envío y recibo de datos utilizando la respectiva tecnología de comunicación inalámbrica .Aquí se define identifica el tipo de dato a enviarse por la red de sensores ,técnicamente se llama Identificador de propiedad de sensor y tiene un tamaño específico. Para el desarrollo de este proyecto se usara los valores 0x0059 , 0x005D ;estos valores se define como generales ya que hace referencia a un voltaje de entrada y corriente de entrada pero en si se eligen porque solo utilizan 2 bytes de carga útil en la capa más alta del estándar de comunicación de red .También se define el rol del nodo en la red ,las funciones que permitirá y los respectivos modelos que soportará .El usuario indirecto y administrador podrán revisar y realizar las configuraciones de los nodos.

En la tercera fase se define como será la interacción entre el nodo central de la red ,la red de sensores y la plataforma BaaS. El nodo central será un smartphone por lo cual deberá conectarse a la red de sensores utilizando Bluetooth mesh y a su vez también se debe conectar al internet con una conexión inalámbrica mediante Wi-Fi. Se utilizará una aplicación móvil

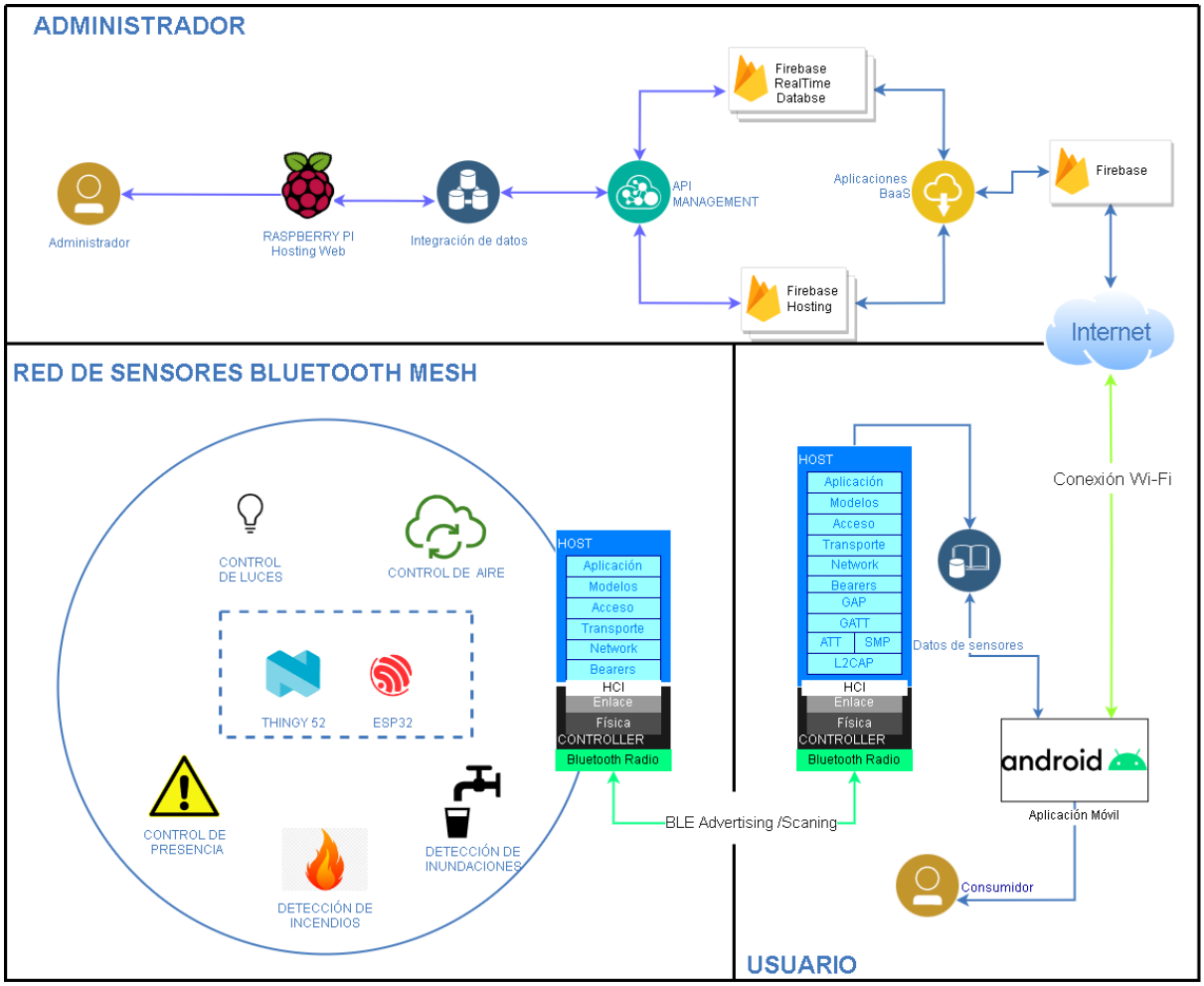
que permitirá la interacción con la red de sensores y el envío de datos a una base de datos en la nube de forma casi simultánea .Así como la presentación de datos a los usuarios directos cumpliendo accesibilidad e inclusión .El administrador será el encargado total de esta fase del sistema .

La cuarta fase comprenderá la interacción del nodo central con una plataforma BaaS ,la cual brindara servicios para el envío de datos a la nube y web Hosting .Los datos se guardarán en la base de datos en tablas ,la tabla contendrá información del nombre del nodo así como los valores de cada sensor de acuerdo al nodo .Ademas se guardará alertas de acuerdo a valores dictados por el nodo central ,la alerta constara de la fecha en donde se registró así como la identificación de nodo y sensor. El administrador de red también será el encargado total de esta fase del sistema .

La fase quinta y final se mostrará los datos a los usuarios directos e indirectos ,para los usuarios directos se utilizará la propia aplicación móvil .Las alertas serán audios grabados con un sintetizador de voz que se reproducirán en base a las alertas guardadas en la base de datos en la nube .Mientras que se utilizara una página web para mostrar los datos a los usuarios indirectos

Para una mayor extensión del funcionamiento del sistema se presenta la arquitectura de este que se puede observar en la Figura 46.En donde se podrá observar a detalle cada uno de los módulos, así como a la tecnología de comunicación inalámbrica se utilizará.

Figura 46. Arquitectura del sistema.



Fuente: Autoría.

3.3.2. Descripcion lógica de los bloques del sistema

En base a los bloques del sistema es necesario definir una lógica que permita lograr con éxito la tarea encomendada a cada bloque ,es decir la estructura lógica de los códigos de programación a implementar ,a continuación

En algunos casos es necesario la definición de roles .Los roles se calificarán en categoría :General ,Usuarios indirectos y usuarios directos.

El rol general indica las acciones que se debe realizar o tomar en cuenta para todo el funcionamiento del sistema que involucra al nodo central ,el rol de usuarios indirectos muestra todas las acciones que el nodo central debe permitir a los usuarios indirectos y el administrados del sistema ,el rol de usuarios directos se refiere a las acciones específicas que el sistema deberá

cumplir con los usuarios directos .En los roles no se tomara en cuenta la presentación de datos ,dado que se trata de otro bloque del sistema .

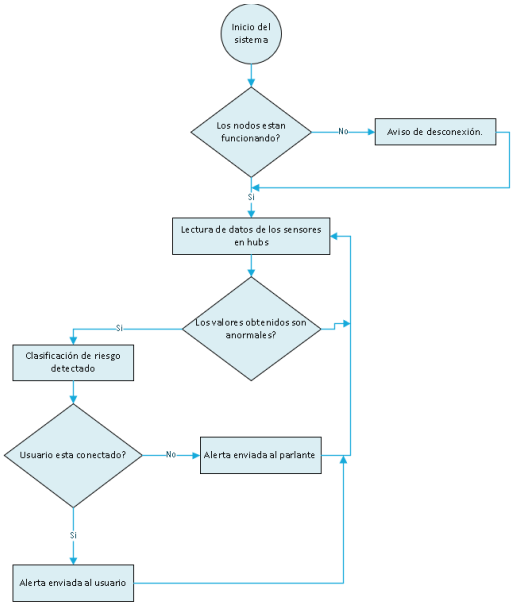
Para demostrar de forma lógica el funcionamiento de cada una de las fases del sistema se utilizará más de un tipo de diagramas ,cada diagrama se ajustará para explicar dicha fase del sistema.

A continuación, se detalla cada una de las funciones lógicas que debe incluir cada una de las fases del proyecto

3.3.2.1. Obtención de datos

Se trata de una de las fases más simples del sistema ,el usuario ya conectado al sistema solicitara los datos al sistema .Es posible realizar un diagrama de flujo explicar el funcionamiento de esta fase. Para lo cual con el sistema inicializado y funcionando con normalidad, cada uno de los nodos comenzaran con la lectura de datos de acuerdo con los sensores instalados en cada nodo . Como primer análisis se detecta que todos los nodos estén activos y con los respectivos sensores funcionando, en el caso del no funcionamiento de un nodo no se actualizarán los valores anteriores. A continuación, se realiza la lectura de los sensores y se compara los valores obtenidos. En caso de que alguno de los nodos presente un valor de alerta se identifica el valor y se enviara la alerta. Dicha alerta se envía a través de la red para avisar al usuario. La alerta seguirá emitiéndose hasta que el evento que la desencadeno pare. El diagrama de flujo de este proceso se observa en la Figura 47.

Figura 47. Diagrama de flujo del proceso de lectura de sensores.



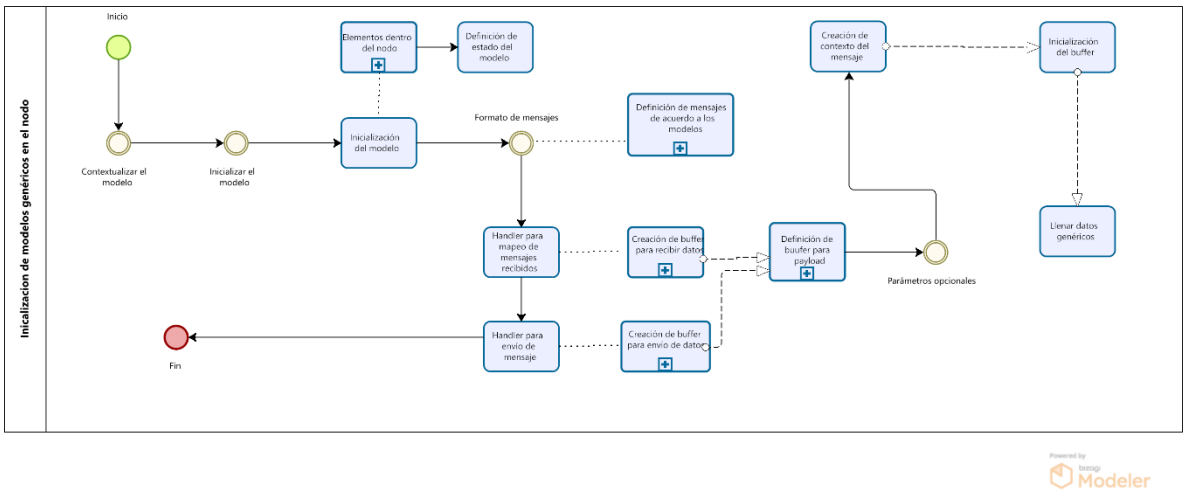
Fuente: Autoría.

3.3.2.2. Nodos aprovisionados de la red de sensores

3.3.2.2.1. Usuarios indirectos :Configuración de parámetros genéricos para modelos en nodos de la red Bluetooth Mesh

Ciertos nodos de la red ademas de albergar sensores, luces, ademas deberán cumplir con la función de proxy o futuro aprovisionador, cada uno de estos roles está determinado por un cierto modelo. Estos modelos permiten inicializar cada una de las características de los equipos de acuerdo al rol en la red ,por lo que algunos parámetros pueden considerarse genéricos para todos los equipos de la red .La programación de estos equipos comparte esta misma lógica por lo que se realiza un callbacks a APIs definidas por los fabricantes ,los equipos utilizados se basan en las APIs de Zephyr para implementar Bluetooth Mesh .En la Figura *48* se resume la configuración típica que se debe seguir para levantar cualquier tipo de modelo en cualquiera de los nodos de la red .

Figura 48. Estructura de programación para definir los paramentos genéricos de un nodo dentro de la red.



Fuente: Autoría

Como se pudo observar en la estructura de programación presentada ,se inicia con la contextualización del modelo .Esta operación generalmente define el Setup del nodo ,ya sé que se trate de un modelo de servidor o cliente .Con la contextualización del modelo completo se instancia el o los modelos dentro del nodo en cuestión ,en esta fase se inicializan todos los parámetros del nodo en cuestión es decir se inicializa todas las variables para que a futuro el nodo se aprovisione en una red Bluetooth mesh. Generalmente toda la información necesaria está contenida en estructuras de datos que se alojan en las APIs de Zephyr, por lo que se hace el llamado a distintas APIs de acuerdo a los modelos que va a contener el nodo y su futuro rol en la red. Estas estructuras ayudan a definir a los elementos del nodo, dichos elementos van a ser llamados en métodos propios de Zephyr de acuerdo a si se está utilizando algún modelo genérico o se está utilizando un modelo propio.

La construcción de un modelo propio es necesario cuando se va a realizar una tarea que no está definida en los modelos de acuerdo a la Bluetooth SIG. Para el desarrollo de este proyecto no se va a utilizar los llamados modelos de vendedor, por lo que solo se utilizara los modelos ya definidos. Caso seguido se procede a definir los elementos que tendrá el modelo, los elementos están altamente relacionados al tipo de modelo utilizado y generalmente se

declaran en arrays de acuerdo al número de elementos del nodo. Hay que recalcar que en caso de que se declare varios elementos dentro de un mismo nodo se deberá realizar la declaración de los modelos para cada uno de los elementos dentro del nodo, esta etapa finaliza con la definición de parámetros del estado del modelo.

La siguiente fase de programación define el formato de los mensajes que se van a transportar a través de la red, para definir los mensajes se utiliza la información de acuerdo al modelo programado en dicho nodo. Generalmente se define tres tipos de mensajes: GET, SET y ACK. Los mensajes de GET sirven para recibir mensajes de la red, los mensajes de SET envían mensajes a través de la red y los mensajes de ACK son utilizados para la confirmación de recepción o Envío de mensajes. También se puede definir más tipos de mensajes de acuerdo al funcionamiento del nodo en la red, tomando en cuenta que el formato ha sido definió se utiliza los llamados handler para recibir y enviar mensajes.

Los handler definen un buffer de red, estos buffers se utilizan mediante macros de Zephyr. Los buffers deben tener ciertos parámetros para el envío y recepción de mensajes, la diferencia radica en el contexto del mensaje. El contexto definirá algunos parámetros que difieren al envío o recibo de mensajes del nodo, de acuerdo al contexto se inicializa el buffer y se llena los datos para en una etapa posterior enviar o recibir mensajes.

Por lo que al finalizar esta fase se ha conseguido: un dispositivo programado en base a un modelo con todos los parámetros para funcionar y aprovisionarse dentro de una red, así como enviar o recibir mensajes. Es decir, se ha programado un nodo con la funcionalidad *plug and play*.

3.3.2.2.2. Usuarios indirectos : configuración de lectura de datos de los sensores utilizados

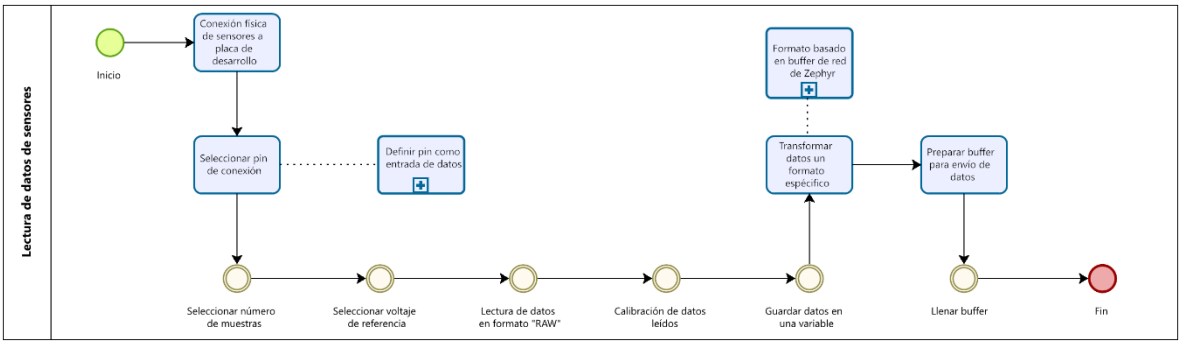
Para la lectura de sensores se debe realizar la interpretación de los datos recogidos por los sensores, además el estándar de comunicación de red define propiedades de dispositivos

para ayudar a identificar los tipos de datos recogidos por los sensores. Estas propiedades deberán coincidir con los tipos de sensores utilizados.

Los sensores utilizados para medir la calidad de aire y la presencia de GLP en el aire serán: sensor MQ6 y sensores internos del Thingy :52. Para el sensor MQ6 es posible utilizar la placa de desarrollo ESP 32, mientras que el Thingy: 52 debe funcionar por su cuenta. Para el caso del sensor de agua y sensor de detección de fuego se puede usar las lecturas analógicas para obtener datos. Hay que recordar que el firmware ya viene precargado en la bombilla inteligente usada en el desarrollo de este proyecto por lo cual no es necesario realizar ningún cambio.

La lectura de datos de los sensores de la red se realizará a través del propio código de cada uno de los equipos utilizados, pero en si se utilizará el conversor analógico a digital de las placas de desarrollo utilizadas. Caso seguido se debe realizar la transformación de los datos recibidos a un formato que sea compatible con el código de programación respectivo para ser enviados a través de la red. Este proceso se realizará a través de un buffer de red que se define en la documentación de Zephyr, dicho buffer tiene un formato específico para que cualquier equipo de la red pueda entender los datos enviados y actuar de acuerdo a la configuración predeterminada de la red. En la se resume la lógica de programación utilizada para realizar la lectura de sensores y después configurar los datos para que sean enviados por la red Bluetooth mesh.

Figura 49. Estructura de programación para definir los paramentos genéricos de un nodo dentro de la red.



Fuente: Autoría

Para algunos de los sensores algunas de las etapas presentadas se pueden omitir. Como se puede observar en la lógica de programación presentada para la lectura de sensores, se inicia desde la conexión física del sensor con la placa de desarrollo. Posteriormente se selecciona el pin al cual el sensor ha sido conectado así como el tipo de pin que se le asigna ,para la lectura de sensores siempre se configura a los pines como entrada de datos .En este punto se debe realizar la configuración del ADC ;se inicia con la configuración de las muestras utilizadas ,a continuación se define un voltaje de referencia para realizar posteriormente la calibración de los datos leídos por el sensor, como paso seguido se realiza la lectura de los datos enviados por el sensor en lo que se podría llamar formato RAW, después se realiza la calibración en base al valor de voltaje mencionado para finalmente guardar la información en una variable y terminar el proceso de ADC .

Con este primer proceso terminado se debe realizar las conversiones de los datos digitales a un formato que sea reconocido en la red. Es decir, se ingresará los datos a un buffer de red para posteriormente enviar los datos a la red, dicho proceso finaliza la lectura de datos. A continuación, se debe realizar las configuraciones en los modelos para ingresar los respectivos datos en la red.

3.3.2.3. Rol del nodo central del sistema

El nodo central de la red será un smartphone con la aplicación desarrollada ya instalada ,llevará a cabo muchas tareas dentro del sistema .Dado que se encargará de la conexión a la red hasta el envío de datos hacia la plataforma en la nube se utilizarán algunos de los roles ya

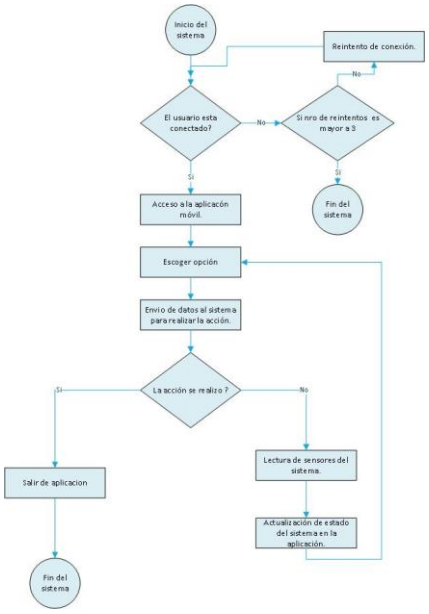
General :Conexión a la red de sensores desde el nodo central

3.3.2.3.1. General : Reconexión a la red de sensores

Al estar en funcionamiento el sistema se entra a la aplicación mediante el smartphone; se toma en cuenta que el usuario debe estar dentro de la red para recibir la información y este proceso se realizara antes de entrar a la sección .Se tendrá dos opciones : apagar las luces encendidas dentro del hogar y mostrar el estado del sistema .Después de escoger una de las dos opciones se ejecutara la opción ,se realizara una lectura de sensores para actualizar la aplicación con el estado actual del sistema y comprobar si se realizó la acción o no .Para finalmente avisar por un comando de voz.

Se añade un proceso de conexión al usuario para que solo se le permita un cierto número de intentos para acceder a la red antes de desconectarlo. . El diagrama de flujo de este proceso se muestra a continuación.

Figura 50. Diagrama de flujo del proceso de control de sistema.

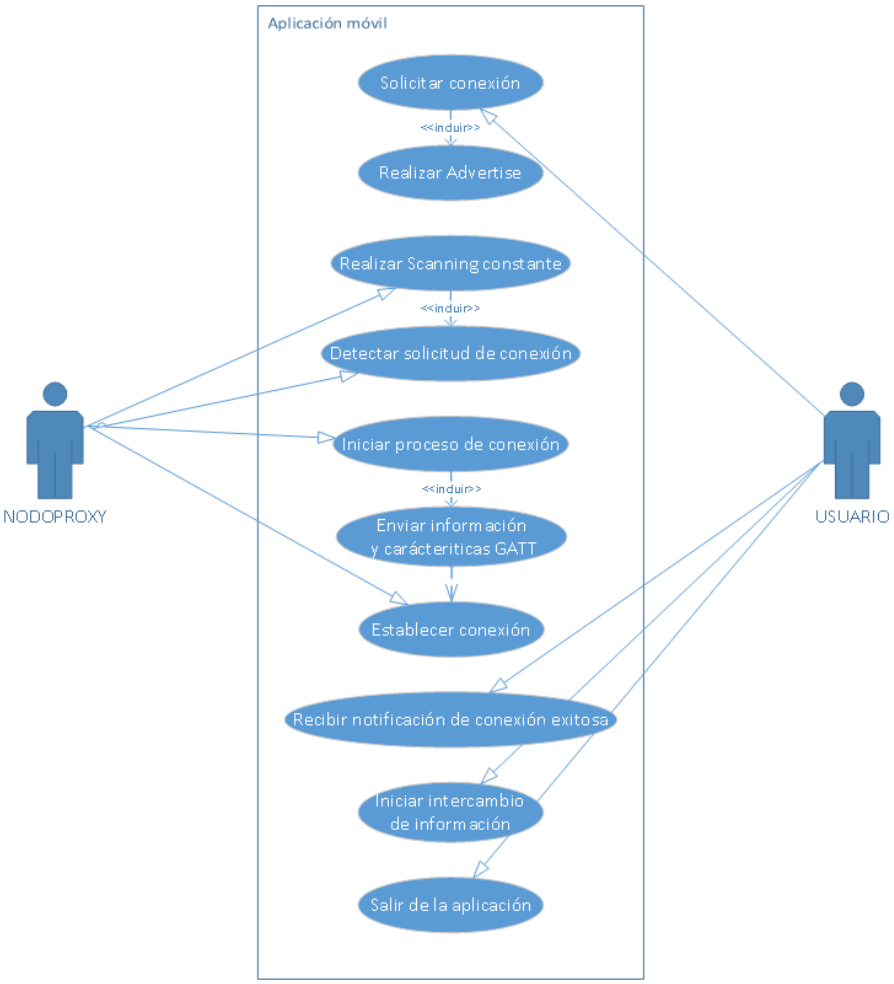


Fuente: Autoría.

3.3.2.3.2. General : Conexión a la red de sensores mediante un dispositivo externo

Esta funcionalidad de la aplicación permitirá que un dispositivo móvil que no funcione bajo el estándar Bluetooth mesh pueda recibir datos a través de una conexión usando Bluetooth Low Energy. Para lograr este cometido es necesario un nodo en la red que funcione como un servidor GATT y a la vez este dentro de la red ,El cliente realizara la conexión a este nodo y pasara a realizar el intercambio de información para establecer una conexión a través de Bluetooth Low Energy para que el cliente finalmente reciba una notificación de conexión exitosa e inicie con el intercambio de información ,el proceso puede finalizar cuando el usuario salga de aplicación luego de recibir la información . Estas acciones se resumen en la Figura *51*

Figura 51. Diagrama de casos de uso para entrar y salir a la aplicación móvil



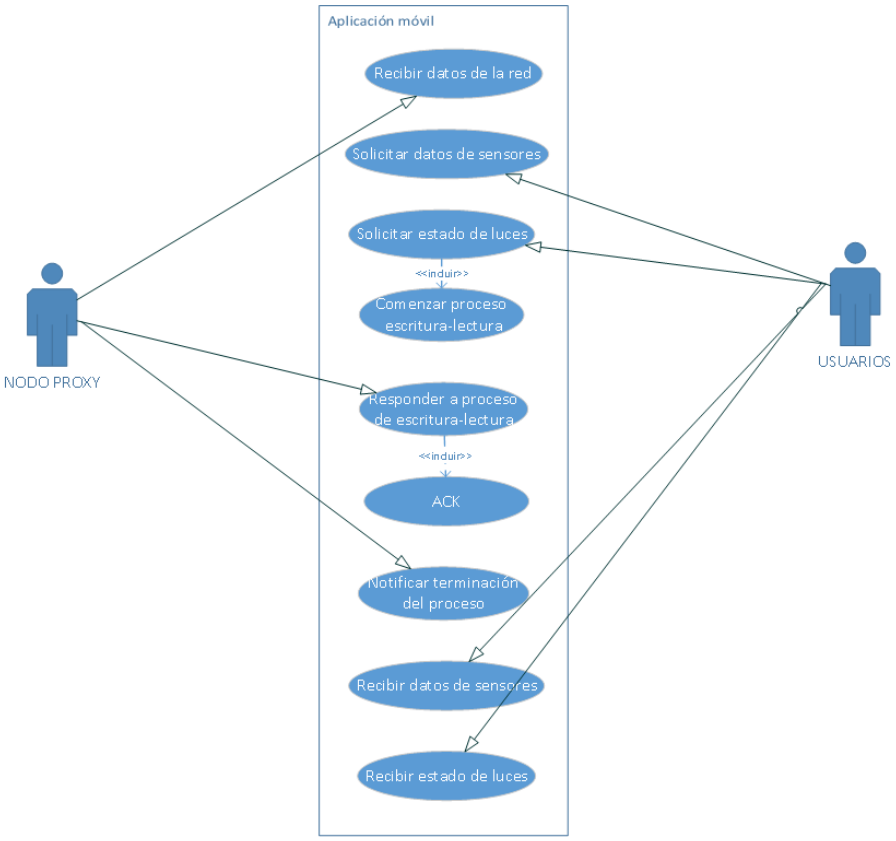
Fuente: Autoría.

3.3.2.3.3. Usuario directo :Recibir notificaciones de la red de sensores

Para este proceso se toma en cuenta que el usuario ya ha establecido la conexión con el nodo proxy y ya se ha intercambiado todos los parámetros para él Envío y recibo de datos hacia la aplicación móvil. Los datos ya están siendo enviados al nodo proxy por lo que es necesario que la aplicación móvil pueda recibir información a través de una conexión GATT maestro-esclavo. Cuando la aplicación solicita la información se inicia un proceso de lectura-escritura en los dos nodos, dependiendo del tipo de información se recibe un ACK, el proceso finaliza cuando el servidor envía una notificación de la terminación del proceso que no tiene un ACK por parte de la aplicación móvil. Este proceso se resume en la

Figura *52*.

Figura 52. Diagrama de casos de uso para recibir las notificaciones en la aplicación móvil.

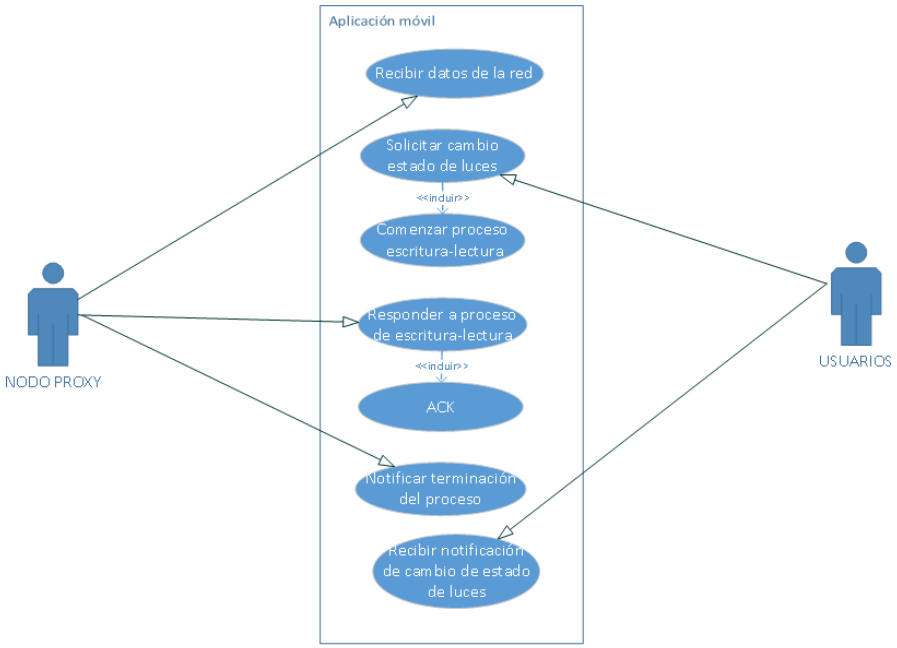


Fuente: Autoría.

3.3.2.3.4. Usuario directo: Control de luces en el sistema

El control de luces será muy similar a la funcionalidad de recibir notificaciones solo que, para este caso específico, el proceso de escritura-lectura realizará el cambio de estado de las luces del sistema. El proceso terminara al recibir una notificación del cambio de estado de las luces. Se resume todo el proceso en la Figura *53*.

Figura 53.Diagrama de casos de uso para controlar las luces del sistema .

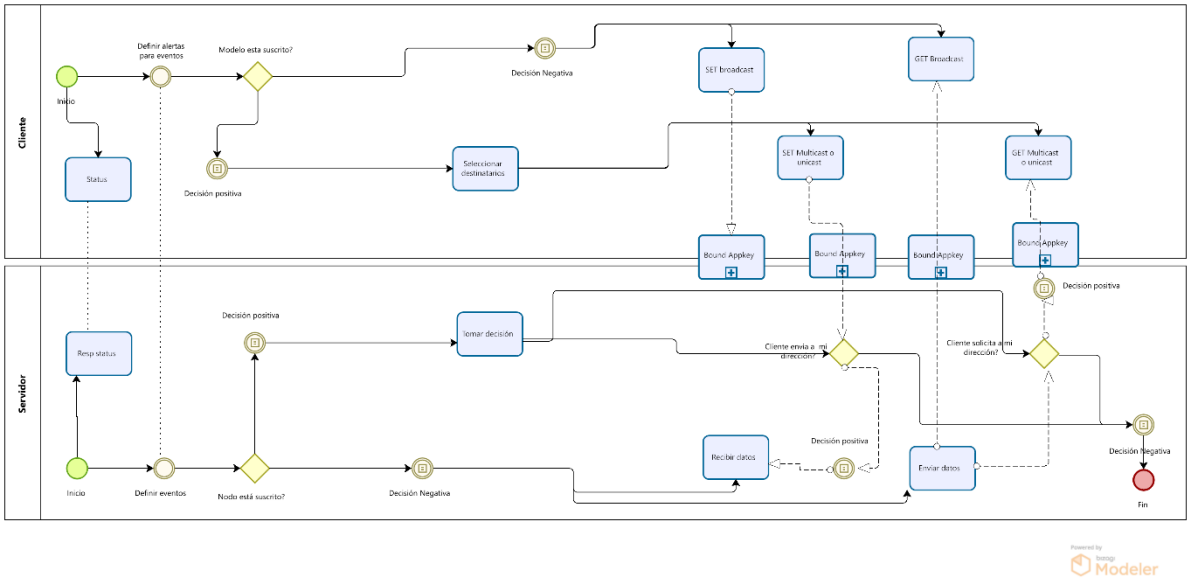


Fuente: Autoría.

3.3.2.3.5. Usuarios indirectos :Configuración de modelo servidor y modelo cliente para la lectura y recibido de datos de los sensores

El servidor será el encargado de la lectura de sensores mientras el cliente solicitará la lectura del estado de estos a través de peticiones. Anteriormente se realizó la configuración de un modelo genérico, así como la lectura de datos de los sensores mencionados por lo que ahora se debe construir las líneas de código para que un modelo de sensores sea capaz de funcionar y enviar los datos al cliente. Por otro lado, el cliente será el equipo que tenga configurado este modelo para recibir los datos de los modelos servidores, pudiendo ser de uno o varios equipos. Para realizar este proceso se hace uso de un buffer de red antes mencionado, se puede manejar a los servidores mediante operaciones de GET y SET. La lógica de programación se puede observar en la Figura *54*

Figura 54. Diagrama de procesos para la configuración de modelo servidor y cliente para la lectura de datos desde los sensores.



Fuente: Autoría.

Como se observó en la lógica de programación presentada para este proceso ,ahora se debe configurar el modelo cliente y servidor con sus respectivas relaciones .Se inicia definiendo eventos en el lado del servidor mientras en el lado del cliente será posible definir alertas para estos eventos .Se define dos tareas principales en el lado del cliente : GET y SET ,estas tareas se encargaran de leer el estado actual de los sensores así como enviar datos a los servidores para cambiar el estado de algún servidor o actuador dentro de la red. Antes de solicitar datos se debe la AppKey al modelo, cuando se realiza este proceso se puede enviar y recibir datos.

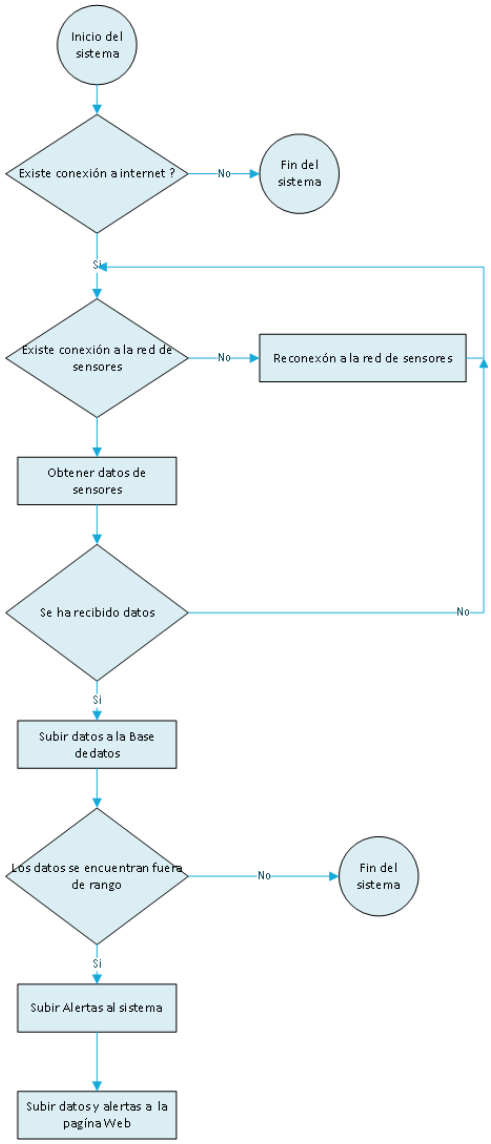
En si GET y SET son las únicas opciones que se deben programar en esta sección. Solo hay que tomar como parámetro adicional a la direccion a la cual estos servidores están suscritos para conocer hacia donde se estarán publicando los datos.

3.3.2.4. Plataforma BaaS

La plataforma BaaS servirá como puente entre los datos que se envían a la base de datos desde el nodo central del sistema .También se utilizara un servicio de web hosting para alojar la presentación de información para los usuarios indirectos. No es necesario la definición de

roles ,el proceso inicia consideran que antes se ha conectado al sistema y se ha solicitado datos a los sensores .El sistema inicia ,se recibe los datos de todos los sensores identificados con direcciones unicast .Los datos se guardaran en una base de datos en tiempo real con un identificativo de nombre de sensor y el número de sensor en el nodo .También se envía datos cuando se enciende una alerta ,la alerta se guarda en la misma tabla pero se solicita el tiempo en formato dd/mm/yy ademas se guarda un identificador del nodo .Para esta operación se necesita una conexión estable a internet .Por lo que se utilizara un diagrama de flujo para explicar lógicamente el funcionamiento de esta fase. Se puede observar dicho diagrama en la *Figura 55* .

Figura 55. Diagrama de flujo para la plataforma BaaS .

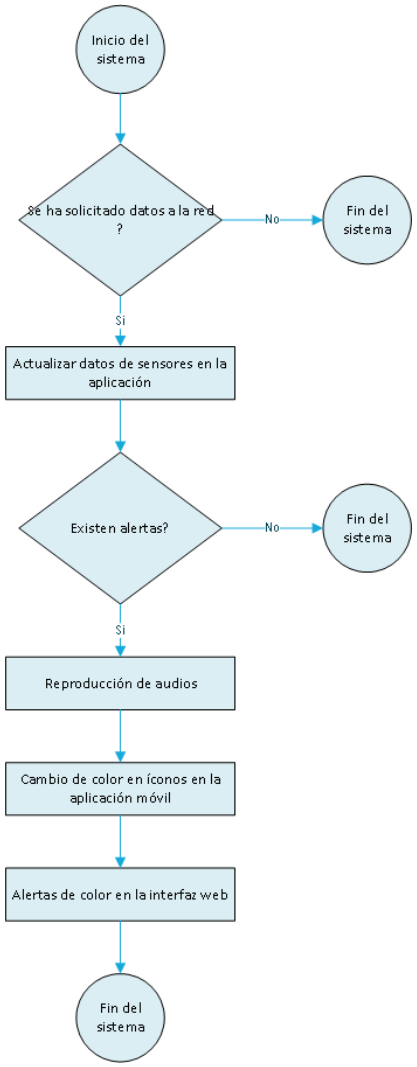


Fuente: Autoría.

3.3.2.5. Presentación de información a usuarios directos e indirectos

Llegados a este punto del funcionamiento del sistema ,se considera que el usuario directo se ha conectado hacia la red de sensores ,luego ha solicitado los datos de los sensores ,dichos datos han sido recibidos y se han subido correctamente en la base de datos en tiempo real .Es la última fase del proyecto ,se usara un diagrama de flujo para explicar cómo se realizara las acciones a realizar en esta fase, el diagrama se puede observar en la Figura *56*.

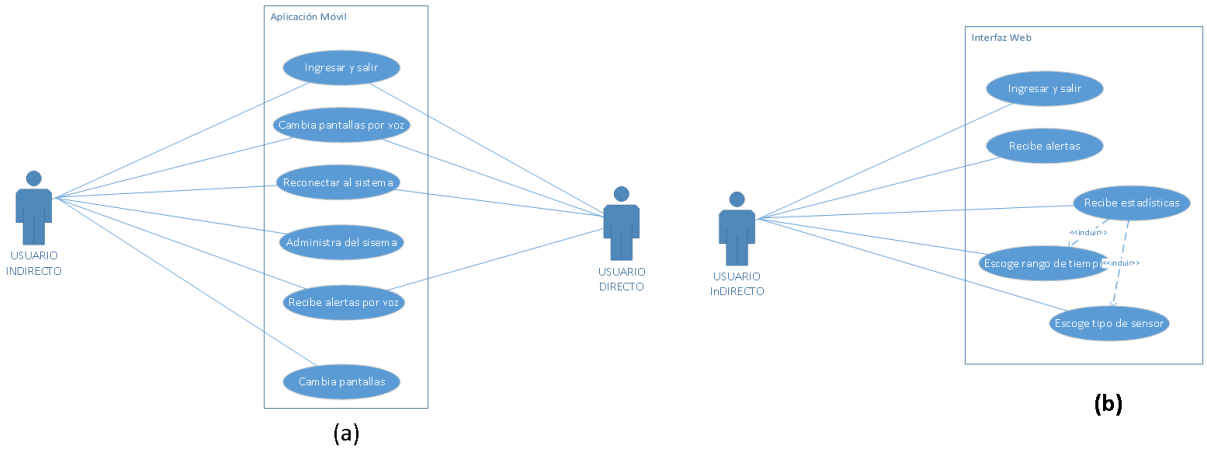
Figura 56. Diagrama de flujo para la fase de presentación de información al usuario.



Fuente: Autoría.

Notar que aún no se ha hablado de la accesibilidad al usuario en la aplicación móvil a desarrollar ,la aplicación podrá ser usado por los usuarios directos e indirectos ,por lo que contará con la reproducción de audios cuando existan alertas indicando en donde se dio a la alerta y el sensor que la genero ,estas alertas estarán disponibles para los usuarios directos) .Para los usuarios directos se diseñara una pantalla que cambiara de color cuando exista una alerta indicando la misma información que para los usuarios directos ,ademas se construirá una página web que estará disponible fuera del hogar para los usuarios indirectos que presentara las mismas características de la aplicación móvil .Para explicar de forma lógica se ha construido un diagrama de casos de uso de la Figura *57* ,el cual explica los métodos de presentación de información para los usuarios del sistema.

Figura 57.Diagrama de casos de uso para las formas de presentación del sistema para los usuarios.



Nota : (a) caso de uso de los usuarios directo e indirecto para la aplicación móvil,(b) casos de uso del usuario indirecto para la interfaz web.

Fuente: Autoría.

3.3.3. Descripción General de los módulos del sistema

El presente sistema domótico estará formado por 5 módulos: control de posición de objetos, control de luces, control de aire, control de inundaciones y control de incendios. Cada uno de los módulos tendrá asociado una serie de sensores asociados en un centro de operaciones, cada hub será un nodo de la red Mesh que utilizará Bluetooth Low energy para la intercomunicación basándose en el estándar Mesh. Cada uno de los nodos tendrá una batería

con un tiempo útil determinado por la batería o dispositivo escogido. Las alertas para cada uno de los módulos se presentarán al cliente mediante audios, cada sensor ubicado en cada nodo podrá generar una alerta cuando el valor de dicho sensor supere un determinado umbral.

Para la presentación de información al usuario, se desarrollará una aplicación móvil que podrá ser instalada en un smartphone común preferiblemente con el sistema operativo Android, la interacción será a través de comandos de voz al usuario final.

El funcionamiento de cada uno de los módulos se explica a continuación

3.3.3.1. Control de posición y objetos

Este tendrá dos submódulos: el submódulo de control de posición de objetos y el submódulo de control de estado de objetos. El submódulo de control de posición de objetos se basará en la actualización del estado de una habitación mediante la detección de cuando un tercero entra a una habitación. Mientras el submódulo de control de estado de objetos se encargará de información al usuario de situaciones tales como: planchas conectadas, neveras abiertas y similares dentro de toda la casa. El primer submódulo se implementará con la ayuda de botones pulsadores colocados en zonas cercanas a la entrada de las habitaciones que serán presionadas por un tercero siempre que se ingrese a dicho espacio mientras que el segundo submódulo se implementara usando sensores de circulación de corriente o voltaje.

3.3.3.2. Control de luces

Este módulo se encargará de realizar el proceso de encendido y apagado de luces en las habitaciones del hogar, tomando en cuenta que para los usuarios del sistema es complicado detectar este fenómeno. Dado que la mayoría de los hogares no permiten una integración del módulo con el sistema eléctrico del hogar a gran escala, entonces se utilizará bombillas que soportan el estándar. Siendo estas una opción mucho más barata.

3.3.3.3. Control de aire

El módulo de control de aire será encargado de controlar fugas de gas, ya sea que estas se traten de cilindros de gas, calefones o incineración de materiales. Este módulo será instalado en habitaciones criticas como: cocina, baño y sala. Podrá ser implementado utilizando sensores de medición de GLP o calidad del aire, la elección se realizará de acuerdo a la ubicación del sensor.

3.3.3.4. Detección de inundaciones

Este módulo se encargará de detectar el nivel de agua de los cuartos de baño y cocina, dado que para los usuarios del sistema es complicado detectar también este fenómeno. La ubicación de este módulo dependerá de la estructura física del domicilio del usuario y será implementado con sensores que midan la presencia de agua en una superficie.

3.3.3.5. Detección de incendios

Este módulo se encargará de detectar humo y alertar al usuario, funcionará en conjunto con el módulo de control de aire. La ubicación de este módulo será esencial para detectar de forma temprana un incendio, por lo que de forma obligatoria se deberá colocar al menos un módulo de este tipo en la cocina. Actualmente existen sensores que facilitarían la implementación de este módulo.

3.3.4. Selección de hardware

La selección de equipos se realizará en base a la lista de requerimientos listados anteriormente, se debe escoger: la placa de desarrollo, los sensores de aire, los sensores de agua y el tipo de sensor de incendios. Para la elección de cada uno de los equipos se realizará una comparativa para conocer si el dispositivo listado cumple o no con los requisitos. También se podrá utilizar alguno de los dispositivos con mayor calificación en la comparativa para ser usado a posterior en mejoras del sistema.

3.3.4.1. Elección de placa de desarrollo

Hay que tomar en cuenta que los fabricantes aun no desarrollan una placa de desarrollo optima que soporte el estándar Mesh de Bluetooth. Las opciones en el mercado a la fecha de realización del sistema son aún basadas en placas de desarrollo enfocadas para “developers”. Dada la naturaleza del proyecto, no se escogerá solo una placa de desarrollo. Las placas escogidas para esta tarea serán las placas con mejor valoración.

Se listan las placas de desarrollo en la

Tabla *7* de acuerdo a los requisitos a analizar.

Tabla 7. Listado de opciones para placas de desarrollo.

Tipo	Requerimientos														Valoración
	StRS2	**StRS5**	**StRS9**	**SyRS12**	**SyRS13**	**SyRS14**	**SyRS19**	**SRS1**	**SRS10**	**SRS18**	**SRS19**	**SRS21**	**SRS24**	**SRS25**	
LAUNCHXL-CC26X2R1	**X**	--	**X**	**X**	--	**X**	--	**X**	**X**	**X**	**X**	--	**X**	**X**	10
ESP 32-DEVKITC v4	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	14
Nordic Thingy 52	--	--	**X**	**X**	**X**	**X**	**X**	**X**	--	**X**	**X**	**X**	**X**	**X**	11
nRF52 DK	**X**	--	**X**	**X**	--	**X**	--	**X**	**X**	**X**	**X**	**X**	**X**	**X**	11
UG509: BG2 EK4108A	**X**	--	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	--	--	**X**	**X**	11
ESP 32-S3-DEVKITC-1	--	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	13
Si cumple = X										No cumple = --					
Elección															
ESP 32-DEVKITC v4				Nordic Thingy 52						ESP 32-S3-DEVKITC-1					

Fuente: Autoría

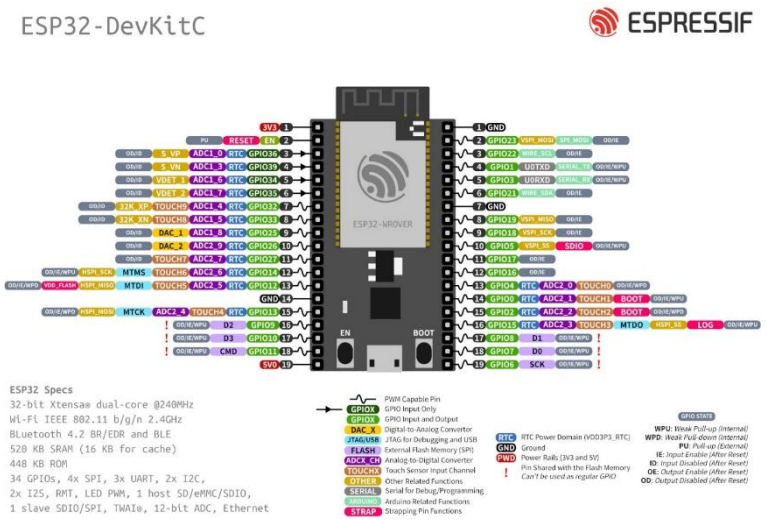
Se ha escogido las placas de desarrollo : ESP 32-DEVKITC v4, Nordic Thingy 52. También se podría utilizar la placa nRF52 DK ,pero no existe disponibilidad a corto plazo en el país. A continuación, se listan las características de cada una de las placas de desarrollo escogidas para el desarrollo del sistema.

- Espressif ESP 32 DEVKITC v4

Esta placa es desarrollada por Espressif. Se trata de una placa de desarrollo de bajo costo con soporte para un sin número de aplicaciones en especial de IoT. La version escogida presenta compatibilidad con wifi y bluetooth en las versiones más recientes. La placa cuenta con una extensa documentación y soporte a través de los canales oficiales; según los requerimientos presentados esta placa no cuenta con una antena integrada, pero algunas de las nuevas versiones cuentan con este requerimiento. Se puede configurar la placa a través de una conexión a una computadora utilizando un cable USB. Las dos placas de esta marca listadas tienen las propiedades para ser usadas como nodos de la red y al ser económicamente accesibles se utilizará al menos una de cada una para el desarrollo de este proyecto.

En la Figura 58 se puede observar la estructura física de la placa típica de desarrollo en cuestión.

Figura 58. Distribución de pines de placa de desarrollo ESP 32 típica



Fuente :(Espressif Systems, 2021a), ESP32-DevKitC V4 Getting Started Guide - ESP32 - — ESP-IDF Programming Guide latest documentation. Recuperado el 1 de noviembre de 2021, de https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html

En la Tabla 8 se puede observar las características de la placa de desarrollo, se toma en cuenta los parámetros listados en los requerimientos ya analizados previamente.

Tabla 8. Características de las palcas ESP32 que se van a utilizar como módulos.

Nombre	ESP32-DEVKITC v4	ESP32-S3-DEVKITC-1
Precio	$10	$15
Conectividad	IEEE 802.11 b/g/n, Bluetooth v4.2 BR/EDR y BLE	IEEE 802.11 b/g/n, Bluetooth LE: Bluetooth 5 y Bluetooth mesh
Núcleo	ESP32 D0WD	ESP32-S3
Oscilador	40 MHz	40 MHz
Chip	ESP32-WROOM-32D	ESP32-S3-WROOM-1
Especificaciones de procesador	Dual core Tensilica LX6 de 32 bits con 240 MHz de frecuencia de reloj,	Dual core Xtensa LX7 de 32 bits con 240 MHz de frecuencia de reloj
Especificaciones de memoria interna	448 KB en memoria ROM 520 KB en memoria SRAM 4 MB en memoria Flash	384 KB en memoria ROM 512 KB en memoria SRAM 32 MB en memoria Flash
Características eléctricas	3.0 V - 3.6 V, 500 mA mínimo.	3.0 V - 3.6 V, 500 mA como mínimo.

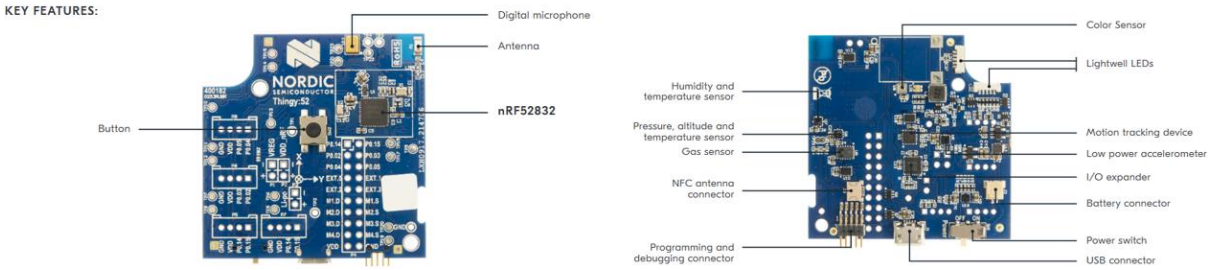
Fuente : Basado en (Espressif Systems, 2021c), ESP32-WROOM-32D & ESP32-WROOM-32U Datasheet. Recuperado el 1 de noviembre de 2021, de https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf y (Espressif Systems, 2021b),ESP32-S3-WROOM-1 ESP32-S3-WROOM-1U Datasheet. Recuperado el 1 de noviembre de 2021, de https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf

- Nordic Thingy 52

Se trata de una placa de desarrollo que contiene un kit de sensores para uso de aplicaciones IOT, es desarrollada por Nordic Semiconductor. Cuenta con el SoC nRF 52832, el cual tiene características para funcionar bajo el estándar de comunicación inalámbrico Bluetooth LE y NFC. Es posible realizar las configuraciones de aplicaciones a través de una aplicación móvil o a través de una conexión utilizando un dispositivo J-Link Debug.

En la se puede observar la estructura física del módulo Nordic Thingy 52, nótese la entrada diferenciada para alimentación eléctrica de la entrada para configurar el dispositivo.

Figura 59. Estructura física del Nordic Thingy :52.



Fuente :(Nordic Semiconductor, 2018d), Nordic Thingy:52 User Guide. Recuperado de https://infocenter.nordicsemi.com/pdf/Thingy_UG_v1.1.pdf

En la Tabla 9 se puede observar las características de hardware y software del kit Nordic Thingy 52, el parámetro más notable es la conectividad del dispositivo.

Tabla 9. Características de las palcas ESP32 que se van a utilizar como módulos.

Nombre	**Precio**	**Conectividad**	**Tamaño**	**Procesador**	**Especificaciones de memoria**	**Alimentación eléctrica**
Nordic Thingy :52	$39	Bluetooth Low Energy, Bluetooth mesh	6 x 6 cm	ARM Cortex M4(SoC nRF 52832)	512/256 KB en memoria flash, 64/32 KB en memoria ROM	1.7 V a 3.6 V través de batería interna.

Fuente :(Nordic Semiconductor, 2018c), Nordic Thingy:52 Product Brief Version 2.0. Recuperado el 1 de noviembre de 2021, de https://www.nordicsemi.com/-/media/Software-and-other-downloads/Product-Briefs/Nordic-Thingy52-product-brief.pdf?la=en&hash=4976338D3BEF6549B2C9470834A2EF0094F785D4

3.3.4.2. Elección de sensores de aire

El sensor de calidad de aire estará enfocado en detectar fugas de GLP (Gas licuado de petróleo) que generalmente suele encontrarse en la cocina de los futuros usuarios. De acuerdo a los requerimientos presentados se tiene la que muestra la comparativa realizada.

Tabla 10. Listado de opciones para sensores de aire

Tipo	**Requerimientos**						Valoración
	StRS5	**StRS6**	**SyRS2**	**SRS12**	**SRS24**	**SRS25**	
Sensor MQ2	**X**	**X**	**X**	**--**	**X**	**X**	5
Sensor MQ5	**X**	**X**	**X**	**--**	**X**	**X**	5
Sensor MQ6	**X**	**X**	**X**	**X**	**X**	**X**	6
Sensor MQ135	**X**	**X**	**X**	**--**	**X**	**X**	5
Sensor BME680	**X**	**X**	**X**	**--**	**X**	**X**	5
Sensor CCS811	**X**	**X**	**X**	**--**	**X**	**X**	5
Hub SMOD711KITV1	**--**	**--**	**---**	**--**	**X**	**X**	2
Si cumple = X				No cumple = --			
Elección							
Sensor MQ6							

Fuente: Autoría

El sensor escogido es el sensor MQ6, se trata de un sensor de bajo costo con la capacidad de medir concentración de gas licuado de petróleo específicamente. Tiene un rango de detección de 200 y 1000 ppm y se puede usar en todo tipo de entornos. En la se puede observar la estructura física del sensor.

Figura 60.Estructura física del sensor de aire MQ6.

Fuente :Basado en (TechnoElectronics44, 2021), MQ6-GAS SENSOR. Recuperado el 1 de noviembre de 2021, de https://www.technoelectronics44.com/2021/01/MQ6-Gas Sensor.html

A continuación, se lista las características del sensor escogido para controlar las fugas de GLP.

Tabla 11. Características del sensor de aire escogido

Nombre	**Precio**	**Tamaño**	**Parámetro medido**	**Lugar de uso y rango de detección**	**Alimentación eléctrica**
MQ6-6 sparkfun sen 09405	$5	3.2 x 2.2 x 2.7 cm	GLP (propano - butano)	Entorno domésticos e industriales (lugares de baja altura), 300～10000 ppm (Propano)	4.9 V ~ 5.1 V en DC 950 mW

Fuente : Basado en (Sparkfun, 2014), HANWEI SENSORS MQ-6. Recuperado el 1 de noviembre de 2021, de https://cdn.sparkfun.com/datasheets/Sensors/Biometric/MQ-6 Ver1.3 - Manual.pdf?__hstc=175869181.c80e8a76ede39d23c9f7282049f19df0.1635809304281.1635809304281.1635809304281.1&__hssc=175869181.2.1635809304281&__hsfp=2445552440

3.3.4.3. Elección de sensores de agua

Para la elección de los sensores de nivel de agua se realiza el análisis de los respectivos requerimientos y tomando en cuenta los lugares donde se van a ubicar los sensores. Se listan las placas de desarrollo en la Tabla 12de acuerdo a los requisitos a analizar.

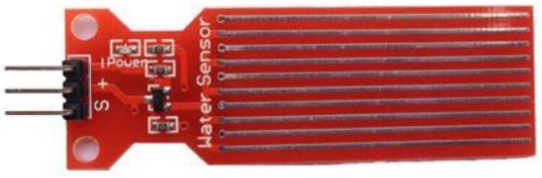
Tabla 12. Listado de opciones para sensores de nivel de agua.

Tipo	**Requerimientos**						Valoración
	StRS5	**StRS6**	**SyRS2**	**SRS13**	**SRS24**	**SRS25**	
Sensor GAOHOU A930	**X**	**X**	**X**	**X**	**--**	**X**	5
Sensor ZP5210	**X**	**--**	**--**	**X**	**--**	**X**	3
sensor YF-S201	**X**	**--**	**--**	**X**	**--**	**X**	3
Sensor Jsn-sr04	**--**	**--**	**X**	**X**	**X**	**X**	4
Sensor Nivel Agua Infrarrojo	**--**	**--**	**X**	**X**	**X**	**X**	4
Si cumple = X					No cumple = --		
Elección							
Sensor GAOHOU A930							

Fuente: Autoría

El sensor GAOHOU A930 es el escogido debido a sus características, se toma en cuenta también que se desea una respuesta rápida a las fugas de agua de recipientes, duchas, lavabos por lo que no será necesario medir distancia (sensor infrarrojo y ultrasónico) y se toma en cuenta el precio del sensor, así como los requerimientos ya listados. Se puede observar en la estructura física del sensor escogido.

Figura 61.Estructura física del sensor de agua



Fuente :(desensores.com, 2018), High Sensitivity Water Sensor -Red Version. Recuperado el 1 de noviembre de 2021, de https://desensores.com/datasheets/39-Datasheet-Sensor-de-Agua.pdf

A continuación, se lista las características del sensor escogido para medir los excesivos niveles de agua en el sistema.

Tabla 13. Características del sensor de agua escogido

Nombre	Precio	Tamaño	Parámetro medido	Lugar de uso y rango de detección	Alimentación eléctrica
GAOHOU A930	$5	5 x 2 cm	Nivel de agua	Entorno doméstico 4 x 1.6 cm (placa de sensor)	3 V ~ 5 V en DC Menor a 20 mA

Fuente : Basado en:(desensores.com, 2018) High Sensitivity Water Sensor -Red Version. Recuperado el 1 de noviembre de 2021, de https://desensores.com/datasheets/39-Datasheet-Sensor-de-Agua.pdf

3.3.4.4. Elección del sensor de incendios

Para la elección de los sensores de nivel de agua se realiza el análisis de los respectivos requerimientos tomando en cuenta los lugares donde se van a ubicar los sensores. Se listan las alternativas de sensores propuestos en la Tabla 14 .

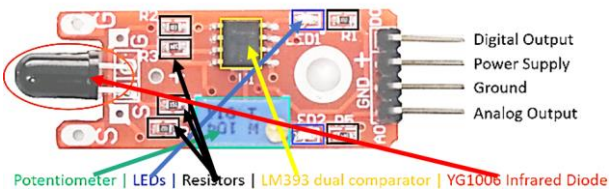
Tabla 14. Listado de opciones para sensores de detección de incendios.

Tipo	Requerimientos						Valoración
	StRS5	StRS6	SyRS2	SRS14	SRS24	SRS25	
Sensor DHT11	X	X	X	--	X	--	4
Sensor LM35	X	X	X	--	X	--	4
Sensor DS18B20	X	--	X	--	X	X	4
Sensor Gy-ml8511	X	X	X	X	X	--	5
Sensor KY-26	X	X	X	X	X	X	6
Si cumple = X					No cumple = --		
Elección							
Sensor KY-26							

Fuente: Autoría

El sensor KY-26 es el escogido debido a sus características, se toma en cuenta también que se desea una respuesta rápida a la detección de llamas, así como los requerimientos ya listados. Se puede observar en la estructura física del sensor escogido.

Figura 62.Estructura física del sensor de detección de incendios



Fuente :basado en (David, 2020), KY-026 Flame Sensor Tutorial for Arduino, ESP8266 and ESP32. Recuperado el 1 de noviembre de 2021, de https://diyi0t.com/flame-sensor-arduino-esp8266-esp32/

A continuación, se lista las características del sensor escogido para detectar si existe un incendio en el hogar del usuario.

Tabla 15. Características del sensor de detección de incendios escogido

Nombre	Precio	Tamaño	Parámetro medido	Angulo de detección	Alimentación eléctrica
KY-26	$3	9 × 5 × 10 cm	Luz infrarroja 760 nm a 1100 nm	60 grados	3.3 a 5.5 V 15 mA

Fuente : Basado en (JOY-IT, 2017), KY-026 Flame-sensor module. Recuperado el 1 de noviembre de 2021, de https://moviltronics.com/wp-content/uploads/2019/10/KY-026.pdf

3.3.4.5. Elección del sensor de corriente eléctrica

Para la elección del sensor de medición de corriente eléctrica se analiza requerimientos de acuerdo a los criterios listados en incisos anteriores. Se listan las placas de desarrollo en la Tabla 12 de acuerdo a los requisitos a analizar.

Tabla 16. Listado de opciones de sensores de medición de corriente eléctrica.

Tipo	Requerimientos							Valoración
	StRS5	StRS6	SyRS2	SRS15	SRS16	SRS24	SRS25	
SCT 013	**X**	**X**	**X**	**X**	**X**	**X**	**X**	7
ACS 712	**X**	--	**X**	--	--	**X**	--	3
PZEM 400T	**X**	--	**X**	--	--	**X**	--	3
ZMPT101B	**X**	--	**X**	--	---	**X**	--	3
WCS1800	**X**	**X**	**X**	**X**	--	**X**	--	5
CSNB Honeywell	--	--	**X**	--	--	**X**	**X**	3
Si cumple = X						No cumple = --		
Elección								
SCT 013								

Fuente: Autoría

El sensor SCT 013 en la version 100 A/50 mA es el escogido debido a sus características. Existen varias variaciones de este sensor, pero se escogerá la version mencionada. Especialmente dado que esta version del sensor puede configurarse con un circuito externo de acuerdo a lo que quiere medirse, se puede configurar dicho circuito con un

margen mínimo o máximo de medidas Se puede observar en la estructura física del sensor escogido en la siguiente figura.

Figura 63.Estructura física del sensor de detección de incendios

Fuente: Autoría

A continuación, se lista las características del sensor escogido para detectar si existe un incendio en el hogar del usuario.

Tabla 17. Características del sensor de detección de corriente escogido

Nombre	Precio	Tamaño	Parámetro medido	Resultado de medición	Alimentación eléctrica
SCT 013 100 A/50 mA	$12	9 × 5 × 10 cm	Corriente eléctrica	Señal alterna de corriente	3.3 a 5.5 V 15 mA

Fuente : Basado en (YHDC, 2018), SCT-013. https://www.mcielectronics.cl/website_MCI/static/documents/Datasheet_SCT013.pdf

3.3.4.6. Elección del sensor de presencia

Como se indicó en incisos anteriores, para medir la presencia en determinados sectores de la casa del cliente, se va a utilizar los pulsadores integrados en los Thingy 52 a utilizar. Pero al configurar el nodo de esta forma, no se asegura una correcta detección de personas; por lo que se necesita un sensor adicional que sirve específicamente para el rol. Se analiza los requerimientos de acuerdo a la Tabla *18* de acuerdo a los requisitos a analizar.

Tabla 18. Listado de opciones de sensores de detección de movimiento

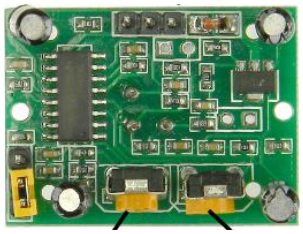
Tipo	Requerimientos	Valoración

	StRS5	StRS6	SyRS2	SRS17	SRS24	SRS25	
HC-SR501	**X**	**X**	**X**	**x**	**X**	**X**	6
HC-SR04	**X**	**X**	**X**	**--**	**X**	**X**	5
GP2Y0A51SK0F 2	**X**	**X**	**X**	**--**	**X**	**X**	5
KY-032	**X**	**X**	**X**	**--**	**X**	**X**	5
KY-002	**X**	**X**	**X**	**--**	**X**	**X**	5
leviton O2C04-IDW	**--**	**X**	**--**	**X**	**X**	**--**	3
Si cumple = X					No cumple = --		
Elección							
Sensor HC -SR501							

Fuente: Autoría

El sensor escogido es el sensor HC-SR501, el mayor peso de la elección del sensor es la distancia de detección. Este sensor será instalado en el módulo correspondiente. Se puede observar en la estructura física del sensor escogido en la siguiente figura.

Figura 64.Estructura física del sensor de detección de presencia



Fuente : (Marlin P. Jones & Assoc. Inc., 2011), HC-SR501 Pir Motion Detector. En Marlin P. Jones & Assoc. Inc. (pp. 3–5). https://www.mpja.com/download/31227sc.pdf

A continuación, se lista las características del sensor escogido para detectar si existe un incendio en el hogar del usuario.

Tabla 19. Características del sensor de detección de corriente escogido

Nombre	**Precio**	**Tamaño**	**Parámetro medido**	**Rango**	**Alimentación eléctrica**
HC-SR04	$3.50	5 cm x 5 cm x 2 cm	Obstáculo infrarrojo	3-7 m	3.3 v o 5 v

Fuente : Basado en (Marlin P. Jones & Assoc. Inc., 2011), Hc-Sr501 Pir Motion Detector. En Marlin P. Jones & Assoc. Inc. (pp. 3–5). https://www.mpja.com/download/31227sc.pdf

3.3.4.7. Elección de batería

El uso de baterías puede ser opcional de acuerdo a los criterios del cliente, así como al precio de estas. Dado que es posible que el cliente posea tomas de corriente libres y cargadores USB que pueden ser utilizados para alimentar el sistema.

Para la elección de la batería se toma en cuenta el consumo energético de cada una de las placas que se va a utiliza, así como los valores recomendados por el fabricante para dichas placas. Considerando ya estos parámetros se incluyó los requerimientos en incisos anteriores, en la Tabla 20 se puede observar los requerimientos considerados.

Tabla 20. Listado de opciones para baterías usadas por el sistema

Tipo	**Requerimientos**					**Valoración**
	StRS5	**StRS6**	SyRS19	**SRS24**	**SRS25**	
Batería de Ni-MH	**X**	**X**	**X**	**X**	**X**	2
Banco de energía portátil	**--**	**X**	**X**	**X**	**X**	2
Batería de litio	**X**	**X**	**X**	**X**	**X**	3
Batería de polímero de litio	**X**	**X**	**X**	**X**	**X**	4
Si cumple = X				No cumple = --		
Elección						
A criterio del administrador						

Fuente: Autoría

Como se puede observar ,la mayoría de las opciones presentadas es útil por lo que se deberá escoger la batería en base al stock local .Por lo que se escoge las baterías de polímero de lito para el uso dentro del sistema, se toma en cuenta ademas estas tienen una alta vida útil y pueden cargarse con facilidad.

Se lista las características eléctricas de cada una de las placas y la elección de baterías para cada una de ellas en la Tabla 21 .Hay que recalcar que también es viable usar las baterías de polímero de litio para la alimentación individual de los sensores a utilizar en el sistema.

Tabla 21. Baterías para utilizar por cada una de las placas de desarrollo.

Placa de desarrollo	Características eléctricas	Batería para utilizar
ESP32-DEVKITC	2.5 a 3.3 v 80-180 mA (en máxima operación)	Batería de litio
Nordic Thingy :52	1.7 V a 3.6 V 100 mA (en máxima operación)	Batería interna de polímero de litio

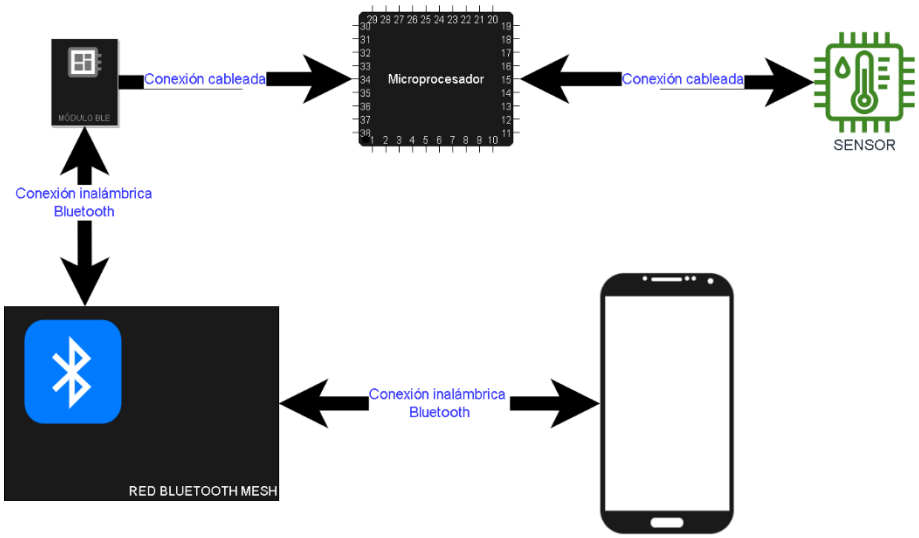
Fuente: Autoría

El uso de baterías en el sistema será opcional ,pero la construcción de un sistema para realizar el proceso de carga se ha presentado en este inciso.

3.3.4.8. Elección de equipo complementario

Los elementos faltantes para la construcción del sistema son :smartphone , conectores y bombillas .Para el caso del smartphone es necesario solo conectividad por medio de Bluetooth y una version del sistema operativo Android 8.x .Por otro lado los conectores serán escogidos en base al diagrama de pines de cada uno de los sensores a utilizar, se considera que los sensores utilizados están organizados en módulos por lo que no es necesario etapas de filtrado de datos .En la Figura 65 se puede observar la función del smartphone y conectores dentro del sistema.

Figura 65. Diagrama de ubicación de conectores y smartphone en el sistema.



Fuente: Autoría

Elección de bombillas

Para el caso de las lámparas ,actualmente en el mercado ya existen productos que ya cuentan con el soporte del estándar .Se puede consultar esta información en la página oficial de la SIG de Bluetooth donde se puede consultar que productos ya están confirmados con la certificación (Bluetooth S.I.G., 2019b).Se considera a los requerimientos presentados y se realiza la comparación en la Tabla 22.

Tabla 22. Listado de opciones para luminarias

Tipo	**Requerimientos**				**Valoración**
	StRS5	**StRS6**	**SRS3**	**SRS24**	
Philips Hue A19	**X**	**X**	**--**	**X**	2
Sengled A19 E26	**X**	**X**	**X**	**X**	3
Sengled E12	**X**	**X**	**X**	**X**	3
ENERGETIC Smart A19	**X**	**X**	**--**	**X**	2
Si cumple = X				No cumple = --	
Elección					
Sengled A19 E26 o Sengled E12					

Fuente: Autoría

La lampara cumple con los requerimientos del sistema y ademas puede ser instalada como una bombilla común y corriente, además que existe en el mercado local; se puede notar que la mayoría de las opciones presentadas no cumple con el requisito de costo dado que ademas del costo del producto hay que sumar el caso de importación al país. Se espera que a futuro los precios de estos elementos tengan un decremento considerable, para este proyecto solo se usara una bombilla inteligente. Otra de las opciones es buscar una variante de la bombilla escogida, para este caso se puede usar la bombilla común para luz diurna. La bombilla en cuestión se puede observar en la, notar que se trata de la version normal y no de la versión RGB. La versión normal cumple con los requisitos de costo para este proyecto.

Figura 66. Bombillas inteligentes usadas en el proyecto.

Fuente : Basado en (Sengled, 2021), Sengled Smart Bluetooth LED Daylight A19 Bulb – Sengled USA. Recuperado el 9 de noviembre de 2021, de https://us.sengled.com/products/sengled-smart-bluetooth-led-daylight-a19-bulb

Mientras que las características técnicas de la lampara se pueden observar en la Tabla 23.

Tabla 23. Características de la lampara led escogida

Nombre	**Precio**	**Tamaño**	**Conectividad**	**Potencia**	**Características eléctricas**
Sengled A19 E26	$30	6 × 6 × 11 cm	Bluetooth, Bluetooth Mesh, Bluetooth Low Energy	800 lumen	120 V 60 W

Fuente : Basado en (Sengled, 2021), Sengled Smart Bluetooth LED Daylight A19 Bulb – Sengled USA. Recuperado el 9 de noviembre de 2021, de https://us.sengled.com/products/sengled-smart-bluetooth-led-daylight-a19-bulb

Elección de regulador de voltaje

Otro elemento que considerar es un regulador de voltaje ,específicamente para las placas ESP32. Estas placas pueden ser alimentadas externamente con voltajes de 5 voltios en corriente y 3.3 voltios respectivamente ,cuando se utiliza 5 voltios :el regulador interno de la placa realiza una reducción a 3.3 voltios ;por lo que existe un desperdicio de energía considerable ,tomando en cuenta que se está utilizando baterías para la alimentación externa .Si se utiliza el voltaje de 3.3 voltios ,la operación anterior no se da ,pero la placa solo admite un límite de ± 0.1 voltios .Un valor mayor puede dañar la placa ,por lo que se es necesario el uso de un regulador de voltaje que reduzca el voltaje de 3.7 voltios ,típico de una batería escogida a un valor de 3.3 voltios exactos . Se considera a los requerimientos presentados

Tabla 24. Listado de opciones para convertidores de voltaje

Tipo	Requerimientos				Valoración
	StRS5	SyRS19	SRS25	SRS6	
Step DownXL4015	**X**	**X**	**X**	**X**	4
Step Down LM2596	**X**	**--**	**X**	**X**	3
Boost Buck 3.3V S09	**X**	**--**	**--**	**X**	2
Pololu Adjustable Boost Regulator 2.5-9.5V	**--**	**X**	**X**	**X**	3
Si cumple = X				No cumple = --	
Elección					
Step DownXL4015					

Fuente: Autoría

El convertidor escogido es Step DownXL4015, las demás opciones pueden configurarse para obtener un rango de voltajes de salida de acuerdo a la placa de desarrollo, pero es necesario al menos 700 miliamperios al momento de boot de la placa, por lo que esta opción es la más fiable. Además, es posible configurar al dispositivo con más voltaje en caso de ser necesario. En la Figura *67* se puede observar la estructura física del convertidor en cuestión con las dos configuraciones mencionadas.

Figura 67. Convertidor de voltaje utilizado en el proyecto para las placas ESP32.

Fuente: (XLSEMI, 2017) Datasheet 5A 180KHz 36V Buck DC to DC Converter XL4015. www.xlsemi.com

Mientras que las características técnicas del conversor de voltaje se pueden observar en la Tabla 25.

Tabla 25. Características del conversor de voltaje escogido.

Nombre	Precio	Tamaño	Voltaje de Entrada	Salida
Convertidor de voltaje XL4015 5A	$3.00	54 mm x 23 mm x 15 mm	4 a 38 v DC	1.25 a 36 v DC 5 A

Fuente :Basado en (XLSEMI, 2017) Datasheet 5A 180KHz 36V Buck DC to DC Converter XL4015. www.xlsemi.com

Elección de cargador de batería

Dado que se utilizará un tipo de batería recargable, se deberá designar un cargador de esta de acuerdo a la batería escogida para el funcionamiento del sistema. El cargador deberá poder realizar la carga para los valores de voltaje adecuados, además del tamaño de este.

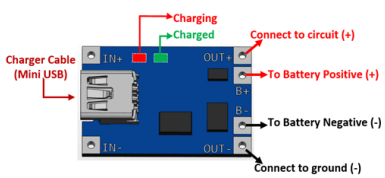
Tabla 26. Listado de opciones para cargador de batería

Tipo	Requerimientos						Valoración
	StRS5	SyRS14	SyRS18	SyRS19	SRS24	SRS25	
BMS 18650 3S 10A	**X**	**X**	**--**	**X**	**--**	**X**	4
Cargador USB Pilas GP Recyko	**--**	**X**	**--**	**--**	**X**	**X**	3
Hobbyking DC-4S	**--**	**X**	**X**	**--**	**--**	**--**	2
TP4056	**X**	**X**	**X**	**X**	X	X	5
Si cumple = X					No cumple = --		
Elección							
TP4056							

Fuente: Autoría

El cargador escogido es el TP4056, este dispositivo ademas tiene protecciones contra sobrevoltajes y permite una conexión USB a través de una interfaz micro B. Con un circuito adecuado se puede juntar el cargador de batería y el regulador de voltaje con el fin de lograr un circuito de carga y alimentación de las placas de desarrollo utilizadas. En la se puede observar la estructura física del dispositivo escogido.

Figura 68. Estructura física del cargador de batería escogido.



Fuente : (Components101, 2018)

Mientras que las características técnicas del conversor de voltaje se pueden observar en la Tabla *27*,así como el precio y otras características.

Tabla 27. Características del cargador de batería escogido

Nombre	Precio	Tamaño	Voltaje de Entrada	Voltaje /Corriente de carga
TP4056 A Li-ion Battery Charging/Discharging Module	$3.00	30 mm X 20 mm X 4 mm	5 V DC	4.2 v DC 1 A

Fuente :Basado (NanJing Top Power ASIC Corp., 2017). TP4056 1A Standalone Linear Li-lon Battery Charger with Thermal Regulation in SOP-8 DESCRIPTION.

3.3.5. Selección de software

3.3.5.1. Selección de software para programación de módulos

Para la selección de software de los módulos se toma en cuenta la placa de desarrollo usada, en este caso se utilizarán más de una placa de desarrollo por lo cual se debe emplear los entornos de programación recomendados por los fabricantes que permitan codificar las instrucciones de cada uno de los nodos. La muestra un listado de los softwares de cada una de las placas de desarrollo utilizadas.

Tabla 28. Listado de software a utilizar para la programación en las placas de desarrollo utilizadas.

Nombre del entorno	Placa de desarrollo	Uso de repositorios	Grabación de código
ESP-IDF	ESP32-DEVKITC V4 ESP32-S3-DEVKITC-1	Si, repositorios públicos	Si mediante conexión USB.
SEGGER Embedded Studio, nRF connect	Nordic Thingy 52	Si, repositorios públicos	Si mediante conexión JTAG

Fuente: Autoría

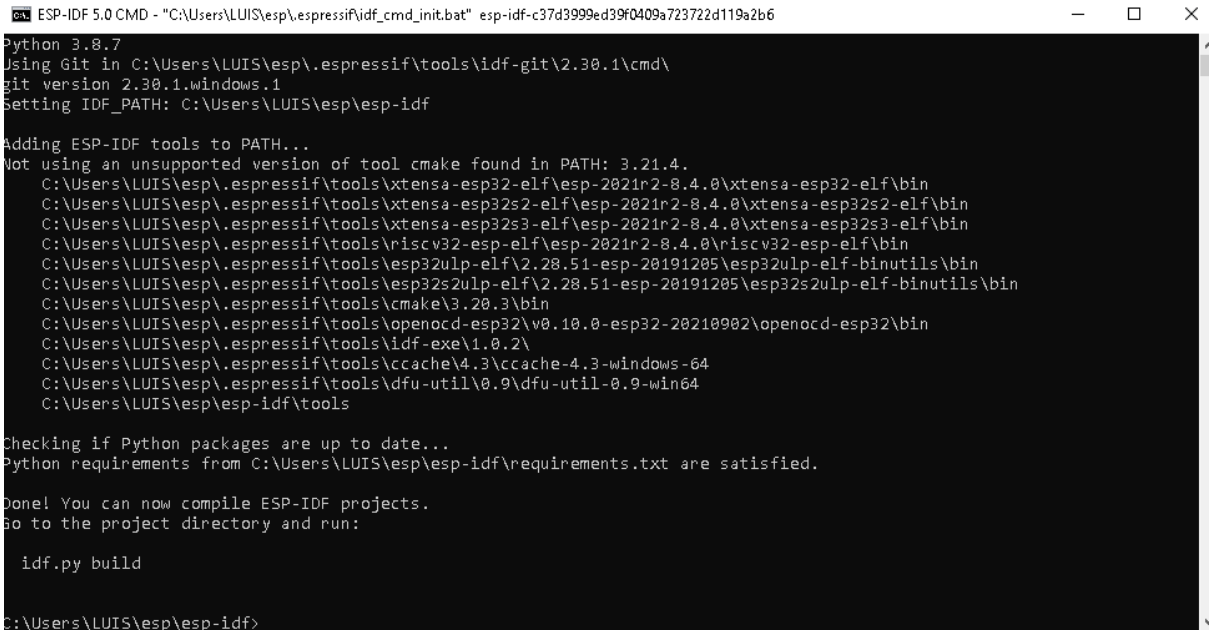
A continuación, se muestra las información adicional y configuraciones realizadas en cada uno de los entornos para iniciar con la programación de aplicaciones.

- ESP-IDF y configuraciones básicas

Se trata del software de Espressif para el desarrollo de aplicaciones usando las placas ESP32. Contiene las librerías y código fuente para funcionar, puede funcionar con Visual Estudio code y Eclipse IDE como entorno de programación. Las configuraciones para poder

programar la placa de desarrollo ESP32 se pueden observar en el Anexo 7 En la se puede observar la pantalla de inicio del programa en cuestión.

Figura 69. Pantalla de inicio del software ESP-IDF

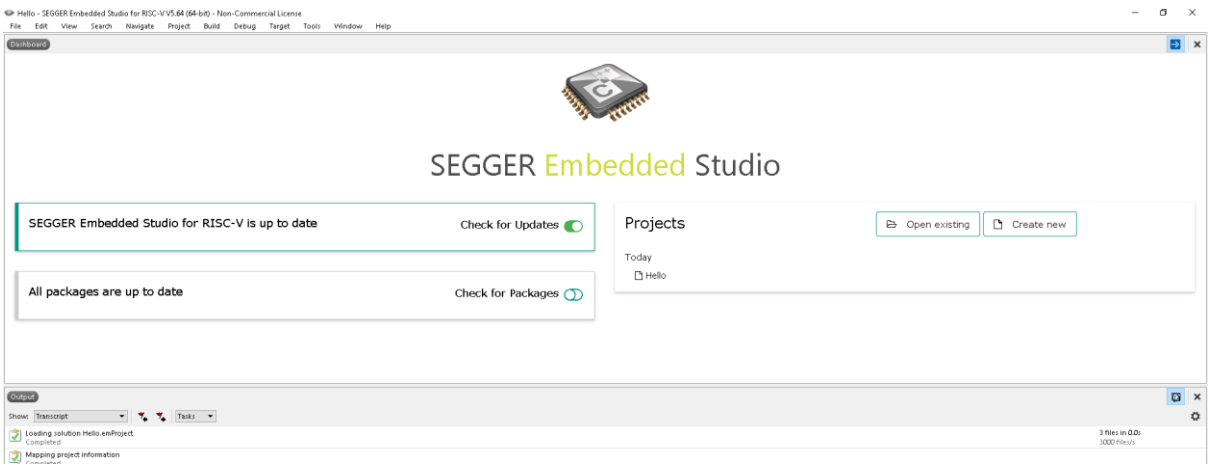


Fuente: Autoría

- SEGGER Embedded Studio, nRF Connect y configuraciones básicas

Es el software desarrollado por SEGGER para configuración, programación y depuración de dispositivos que funcionen con ARM Cortex. Este software es recomendado por Nordic para la configuración de los equipos de la marca. Es posibles descargar la última version desde la página oficial de forma gratuita, para obtener una licencia se debe seguir el proceso especificado en **: Configuraciones básicas de la placa de desarrollo Nordic Thingy :52 en SEGGER Embedded Studio** .Hay que recalcar que se usan en conjunto con algunos softwares adicionales de Nordic por lo que antes de configurar la placa de desarrollo hay que seguir el proceso de forma detallada. En la se puede observar la pantalla de inicio del programa SEGGER.

Figura 70. Pantalla de inicio del software SEGGER Embedded Studio.



Fuente: Autoría

3.3.5.2. Selección de software para el desarrollo de la aplicación móvil

Para la selección del software utilizado en el desarrollo de la aplicación móvil, se toma en cuenta los parámetros listados en la Tabla *29*

Tabla 29. Listado de software a utilizar para el desarrollo de la aplicación móvil.

Nombre	Requerimientos							Valoración
	StRS12	**SySR3**	**SySR10**	**SRS2**	**SRS20**	**SRS22**	**SRS23**	
React Native	**X**	**X**	**X**	**--**	**X**	**--**	**--**	4
Ionic	**X**	**X**	**X**	**--**	**X**	**--**	**--**	4
Android Studio	**X**	**X**	**X**	**X**	**X**	**X**	**X**	7
Flutter	**X**	**X**	**X**	**X**	**X**	**--**	**X**	6
Microsoft Visual Studio Xamarin	**X**	**X**	**X**	**--**	**X**	**--**	**--**	4
Si cumple = X						No cumple = --		
Elección								
Android Studio								

Fuente: Autoría

El software para utilizar es Android Studio , se utilizará debido a las configuraciones de componentes nativos que se puede realizar. Además, se trata del IDE oficial de Google para el desarrollo de aplicaciones en Android y es posible la exportación de proyectos hacia IOS de una manera muy sencilla. . Las demás opciones utilizan complementos para el soporte de elementos nativos móviles por lo que suelen ser más difíciles de administrar, ya sea en la instalación o gestión de proyectos. Para este proyecto, se ha escogido la opción más fiable.

3.3.6. Diagrama de conexiones

Esta sección describe como se realizará la conexión física de todos los dispositivos del sistema, así como el diseño de circuito adicionales para dichos dispositivos. Se considera como parámetro inicial un diagrama esquemático general de la red, y ademas se debe tomar en cuenta que se utilizara los modelos definidos en el estándar Bluetooth Mesh, especialmente los modelos de luces y sensores, los cuales serán utilizados para el desarrollo de este proyecto. Todos los módulos pueden funcionar mediante alimentación eléctrica con baterías o mediante una conexión con un adaptador.

Antes de presentar los diagramas de conexión, se considera el voltaje de alimentación de cada uno de los sensores de acuerdo con su respectivo datasheet. Los valores se pueden observar en la Tabla 30.

Tabla 30. Características de operación de los sensores utilizados

Sensor	**Voltaje de Operación**	**Corriente de operación**	**Salida**
MQ6	5 v DC	150 mA	Voltaje
KY-026	3.3-5 v DC	20 mA	Voltaje
GAOHOU A930	3.3-5 v DC	15 mA	Voltaje
SCT-013	-----	5 mA	Corriente
HC SR501	3.3-5 v DC	65 mA	Voltaje

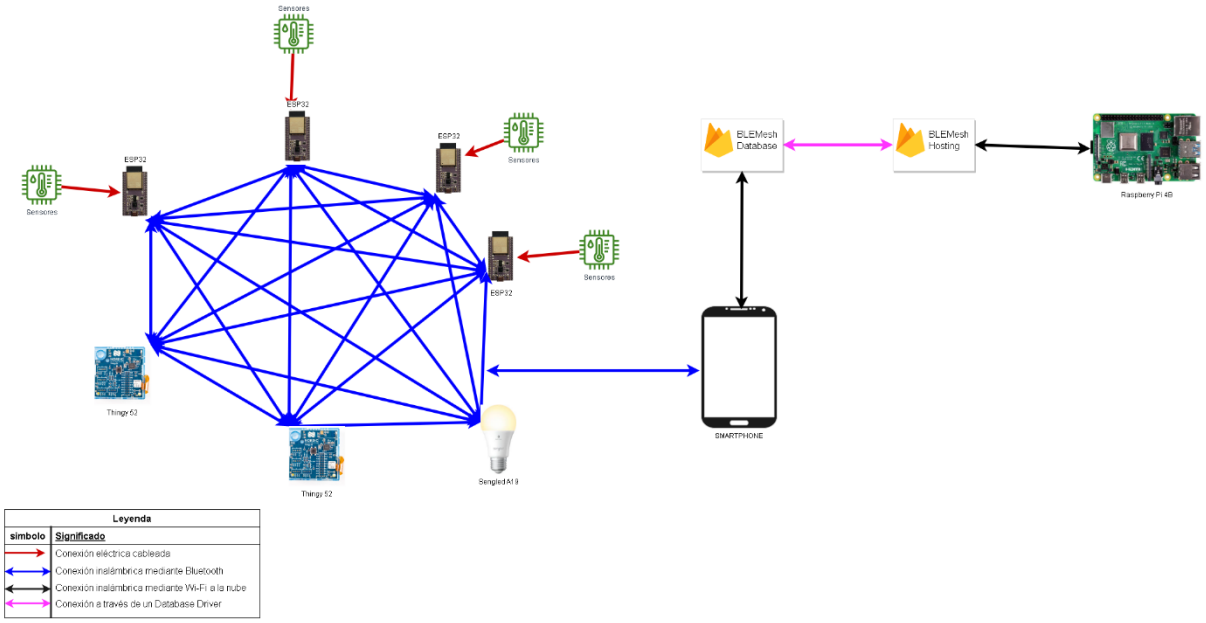
Fuente: Autoría.

Como se puede observar en la 32 los sensores de gas, aire, incendios y presencia arrojan como salida voltaje mientras el sensor de corriente arroja corriente como salida. También hay que notar que el voltaje de operación de algunos sensores no coincide con los valores de voltaje que la placa puede soportar. Por lo que se debe realizar diseños de circuitos acopladores para aquellos sensores mencionados. Como ultima consideración, no se utilizarán ningún tipo de filtro ni amplificador operacional dado la naturaleza de los parámetros a medir.

3.3.6.1. Esquema General de Conexiones

En la Figura 71 se observar el diagrama esquemático general de la red de sensores recordando que el módulo Thingy 52 tiene los sensores integrados dentro de la placa y puede ser usado sin la conexión con equipos adicionales. Además, se tratará a todos los datos de los sensores como entradas digitales con un umbral binario de funcionamiento, es decir se detecta o no el evento, no existirá un evento medio. Además, se puede observar las conexiones realizadas al exterior de la red, así como la participación del smartphone en la red y las conexiones al exterior.

Figura 71. Esquema general de las conexiones de la red de sensores a implementar.



Fuente: Autoría

3.3.6.2. Diseño del circuito acoplador para el sensor de Gas

La placa de desarrollo si posee un pin para alimentación en 5[voltios], el problema reside en que las entradas analógicas y digitales solo soportan un voltaje de 3.3[voltios]. Para resolver este problema se plantea un circuito divisor de voltaje, se utiliza la Ecuación (1):

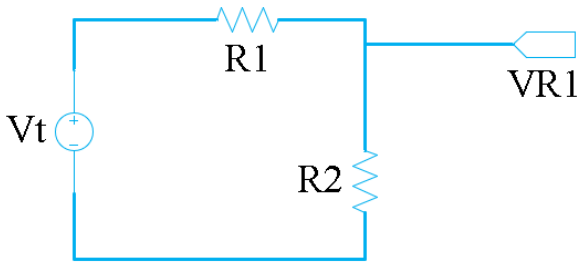
	$$VR1 = Vt * \frac{R2}{R1 + R2}$$	(1)

Donde:

- VR1 será el voltaje que circule por la resistencia R1, el cual se conectará directamente a la placa de desarrollo.
- Vt será el voltaje de alimentación del sensor, para este caso 5 [voltios].
- R2 será el valor de resistencia utilizado para satisfacer la formula presentada.

El circuito de divisor de voltaje se presenta en la Figura 72 .

Figura 72. Circuito divisor de voltaje para alimentación eléctrica del sensor MQ6.



Fuente: Autoría.

Con estas consideraciones se realizan los cálculos sustituyendo los valores de acuerdo a la siguiente formula.

$$VR1 = 5[V] * \frac{17k[\Omega]}{17k[\Omega] + 33k[\Omega]}$$

$$VR1 = 3.3[V]$$

Las resistencias escogidas son 17[K Ω] y 33[K Ω]. Para el funcionamiento del prototipo no importa si el valor de Vt es menor a 5[v] sino que el valor de VR1 sea menor o igual a 3.3[v] para no dañar el pin que recibirá los datos analógicos del sensor MQ6.

3.3.6.3. Diseño de circuito acoplador para el sensor de corriente eléctrica

Para el diseño del circuito acoplador de este sensor se considera que:

- Por datasheet el sensor tiene 2000 vueltas en el devanado secundario.
- El devanado primario es el cable que se va a medir (1 vuelta).

- El valor mínimo de consumo en watios será de 230[W], considerado para una refrigeradora. Teniendo este valor como umbral mínimo.
- El voltaje AC será de 120[v].
- El sensor arroja una onda de voltaje sinusoidal (valores negativos y positivos) que no son reconocidos por las placas de desarrollo utilizadas
- La onda de voltaje variara en función del voltaje de alimentación dado por la placa de desarrollo utilizada, en este caso 3.3 y -3.3 [v].
- El sensor utilizado funciona como un trasformador de corriente por lo que las fórmulas de un trasformador son aplicables.
- La corriente medida del sensor se da en el devanado secundario del mismo.
- Las fórmulas por utilizar se basan en la ley de Ohm.

Como primer paso se realiza los cálculos para calcular la resistencia de carga que va a utilizar el sensor, se comienzan los cálculos. Se calcula el valor de corriente eficaz en base a la potencia y el voltaje de consumo presentado, con la Ecuación (2)

$$I_{RMS} = \frac{Potencia}{Voltaje} \tag{2}$$

$$I_{RMS} = \frac{230\ [W]}{120[V]}$$

$$I_{RMS} = \frac{23}{12}\ [A]$$

Notar que se utiliza el valor en forma de fracción para lograr cálculos más exactos, ahora se procede a buscar la corriente de pico con la formula:

$$I_{pico} = I_{RMS} * \sqrt{2}$$

$$I_{pico} = \frac{23}{12} * \sqrt{2}$$

$$I_{pico} = \frac{23}{12} * \sqrt{2}$$

Este valor será utilizado en el cálculo de la corriente de pico en el devanado secundario del transformador, es decir este valor será el valor mínimo por medir de corriente. Para el caculo se utiliza la siguiente formula:

$$\frac{N_{Primario}}{N_{Secundario}} = \frac{I_{Secundario}}{I_{Primario}} \quad (3)$$

Donde

- N primario es el número de vueltas en el devanado primario
- N secundario es el número de vueltas en el devanado secundario
- I primario es la corriente en el devanado primario, para este caso es el valor de la corriente de pico.
- I secundario es la corriente en el devanado secundario, para este caso es el valor del umbral mínimo de medición.

Se despeja la Ecuación 3 y se obtiene la siguiente ecuación en la cual solo se procede a remplazar los datos.

$$I_{Secundario} = \frac{N_{primario} * I_{Primario}}{N_{secundario}}$$

$$I_{Secundario} = \frac{1[vueltas] * \left(\frac{23}{12} * \sqrt{2}[A]\right)}{2000[vueltas]}$$

$$I_{Secundario} = 1.35528[mA]$$

Como se mencionó en la elección del sensor, el modelo escogido necesita de una resistencia de carga para realizar las mediciones. Esta resistencia de carga será calculada en base al valor de la intensidad pico en el devanado secundario y el valor de voltaje de la onda sinusoidal sobre 2. Se realiza esta división dado que solo se necesitan los valores positivos de la onda. Se utiliza la siguiente fórmula:

$$R_{carga} = \frac{\frac{V_{referencia}}{2}}{I_{\ pico\ secundario}} \quad (4)$$

$$R_{carga} = \frac{\frac{3.3[v]}{2}}{1.35528[mA]}$$

$$R_{carga} = 1217.45\ [\Omega]$$

Con el valor de la resistencia de carga calculado se procede a construir un circuito que añadirá un valor de offset de acuerdo a la placa de desarrollo utilizada ,en este caso es de +1.65[v].Para diseñar este circuito se toma en cuenta que se usara la alimentación de la placa de 3.3 [v] y a esta fuente de alimentación, se le aplicara un divisor de voltaje para obtener el valor de 1.65[v].Este valor será utilizado para sumar dicho valor a la señal obtenida y eliminar la parte negativa de la misma .Ademas se utilizara un capacitor de 10[uF] para evitar el paso de cualquier señal anómala que pueda dañar el circuito. Por lo que se utiliza la formula:

$$VR1 = Vt * \frac{R2}{R1 + R2}$$

Donde se reemplaza los valores correspondientes tomando en cuenta las condiciones, por lo que la formula quedara de la siguiente forma:

$$\frac{Vt}{2} = Vt * \frac{R2}{R1 + R2}$$

Despejando esta fórmula para obtener los valores de R1 y R2 se obtendrá que:

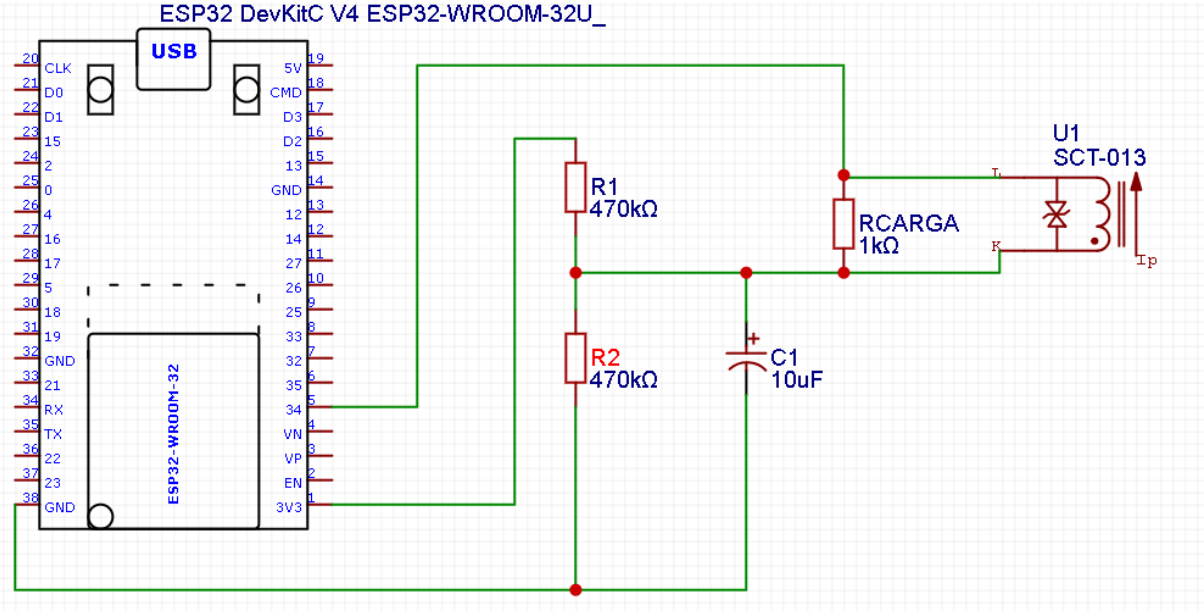
$$\frac{Vt}{2 * Vt} = \frac{R2}{R1 + R2}$$

$$R1 + R2 = 2R2$$

$$R1 = R2$$

Por lo que los dos valore de la resistencia en el puente serán iguales, queda al criterio del administrador el valor de la resistencia a utilizar, en consecuencia, se utilizara una resistencia de 470[k Ω] para reducir al mínimo el consumo de este nuevo circuito. Con todos los parámetros diseñados es posible armar el diseño esquemático del circuito. Este esquema se puede observar en la Figura 73. Notar que la entrada analógica de la placa es referencial.

Figura 73. Diagrama de conexiones para el sensor SCT -013-000 en la placa ESP32 utilizada

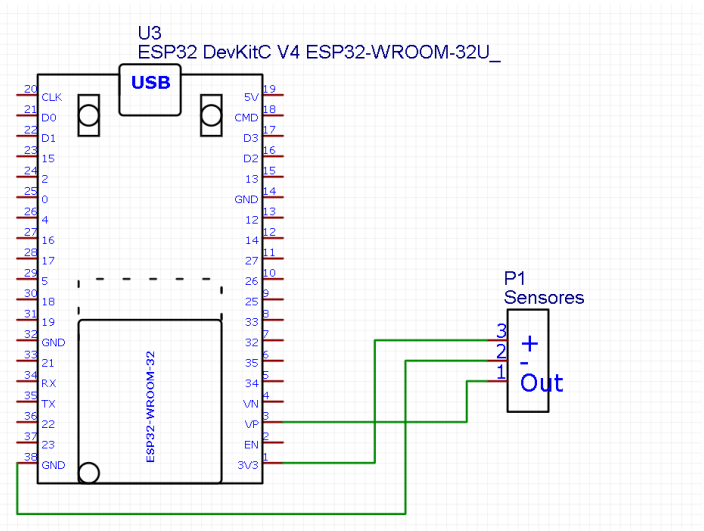


Fuente: Autoría.

3.3.6.4. Diseño de circuito acoplador para los sensores detección de agua e incendios

Para adecuar los sensores de detección de agua e incendios no se necesita un circuito adicional, los sensores mencionados pueden funcionar mediante alimentación eléctrica directa de la placa, por lo que se debe realizar las conexiones de acuerdo a la Figura 74.Ademas los dos sensores poseen una salida analógica por lo que pueden ser conectados a las entradas utilizadas en la placa respectiva.

Figura 74. Conexión de sensores a la placa de desarrollo ESP32.

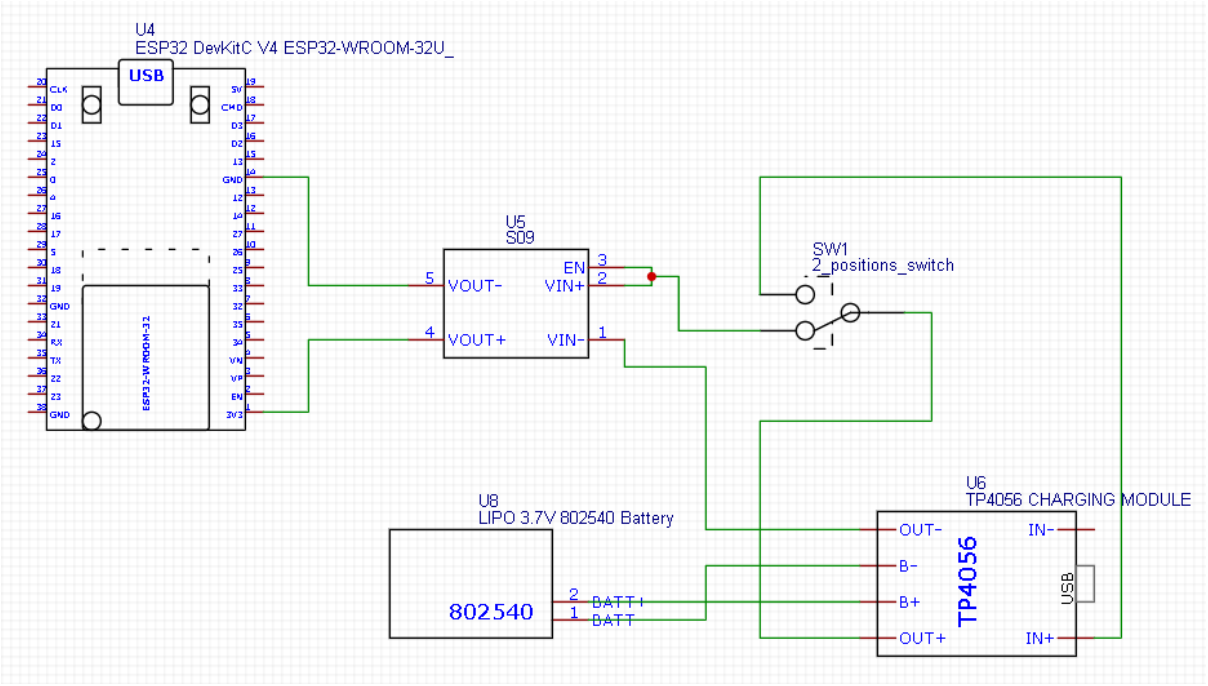


Fuente: Autoría

3.3.6.5. Diseño del circuito de Conexión de batería a la placa de desarrollo ESP32 y cargador

Para la alimentación eléctrica externa de esta placa se toma la recomendación del fabricante con un voltaje máximo de 3.3 v para alimentar la placa con los pines correspondientes. Es posible realizar la conexión utilizando el regulador de voltaje y el cargador de batería escogido en la fase de elección de hardware. Para diseñar el circuito se utiliza ademas un switch que cambiará el estado del circuito, ya sea para cargar el circuito o funcionamiento solo con batería Con todas estas consideraciones se muestra la conexión en la Figura 75,notar que el circuito funciona bajo los parámetros mencionados

Figura 75. Conexión de una batería externa a la placa de desarrollo ESP32.



Fuente: Autoría

En el caso de que utilice este circuito para la alimentación, también se utilizara un cargador USB externo para cada uno de los nodos del sistema.

3.3.7. Configuración de la infraestructura de red

Para esta fase se considera el desarrollo de la codificación de todas las funcionalidades de los módulos del sistema. Se toma en cuenta los respectivos modelos del estándar Bluetooth Mesh y como se van a aplicar dentro de cada uno de los módulos, así como los equipos utilizados. También se considera en cuenta los roles de las placas utilizadas, por ejemplo, no se utilizará el mismo esquema de programación en el dispositivo que es utilizado como Web Hosting, que el mismo utilizado en los módulos de la red de sensores; de hecho, en los módulos tampoco se repiten los esquemas de programación para algunos de ellos.

El lenguaje de programación será C, este es el lenguaje que utilizan los respectivos softwares de cada una de las placas de desarrollo, la arquitectura del estándar en los entornos de programación de cada uno de los SDK se basa en APIs nativas de los fabricantes. Las APIs base utilizadas son las de la empresa Nordic Semiconductor que no permite la modificación del código utilizada, las API de Silicon Labs siguen el mismo principio mientras las de Espressif son de uso libre. Las API permiten programas los procesos de funcionamiento para que un equipo pueda formar de una Bluetooth Mesh. De acuerdo al estándar se implementan nodos con sus respectivos elementos, los cuales están regidos a ciertas características de funcionamiento de acuerdo a modelos. Tomando en cuenta esta conclusión, se debe escoger que métodos ya definidos usar en cada uno de los módulos para construir la arquitectura de red planteada.

Para el módulo de control de luces se utilizará la placa de desarrollo ESP32 y el SmartBuld Sengled A19. Dentro de los ejemplos se tiene el modelo de on-off tanto como para cliente como para servidor, en base a este ejemplo en modo cliente se construiría el código para el módulo. El Smart Bulb ya viene precargado con el firmware que soporta el modelo on-off en modo servidor por lo que se debe resetear cualquier configuración, para ello se enciende y

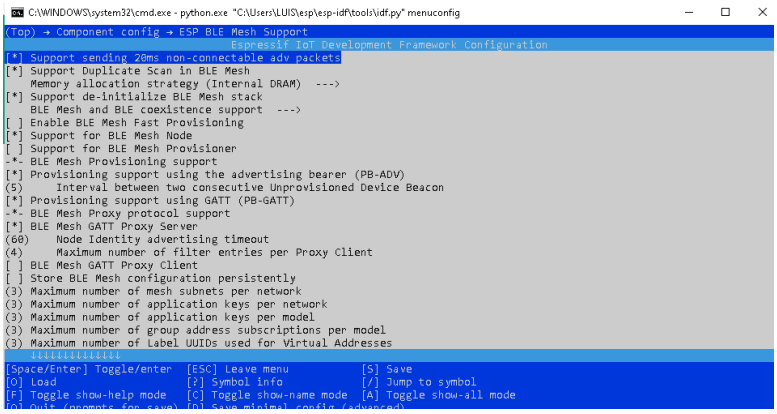
apaga la bombilla 5 veces., si el proceso se realizó correctamente entonces el bombillo se encenderá tres veces antes de regresar a su estado normal.

Para los demás módulos se plantea los respectivos modelos y elementos de acuerdo al estándar oficial de Bluetooth mesh. A continuación, se explica la lógica de programación usada en los archivos y códigos del presente proyecto, así como las herramientas adicionales utilizadas.

3.3.7.1. Configuración de parámetros previos de aprovisionamiento

Los parámetros previos de aprovisionamiento se configuran antes de subir los códigos a las placas de desarrollo utilizadas. Aquí se configura los modelos que soportara el nodo, algunos modelos son mandatorios mientras que, para algunos nodos, estos parámetros vienen configurados de fábrica. En la se observa la configuración de estos parámetros.

Figura 76. Configuración de parámetros de ESP-BLE mesh en una placa ESP32.



Fuente: Autoría

4. Capítulo IV: Implementación y Pruebas de funcionamiento del sistema

4.1. Implementación del aprovisionador

La librería utilizada para la aplicación móvil brinda esta función ya implementada con todo los relacionado. Además, dada la naturaleza del estándar, los datos de sensores y demás se pueden acceder fácilmente si se utiliza el teléfono móvil como aprovisionador, a diferencia de usar un equipo externo y ademas de este modo se asegura una futura escalibidad de la red.

4.2. Implementación física de nodos

En esta sección se realizará la conexión final de los sensores a cada una de las placas utilizadas para el desarrollo del proyecto. Por lo que se considera la cantidad de sensores y cantidad de placas usadas de la. También se debe tomar en cuenta los diferentes tipos de sensores la combinación de estos en cada nodo de la red. En Tabla 31 la se puede observar los materiales y equipos usados para su posterior distribución de nodos

Tabla 31. Materiales usados para la construcción de la red.

Tipo	**Numero**	**Observación**
Placa ESP32	4	Placas usadas para conectar sensores
Sensor MQ6	2	Sensor de GLP
Sensor GAOHOU A930	2	Sensor de presencia de agua
Sensor KY-26	2	Sensor detector de llama
Thingy 52	2	Kit de sensores IoT
SCT 013	1	Sensor de medición de corriente

Fuente: Autoría

A continuación, se detalla los implementos necesarios usados para montar la estructura física de los nodos, tomando en cuenta la información de planificación de módulos y submódulos del sistema.

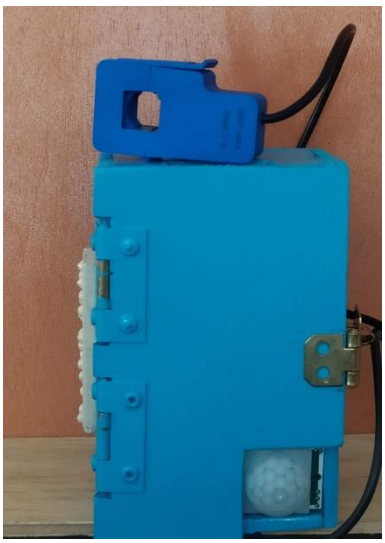
4.2.1. Implementación del módulo de estado de objetos

Dentro de este módulo se tiene a los submódulos de posición y control de estado de objetos. Los dos submódulos se implementarán con la ayuda de las dos placas de desarrollo utilizadas con sus respectivos sensores como se explica a continuación.

El Thingy 52 cuenta con un pulsador, este pulsador puede configurarse como un temporizador. Por lo que se puede utilizar para controlar cuando alguien ha entrado en las zonas donde se implementarán dichos dispositivos. En la se observa el posicionamiento del módulo y los datos recibidos cuando el pulsador es utilizado dentro de la aplicación.

Para el caso del submódulo de control de estado se planificó para alertar el estado de algunas de las conexiones electicas dentro del hogar, se utilizará el sensor SCT 013 conectado a una placa ESP32. La placa también podrá albergar otros sensores en caso de ser necesario como es el caso del sensor de presencia. En la Figura 77 se puede observar la conexión física de los sensores utilizados.

Figura 77. Implementación física del submódulo de control de estado



Fuente: Autoría

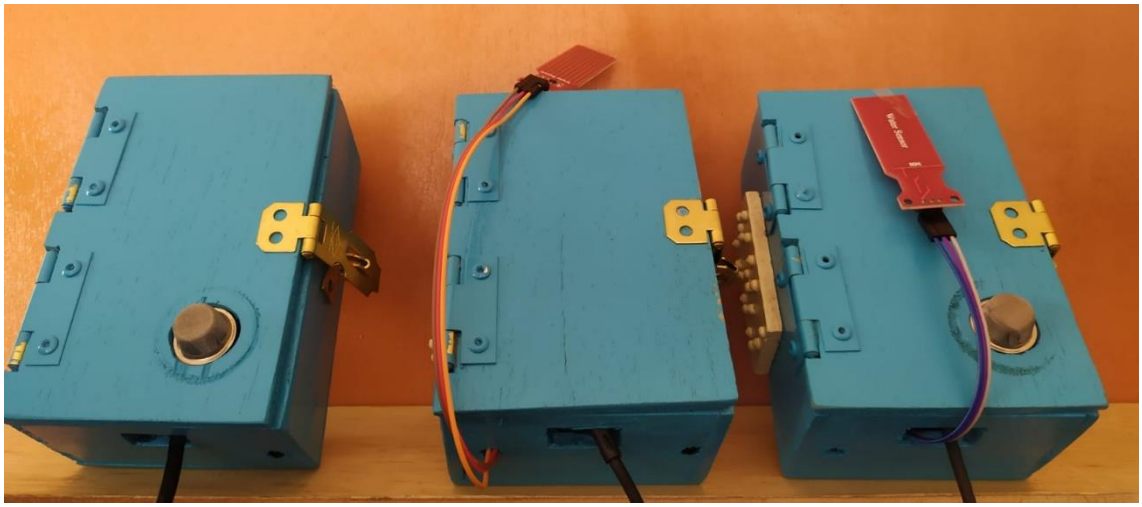
4.2.2. Implementación del módulo de control de Luces

El módulo de control de luces se implementará con la ayuda de las bombillas Smart, estas funcionan con un módulo interno que permite el soporte de bluetooth mesh.

4.2.3. Implementación del módulo de control de Aire, Incendios y detección de agua

Estos módulos serán implementados usando las placas ESP32 restantes con dos sensores en cada uno. En la Figura 78 se observar el diagrama de conexiones planteada para estos módulos

Figura 78. Implementación física de los módulos de control de aire, incendios y detección de agua

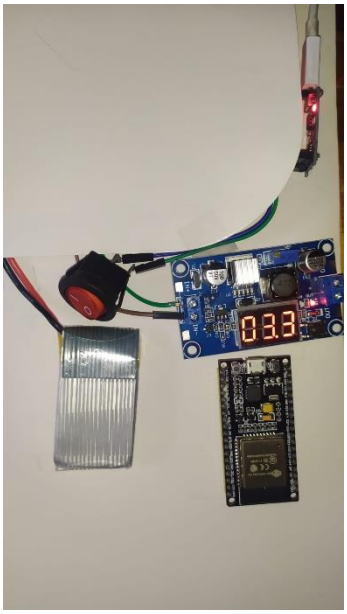


Fuente: Autoría

4.2.4. Implementación de la batería a los nodos

Este diseño se implementará como una opción adicional, solo algunos de los nodos implementaran esta función. Esta funcionalidad podrá ser implementada en todos los nodos dependiendo del rendimiento de la batería en los nodos implementados, en una futura actualización del sistema. En la Figura *79* se puede observar la implementación de la batería dentro de un nodo de la red construida.

Figura 79.Implementación de baterías en los nodos de la red.



Fuente: Autoría

Hay que recalcar que este circuito será opcional para los nodos en los que se decida implementar con una batería.

4.2.5. Listado de nodos

En la sección anterior se mostró como se distribuirán las placas de desarrollo y sensores de acuerdo a los módulos planificados; para identificar de forma sencilla a los nodos se presenta la siguiente tabla en donde se asigna un código a cada uno de los nodos. Los códigos asignados se pueden observar en la Tabla 32

Tabla 32. Código de identificación para los nodos de la red construida.

Identificador de nodo	**Sensores presentes**	**Modulo perteneciente**
N1BLEMesh	Smart Bulb	Control de luces
N2BLEMesh	Detección de agua	Control de inundaciones
	Detección de GLP	Control de aire
N3BLEMesh	Detección de incendios	Control de incendios
	Detección de agua	Control de inundaciones
N4BLEMesh	Detección de GLP	Control de aire
	Detección de incendios	Control de incendios
N5BLEMesh	Detección de corriente	Control de posición y objetos
	Detector de presencia	Control de posición y objetos
N6BLEMesh	Detector de presencia	Control de posición y objetos
	Detector de VOC	Control de aire
N7BLEMesh	Detector de presencia	Control de posición y objetos
	Detector de VOC	Control de aire

Fuente: Autoría

4.2.6. Identificadores para los nodos en Braille

Con todos los parámetros de la fase de diseño terminada, se procede a la construcción de contenedores para cada uno de los nodos, considerando la alimentación eléctrica de la placa utilizada. Para ello se realizó contenedores de madera con el fin de acoplar las conexiones a baterías y alimentación eléctrica, así como colocar sensores dentro de la caja. De esta forma se logra cumplir con los requerimientos de tamaño de nodo, así como la alimentación del sistema. También se realizó la construcción de identificadores de nodo en sistema braille, se utilizó un trozo de tríplex en el cual se incrusto diamantes de plástico con el fin de simular el sistema

Braille en papel. La palabra usada será sensor, que de acuerdo a la traducción a braille se define de acuerdo a la Tabla *33*.

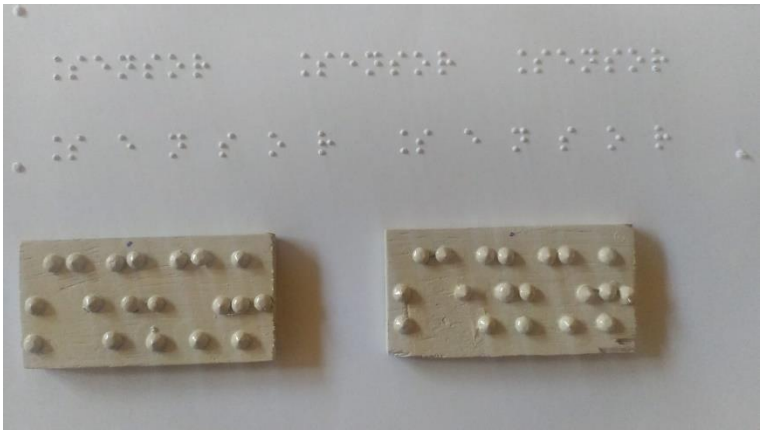
Tabla 33. Palabra sensor en Braille

Palabra letra por letra	Palabra en braille ⠎ ⠑ ⠝ ⠎ ⠕ ⠗
s	⠎
e	⠑
n	⠝
s	⠎
o	⠕
r	⠗

Fuente: Autoría

Como resultado se obtiene, ademas se realiza la comparación de la misma palabra escrita en una cartulina formato A4. Todo esto se puede observar en la figura correspondiente. Notar que la escritura en papel indica en el primer símbolo que todas las letras a continuación son mayúsculas.

Figura 80. Comparación de la palabra sensor escrita en braille sobre una cartulina y sobre tríplex.



Fuente: Autoría.

Con la implementación de los nodos físicos, se procede a realizar las pruebas de validación de los requerimientos, de acuerdo a la metodología planteada para el desarrollo del presente proyecto. Las pruebas se realizarán en un ambiente de funcionamiento, pero con parámetros especiales para comprobar y validar la utilización de la tecnología escogida. Las pruebas de funcionamiento se realizan de acuerdo a los siguientes ítems.

- Rendimiento de la red mesh
- Alertas al usuario
- Costo del sistema

A continuación, se detalla el proceso y resultados obtenidos de las pruebas listadas.

4.3. Desarrollo de la aplicación usando Android Studio

En la Figura 46 se observa la arquitectura del sistema, la aplicación móvil necesita comunicarse con un nodo proxy dentro de la red para que sea capaz de realizar el intercambio de datos con la red. La aplicación podrá enviar datos a solo dichos nodos, al momento del configurar el nodo es posible activar la función de proxy solo si se ha configurado el nodo para soportar dicha función como se mencionó en incisos anteriores. En resumen, se desarrollará una API a la cual se conectará el cliente Android para consumir los servicios programados en esta.

4.3.1.1. Planificación de pantallas de la aplicación

Android Studio utiliza lenguaje XML para diseñar el dashboard que se mostrará al cliente, cada archivo XML puede contener varias pantallas. Estas pantallas se asocian con archivos de configuración propios de la estructura de proyectos en Android Studio, que indican a la aplicación las acciones a realizar cuando: se abre la aplicación, se presiona un botón entre otras más. Las pantallas con las que contará la aplicación se basan en las funcionalidades de aplicación, por lo que la aplicación deberá contar con al menos las siguientes pantallas:

- Abriendo aplicación

Se trata de una pantalla datos con datos informativos que se mostrara durante varios segundos al ingresar en la aplicación.

- Administración de nodos de la red

Se trata de una pantalla usando la mayoría de las opciones de la librería utilizada, servirá para que el administrador configure los nodos para posteriormente realizar el despliegue de la red de sensores.

- Listado de dispositivos activos Bluetooth para conexión y reconexión

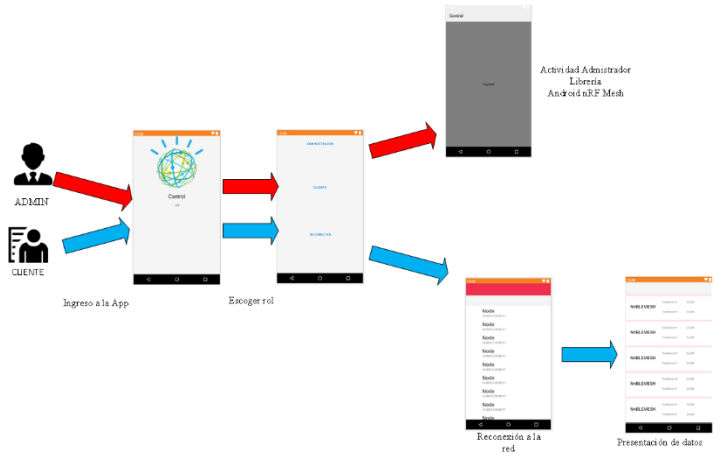
Esta pantalla mostrara el listado de dispositivos que funcionan bajo el estándar Bluetooth mesh y tienen activada la opción de proxy. Se mostrará la dirección MAC del dispositivo y el UUID de red a la cual está asociado y se debe poder escoger el dispositivo en cuestión para posteriormente realizar la conexión al mismo.

- Listado de valores de sensores con los respectivos valores recolectados

Esta pantalla se encargará de mostrar los datos se sensores al cliente, las alertas se realizarán utilizando audios predeterminados dentro de la aplicación, los cuales corresponden a los nodos físicos presentes en la red. Desde esta pantalla se enviran los datos a la plataforma en la nube escogida, así como se clasificará de acuerdo a los nodos donde se produjeron. La interfaz será amigable con el usuario del sistema.

En la se puede observar las pantallas planificada y su rol en el funcionamiento de la aplicación desarrollada, nótese los roles definidos y el uso de la librería para las actividades de administrador

Figura 81. Flujo de pantallas de la aplicación a desarrollar



Fuente: Autoría.

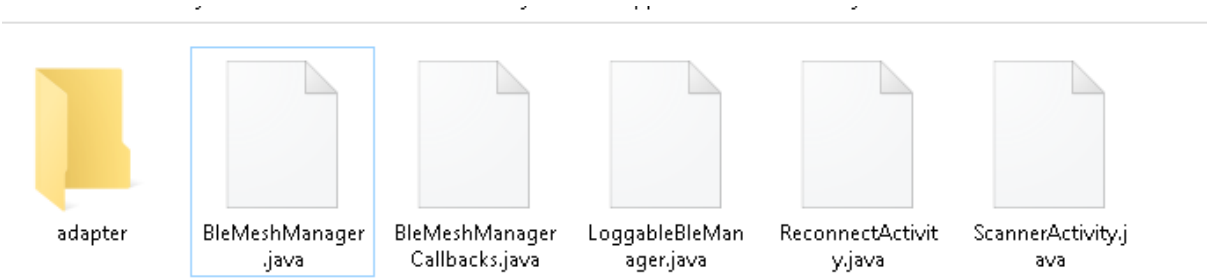
A continuación, se realiza una breve explicación de las funcionalidades esperadas que la aplicación debe soportar, además de los procesos realizados para lograr dichas funcionalidades.

4.3.1.2. Escaneo de dispositivos y permisos de aplicación

Para esta y las siguientes funcionalidades se utilizará funciones propias del core de Android disponibles en el Ide usando el lenguaje Java. Los permisos que deben concederse a la aplicación serán: control de administración global de la interfaz Bluetooth del dispositivo, acceso a la localización y acceso a la interfaz de red.

Las funciones necesarias ya se encuentran contenidas dentro de la librería utilizada, tomando en cuenta las capas y sus respectivos elementos declarados en el estándar de comunicación de red. En la Figura *82* se puede observar el contenido de la carpeta que contiene las funciones básicas de escaneo de dispositivos Bluetooth usando la tecnología de comunicación inalámbrica Bluetooth Low Energy.

Figura 82. Contenido de la carpeta correspondiente a la librería utilizada para escaneo de dispositivos.



Fuente: Autoría.

Por otro lado, se usará los permisos por defecto utilizados en la librería base, dado que se utilizarán las mismas interfaces del smartphone donde se instalará la aplicación.

4.3.1.3. Programación de recibo de datos y Envío de datos

Se utilizará al smartphone como cliente de todos los nodos de la red. El objetivo es conseguir los servicios y características que tienen el Proxy server (GATT server) ubicado en la red, estos valores serán des encapsulados dentro de la aplicación móvil y transformados a

datos útiles que puedan ser utilizados posteriormente. Estos datos serán usados para guardar cada alerta como un evento y posterior Envío a la nube para uso del administrador .Para lograr el control de mensajes que envía y recibe el smartphone ,la librería implementa una API .Con la ayuda de esta API ,se configuro él Envío de sensor Get y recibo de sensor status para los nodos de sensores .Para el modelo de on off se realizó el proceso respectivo para él Envío de onoff get y recibo de onoff status ,además se envía un onoff set para apagar el nodo en cuestión cada vez que el cliente entre en la aplicación .En la Figura *83* se puede observar

Figura 83. Envío y recibo de datos desde el smartphone hacia la red de sensores.

```
refreshLayout.setOnRefreshListener(
        new SwipeRefreshLayout.OnRefreshListener() {
            @Override
            public void onRefresh() {
                final SensorGet sensorGetn1 = new SensorGet(mViewModel.getNetworkLiveData().getAppKeys().get(0),  property: null);
                final SensorGet sensorGetn2 = new SensorGet(mViewModel.getNetworkLiveData().getAppKeys().get(0),  property: null);
                final SensorGet sensorGetn3 = new SensorGet(mViewModel.getNetworkLiveData().getAppKeys().get(0),  property: null);
                final SensorGet sensorGetn4 = new SensorGet(mViewModel.getNetworkLiveData().getAppKeys().get(0),  property: null);
                final SensorGet sensorGetn5 = new SensorGet(mViewModel.getNetworkLiveData().getAppKeys().get(0),  property: null);
                final SensorGet sensorGetn6 = new SensorGet(mViewModel.getNetworkLiveData().getAppKeys().get(0),  property: null);
                final SensorGet sensorGetn7 = new SensorGet(mViewModel.getNetworkLiveData().getAppKeys().get(0),  property: null);
                final GenericOnOffGet genericOnOffget = new GenericOnOffGet(mViewModel.getNetworkLiveData().getAppKeys().get(0));


                mViewModel.getMeshManagerApi().createMeshPdu( dst: 2,genericOnOffget);
                mViewModel.getNrfMeshRepository().onMeshMessageReceived( src: 1,genericOnOffget);
                mViewModel.getMeshManagerApi().createMeshPdu( dst: 3,sensorGetn1);
                mViewModel.getNrfMeshRepository().onMeshMessageReceived( src: 1,sensorGetn1);
                mViewModel.getMeshManagerApi().createMeshPdu( dst: 4,sensorGetn2);
                mViewModel.getNrfMeshRepository().onMeshMessageReceived( src: 1,sensorGetn2);
                mViewModel.getMeshManagerApi().createMeshPdu( dst: 5,sensorGetn3);
                mViewModel.getNrfMeshRepository().onMeshMessageReceived( src: 1,sensorGetn3);
                mViewModel.getMeshManagerApi().createMeshPdu( dst: 6, sensorGetn4);
                mViewModel.getNrfMeshRepository().onMeshMessageReceived( src: 1,sensorGetn4);
                mViewModel.getMeshManagerApi().createMeshPdu( dst: 7,sensorGetn5);
                mViewModel.getNrfMeshRepository().onMeshMessageReceived( src: 1,sensorGetn5);
                mViewModel.getMeshManagerApi().createMeshPdu( dst: 8,sensorGetn6);
                mViewModel.getNrfMeshRepository().onMeshMessageReceived( src: 1,sensorGetn6);
                adapter.notifyDataSetChanged();
```

Fuente: Autoría.

4.3.1.4. Programación de alertas

Las alertas se programarán en base a los datos recibidos en el teléfono móvil, de acuerdo a los datos de sensores. Dichas alertas serán audios en formato mp3 que serán reproducidos al momento de recibir un cambio de valor en un determinado sensor. Los cambios que se van a leer desde la base de datos en la nube, esto con el fin de no sobrecargar al smartphone. Además, las alertas se reproducirán una después de otra de acuerdo a los nodos de la red. EnFigura *84* la se puede observar un extracto de código de la reproducción de alertas por medio del smartphone.

Figura 84. Alertas reproducidas después de un evento.

```
277
278             String val2_node7 =snapshot.child("Sensor").child("N7BLEMesh").child("Sensor2").getValue().toString();
279             int valor2Nodo7 = Integer.parseInt(val2_node7);
280             ValorDatos1.set(6,val2_node7);
281             if(valor2Nodo7>=1){
282                 saveNotification( sensor: 2, nodo: 7,snapshot);
283             }
284             int ns11= verifyWarning( nodoSensor: 11,valorNodo1);
285             int ns21=verifyWarning( nodoSensor: 21,valor1Nodo2);
286             int ns22=verifyWarning( nodoSensor: 22,valor2Nodo2);
287             int ns31=verifyWarning( nodoSensor: 31,valor1Nodo3);
288             int ns32=verifyWarning( nodoSensor: 32,valor2Nodo3);
289             int ns41=verifyWarning( nodoSensor: 41,valor1Nodo4);
290             int ns42=verifyWarning( nodoSensor: 42,valor2Nodo4);
291
292             int ns51=verifyWarning( nodoSensor: 51,valor1Nodo5);
293             int ns52=verifyWarning( nodoSensor: 52,valor2Nodo5);
294             int ns61=verifyWarning( nodoSensor: 61,valor1Nodo6);
295             int ns62=verifyWarning( nodoSensor: 62,valor2Nodo6);
296             int ns71=verifyWarning( nodoSensor: 71,valor1Nodo7);
297             int ns72 = verifyWarning( nodoSensor: 72, valor2Nodo7);
298
299             int valoresNodos[] = {ns11,ns21,ns22,ns31,ns32,ns41,ns42,ns51,ns52,ns61,ns62,ns71,ns72};
300             nodoSensorAlert= nodoSensorAlert(valoresNodos);
301             //playBeep(nodoSensorAlert);
302             adapter.notifyDataSetChanged();
303         }
```

Fuente: Autoría.

La transformación ADC de la placa de desarrollo no es perfecto por lo que existe cierto ruido en el circuito que en algunos casos no permite que el valor cuando no existe el evento baje a cero .Por lo cual después de las pruebas se realizadas es posible concluir que: los valores de la son los valores de reposo de los sensores cuando no existe el evento a medir, los valores recolectados para cada sensor se pueden observar en la Tabla 34.

Tabla 34.Valores de los sensores cuando no existe ninguna detección .

Sensor	Valor medido con evento (ADC)	ADC sin evento	ADC con evento
Sensor de GLP	80 a 86 mV/2 sin evento	-86(nodo 2) 2(nodo 4)	-1(nodo 2) Mayor a 20(nodo 4)
Sensor de agua	-76 a -114	Mayor a 50(nodo 2) -100(nodo 3)	27(nodo 2) 81(nodo 3)
Sensor de incendios	45 sin evento 75 con evento /52 sin evento	45(nodo 3) 52(nodo 4)	75(nodo 3) -108(nodo 4)
Sensor de presencia	75	33	75
Sensor de corriente eléctrica 1	118	117	78
Sensor de VOC	300	300	

Fuente : Autoría

4.3.1.5. Accesibilidad para personas con discapacidad visual

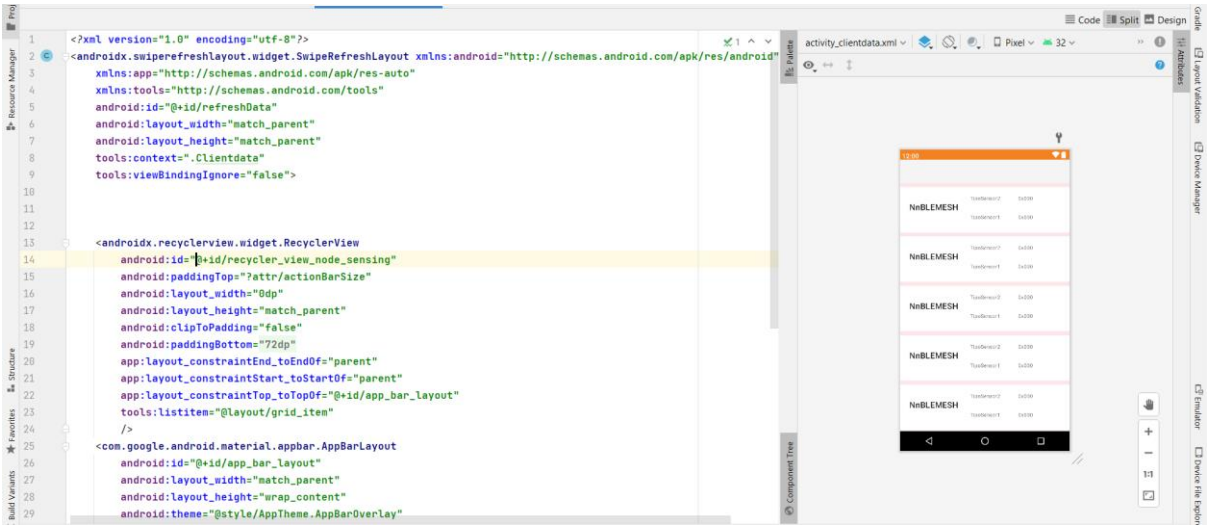
Con la ayuda audios de voz pregrabados se reproducirá alertas por voz en base a los valores recibidos por los sensores. Por ejemplo, en la sección para el ingreso a los roles que

brinda la aplicación, se utiliza un audio con las instrucciones para acceder a dichos roles y alguna explicación muy rápida de las funcionalidades de la aplicación.

También se ha implementado la navegación por voz para la reconexión de dispositivos y el acceso a la pantalla de clientes ,para ello se utiliza la función SpeechRecognizer .Esta función permite utilizar el micrófono del smartphone y traducir las palabras a texto para su posterior uso en la aplicación .

Ademas se debe asegurar que la interfaz para visualización de datos posea un diseño simple, así como la navegación dentro de la aplicación para acceder a las funciones del cliente. De igual forma ,la pantalla que muestra y presenta alerta funcionara mediante la función refresh .Esta función permite al usuario realizar operaciones al deslizar un dedo en la pantalla .Todas las operaciones se realizaran en base a esta acción ,de tal forma que cada vez que el usuario deslice sus dedos en la pantalla entonces comenzara el envío de datos ,recibimiento de mensaje ,alertas y demás presentes en la aplicación. En la se puede observar el tipo de layout utilizada para la pantalla de cliente con el fin de cumplir los estaos mencionados.

Figura 85.Layout utilizado para implementar la función swipe to refresh.



Fuente: Autoría.

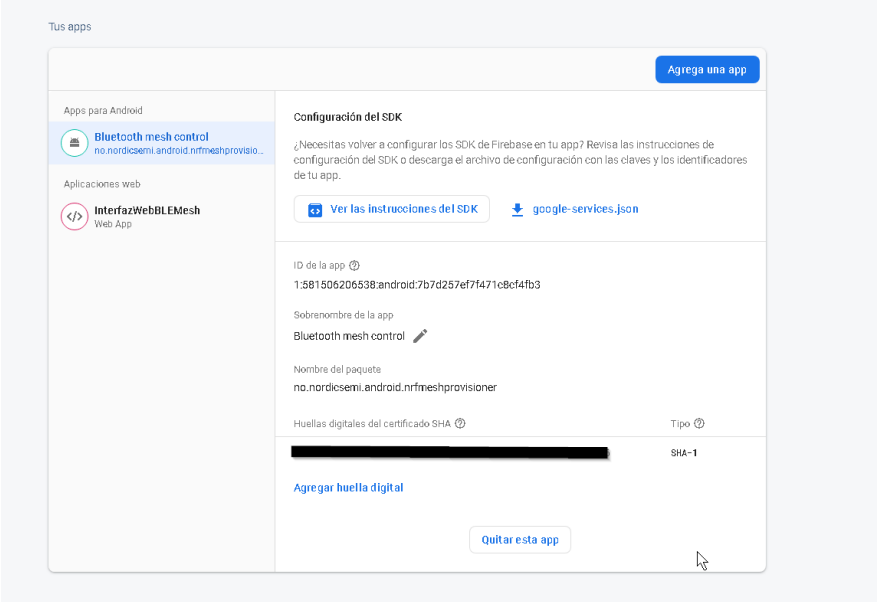
4.3.1.6. Conexiones a la nube

Según los requerimientos presentados, es necesarios la exportación de datos de cada uno de los sensores dentro de la red para presentar los datos al administrador del sistema, para este fin se utilizará una herramienta complementaria de Google llamada Firebase, específicamente la herramienta de base de datos en tiempo real. Firebase utiliza el protocolo FCM HTTP v1 API para la mayoría de las aplicaciones, lo que permite integrar varios tipos de dispositivos para un mismo proyecto, en este caso se utiliza el mismo proyecto para utilizar las funciones de base de datos en la aplicación móvil y la función de hosting en una aplicación web.

Los datos guardados también se utilizarán para presentar las alertas, por lo que se realizara el siguiente proceso: envío y recibo de datos de sensores desde la red Bluetooth Mesh, envío de datos de sensores a una base de datos en tiempo real de Firebase, lectura de cambios en la base de datos de Firebase y alertas de acuerdo a los cambios realizados en la base de datos de Firebase. Estos datos se presentarán al administrador usando otra función de Firebase.

Al crear un proyecto en Firebase es posible realizar las conexiones a otros tipos de aplicaciones, para este caso se realizará la conexión a una aplicación Android y una aplicación Web. En la Figura 86 se puede observar las aplicaciones vinculadas al proyecto creado

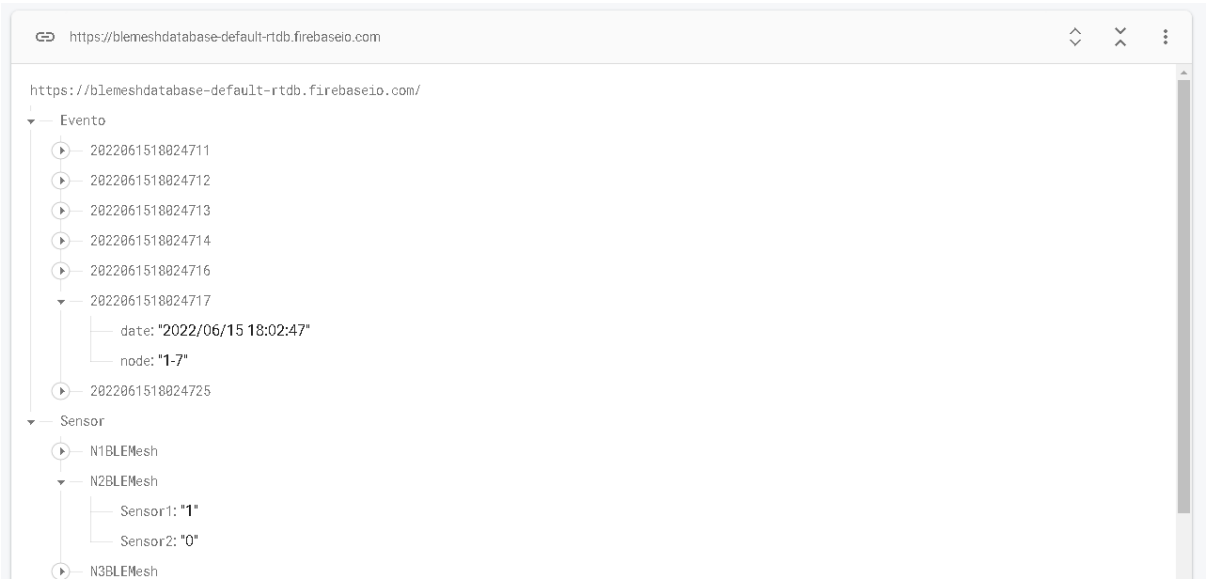
Figura 86. Aplicaciones vinculadas al proyecto creado en Firebase.



Fuente: Autoría.

En si la base de datos guardar dos campos: los valores de nodos y eventos. Los valores de nodos contendrán cada uno de los valores de los sensores en tiempo real conforme se actualicen en el smartphone mientras los eventos guardaran los eventos en donde los valores de un sensor sobrepasaron ciertos límites establecidos. Se guardará una fecha y el sensor, así como el nodo involucrado en el evento. En la Figura 87 se puede observar la estructura de la base de datos.

Figura 87. Estructura de la base de datos usada



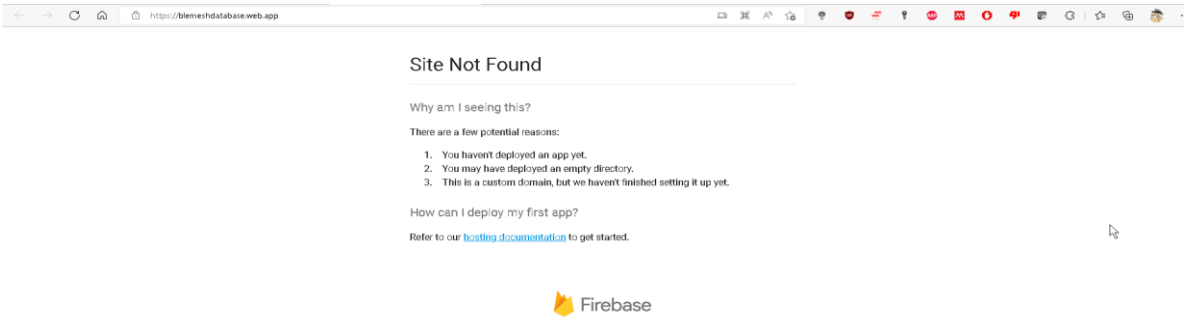
Fuente: Autoría.

4.4. Desarrollo de la interfaz web para usuarios indirectos

Con los datos listos para usarse en la base de datos en Firebase, se procederá a levantar un servidor web con la ayuda de la función de Web Hosting de Firebase. El servidor web contendrá una página web para visualización de eventos de la red de sensores construida, en donde se especificará: hora, sensor y nodo del evento detectado. Con la utilización de este servicio de Google se evita realizar las configuraciones de TTLS, así como DNS del servidor web por lo que se asegura un servidor web seguro, fácil de implementar, fácil de actualizar y mantener.

Se trabajará con el mismo proyecto de la base en tiempo real usada para la aplicación móvil, para ello se vinculará un proyecto de Firebase a la aplicación desarrollada. Android Studio permite realizar conexiones de este tipo para vincular la base de datos en tiempo real. En la Figura *88***¡Error! No se encuentra el origen de la referencia.** se puede observar la URL del hosting, nótese que no se tiene ningún proyecto aún.

Figura 88. Hosting levantado usando Firebase sin ningún contenido

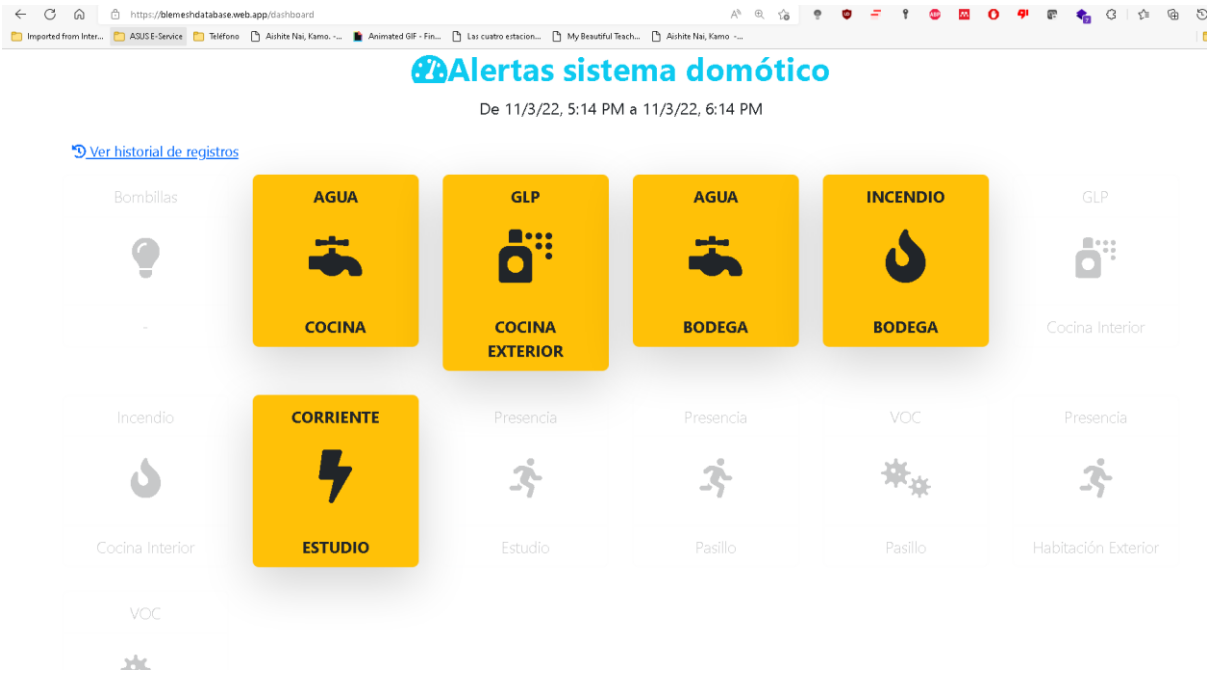


Fuente: Autoría.

La interfaz de visualización permitirá graficar los eventos con los datos que se están guardando en la base de datos. Para ello se diseñará y construirá una aplicación web usando Angular. La aplicación web realizara consultas al igual que la aplicación móvil, pero a un campo diferente de la base de datos. Se realizará la consulta al campo eventos, se analiza la cadena guardada y cada uno de los datos para construir una línea de tiempo de cada uno de los eventos suscitados, en cada uno de los nodos para presentar esta información de acuerdo a los

requisitos del proyecto. Además, se incluirá una tabla con los eventos recogidos por el sistema. En la se puede observar el despliegue de la aplicación web usando Firebase.

Figura 89. Aplicación Web en funcionamiento.



Fuente: Autoría.

4.5. Construcción del entorno de pruebas de verificación del sistema

4.6. Pruebas realizadas al sistema :Verificación de algoritmos usados en el sistema

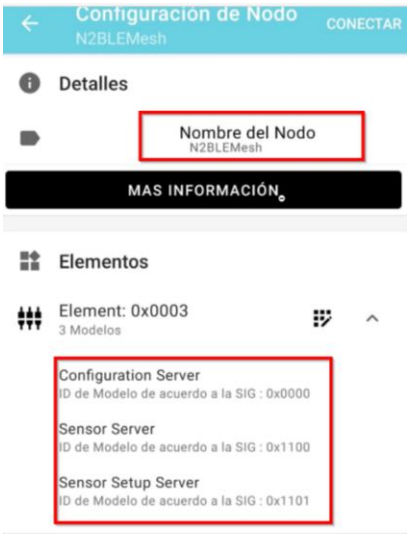
En esta prueba se realizará las verificaciones de cada una de las funciones de los bloques del sistema en base a los algoritmos planeados y diseñados para cada funcionalidad .En algunos casos será necesario el análisis de paquetes usando un sniffer en las secciones posteriores se explica cómo construir el entorno de pruebas ,las pruebas realizadas se enumeran a continuación :

4.6.1. Verificación de la correcta aplicación de modelos a los nodos de la red

Dentro de los nodos de la red es obligatorio la implementación del modelo Sensor Server ,este modelo debe ser implementado con el modelo de Sensor Setup Server de forma mandatorias .Otros modelos son opcionales ,en la red se ha implementado estos dos modelos y algunos modelos más en ciertos nodos .Esta configuración se puede comprobar con la ayuda

de la librería utilizada ,al entrar en la configuración de un nodo .Se mostraran los elementos del sensor ,en donde se encuentran los modelos implementados .En la Figura *90* se puede observar la presencia de los modelos en un nodo sensor de la red implementada .

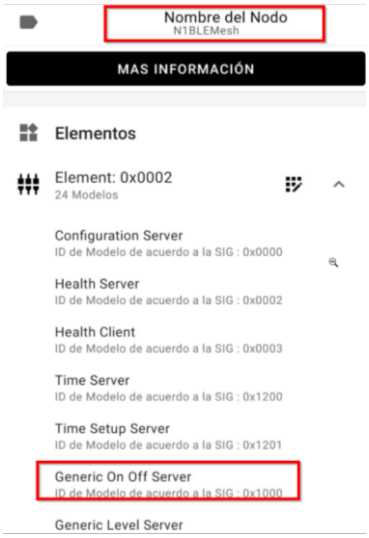
Figura 90.verificación de la implementación del modelo Sensor Server y sus complementos en un sensor de la red.



Fuente: Autoría.

De igual forma se debe implementar el modelo Generic OnOff Server ,este modelo viene implementado por defecto en la bombilla escogida para el control de luces .En la Figura *91* se puede observar algunos de los modelos implementado en la bombilla utilizada para el sistema.

Figura 91.Verificación de la implementación del modelo Generic OnOff Server en la bombilla

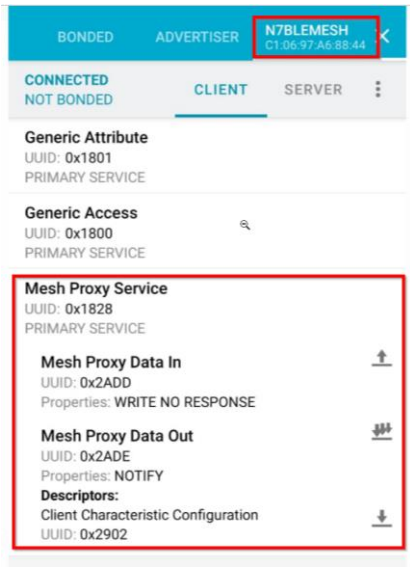


Fuente: Autoría.

4.6.2. Verificación del funcionamiento de la función proxy en los nodos de la red

La verificación de esta funcionalidad es propia de la librería utilizada ,en las placas de desarrollo utilizadas también es posible activar dicha funcionalidad .Para verificar esta funcionalidad es necesario el uso de la aplicación nRF Connect .Se realiza un escaneo y a continuación se realiza la conexión al nodo .Dentro del apartado del cliente deberá aparecer un servicio llamado Mesh Proxy Server ,este servicio funciona para la comunicación con un nodo externo ,que en este caso es el smartphone ,se puede observar la aparición de este servicio en el nodo N7BLEMESH en la .

Figura 92. Verificación del funcionamiento Mesh Proxy Server



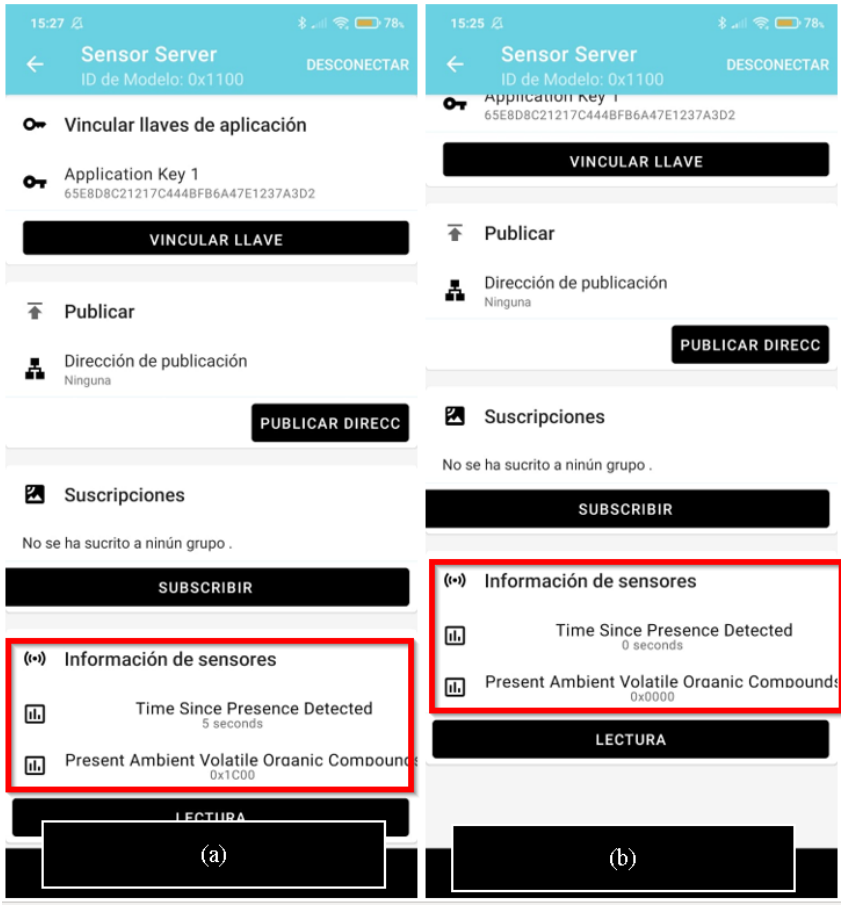
Fuente: Autoría

4.6.3. Verificación de lectura de datos de sensores

Para la verificación de lectura de datos de los sensores se utilizará la pantalla de administración de la aplicación móvil diseñada .También se ha realizado el correcto funcionamiento de los sensores de forma previa ,por lo que para esta prueba se solicita el Envío de datos de un determinado sensor .Específicamente se utiliza un nodo N7BLEMesh ,este nodo deberá enviar datos de presencia y VOC ,el valor de presencia deberá estar en cero cuando no exista presencia alguna. Los sensores enviaran un nuevo valor cuando exista el evento en

cuestión .Para el sensor de VOC ,de acuerdo a la documentación ,un valor superior a mil será mayor a la media .En la se puede observar los datos recibidos en la aplicación móvil de acuerdo con la prueba realizada cuando existe el evento.

Figura 93. Verificación de lectura de sensores ,cuando existe el evento y no existe el evento .



Nota :En(a) el sensor no ha recibido datos y en (b) el sensor ha detectado el fenómeno .

Fuente : Autoría

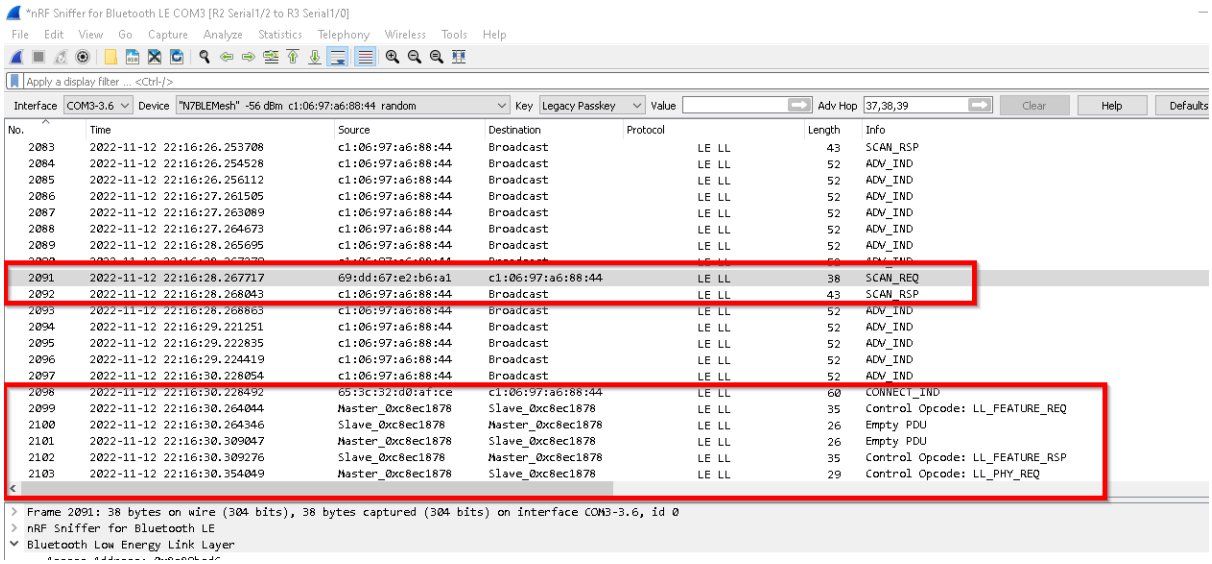
Gracias a esta prueba se puede confirmar que los valores de la Tabla *34* se cumplen a cabalidad.

4.6.4. Verificación de la reconexión al sistema

Para la verificación de esta funcionalidad es necesario la captura de paquetes durante el proceso. Es necesario verificar la presencia de los paquetes SCAN_REQ de parte del smartphone utilizado y la respuesta se dará por medio de un SCAN_RSP por parte del nodo. Para no muchos paquetes después ,se inicie una conexión de esclavo maestro entre los dos

dispositivos .En un caso de conexión por primera vez deben existir los paquetes de CONNECT_REQ ,los cuales no se visualizan en la captura. En la Figura *94* se puede observar la captura de paquetes realizados.

Figura 94. Captura de paquetes para la verificación de la reconexión del smartphone al sensor .

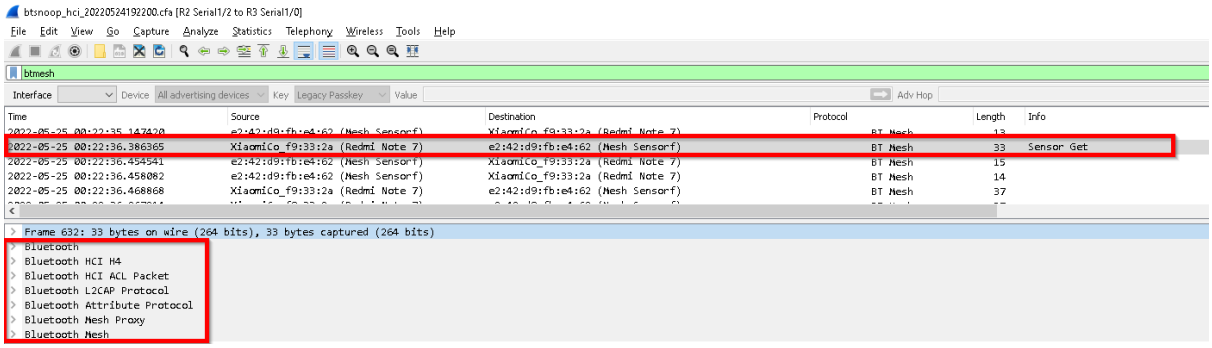


Fuente : Autoría

4.6.5. Verificación de la conexión de un dispositivo externo a la red de sensores

Los dispositivos que soportan el estándar de red Bluetooth mesh son pocos ,en este proyecto se utilizó algunos IDEs basados en Zephyr .el sistema operativo Android no los soporta de forma nativa pero la librería utilizada permite la encapsulación del estándar en las capas más bajas de la tecnología de comunicación Bluetooth Low Energy .Esto se puede comprobar mediante una captura de paquetes desde el smartphone ,se podrá observar a los protocolos de las capas física y enlace correspondientes al bloque de controlador y a al estándar de comunicación de red mesh en las capas altas .Para la comprobación se usa un paquete correspondiente a una operación Sensor Get, como se observa en la Figura *95*.

Figura 95.Comprobación del encapsulamiento del estándar de comunicación de red bluetooth mesh sobre el stack de protocolos de Bluetooth Low energy



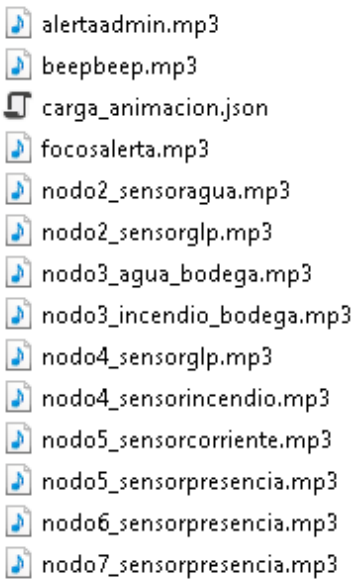
Fuente : Autoría

4.6.6. Verificación de la presencia de notificaciones después de alertas a usuarios directos e indirectos

Las alertas al usuario se realizan en base a los valores que se envían en tiempo real a la base de datos alojada en Firebase ,las alertas deberán tienen un nombre especificado para cada archivo que identifica al sensor y tipo de nodo. Para optimizar el tiempo de duración de alertas ,cada archivo mp3 durara un máximo de cinco segundos.

Se podrá comprobar que las alertas se reproducen de acuerdo con los eventos adecuados usando la pantalla de Debug de Android Studio .Considerando un ambiente de pruebas ,las alertas estarán ubicadas en una carpeta bajo el nombre de assets .Como se puede observar en la Figura *96*.

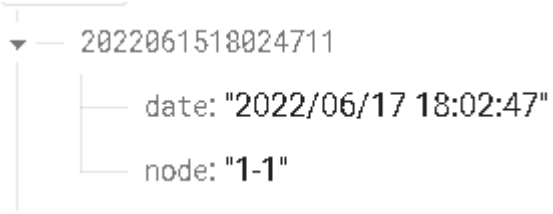
Figura 96.Archivos de audio que se reproducirán cuando suene una alerta .



Fuente : Autoría .

No se incluye a los sensores de VOC debido a que su tiempo de medición es diferente a los demás sensores ,por lo que las alertas se clasificaran de otra manera .Las alertas sonaran de acuerdo al evento detectado .Los valores se medirán de acuerdo a la base de datos en tiempo real de Firebase .Las alertas tendrán el formato que se observa en la Figura *97*.Se tendrá una fecha con la hora respectiva y el nodo, así como el sensor en donde se produjo el evento .

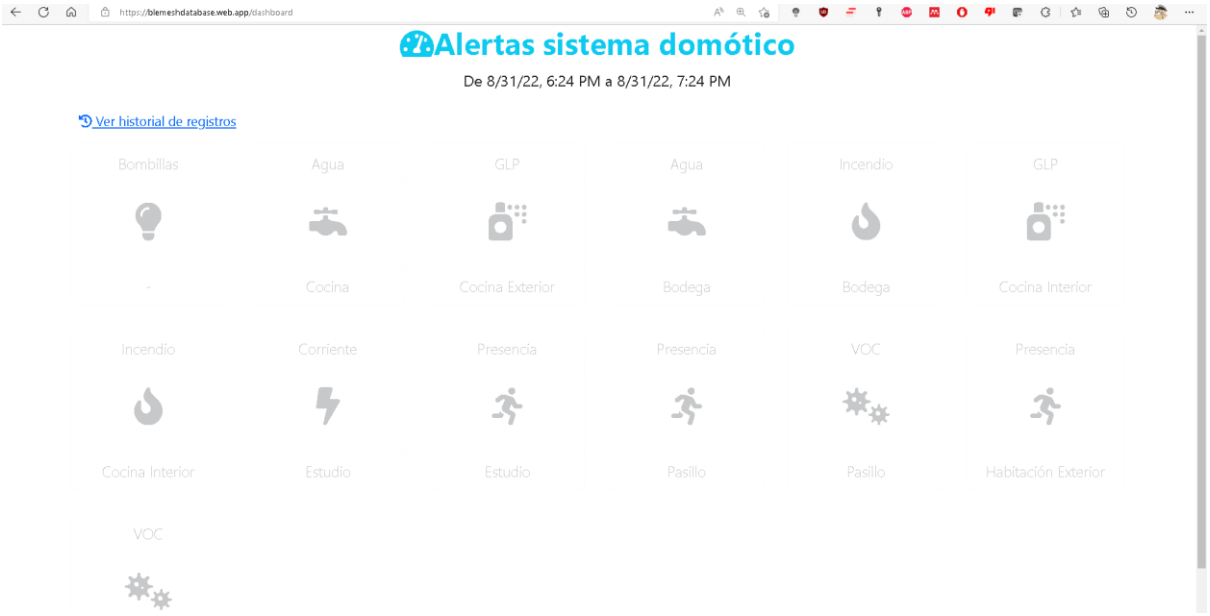
Figura 97.Formato del evento guardado en la base de datos de Firebase.



Fuente : Autoría .

De igual manera se actualizará la página web diseñada con este evento ,los mismos valores aparecerán en una línea de tiempo ,en base a los valores de fecha .Se ha diseñado el grafico para la fácil identificación del evento que se observa en la Figura *98*.

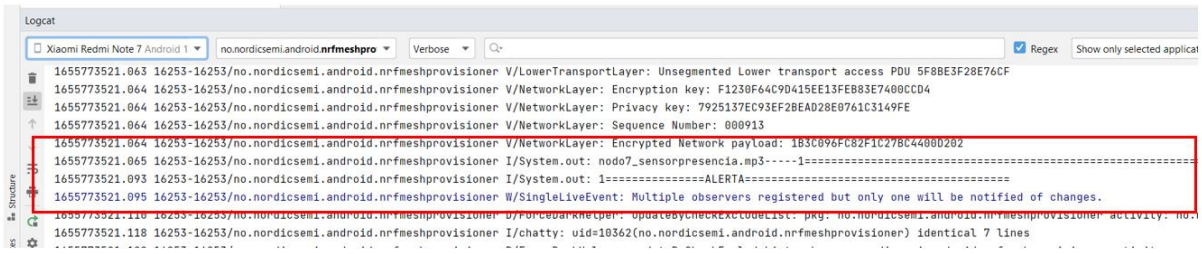
Figura 98. Página web después de recibir algunos eventos



Fuente : Autoría .

Mientras que, en la aplicación ,se convocara a un objeto del tipo Media player ,dicho objeto se encargara de reproducir los audios de acuerdo con los eventos detectados .En la Figura *99* ,se puede observar ,la presencia del archivo de audio que se ha reproducido con la ayuda de la consola de Android Studio. Comprobando de esta manera que las alertas suenan en el momento adecuado ,cumpliendo con los requisitos de alertas.

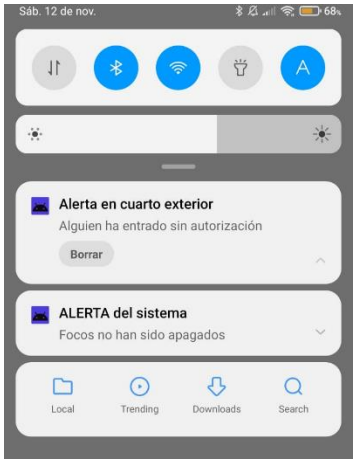
Figura 99. Reproducción de audio en la aplicación móvil .



Fuente : Autoría .

Ademas es posible la visualización en pantalla de notificaciones propias del sistema operativo Android ,en donde se indica :el tipo de alerta y donde sucedió .Como se observa en la Figura *100*.

Figura 100. Presencia de notificaciones en el smartphone del usuario.

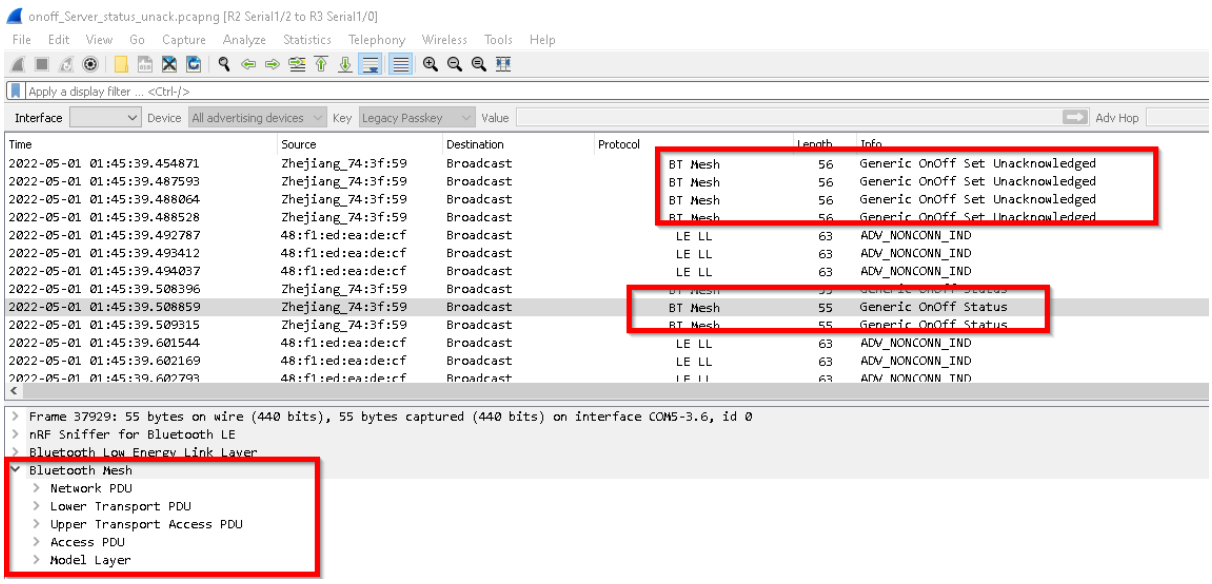


Fuente : Autoría .

4.6.7. Verificación de control de luces en el sistema

Para esta verificación también se utilizó una captura de paquetes ,las luminarias a utilizar son compatibles con el perfil Mesh .Se podrá observar los paquetes de Generic OnOff Status y Generic OnOff Set ,en la Figura *101* se puede observar la presencia de estos paquetes.

Figura 101.Comprobación de paquetes del modelo OnOff server en el sistema



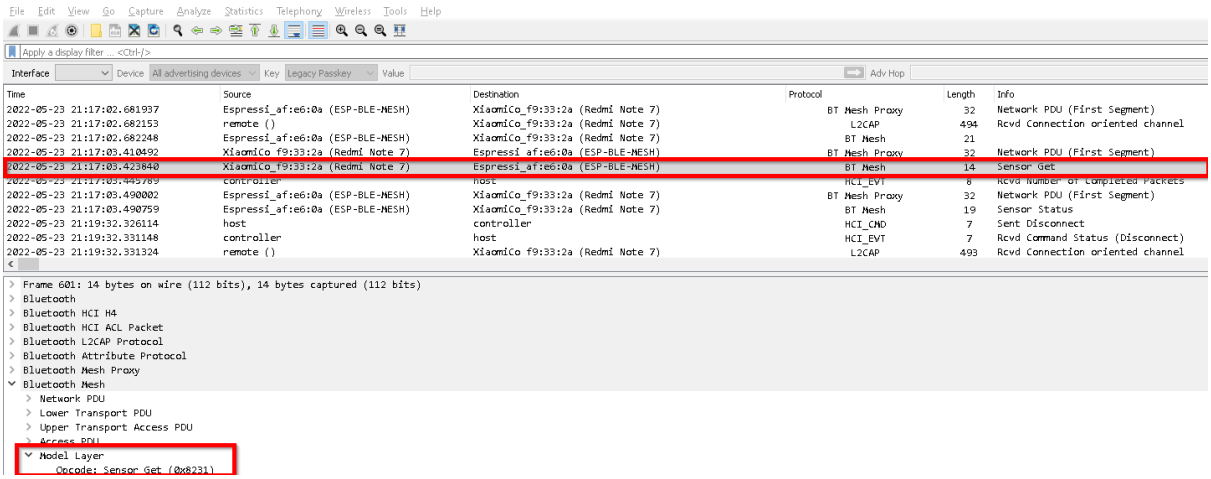
Fuente : Autoría .

4.6.8. Verificación de presencia de modelo servidor en los sensores de la red

De acuerdo con el diseño construido ,los sensores trabajaran en un esquema servidor-cliente con el smartphone ,siendo el smartphone el que consume los datos .En resumen ,si se realiza una solicitud a los sensores ,estos deberán responder con una respuesta .Para la

comprobación de esta funcionalidad ,se utilizara el modelo de servidor .En este modelo se utilizan el cliente (smartphone) solicita los datos a los sensores con un mensaje de Sensor Get, como se observa en la Figura *102*.

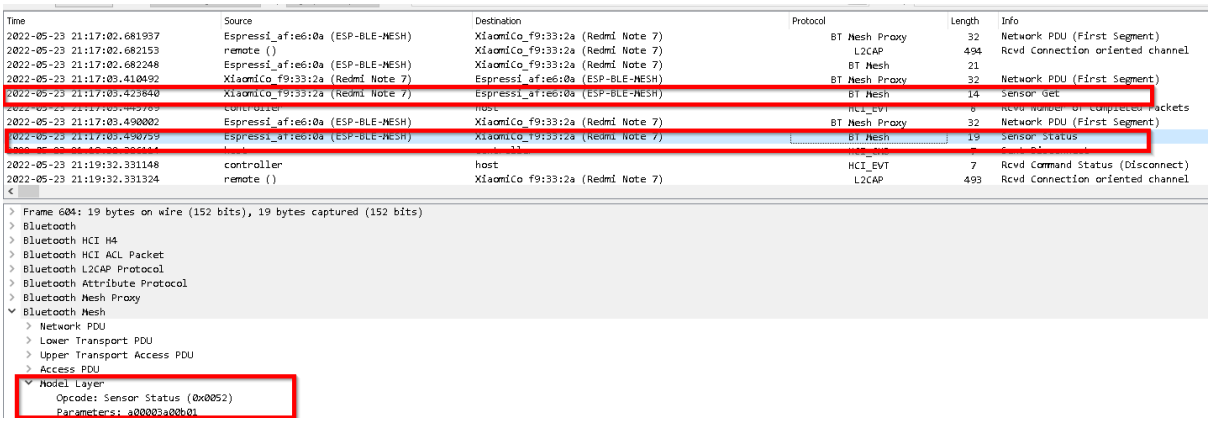
Figura 102.Solicitud de datos a un sensor desde el smartphone.



Fuente : Autoría .

Por su parte el servidor responderá con un Sensor Status ,en donde se enviarán el estado de los sensores en el momento en donde se realizó la petición. Como se puede observar en la Figura 103 .Notar que se utiliza el protocolo mesh proxy para recibir los datos.

Figura 103. Respuesta enviada al smartphone desde el sensor .



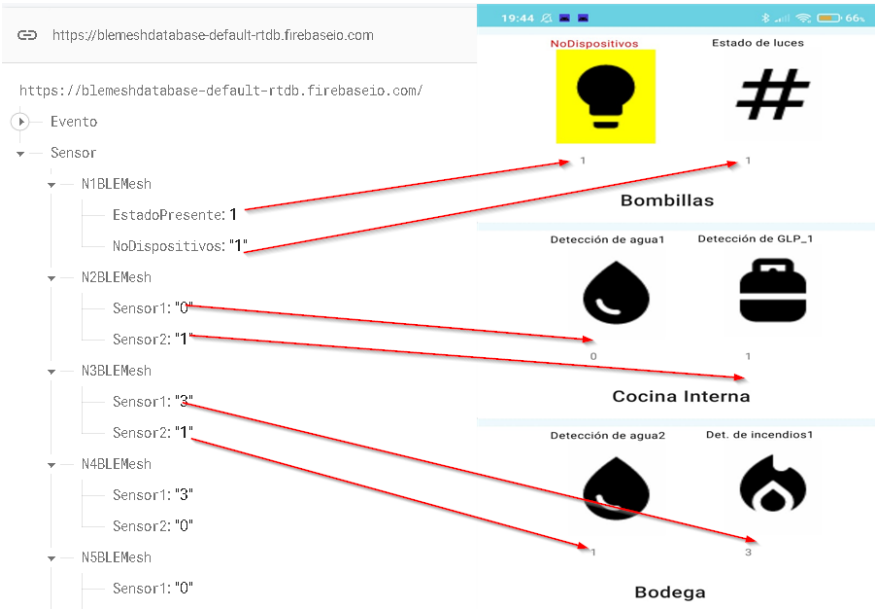
Fuente : Autoría .

4.6.9. Verificación del Envío y recibo de datos desde y hacia la plataforma BaaS

Para realizar la verificación de la plataforma BaaS donde se alojó la base de datos en tiempo real como el servicio de hosting ,es posible comparar los valores de la base de datos y

su actualización en tiempo real .La pantalla de información de sensores refleja los datos dentro de la base como se observa en la Figura 104.

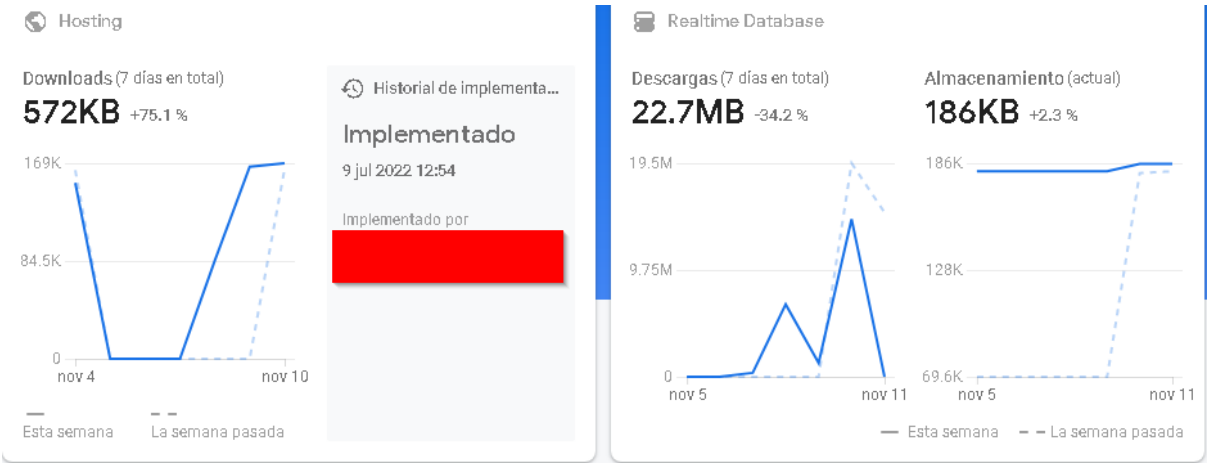
Figura 104. Comparación de los datos mostrados en la aplicación móvil con los datos en la base en la nube.



Fuente : Autoría .

Ademas también se tiene a las métricas del proyecto en Firebase ,en donde se podrá observar las descargas de la base de datos y de la página web realizada. Las métricas se pueden observar en la Figura 105

Figura 105. Métricas del proyecto creado en la plataforma BaaS



Fuente : Autoría .

4.6.10. Validación de requerimientos de arquitectura

En base a las pruebas realizadas en la presente sección es posible realizar la validación de requerimientos de arquitectura del sistema ,estos validan la correcta construcción lógica del sistema .Para realizar este análisis se usa la tabla de verificación de requerimientos de arquitectura del sistema ,se puede observar en la Tabla *35*.

Tabla 35. Verificación de requerimientos de arquitectura del sistema.

SRS					
Requerimientos de Arquitectura					
No	**Requerimiento**	**Prioridad**			**Observación**
		SI	**NO**	**N/A**	
	Requerimientos Lógicos				
SRS1	El sistema debe ser capaz de enviar y recibir datos desde y hasta al cliente en un esquema de acuerdo con el Perfil Mesh	X			
SRS2	El sistema debe funcionar en conjunto a una aplicación móvil compatible con el Perfil Mesh	X			
	Requerimientos de diseño				
SRS3	Las placas y equipos usadas en el sistema deben contar con características de software y hardware para la construcción de una red Bluetooth Mesh sin la necesidad de equipos adicionales.	X			
SRS4	El nodo central emitirá alertas de audio por cada sensor presente en el sistema.	X			
SRS5	El sistema podrá controlar la iluminación de la vivienda sin interferir con las instalaciones eléctricas actuales.	X			
SRS6	Cada módulo del sistema tendrá asociado una serie de sensores dentro de en un hub.	X			
SRS7	El proceso de cambio de sensores de un módulo puede ser realizado con el módulo en funcionamiento	X			
SRS8	Los módulos deben contar con un identificador en código Braille con la palabra sensor o similar.	X			
	Requerimientos de Hardware				
SRS9	Las placas de desarrollo para la construcción de módulos usados deben contar con conectividad Bluetooth version 4.2 o superior.	X			

SRS10	Placa de desarrollo debe contar con entradas analógicas y digitales para conexión de sensores.	X
SRS11	Sensores compatibles con las placas de desarrollo utilizadas.	X
SRS12	Los sensores de aire deben detectar principalmente GLP.	X
SRS13	El sensor de agua debe medir nivel de agua y realizar la detección sin ser introducido totalmente al medio o en su defecto el circuito de detección debe estar alejado y ser impermeable.	X
SRS14	El sensor de incendios debe estar a una distancia segura del flagelo.	X
SRS15	El sensor de corriente debe obtener los valores de forma no invasiva en el ambiente de medición	X
SRS16	El sensor de corriente podrá desacoplarse de forma fácil de un lugar de medición para localizarlo en otro	X
SRS17	El sensor de detección de personas debe tener un alcance de al menos 6 metros	X
	Requerimientos de software	
SRS18	La placa de desarrollo debe contar con soporte para librerías de los sensores a utilizar.	X
SRS19	El lenguaje de programación deberá ser basado en código abierto y sin costo.	X
SRS20	La aplicación móvil debe permitir accesibilidad para personas con discapacidad visual.	X
SRS21	Las placas de desarrollo utilizadas deben contar con un entorno de programación para el desarrollo de aplicaciones y de uso libre	X

SRS22	El lenguaje de programación usado para el desarrollo de la aplicación móvil debe ser intuitivo e integrar funciones nativas sin la necesidad de complementos adicionales	X	
SRS23	El SDK para el desarrollo de la aplicación móvil debe tener librerías o plugins que permitan administrar conexiones inalámbricas usando Bluetooth Low Energy.	X	
	Requerimientos eléctricos		
SRS24	Los dispositivos del sistema deben funcionar con una alimentación eléctrica de entre 3.3 v y 5 v D.C a excepción de las luminarias .	X	
SRS25	El sistema de alimentación eléctrica de los módulos del sistema debe ofrecer 600 mA de corriente mínima, a excepción de las luminarias	X	Se debe usar cierto tipo de cargador USB para alimentar a cada nodo

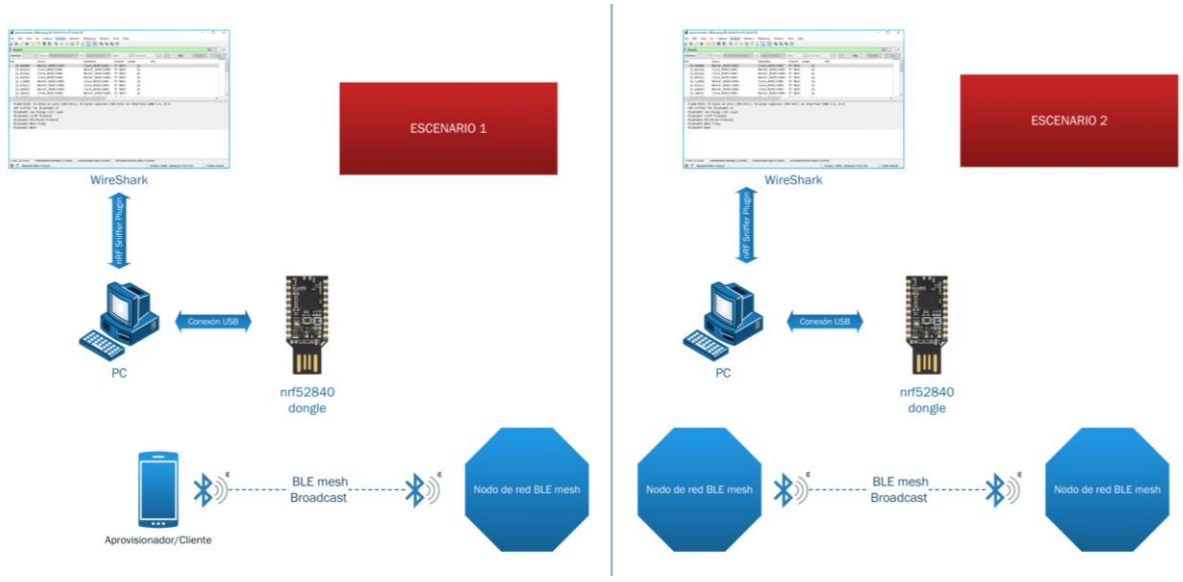
Fuente : Autoría .

4.7. Pruebas realizadas al sistema :Funcionamiento de la red de sensores

Esta sección verificara el intercambio de datos que se está realizando en la red construida, señalando que el hardware utilizado no puede capturar toda la información. Para realizar la captura de paquetes, se debe colocar el hardware como oyente de un intercambio de paquetes en un esquema peer to peer en la red. Es decir, se realizará la captura de paquetes para un determinado evento, en este evento pueden participar distintos nodos de la red incluyendo el dispositivo aprovisionador como se observa en la

Figura 106.

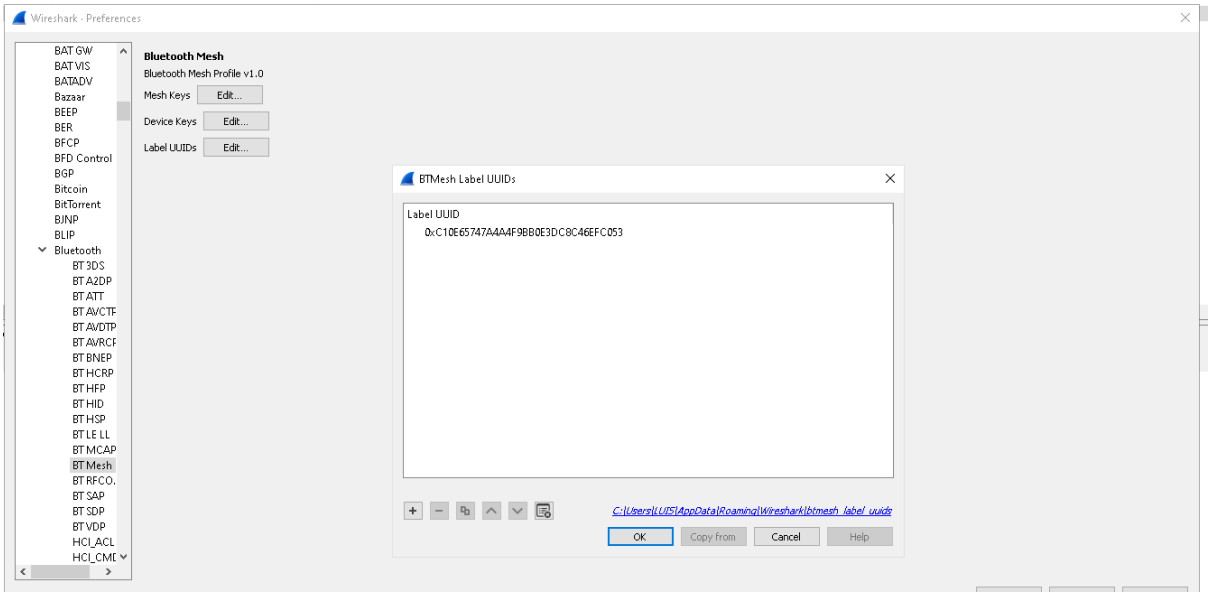
Figura 106. Captura de paquetes en la red usando el módulo nRF sniffer y Wireshark



Fuente: Autoría.

Todos los paquetes capturados están encriptados por lo que será necesario las llaves de red, aplicación y algunos casos también de dispositivos. Por lo que será necesario añadir estas llaves al software utilizado, en este caso wireshark. En la Figura 107 se observa donde se añade las claves

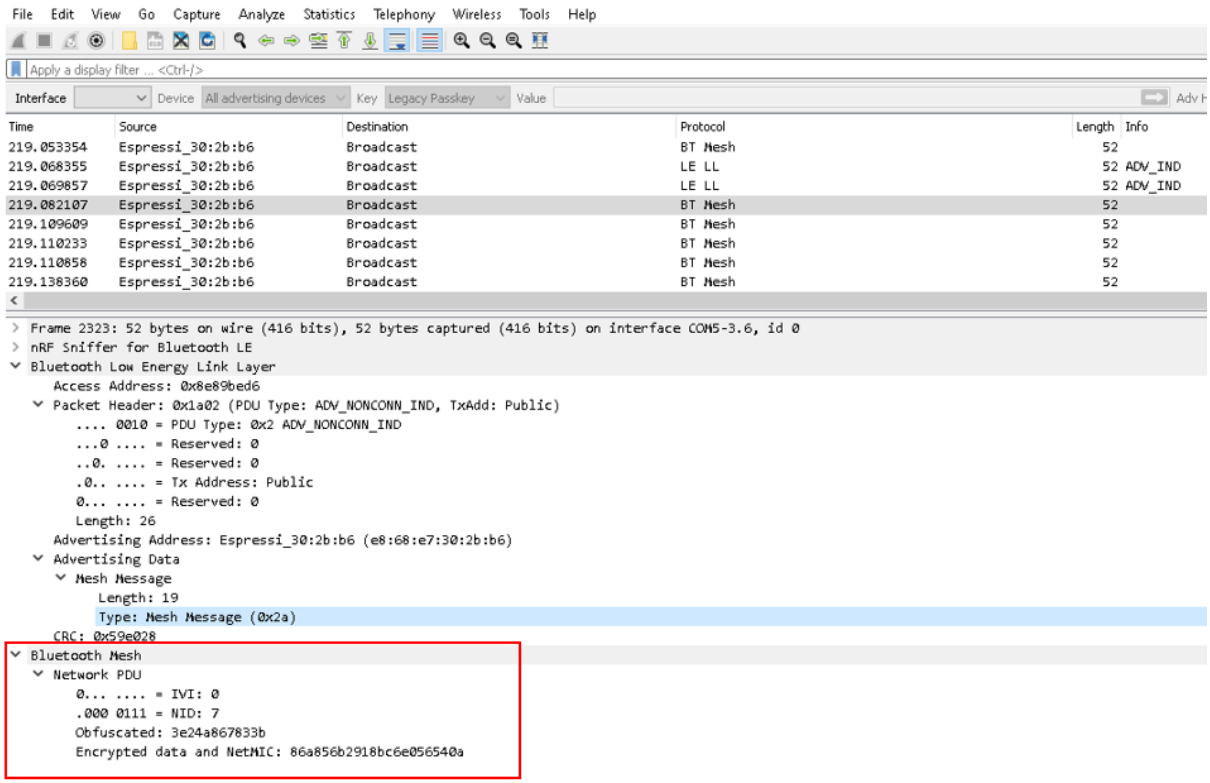
Figura 107. Parámetros para sniffer de BLE Mesh a añadir y donde añadirlos en Wireshark.



Fuente: Autoría.

Si este proceso no se realiza entonces los paquetes capturados como se observa en la Figura 108 .Hay que recalcar nuevamente que las capturas realizadas no pueden capturar todos los paquetes ni en todos los canales de transmisión. Todo esto debido a la limitación del hardware utilizado

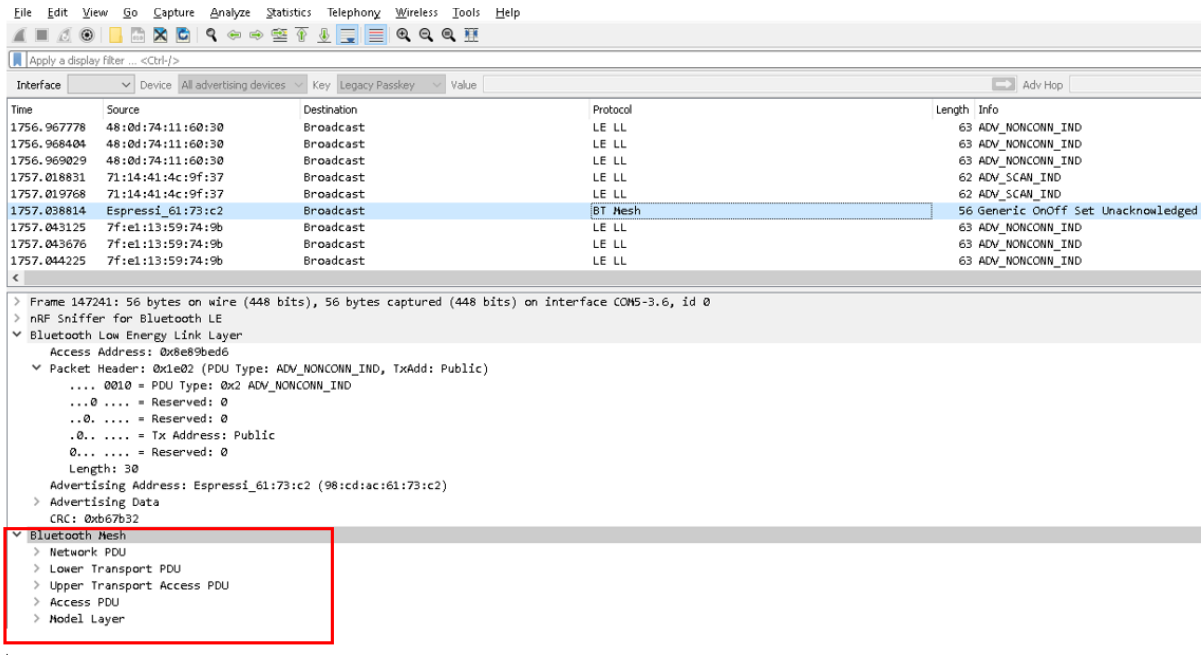
Figura 108 .Paquetes capturados antes de añadir las llaves a wireshark



Fuente: Autoría.

Mientras tanto si el proceso anterior se realiza y se añade con éxito todos los parámetros, entonces el paquete capturado tendrá los campos mostrados en la Figura 109 Notando que ya es posible observar las capas de: red, transporte, acceso y modelo. Y ademas es posible observar que tipo de paquete es en la pestaña Info. Para el ejemplo se puede observar que se trata de un paquete set en la comunicación servidor cliente utilizando el modelo On-Off.

Figura 109. Paquetes capturados después de añadir las llaves a wireshark



Fuente: Autoría.

4.7.1. Prueba 1: verificación del Aprovisionamiento

Para este apartado se comprobará los mensajes que intercambian los nodos con el aprovisionador utilizado.

4.7.1.1. Captura de paquetes entre el smartphone y nodo a aprovisionar

Dentro del proceso de aprovisionamiento participan dos actores: el smartphone que servirá como aprovisionador y un dispositivo no aprovisionado con un firmware que soporta el estándar de comunicación de red Bluetooth mesh. Para demostrar este procedimiento se utilizarán los dispositivos descritos en la *Tabla 36*. Cabe destacar que en esta sección solo se analizará el flujo de paquetes entre los dos dispositivos participantes, mas no se analizará a fondo el contenido de los mismo. Este análisis se realizará en secciones posteriores.

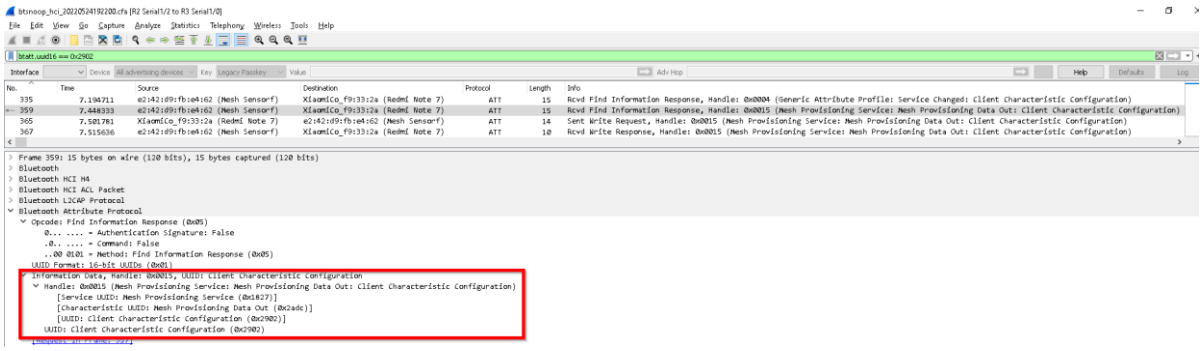
Tabla 36. Dispositivos participantes en la demostración del proceso de aprovisionamiento.

Nombre	Direccion	Rol
Xiaomi Redmi note 7	b4:c4:fc:f9:33:2a	Aprovisionador
Thingy 52	e2:42:d9:fb:e4:62	Nodo por aprovisionar

Fuente: Autoría.

El proceso inicia con el escaneo de nodos no aprovisionados ,los nodos nos aprovisionados enviaran paquetes con información para aprovisionarse a la red .En un proceso entre nodo y nodo se enviarán parámetros como UUID, tipo de OOB y otros más ,pero en este caso se debe establecer un conexión BLE y buscar el servicio Mesh Provisioning con la característica Mesh Provisioning Data Out con el UUID respectivo .En otras palabras ,el nodo puede aprovisionarse .Para observar estos paquetes se usa el filtro btatt.uuid16 == 0x2902(filtro para UUID Client Characteristic Configuration).Los paquetes capturados se pueden observar en la Figura *110*.

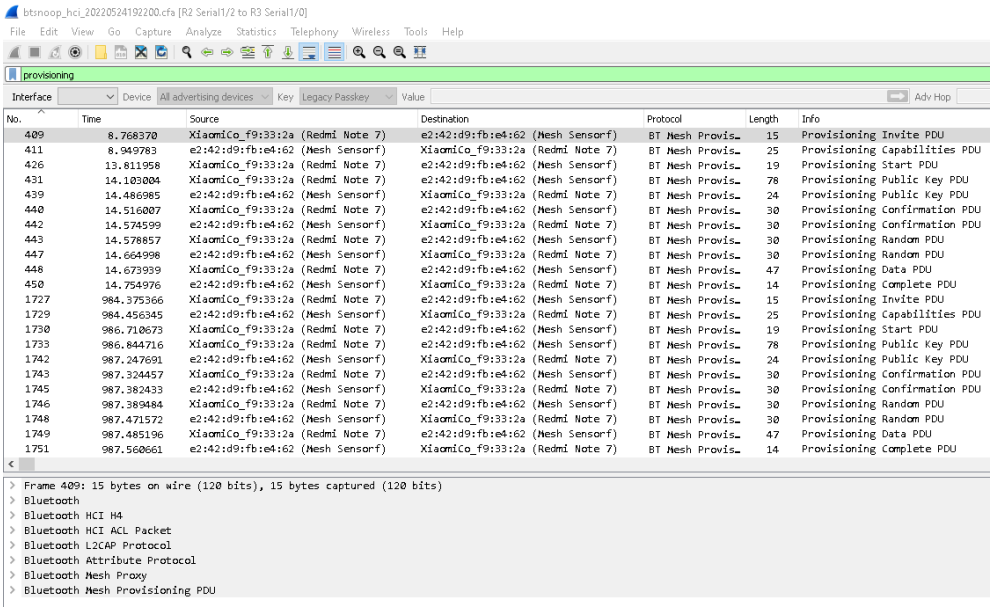
Figura 110 .Paquetes capturados en el proceso de lectura de servicios, características y UUID.



Fuente: Autoría.

Con el establecimiento de la conexión, se inicia el proceso de aprovisionamiento. Para facilitar la observación de paquetes de este proceso se utiliza el filtro *provisioning* en wireshark, este filtro desplegara todos los paquetes capturados durante el proceso. El uso y resultado de este filtro se puede observar en la *Figura 111* .

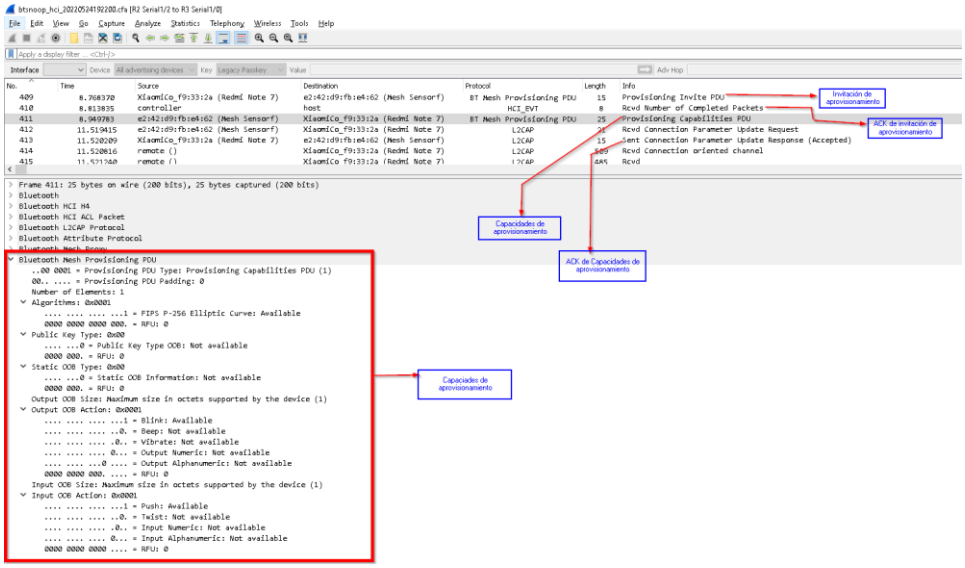
Figura 111. Paquetes capturados en el proceso de aprovisionamiento.



Fuente: Autoría.

El primer paso en el proceso en él Envío de la invitación de aprovisionamiento de parte del aprovisionador al nuevo nodo a aprovisionar. El nuevo nodo por aprovisionar enviará un ACK al aprovisionador y a continuación enviará las capacidades de aprovisionamiento que posee y de igual forma este paquete tendrá un ACK de respuesta. En la Figura 112 se observa este intercambio de paquetes. Notar que, dado que se está utilizando el smartphone como aprovisionador, se observa el uso del protocolo Proxy. También se observa las capacidades en detalle del nodo en cuestión.

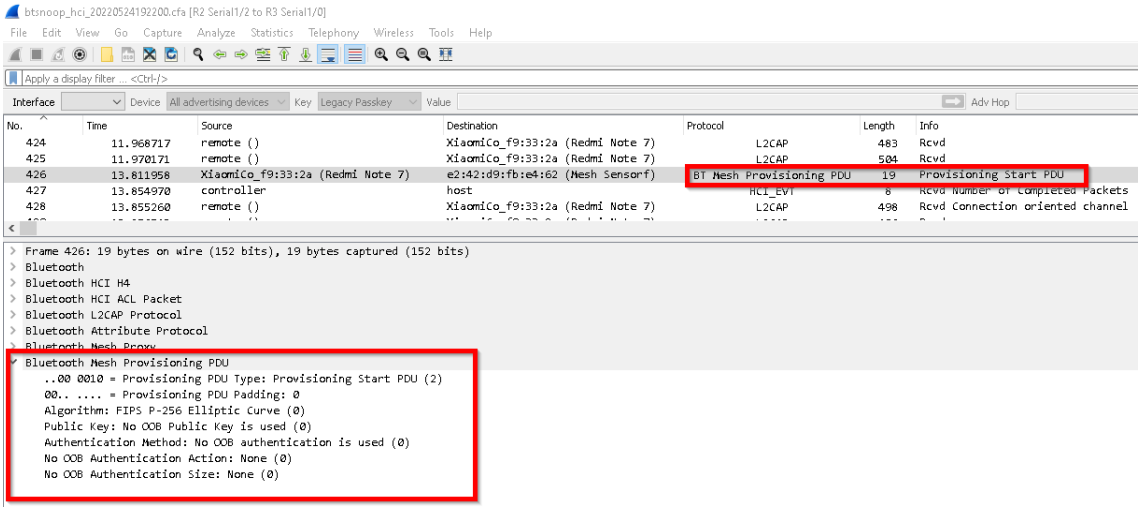
Figura 112. Captura de paquetes para Invitacion de aprovisionamiento y envío de capacidades de aprovisionamiento



Fuente: Autoría.

A continuación, el proceso de aprovisionamiento inicia. Se decide el tipo de aprovisionamiento en base a las capacidades de aprovisionamiento intercambiadas en el paso anterior. El cliente responde con un ACK al paquete correspondiente. A continuación, el aprovisionador envía la llave publica de aprovisionamiento, el nodo responde con un ACK para después reenviar la llave en cuestión, el aprovisionador confirma y en caso de que dicho valor coincida entonces envía un ACK de confirmación. En la Figura 113 se observa el paquete final en cuestión.

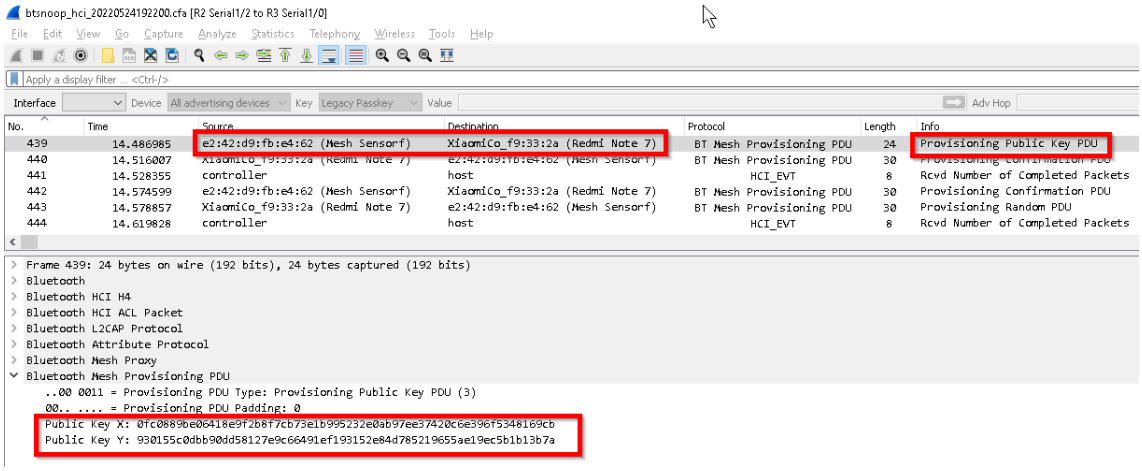
Figura 113. Captura de paquetes para el inicio de aprovisionamiento e intercambio de la llave publica de aprovisionamiento



Fuente: Autoría.

Con el intercambio de llaves públicas del nodo y aprovisionador, se utiliza El protocolo Elliptic-curve Diffie–Hellman (ECDH) para intercambiar todas las llaves de la red hacia el nodo, ya sean llaves de: red, dispositivo y aplicación. Después de intercambiar cada llave se envía un ACK entre los dos dispositivos participantes. En la Figura 114 se puede observar cómo se realiza el intercambio de llave públicas.

Figura 114 .Captura de paquetes del uso de ECDH y ACKs correspondientes.

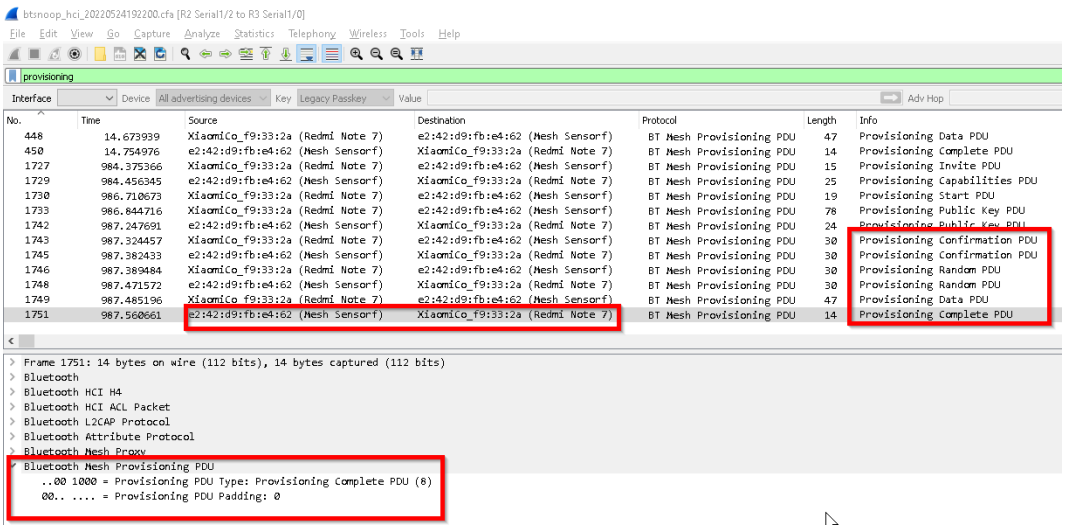


Fuente: Autoría.

Finalmente se utiliza una llave de inicio de sesión para encriptar la sesión del nuevo nodo, existen ACK de respuesta para cada uno de los intercambios y se envía el paquete de aprovisionamiento completo con el cual el proceso termina. Se cierra el enlace entre los nodos al acabar todo el proceso. El proceso final, así como el paquete final se puede observar en la Figura *115*.

El aprovisionador también puede acceder a los datos de los nodos, sin ningún tipo de configuración de publicación -suscripción.

Figura 115. Captura de paquetes de intercambio de llave de sesión



Fuente: Autoría.

4.7.1.2. Datos resultantes del aprovisionamiento en los dispositivos participantes

Se inicia con los mensajes por defecto que se muestran en los códigos subidos a las placas, estos mensajes indicaran la comprobación del almacenamiento de datos de aprovisionamiento en las placas de desarrollo utilizadas, todo este proceso se observa en la Figura 116.

En la línea 1181 se observa que al iniciar el proceso de boot del nodo, primero se verifica los datos guardados de red, si la red sigue activa entonces solo restaría conectarse nuevamente. En la línea 18321 se inicia nuevamente el aprovisionamiento a través de PB-GATT.

En las líneas de consola siguientes se puede observar los nuevos valores de: dirección, llave de red, IV índex y otros más que se asignan al nodo en el aprovisionamiento.

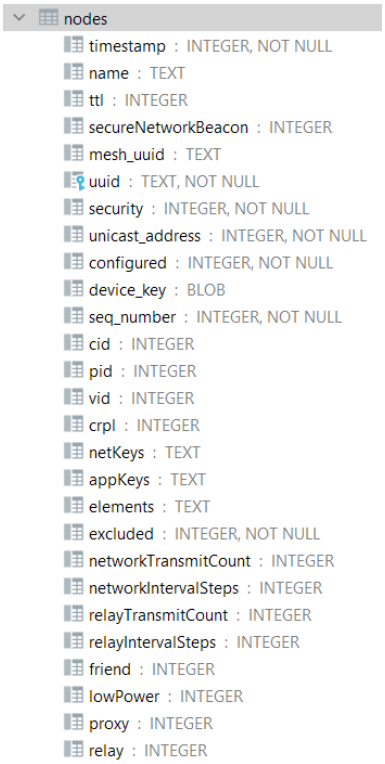
Figura 116. Aprovisionamiento de un nodo que utilizaba la placa ESP32

```
I (671) BTDM_INIT: BT controller compile version [ba07b06]
I (671) system_api: Base MAC address is not set
I (671) system_api: read default base MAC address from EFUSE
I (681) phy_init: phy_version 4670,719f9f6,Feb 18 2021,17:07:07
I (1091) EXAMPLE_NVS: Open namespace done, name "mesh_example"
I (1181) EXAMPLE: ESP_BLE_MESH_PROV_REGISTER_COMP_EVT, err_code 0
I (1181) EXAMPLE_NVS: Restore, key "onoff_client", length 6
I (1181) EXAMPLE_NVS: Restore, data: 00 00 00 00 01 17
I (1191) EXAMPLE: Restore, net_idx 0x0000, app_idx 0x0000, onoff 1, tid 0x17
I (1201) EXAMPLE: BLE Mesh Node initialized
I (1211) EXAMPLE: ESP_BLE_MESH_NODE_PROV_ENABLE_COMP_EVT, err_code 0
I (18321) EXAMPLE: ESP_BLE_MESH_NODE_PROV_LINK_OPEN_EVT, bearer PB-GATT
W (24271) BLE_MESH: No Health Server context provided
W (30531) BLE_MESH: No Health Server context provided
I (30561) EXAMPLE: ESP_BLE_MESH_NODE_PROV_LINK_CLOSE_EVT, bearer PB-GATT
I (30561) EXAMPLE: ESP_BLE_MESH_NODE_PROV_COMPLETE_EVT
I (30561) EXAMPLE: net_idx: 0x0000, addr: 0x001b
I (30561) EXAMPLE: flags: 0x00, iv_index: 0x00000000
W (36791) BLE_MESH: Composition page 255 not available
W (37331) BLE_MESH: No matching TX context for ack
I (40751) EXAMPLE: ESP_BLE_MESH_MODEL_OP_APP_KEY_ADD
I (40751) EXAMPLE: net_idx 0x0000, app_idx 0x0000
I (40751) AppKey: 72 66 42 a3 d8 b7 9e 75 89 c2 cf d8 f0 aa ae 6e
```

Fuente: Autoría.

De acuerdo a la librería para Android utilizada, los datos se guardan en una base de datos para acceder más fácilmente a los datos. Se utiliza SQLite, los datos guardados son varios pero el campo de interés es la tabla nodos. La tabla nodos poseen campos: tiempo de aprovisionamiento, nombre, TTL y otros más que se muestran en la Figura 117.

Figura 117. Campos de la tabla nodos en la librería utilizada en la aplicación móvil.



Fuente: Autoría.

En si esta tabla guarda la lista de nodos aprovisionados de la red creada, para comprobar que el proceso de aprovisionamiento ha sido completado con éxito se verifica el campo elementos de la tabla. Para el nodo aprovisionado de prueba, el contenido del campo es el siguiente

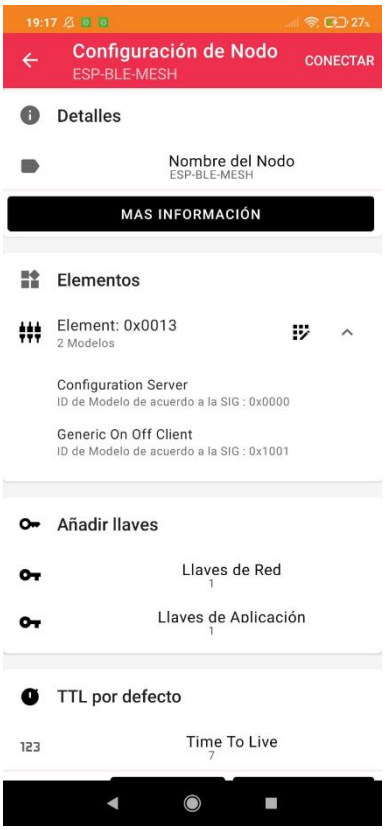
{"19":{"elementAddress":19,"locationDescriptor":0,"meshModels":{"0":{"currentScene":0,"labelUuids":[],"mBoundAppKeyIndexes":[],"mBoundAppKeys":{},"mModelId":0,"sceneNumbers":[],"subscriptionAddresses":[],"targetScene":0},"4097":{"currentScene":0,"labelUuids":[],"mBoundAppKeyIndexes":[],"mBoundAppKeys":{},"mModelId":4097,"sceneNumbers":[],"subscriptionAddresses":[],"targetScene":0}},"name":"Element : 0x0013"}}

Donde

- 19 es el número de nodo asignado después de finalizar el proceso de aprovisionamiento
- 4097 es el número decimal de modelo dentro del elemento ,corresponde al modelo Generic On off Client(0x1001)
- 0 es el número de modelo dentro del elemento ,corresponde al modelo de configuración del servidor
- 0x0013 es el identificador de elementos del nodo

Se puede comprobar estos valores en la pantalla de configuración del nodo ,donde dichos valores coinciden como se observa en la Figura *118*.

Figura 118 .Elementos y modelos del nodo recién aprovisionado.



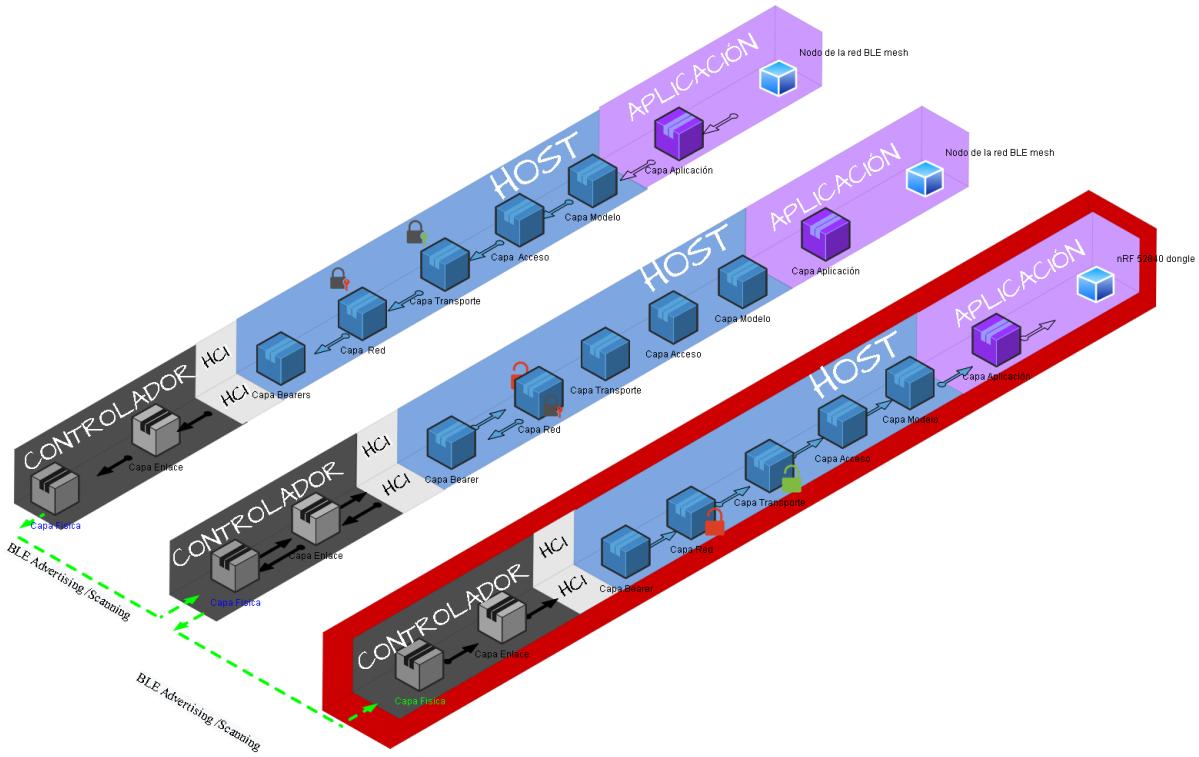
Fuente : Autoría .

4.7.2. Prueba 2 : Verificación de la Comunicación de los nodos de acuerdo al estándar de comunicación de red

La comunicación en la red se da de acuerdo al esquema cliente servidor con base al estándar utilizado ,el cliente realiza peticiones a los servidores de acuerdo al modelo utilizado pero debido a como se controla el sistema ,casi no se utiliza modelos de clientes en el sistema planteado. Solo se ha utilizado los grupos de modelos de Generic y Sensor .Cada uno de estos grupos tiene asociados varios modelos ,. Los modelos utilizan mensajes para comunicarse que dependen del tipo de modelo implementado. La mayoría de estos mensajes se pueden capturar utilizando un sniffer apropiado. Para entender los paquetes capturados de la red ,se debe entender cómo funciona el stack de Bluetooth mesh ;para ello se utilizará un gráfico de acuerdo

a las capas y los paquetes capturados en ellas ,para ello se utiliza como referencia la .Donde se nota que la captura de paquetes se debe hacer en el nRF 52840 dongle ,es decir los paquetes serán capturados en la capa más baja y subirán a la más alta; podrá existir un nodo Relay antes de la captura de paquetes respectiva.

Figura 119.Esquema de comunicación del nodo de acuerdo a las capas definidas en el estándar .



Fuente : Autoría

En la siguiente sección se demuestra la captura de los datos con la ayuda de wireshark ,donde se podrá observar todos los parámetros técnicos junto a las respectivas capturas

4.7.2.1. Generic On-off

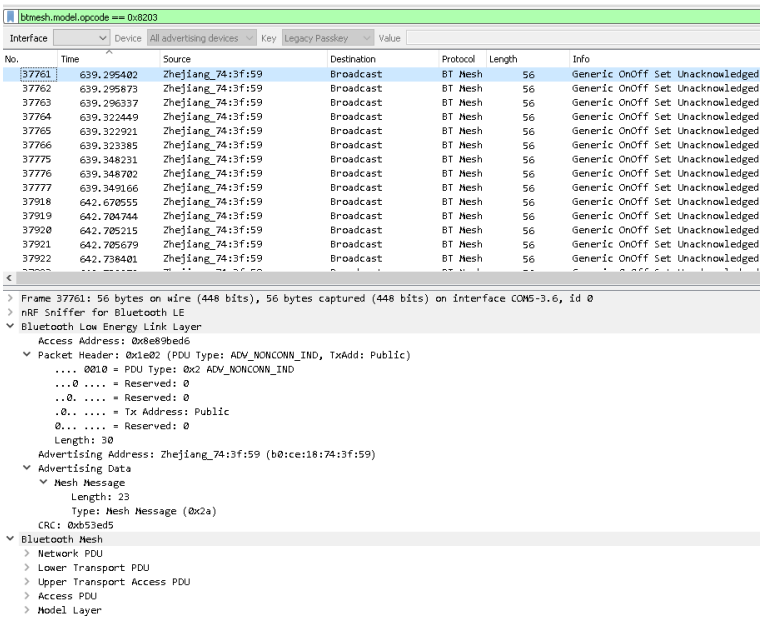
Este modelo fue implementado en el módulo de control de luces ,se implementa el modelo de servidor ,el cual tiene un código de operación de 0x82 0x0n donde n son los números 1 a 4. Cada uno de estos mensajes corresponden a una acción especifica usada en el modelo ,este modelo se puede implementar usando o no un ACK de respuesta.

Para el cambio de estado de luces usando este modelo se utiliza el mensaje de Generic OnOff set Unacknowledged ,este mensaje se utiliza cuando no se recibe una respuesta del

servidor .A continuación se analiza un paquete capturado de este tipo en el cual se recibe el paquete .

En la Figura 120 se observa el paquete con todos los campos disponibles de acuerdo al sniffer utilizado y la version soportada por el software.

Figura 120. Captura de un mensaje Generic OnOff set Unacknowledged



Fuente : Autoría .

Capa BLE y Capa Bearer

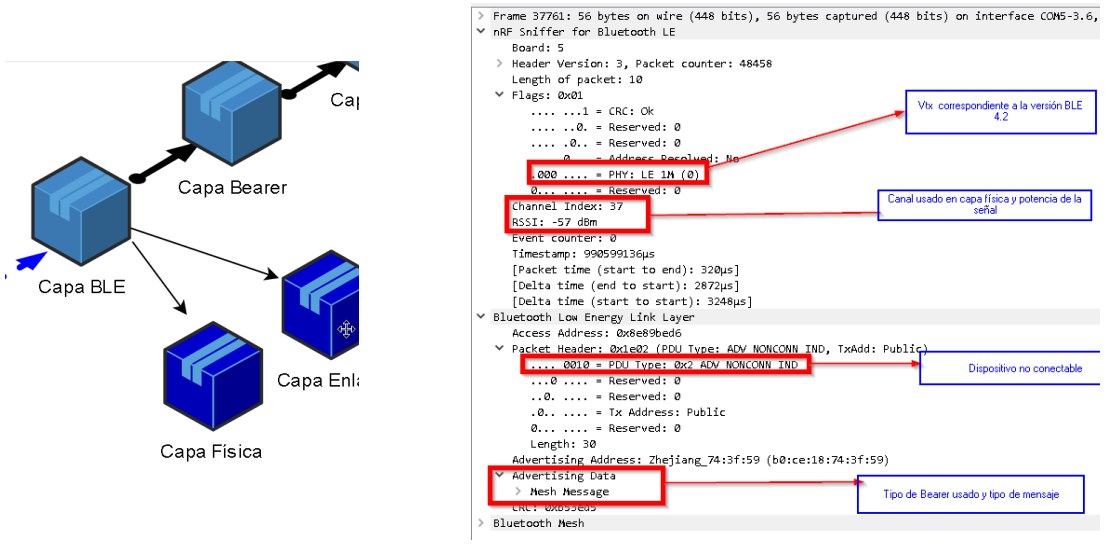
De acuerdo al análisis de paquetes planteado ,se inicia en la capa más baja del paquete es decir en la capa perteneciente a BLE ;donde se encuentran dos subcapas : la capa física y la capa de enlace .Esta capa funciona bajo los estándares definidos por la tecnología de comunicación inalámbrica Bluetooth Low Energy. En la siguiente capa se tiene la capa Bearer que ya es propia del estándar Mesh ,y define como se envían los paquetes BLE a los diferentes nodos .En conclusión ,se estará utilizando Advertising Bearer dado que se está realizando la comunicación entre dos nodos de la red.

En la captura de wireshark correspondiente se podrá observar el canal de comunicación ,siendo el canal 36 .También se puede observar la intensidad de la señal del paquete capturado la cual es -57[dBm].Otro dato importante es el valor de velocidad de transmisión máxima del

medio físico, la cual es de 1[Mbps] correspondiente a la version de Bluetooth 4.2; todos estos parámetros correspondientes a capa Física .

Por otro lado se tiene un apartado completo para la capa enlace ,los parámetros a recalcar son : se trata de un paquete del tipo ADV_NONCONN_IND,el cual está anunciando que se trata Mesh Message utilizando Advertising Bearer .Con esta información se puede concluir que :el paquete en capa enlace es un paquete capturado de un dispositivo no conectable que envía un mensaje a una red mesh ,dicho dispositivo está usando Advertising Bearer por lo que el paquete se está enviando a otro dispositivo que soporta el estándar de comunicación de red directamente.

Figura 121. Mensaje Generic OnOff set Unacknowledged en capa física ,enlace y bearer



(a) (b)

Nota: (a) Ubicación del paquete en la arquitectura de red ,(b) Paquete capturado
Fuente : Autoría .

Capa Red

Esta capa ya pertenece al estándar de comunicación de red Bluetooth mesh ,define el formato de paquetes que envía y recibe de la capa de transporte ,para esta operación se debe usar la llave de red .Por lo que los paquetes se encriptan en origen y se desencriptan en destino ,siendo origen el nodo que genera los paquetes ;según el estándar de comunicación ,siempre serán nodos servidores .Mientras destino serán una direccion de broadcast dado que el estándar de comunicación de red utiliza inundación de paquetes como método de Networking .

Los campos del PDU son los siguientes :IV índex, este campo indica una operación de autenticación para una subred cuando el número de secuencia de la red está alcanzando el límite. Dicho valor suele estar en 0 al inicio de operación de la red .

El siguiente campo es el NID ,este valor está asociado a los valores de seguridad de la llave de red ;en si sirve para identificar los parámetros usados para autenticación usando la llave de red. El campo siguiente corresponde a la identificación del tipo de mensaje ,donde se tiene mensaje de control o mensaje de acceso .A continuación se tiene el valor de TTL ,este valor es utilizado cuando el nodo ha sido configurado como Relay ,de acuerdo al valor se puede conocer si el paquete ha pasado, va a paso o definitivamente no va a ser retransmitido por un Relay. Finalmente se tiene :un numero de secuencia, las direcciones de origen y destino (16 bits cada una),y la carga útil del paquete.

Con toda esta información complementaria es posible analizar el paquete capturado ,en el campo de IV Index si tiene un valor de 0 es decir :los números de secuencia de la red aún están lejos de acabarse para el nodo. En el campo de NID se tiene un valor de 113 ,el cual será único para este nodo ;más adelante se podrá comprobar que cambiará para otro nodo y modelo. En el campo de Control se tiene que el paquete capturado es del tipo de mensaje de acceso .

Continuando se tiene que el valor del campo de TTL es de 5 ,según la documentación oficial este valor corresponde a que el paquete puede ser retransmitido y debe ser retransmitido. En el campo de numero de secuencia se tiene un valor de 79, el cual es propio de este intercambio y se mantendrá el número de veces especificado en la retransmisión del mensaje, así como en cada canal por donde se envía el mensaje.

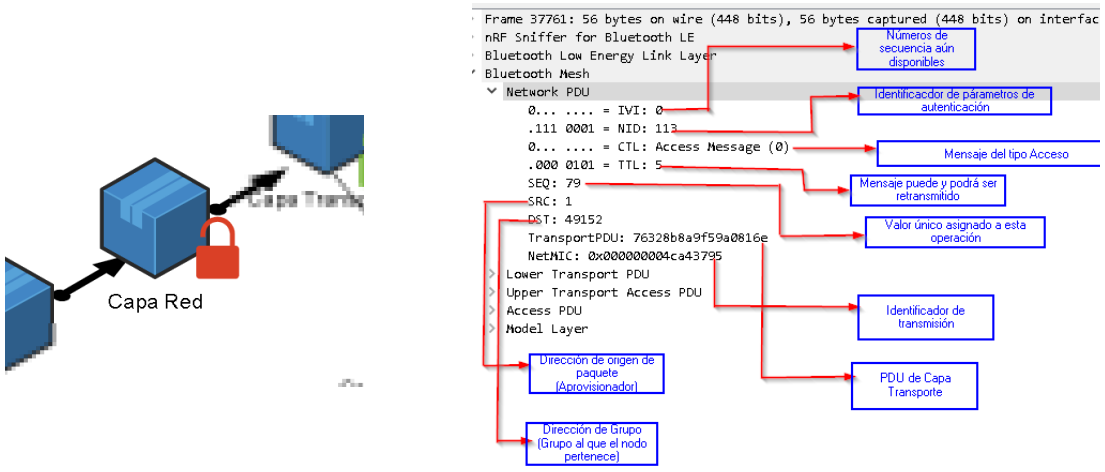
Los campos de direcciones de origen y destino tienen un valor de 1 y 49152, estos valores deben traducirse a hexadecimal para obtener el respectivo valor. La direccion de origen en este caso corresponde al nodo aprovisionador ,por defecto se ubica en la direccion 0x0001.

Mientras la direccion de destino es la direccion 0xC000, este tipo de direccion pertenece a un grupo .

Finalmente se tiene los campos de Transport PDU y NetMIC ,en consecuencia, se tratan de un campo brindado por la capa inferior y un identificador de la transmisión realizada para ese mensaje respectivamente.

Todo esto se puede observar a detalle en la Figura 122.

Figura 122.Capa de red para paquete capturado para un mensaje Generic OnOff set Unacknowledged



(a) (b)

Nota: (a) Ubicación del paquete en la arquitectura de red ,(b) Paquete capturado

Fuente : Autoría .

Capa Transporte y Acceso

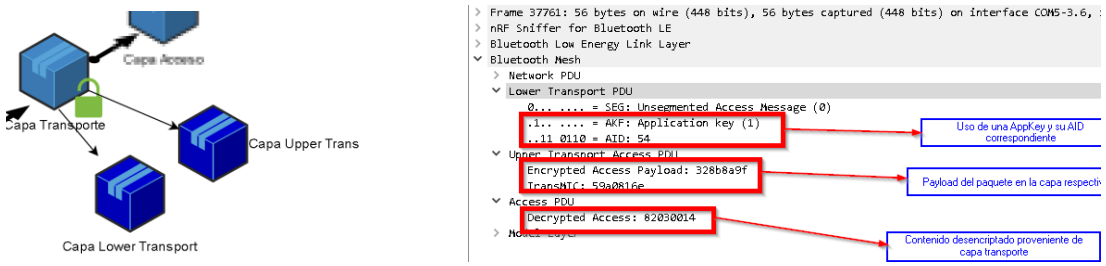
Esta capa se subdivide en dos subcapas ,se tiene las capas de : Lower y Upper Transport Upper, y Access. En la capa Lower transport ,en esta capa se realiza la intercomunicación con otros nodos en esta misma capa .En la captura realizada se puede observar los tres campos : Unsegmented Access Message ,el cual al tener el valor de 0 indica que se trata de un paquete que no ha sido segmentado; Application Key ,el cual al tener el valor en 1 indica que el paquete ha sido vinculado con una llave de aplicación en capas superiores ;AID ,el cual es el identificador de llave de aplicación, este valor es único para la llave de aplicación vinculada al modelo.

En la siguiente capa se tiene a la Upper Transport Acces ,esta capa proporciona todos los datos para realizar la encriptación de los paquetes, con la ayuda de la llave de aplicación o llave de dispositivo del modelo correspondiente; en el paquete capturado se puede observar los dos campos existentes en esta capa ,los cuales son :Payload (Carga Útil de la capa de Acceso) y TransMIC (Comprobación de integridad de mensaje para capa transporte).El Payload es dado por la capa de acceso y el valor de TransMIC es usado para comprobar que el Payload antes mencionado no se ha alterado.

La capa de acceso tiene dentro los campos analizados en la capa modelo ,ya que esta encapsula dichos campos. Wireshark no permite observar todos los parámetros de esta capa, pero en esta capa recibe los paquetes encriptados y realiza el proceso de desencriptación ,para este caso se puede observar que el PDU ha sido desencriptado y el valor resultante.

Todo esto se puede observar en la Figura 123.

Figura 123. Capas de acceso ,Lower y Upper Transport paquete capturado para un mensaje Generic OnOff set Unacknowledged



(a) (b)

Nota: (a) Ubicación del paquete en la arquitectura de red ,(b) Paquete capturado
Fuente : Autoría .

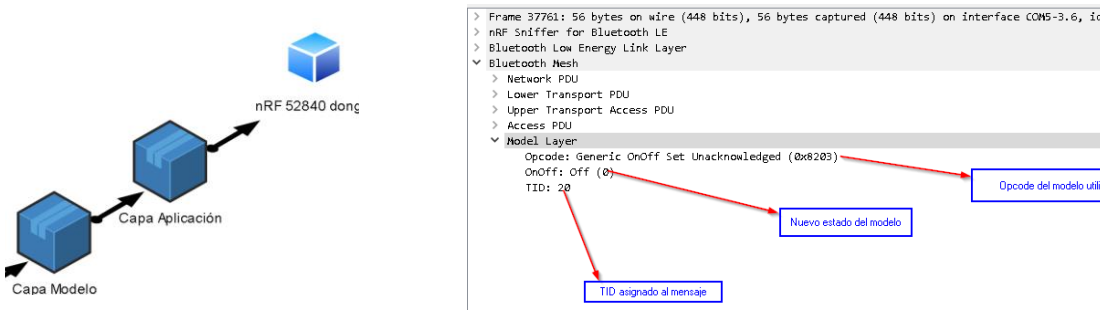
Capas Modelo y Aplicación

Se termina el análisis en la capa más alta ,en esta capa se podrá observar el opcode (código de operación) 0x8203 ,el cual corresponde a un mensaje de Generic OnOff set Unacknowledged. También se tiene el nuevo valor dado en este mensaje que para este caso es 0 ,es decir se apagó la bombilla .Finalmente se tiene un valor de TID (identificador de transacción ,el cual es usado para cada una de las operaciones de la red para evitar repeticiones. En los posteriores análisis se podrá notar que este valor no se repite.

La capa de aplicación define como se usan los modelos ,es decir que será la aplicación móvil. La aplicación móvil envía el mensaje para apagar el SmartBuld y el módulo nRF 52840 captura esta operación.

Los campos explicados, así como las capas donde se realizó el análisis se observan en la Figura 124.

Figura 124. Capa de modelo del paquete capturado para un mensaje Generic OnOff set Unacknowledged



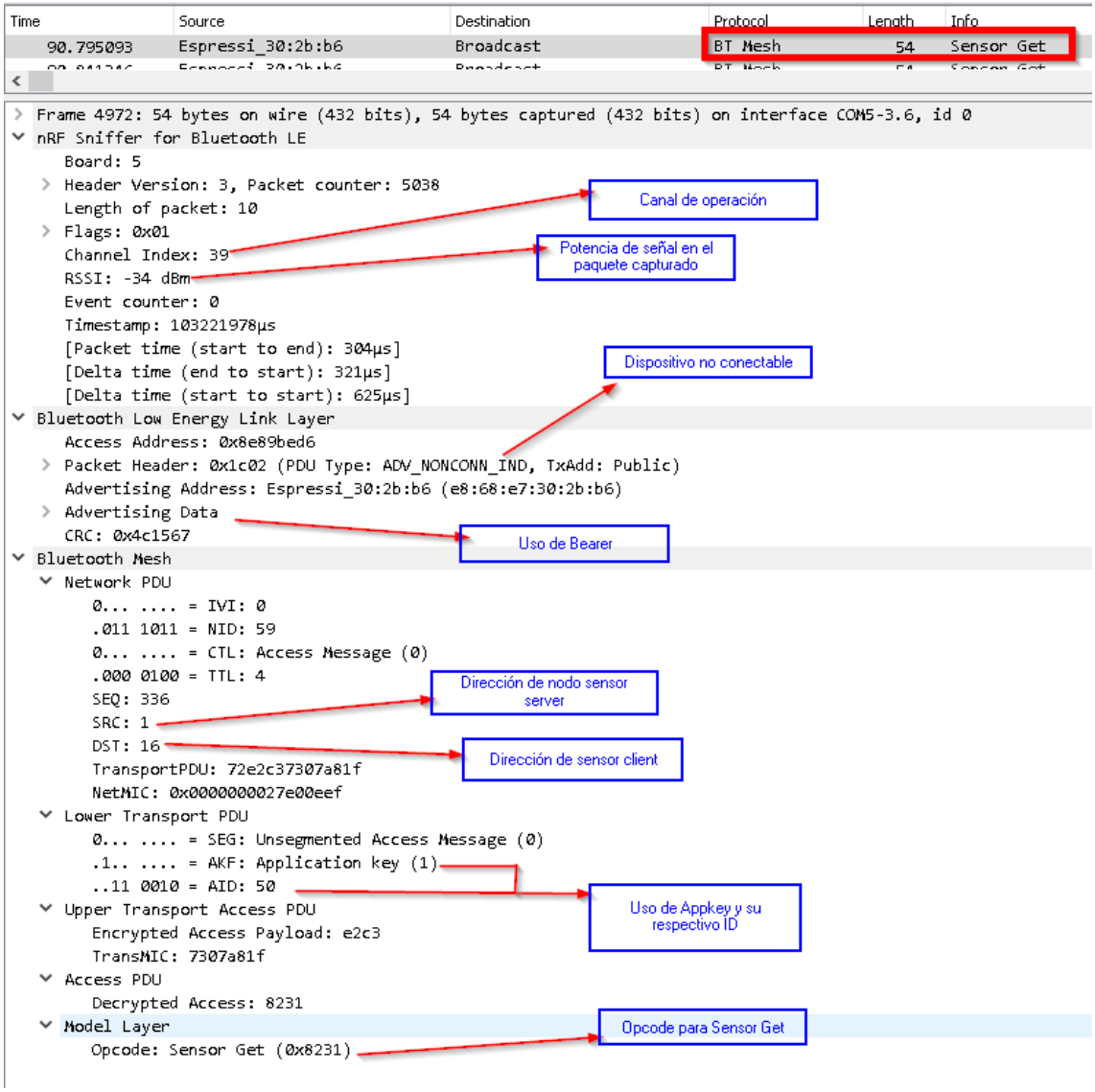
(a) (b)

Nota: (a) Ubicación del paquete en la arquitectura de red ,(b) Paquete capturado
Fuente : Autoría .

4.7.2.2. Sensor

Los paquetes del modelo sensor serán similares a los capturados por el modelo Generic OnOff. Los modelos se definen en capas superiores por lo que los paquetes capturados para el modelo de sensor tendrán sus propios valores para bearer ,red ,transporte y acceso pero tendrán obviamente la misma estructura .La mayor diferencia radicará en el modelo en capa modelo con su respectivo opcode .Esto se puede observar en la Figura 125.Donde se utiliza el opcode 0x8231 correspondiente a un mensaje sensor Get, también se podrán observar otros valores donde se puede comprobar como son utilizados para identificar :llaves ,transmisiones y otros justo como se explicó en la sección anterior.

Figura 125. Captura de paquete para el modelo sensor



Fuente : Autoría .

4.7.3. Función Proxy

La función proxy siempre se debe implementar cuando se utiliza un dispositivo externo como aprovisionador ,las placas de desarrollo utilizadas permiten activar esta función en los códigos de implementación ,como se observa en la Figura 126 donde se tiene los roles activados para cada una de las placas de desarrollo utilizadas.

Figura 126. Configuración de roles de nodos en las placas de desarrollo utilizadas .

```c
static esp_ble_mesh_cfg_srv_t config_server = {
    .relay = ESP_BLE_MESH_RELAY_ENABLED,
    .beacon = ESP_BLE_MESH_BEACON_ENABLED,
#if defined(CONFIG_BLE_MESH_FRIEND)
    .friend_state = ESP_BLE_MESH_FRIEND_ENABLED,
#else
    .friend_state = ESP_BLE_MESH_FRIEND_NOT_SUPPORTED,
#endif
#if defined(CONFIG_BLE_MESH_GATT_PROXY_SERVER)
    .gatt_proxy = ESP_BLE_MESH_GATT_PROXY_ENABLED,
#else
    .gatt_proxy = ESP_BLE_MESH_GATT_PROXY_NOT_SUPPORTED,
#endif
    .default_ttl = 7,
    /* 3 transmissions with 20ms interval */
    .net_transmit = ESP_BLE_MESH_TRANSMIT(2, 20),
    .relay_retransmit = ESP_BLE_MESH_TRANSMIT(2, 20),
};
```

(a)

```
# Bluetooth mesh configuration
CONFIG_BT_MESH=y
CONFIG_BT_MESH_RELAY=y
CONFIG_BT_MESH_FRIEND=y
CONFIG_BT_MESH_ADV_BUF_COUNT=13
CONFIG_BT_MESH_TX_SEG_MAX=10
CONFIG_BT_MESH_PB_GATT=y
CONFIG_BT_MESH_GATT_PROXY=y
CONFIG_BT_MESH_PROXY_USE_DEVICE_NAME=y
CONFIG_BT_MESH_DK_PROV=y

# Bluetooth mesh models
CONFIG_BT_MESH_SENSOR_SRV=y
# Temperature and Humidity Sensor
CONFIG_I2C=y
CONFIG_HTS221=y
CONFIG_HTS221_TRIGGER_NONE=y
```

(b)

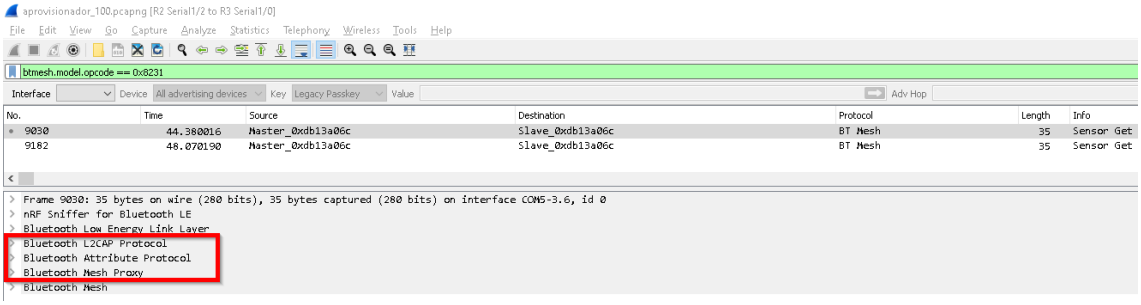
Nota : (a) Configuración de la función proxy en el IDE ESP-IDF ,(b) Configuración de la función proxy en el nRF Connect SDK

Fuente : Autoría .

Para demostrar el funcionamiento de la función proxy se hace uso de un nodo con la función Sensor Server, el smartphone aprovisionador y el sniffer de red. Para esta demostración se usará el mensaje Sensor Get para el modelo Sensor server con el opcode 0x8231. El paquete capturado se puede observar en la Figura 127 ,notar que existen tres nuevas capas : L2CAP Protocol, ATT y Mesh Proxy.

El protocolo L2CAP y el protocolo ATT corresponden capas de la arquitectura de Bluetooth Low Energy en el bloque controlador ,este bloque es reemplazado por las capas estudiadas de Bluetooth mesh .Mientras que el campo Bluetooth mesh proxy corresponde a un servicio GATT propio cuando se utiliza un nodo proxy en una red de este tipo. En consecuencia, se puede concluir que se está encapsulando paquetes de la red Mesh a través de un perfil GATT usando el bloque host de la arquitectura de Bluetooth Low Energy. El paquete capturado en cuestión se puede observar en la Figura 127.

Figura 127. Captura de paquetes para la solicitud de Sensor Get

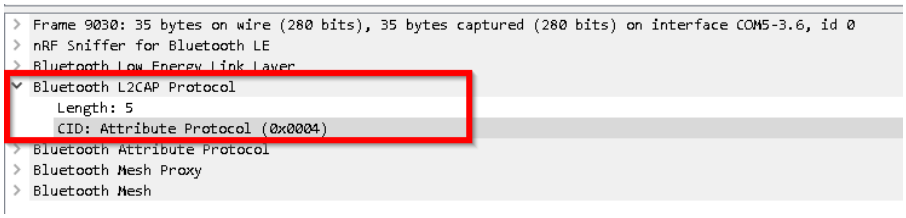


Fuente : Autoría .

En la sección de *Comunicación de los nodos de acuerdo al estándar de comunicación de red* se realizó un análisis del intercambio de paquetes de la red y se concluyó que cada paquete tendrá diferentes valores para las capas de bearer ,red ,transporte y acceso, pero en si la estructura será igual obviamente .De hecho en la Figura *125* se puede observar la estructura de un paquete de modelo Sensor .

Como consecuencia se analizará los nuevos datos ,se inicia con los protocolos L2CAP y ATT. El protocolo L2CAP funciona de tal forma que se asegura que un componente del bloque host utilice correctamente el protocolo asignado en capas superiores, así como se encarga de tareas de segmentación .Por lo que para este caso en específico ,se tiene un valor de CID(Identificador de canal) de 0x00004 correspondiente a ATT(Atribute protocol) .Es decir en capas superiores se utilizará ATT en el servidor GATT .Los campos capturados en cuestión se pueden observar en la Figura 128.

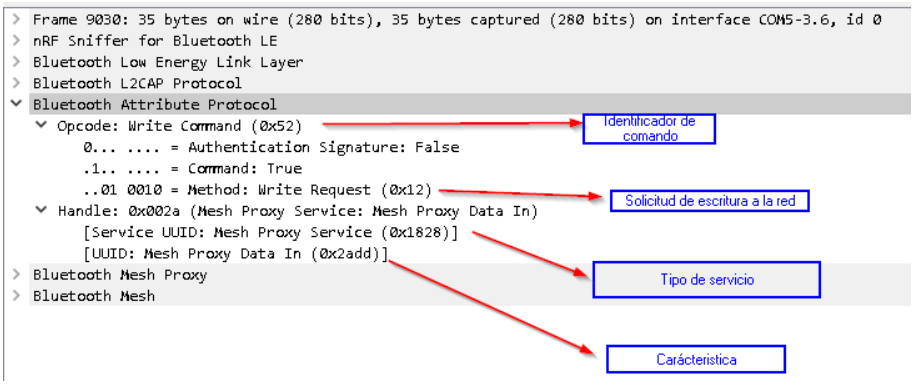
Figura 128. Protocolo L2CAP en la comunicación aprovisionador -nodo para el mensaje sensor get



Fuente : Autoría .

El protocolo ATT solamente guarda atributos ,es el trabajo de protocolos de capas superiores como se procesa ,agrupa dichos atributos .Es decir solo se define el atributo de tal forma que solo lo identifica ,no se comprueba los datos contenidos .Por ello en este caso se tiene una operación de solicitud de escritura mediante el servicio de Mesh Proxy Service usando la característica Mesh Proxy Data In ,dichos campos se pueden observar en la captura de paquetes de la Figura 129.

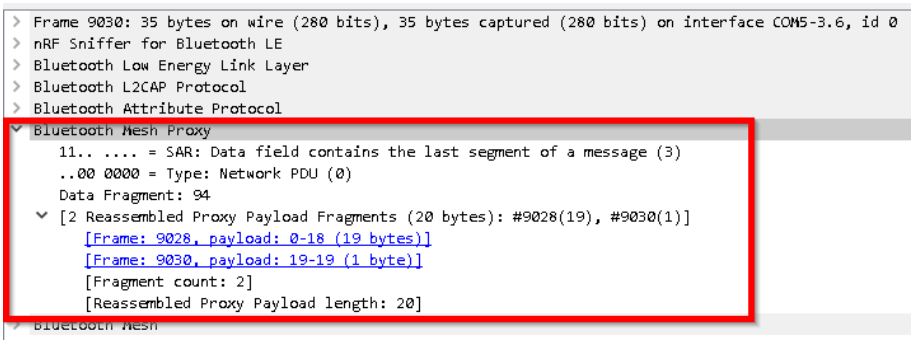
Figura 129. ATT protocol en la comunicación aprovisionador -nodo para el mensaje sensor get



Fuente : Autoría .

En base al campo de ATT ,se sabe que el cliente realiza una solicitud de escritura a la red mediante el servicio de Mesh Proxy .Por lo que se debe generar un paquete de Bluetooth Mesh Proxy y enviarlo a la red para que sea procesado por el bloque Controller .En la Figura *130* se puede observar el paquete de red capturado.

Figura 130. Mesh proxy en la comunicación aprovisionador -nodo para el mensaje sensor get

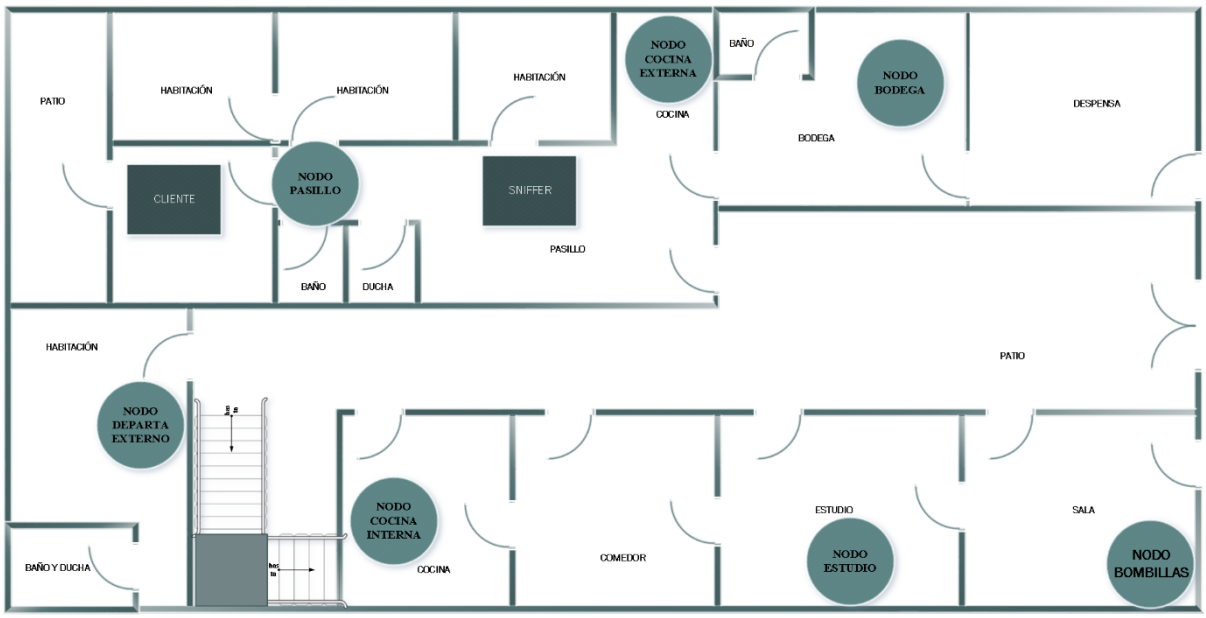


Fuente : Autoría .

4.8. Pruebas realizadas al sistema :Rendimiento y alcance de la red en el entorno de implementación

Las pruebas realizadas para comprobar el rendimiento y alcance de la red se realizaron en una vivienda de trescientos metros cuadrados (25 metros de largo por 12.5 metros de ancho) ,la distribución de habitaciones y dispositivos se pueden observar en la Figura *131*.

Figura 131. Distribución de nodos en la zona de implementación .



Fuente : Autoría .

Las pruebas por realizarse se harán con el sniffer de red ,la ubicación de este puede observarse en la Figura *131*,esta es una ubicación referencial dado que el sniffer se deber ubicar cerca del smartphone que controla la red. De igual forma es necesario conocer las direcciones físicas y de red de los nodos después de terminar el proceso de aprovisionamiento .Estos datos se pueden observar en la Tabla 37 .

Tabla 37. Direcciones físicas y de red de los nodos en el sistema.

Identificador de habitación	Identificador de nodo	Direccion MAC	Direccion Unicast(Hexadecimal)	Direccion unicast(Decimal)
----------	Aprovisionador	1C:80:DD:BA:CB:0F	0001	1
Bombillas	N1BLEMesh	B0:CE:18:74:EF:59	0043	67
Cocina interna	N2BLEMesh	98:CD:AC:61:73:C2	0047	71
Bodega	N3BLEMesh	98:CD:AC:61:73:4E	0049	73
Cocina externa	N4BLEMesh	E8:68:E7:30:2B:B6	0048	72
Estudio	N5BLEMesh	24:0A:C4:AF:E6:0A	004A	74
Pasillo	N6BLEMesh	E2:42:D9:FB:E4:62	004B	75
Departamento externo	N7BLEMesh	C1:06:97:A6:88:44	0040	64

Fuente : Autoría .

También se debe definir algunos parámetros adicionales .Los parámetros usados se pueden observar en la Tabla 38.

Tabla 38.Parámetros generales para realizar las mediciones

Parámetros	Valor
TTL por default	7 (10 para el N1BLEMesh)
BLE radio Throughtput (Bluetooth v4.2)	1 Mbps
Retransmisión por defecto (Network)	3 con intervalos de 20 ms
Retransmisión por defecto (Relay)	3 con intervalos de 20 ms

Fuente : Autoría .

Las pruebas de sensores se realizarán sensores usando las adecuadas propiedades de dispositivos Bluetooth mesh, se usarán las siguientes :

- 0x0059 correspondiente a un sensor que mide voltaje de entrada.
- 0x005D correspondiente a un sensor que mide voltaje de salida.
- 0x0077 correspondiente a un sensor que mide la cantidad presente de CO2.
- 0x004D correspondiente a un sensor que mide la presencia .
- 0x0069 correspondiente al tiempo que paso desde que se detectó presencia .

Todas las propiedades mencionadas definen un tamaño de 1 o 2 octetos en su carga útil ,por lo que se enviará una cantidad de datos pequeña a la red ,justo como se define en el estándar de comunicación de red para que la red funcione de forma óptima.

Para calcular el RTT ,se considerará la diferencia de tiempo entre el primer paquete enviado con una petición get y el primer paquete recibido con el estado del sensor al que se realizó la petición get.

4.8.1. RTT de la red construida

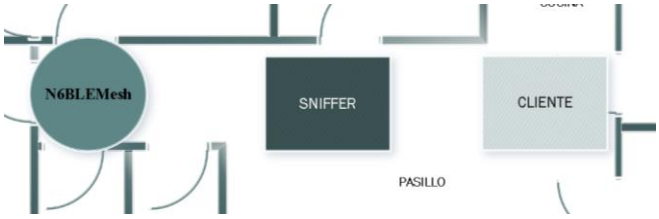
El valor de RTT puede calcularse mediante las capturas realizadas con wireshark ,se usará el modelo de sensor .El modelo de sensor tiene dos tipos de mensajes que serán útiles para estas pruebas ,el mensaje Sensor Get que produce una respuesta Sensor status .

Con las consideraciones previas expuestas ,se procede a realizar dos pruebas bajo dos tipos de respuestas esperadas, las cuales se detallan a continuación .

4.8.1.1. RTT sin ningún salto de red

Para esta prueba se realiza una petición de Sensor Status a un Sensor server de acuerdo a la .Se puede notar que la petición es de 1 a 1 y a una distancia de menos de 2 metros, por lo que el RTT será muy bajo . El esquema utilizado se muestra en la *Figura 132*.

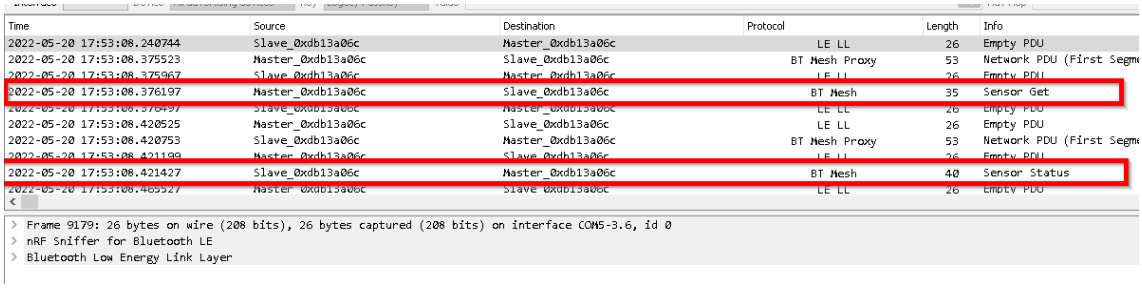
Figura 132. Prueba realizada para obtener el valor RTT en un caso ideal.



Fuente : Autoría .

Para obtener el tiempo se usará el sniffer y la herramienta wireshark ,según los valores configurados se tenía un TTL por default de 7 ,es decir que si los datos se transmiten como el esquema de la Figura *132*. :el valor no deberá disminuir ;esta condición se debe comprobar mensaje de sensor status. Los paquetes capturados para el intercambio de mensajes se pueden observar en la Figura *133*.

Figura 133. Mensajes Sensor Get y sensor Status para obtener el RTT ideal capturados en wireshark



Time	Source	Destination	Protocol	Length	Info
2022-05-20 17:53:08.240744	Slave_0xdb13a06c	Master_0xdb13a06c	LE LL	26	Empty PDU
2022-05-20 17:53:08.375523	Master_0xdb13a06c	Slave_0xdb13a06c	BT Mesh Proxy	53	Network PDU (First Segm
2022-05-20 17:53:08.375967	Slave_0xdb13a06c	Master_0xdb13a06c	LE LL	26	Empty PDU
2022-05-20 17:53:08.376197	Master_0xdb13a06c	Slave_0xdb13a06c	BT Mesh	35	Sensor Get
2022-05-20 17:53:08.376497	Slave_0xdb13a06c	Master_0xdb13a06c	LE LL	26	Empty PDU
2022-05-20 17:53:08.420525	Master_0xdb13a06c	Slave_0xdb13a06c	LE LL	26	Empty PDU
2022-05-20 17:53:08.420753	Slave_0xdb13a06c	Master_0xdb13a06c	BT Mesh Proxy	53	Network PDU (First Segm
2022-05-20 17:53:08.421193	Master_0xdb13a06c	Slave_0xdb13a06c	LE LL	26	Empty PDU
2022-05-20 17:53:08.421427	Slave_0xdb13a06c	Master_0xdb13a06c	BT Mesh	40	Sensor Status
2022-05-20 17:53:08.465527	Master_0xdb13a06c	Slave_0xdb13a06c	LE LL	26	Empty PDU

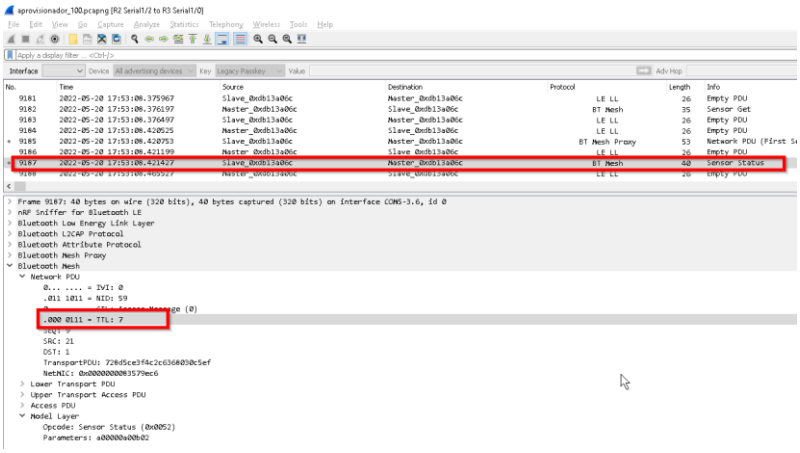
> Frame 9179: 26 bytes on wire (208 bits), 26 bytes captured (208 bits) on interface COM5-3.6, id 0
> nRF Sniffer for Bluetooth LE
> Bluetooth Low Energy Link Layer

Fuente : Autoría .

Como primer valor de verificación se utilizará al tiempo de captura de los dos paquetes ,el tiempo de captura del paquete del mensaje Sensor Get se da a las 17:53:08.376497 mientras el paquete de Sensor Status se obtiene a las 17:53:08.421427. Realizando la diferencia de estos dos valores se obtiene el valor de 44.93 milisegundos ,siendo este el valor de tiempo que se demora el servidor de sensor en responder una petición de obtención de valores .Para comprobar que se trata de un paquete sin saltos se analiza el valor de TTL del paquete del

mensaje Sensor Status ,en la Figura *134* se puede observar que definitivamente el valor de TTL se mantiene en 7.

Figura 134 .Comprobación del valor de TTL



Fuente : Autoría .

En conclusión, el paquete se demora 44.93 milisegundos en llegar de un nodo a él aprovisionador ,este tiempo puede ser usado también como el tiempo de salto .

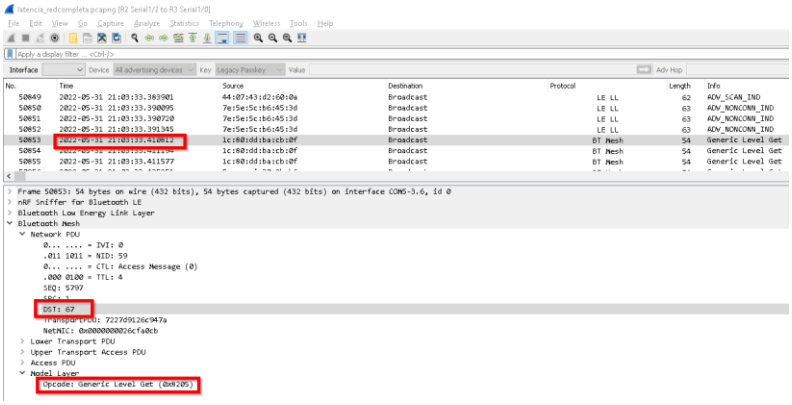
4.8.1.2. RTT con uno o varios saltos de red

Para realizar esta prueba, los sensores, así como el sniffer se ubican de acuerdo a la Figura 131. Se analizará el RTT de cada uno de los nodos de la red de acuerdo a los mensajes intercambiados.

Nodo N1BLEMesh

Este nodo es el SmartBuld y según el escenario de lectura de datos presentado ,es uno de los nodos más alejados del aprovisionador según el esquema presentado . Al ser prefabricado ,contiene varios modelos configurados por lo que se utiliza el modelo Generic Level .Al traducir la direccion unicast de este nodo a notación decimal se obtiene el valor de 67. En la se puede observar el mensaje de Generic Level Get capturado .El tiempo en el que se recibió el paquete en el sniffer es 21:03:33.410812.

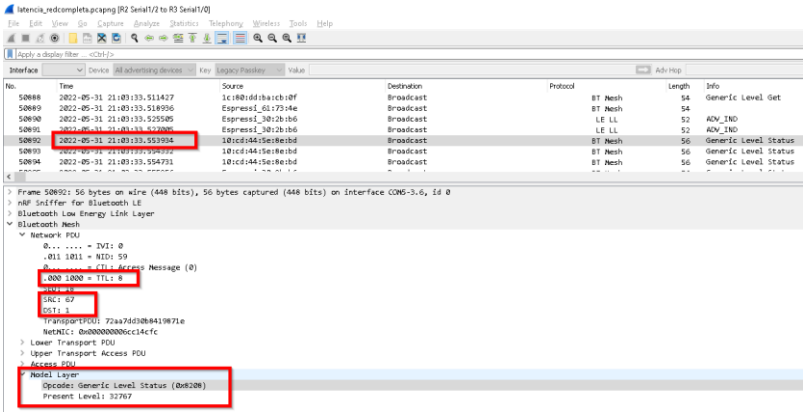
Figura 135 .Envío de un mensaje Generic Level Get al nodo N1BLEMesh



Fuente : Autoría .

De igual forma se realiza la captura del respectivo Generic Level Status ,este paquete se puede observar en la Figura *136*. Nótese que el valor de TTL ha disminuido a 8 es decir que ha realizado dos saltos hasta llegar al smartphone ,el tiempo en el que se recibió el paquete en el sniffer es 21:03:33.553934. La direccion de red de origen es la direccion 67(nodo N1BLEMesh) y la direccion de destino es la direccion 1(Aprovisionador).

Figura 136. Captura de paquete de Generic Level Status desde el nodo N1BLEMesh



Fuente : Autoría .

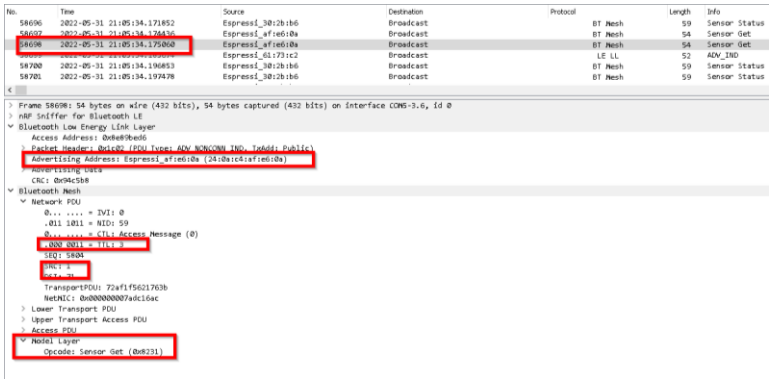
Por lo que se puede concluir que el RTT desde el nodo aprovisionador hacia el nodo N1BLEMesh es de 143.122 milisegundos y se dan 5 saltos de red según el valor de TTL recibido.

Nodo N2BLEMesh

Este nodo está compuesto por dos sensores ,según las propiedades de modelos usados ,los datos en bruto recolectados son 4 octetos. De acuerdo al esquema de ubicación de sensores ,se encuentra ubicado en la cocina .Los mensajes intercambiados corresponden al modelo de

sensor ,específicamente el mensaje sensor Get y Sensor status . .Al traducir la direccion unicast de este nodo a notación decimal se obtiene el valor de 71. El tiempo en el que se recibió el paquete en el sniffer es 21:05:34.175060. Nótese ademas que el primer mensaje capturado corresponde a un mensaje propagado del nodo N4BLEMesh que tiene un TTL de 3, es decir a realizado al menos 4 saltos .Se puede observar en la Figura *137*, el filtro utilizado para obtener el mensaje comprobando la afirmación realizada.

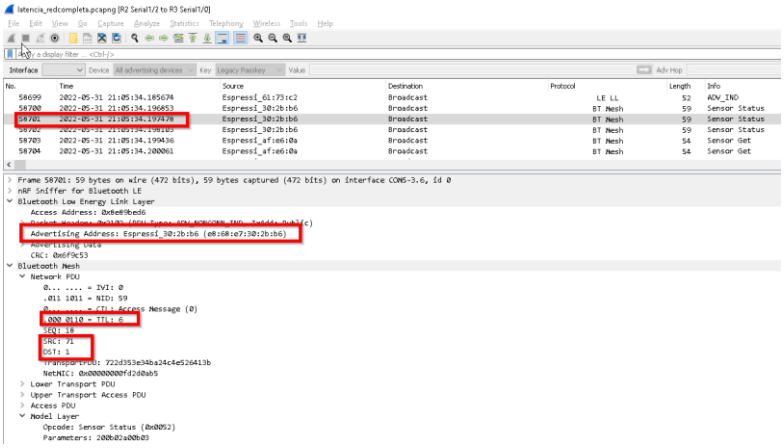
Figura 137.Envío de un mensaje Sensor Get al nodo N2BLEMesh



Fuente : Autoría .

Para obtener el paquete de respuesta se usa el filtro *btmesh.ttl ==6&&btmesh.src==71&&btmesh.dst==1*,se utiliza el valor de TTL de 6 en el filtro debido a que este es el primer paquete de Sensor Status capturados .El primer paquete capturado tiene un tiempo de 21:05:34.197478 y nótese que el paquete recibido es un paquete propagado del nodo N5BLEMesh .En la Figura *138* se puede observar el paquete capturado en cuestión con los parámetros explicados.

Figura 138. Envío de un mensaje Sensor Get al nodo N2BLEMesh



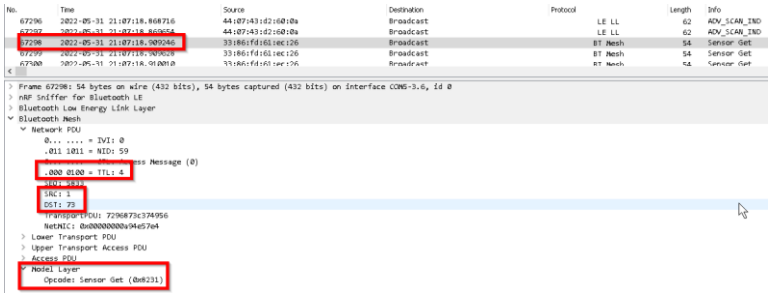
Fuente : Autoría .

En conclusión, el nodo N2BLEMesh tiene un RTT de 22.418 milisegundos con al menos 5 saltos desde que se realizó la petición de Sensor Get hasta que se respondió con un paquete Sensor Status.

Nodo N3BLEMesh

A partir de este modo ,se tendrá dos sensores ,según las propiedades de modelos usados ,los datos en bruto recolectados son 4 octetos .De acuerdo al esquema de ubicación de sensores ,se encuentra ubicado en la bodega .Los mensajes intercambiados corresponden al modelo de sensor ,específicamente el mensaje sensor Get y Sensor status, estas condiciones se repetirán en los nodos siguientes por lo que no se volverán a mencionar .Al traducir la direccion unicast de este nodo a notación decimal se obtiene el valor de 73. El tiempo en el que se recibió el paquete en el sniffer es 21:07:18.909246,y se trata de un mensaje propagado con un TTL de 4 es decir el paquete ha realizado al menos 3 saltos, todos estos parámetros se pueden observar en la Figura *139*.

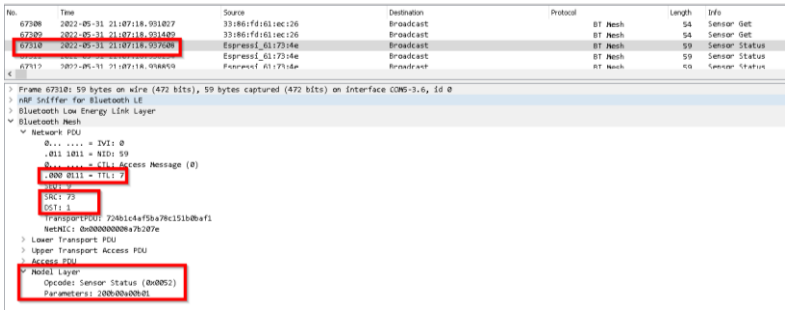
Figura 139. Envío de un mensaje Sensor Get al nodo N3BLEMesh



Fuente : Autoría .

El primer paquete capturado de Sensor status para la petición anterior tiene un tiempo de 21:07:18.937608 y nótese que el paquete recibido es un paquete enviado directamente desde l nodo analizado en cuestión .En la Figura *140* la se puede observar el paquete capturado en cuestión con los parámetros explicados.

Figura 140. Recibo de mensaje Sensor status desde el nodo N3BLEMesh



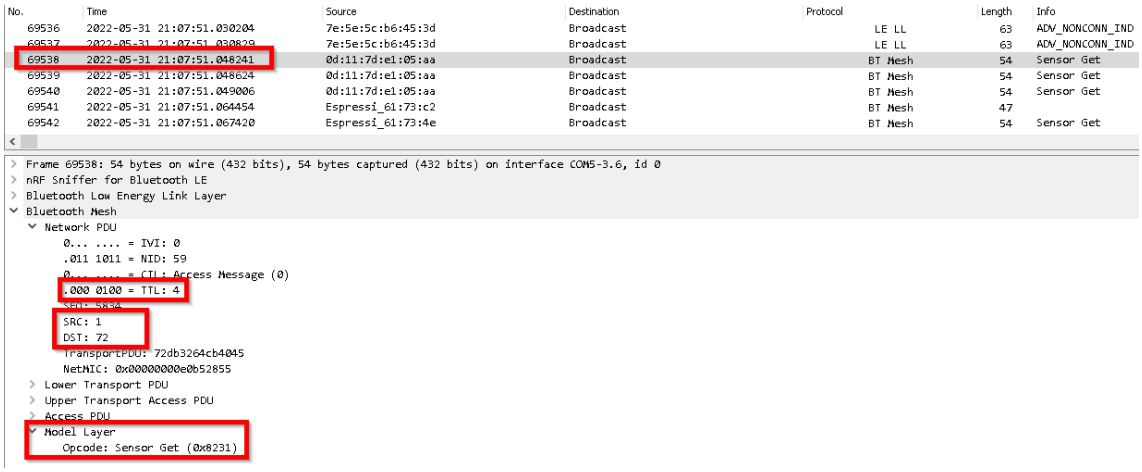
Fuente : Autoría .

En conclusión, el nodo N3BLEMesh tiene un RTT de 28.362 milisegundos con al menos 3 saltos desde que se realizó la petición de Sensor Get hasta que se respondió con un paquete Sensor Status.

Nodo N4BLEMesh

El nodo N4BLEMesh tiene una direccion unicast de 72 y está ubicado en la cocina ,se encuentra muy cerca del lugar de la medición .En la Figura *141* se puede observar el primer paquete capturado cuando de envía un paquete de sensor Get, la marca de tiempo del paquete es de 21:07:51.048241 con un TTL de 4 ,por lo que se puede decir que este paquete ha saltado dentro de la red al menos 3 veces.

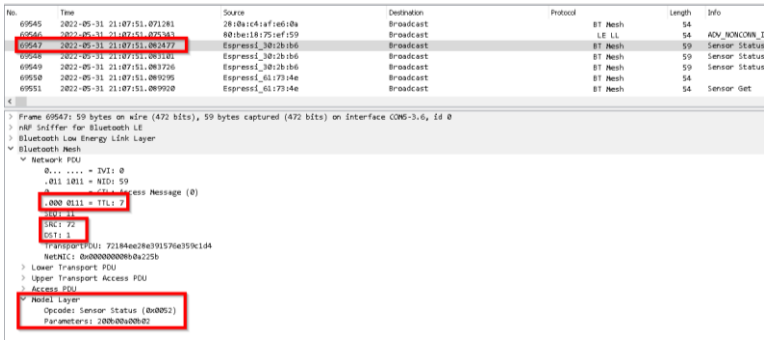
Figura 141.Envío de un mensaje Sensor Get al nodo N4BLEMesh



Fuente : Autoría .

El primer paquete capturado de Sensor status para la petición anterior tiene un tiempo de 21:07:51.082477 y nótese que el paquete recibido es un paquete enviado directamente desde l nodo analizado en cuestión .En la Figura *142* se puede observar el paquete capturado en cuestión con los parámetros explicados.

Figura 142. Recibo de mensaje Sensor status desde el nodo N4BLEMesh



Fuente : Autoría .

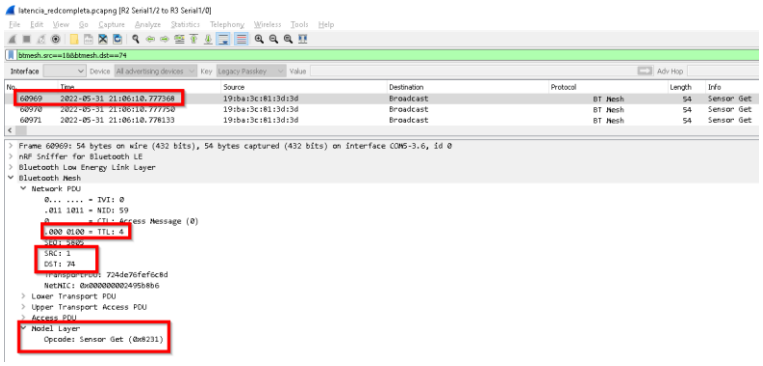
En conclusión, el RTT para los datos recibidos de este nodo es de 34.236 milisegundos para los menos 3 saltos realizados desde la petición y respuesta del estado del sensor.

Nodo N5BLEMesh

El nodo N5BLEMesh tiene una direccion unicast de 74 y está ubicado en el estudio de la vivienda donde se implementara la red de sensores ,es una de las localizaciones más lejanas de los nodos implementados .En la Figura *143* se puede observar el primer paquete capturado

cuando de envía un paquete de sensor Get, la marca de tiempo del paquete es de 21:06:10.777368 con un valor de TTL de 4 ,por lo que se puede decir que este paquete ha saltado dentro de la red al menos 3 veces.

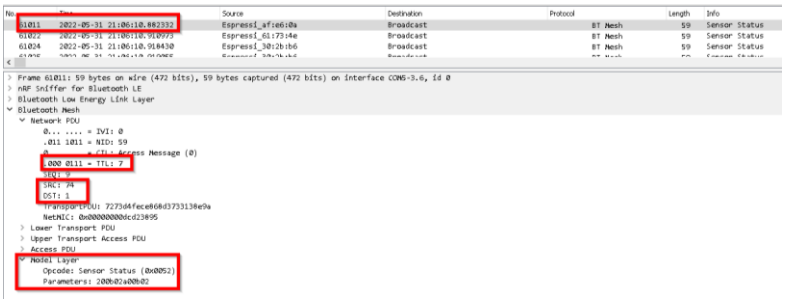
Figura 143 .Envío de un mensaje Sensor Get al nodo N5BLEMesh



Fuente : Autoría .

El primer paquete capturado de Sensor status para la petición anterior tiene un tiempo de 21:06:10.882332 y nótese que igual que en anteriores nodos ,el paquete recibido es un paquete enviado directamente desde el nodo analizado en cuestión .En la Figura *144* se puede observar el paquete capturado en cuestión con los parámetros explicados.

Figura 144. Recibo de mensaje Sensor status desde el nodo N5BLEMesh



Fuente : Autoría .

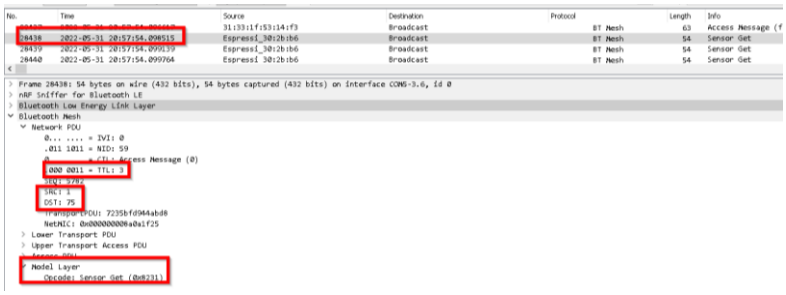
Por lo que el valor de RTT para este nodo es de 104.964 milisegundos para al menos 3 saltos de red entre los dos puntos de comunicación antes de la respuesta del sensor solicitado.

Nodo N6BLEMesh

El nodo N6BLEMesh se encuentra en la misma habitación donde se realiza la medición ,su direccion unicast en decimal es 75, la marca de tiempo del primer paquete capturado de la

operación Sensor Get es de 20:57:54:098515 con un valor de TTL de 3 es decir al menos han existido cuatro saltos. Se puede observar los datos mencionados en la Figura *145*.

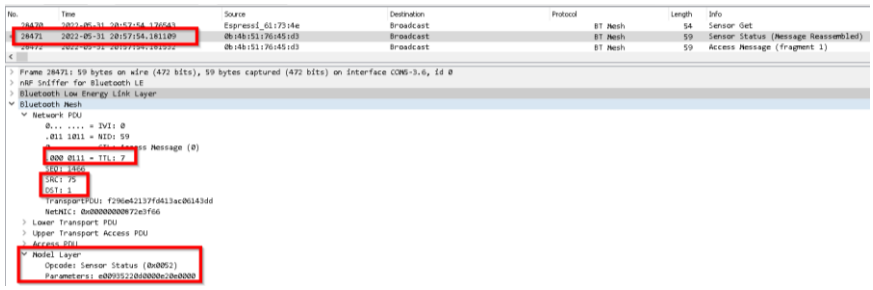
Figura 145.envío de un mensaje Sensor Get al nodo N6BLEMesh



Fuente : Autoría .

El primer paquete capturado de Sensor status para la petición anterior tiene un tiempo de 20:57:54.181109 , de igual forma ,el paquete recibido es un paquete enviado directamente desde el nodo analizado en cuestión .En la Figura *146* se puede observar el paquete capturado en cuestión con los parámetros explicados.

Figura 146. Recibo de mensaje Sensor status desde el nodo N6BLEMesh



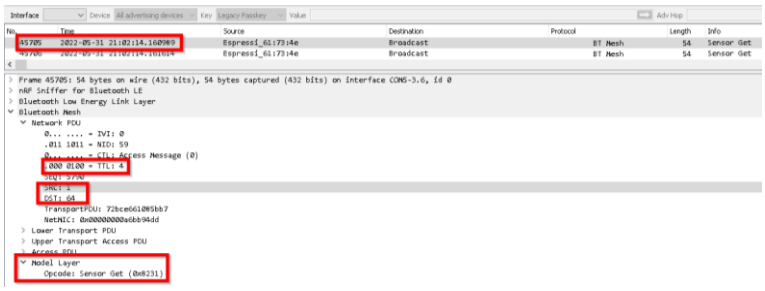
Fuente : Autoría .

Para este nodo el valor de RTT después de recibir la petición de sensor Get es de 82.594 milisegundos con un estimado de 4 saltos hasta que dicha petición llega al sensor .

Nodo N7BLEMesh

El nodo N7BLEMesh se encuentra en la misma habitación donde se realiza la medición ,su direccion unicast en decimal es 64, la marca de tiempo del primer paquete capturado de la operación Sensor Get es de 21:02:14.160989 con un valor de TTL de 4 es decir al menos han existido tres saltos. Se puede observar los datos mencionados en la Figura 147

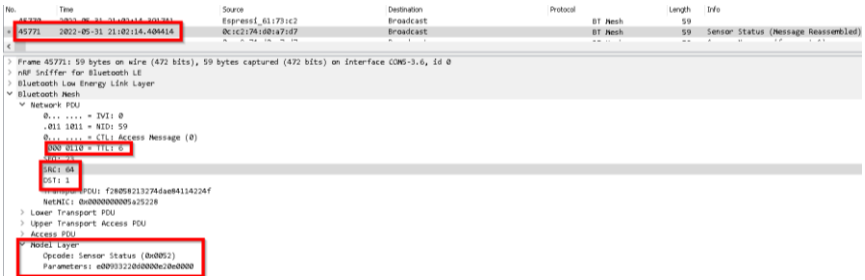
Figura 147.Envío de un mensaje Sensor Get al nodo N7BLEMesh



Fuente : Autoría .

El primer paquete capturado de Sensor status para la petición anterior tiene un tiempo de 21:02:14.400414 , para este caso el valor de TTL es de 6 por lo que se puede deducir que existió un salto en esta operación .En la Figura *148*se puede observar el paquete capturado en cuestión con los parámetros explicados.

Figura 148 .Recibo de mensaje Sensor status desde el nodo N7BLEMesh



Fuente : Autoría .

Para este nodo el valor de RTT después de recibir la petición de sensor Get es de 239.425 milisegundos con un estimado de 4 saltos hasta que dicha petición llega al sensor .

4.8.1.3. Valor RTT y numero de saltos de la red

Con todos los datos de RTT recolectados de cada uno de los nodos se procede a realizar una tabla para relacionar los datos de todos los nodos para realizar la comparativa del RTT de la red .La tabla con los valores se puede observar en la

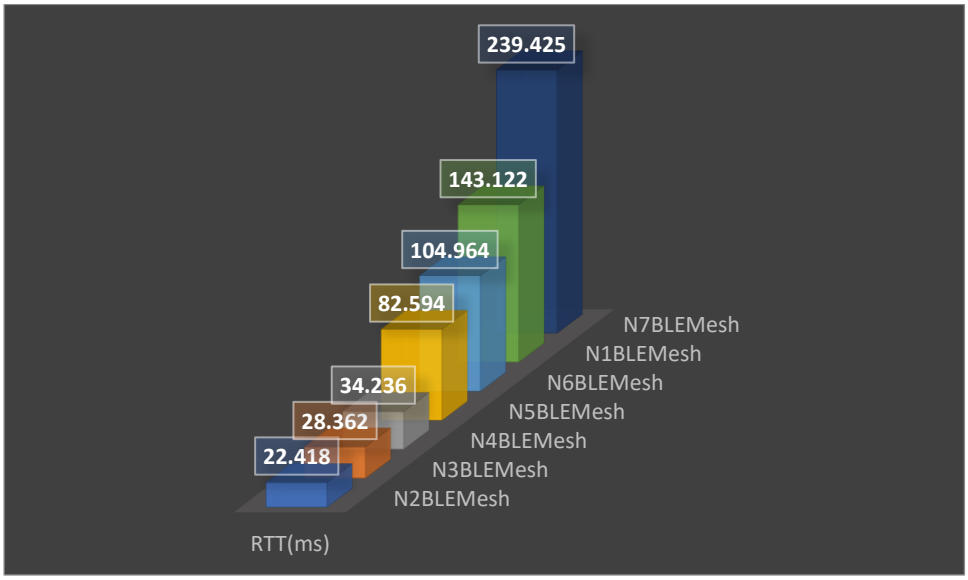
Tabla 39. Valor de RTT obtenidos para cada nodo de la red con su respectivo salto

Nodo	RTT	Saltos
N1BLEMesh	143.122 milisegundos	5
N2BLEMesh	22.418 milisegundos	5
N3BLEMesh	28.362 milisegundos	3

N4BLEMesh	34.236 milisegundos	3
N5BLEMesh	104.964 milisegundos	3
N6BLEMesh	82.594 milisegundos	4
N7BLEMesh	239.425 milisegundos	4

Con estos datos se puede realizar el grafico mostrado en la Figura *149*.Donde se hace una relación del valor de RTT a cada uno de los nodos ,donde ademas se puede observar que mientras más alejado y más objetos haya entre los nodos entonces del valor de RTT sube. Siendo los nodos : 1,7 y 5 los más alejados según el esquema presentado, y el nodo 6 es el nodo con más interferencia de paredes .

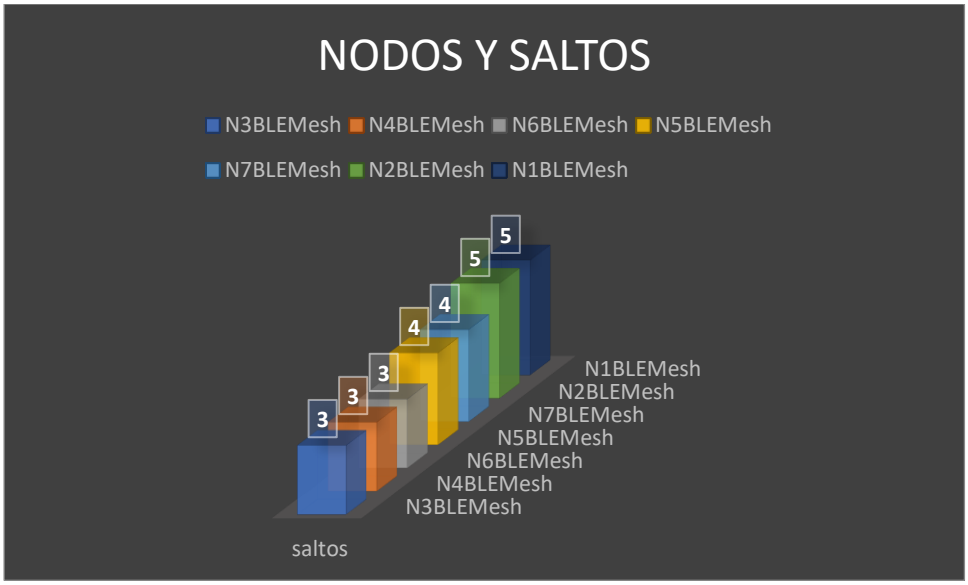
Figura 149 .RTT en cada nodo de la red



Fuente : Autoría .

La comparativa de los nodos y el numero saltos se puede observar en la Figura *150*,l os nodos :1,2,7 y 5 presentan una mayor cantidad de saltos debido a que se encuentran en otro edificio por lo que para llegar a ellos se debe atravesar más paredes .Mientras los nodos :6,4 y 3 se encuentran en el mismo edificio y tienen una menor cantidad de saltos .

Figura 150. Numero de salto de cada nodo de la red.



Fuente : Autoría .

Como conclusión se tiene que el valor de RTT aumenta con el número de saltos y no presentan variaciones notables dado que la cantidad de datos que se está inyectando a la red es similar .Mientras, por otro lado ,la cantidad de saltos está ligada al número de interferencias que los nodos tienen entre ellos ;si el número es mayor entonces la cantidad de saltos aumentará y este factor estará ligado directamente a la ubicación de los nodos en el ambiente de implementación.

El valor máximo de RTT es de 239.425 milisegundos ,con el cual se cumple el requerimiento de que la red debe informar de cualquier evento en menos de 5 segundos .

4.8.2. Validación de requerimientos del sistema

En base a las pruebas realizadas en la presente sección es posible realizar la validación de requerimientos del sistema ,estos validan el correcto funcionamiento del sistema con todos sus elementos .Para realizar este análisis se usa la tabla de verificación de requerimientos del sistema ,se puede observar en la Tabla 40 .

Tabla 40. Validación de requerimientos del sistema.

SyRS					
Requerimientos del sistema					
No	**Requerimiento**	**Estado**			**Observación**
		SI	**NO**	**N/A**	
Requerimientos de uso					
SyRS1	El usuario recibirá notificaciones de audio mediante el smartphone usando archivos mp3 alojados dentro de la aplicación móvil	X			
SyRS2	Recabar datos de: aire, estado de conexiones eléctricas, niveles de agua y posición de personas para mostrarlos en una aplicación móvil.	X			
SyRS4	La aplicación móvil tendrá funciones de accesibilidad que permitan a personas con discapacidad visual hacer uso de las funciones.	X			
SyRS5	El sistema presentará un resumen de estadísticas y eventos al desarrollador para realizar mejoras a futuro.	x			
SyRS6	El sistema debe permitir la coexistencia con otras tecnologías de comunicación inalámbricas para realizar mejoras a futuro.	X			
Requerimientos de rendimiento					
SyRS7	La aplicación móvil tendrá un retardo máximo de 1 minuto al activar cualquiera de sus funciones de alerta.	X			
SyRS8	Los sensores deben tener el tiempo de respuesta corto y con la menor cantidad de errores	X			
SyRS9	La interfaz web de presentación de datos tendrá bajo consumo de datos y corto tiempo de respuesta en el proceso de actualización de datos	X			
SyRS10	La comunicación entre los sensores y el nodo central tendrá un retardo máximo de 1 minuto	X			
SyRS11	La aplicación móvil tendrá un tiempo de respuesta corto en el proceso de interacción con el usuario	X			
SyRS12	Las alertas de voz tendrán una duración corta y un mensaje claro en la medida de lo posible	X			
Requerimientos de interfaces					

SyRS13	Las placas de desarrollo utilizadas en el sistema deben tener una entrada USB o SWD para que puedan ser configuradas mediante una conexión serial a un ordenador.	X	
SyRS14	Las placas de desarrollo utilizadas deben contar con antenas integradas para la conexión inalámbrica a través de Bluetooth.	X	
SyRS15	Un nodo del sistema debe funcionar con alimentación eléctrica ya sea usando cables USB o baterías	X	
SyRS16	El nodo central de la red debe contar con una interfaz inalámbrica para conexión Bluetooth y una interfaz inalámbrica para conexión a Internet de forma simultanea	X	
SyRS17	El sistema tendrá una interfaz de presentación de información para usuarios directos e indirectos	X	
SyRS18	La base de datos guardará las notificaciones del sistema en registros organizados y legibles	X	
	Requerimientos de modos /Estados		
SyRS19	El sistema consumirá los datos que la aplicación el envié cuando se encuentre activo	X	
	Requerimientos físicos		
SyRS20	El tamaño de los módulos del del sistema no debe ser mayor a 15 cm.	X	
SyRS21	Los sensores no se conectarán de forma permanente a la placa de desarrollo para un posterior remplazo de componentes	X	
SyRS22	Los módulos del sistema deben instalarse en una caja empotrada en la pared o similar.	X	
SyRS23	Los nodos de la red deben ser portátiles para realizar la reubicación de estos en caso de ser necesarios	X	

Fuente : Autoría .

4.9. Pruebas realizadas al sistema : verificación de la presentación de información al usuario

Las alertas al usuario se realizan en base a los valores que se envían en tiempo real a la base de datos alojada en Firebase ,las alertas deberán tienen un nombre especificado para cada archivo que identifica al sensor y tipo de nodo. Para optimizar el tiempo de duración de alertas ,cada archivo mp3 durara un máximo de cinco segundos.

4.9.1. Presentación de información a los usuarios directos

La presentación de información para los usuarios directos se da mediante alertas de voz ,las cuales se han comprobado en apartados anteriores .Las alertas se dan de acuerdo con la lectura de los sensores .Ademas el usuario directo puede navegar por voz dentro de la aplicación ,para la realización de esta validación se ha presentado la aplicación a la persona a quien se realizó la entrevista .

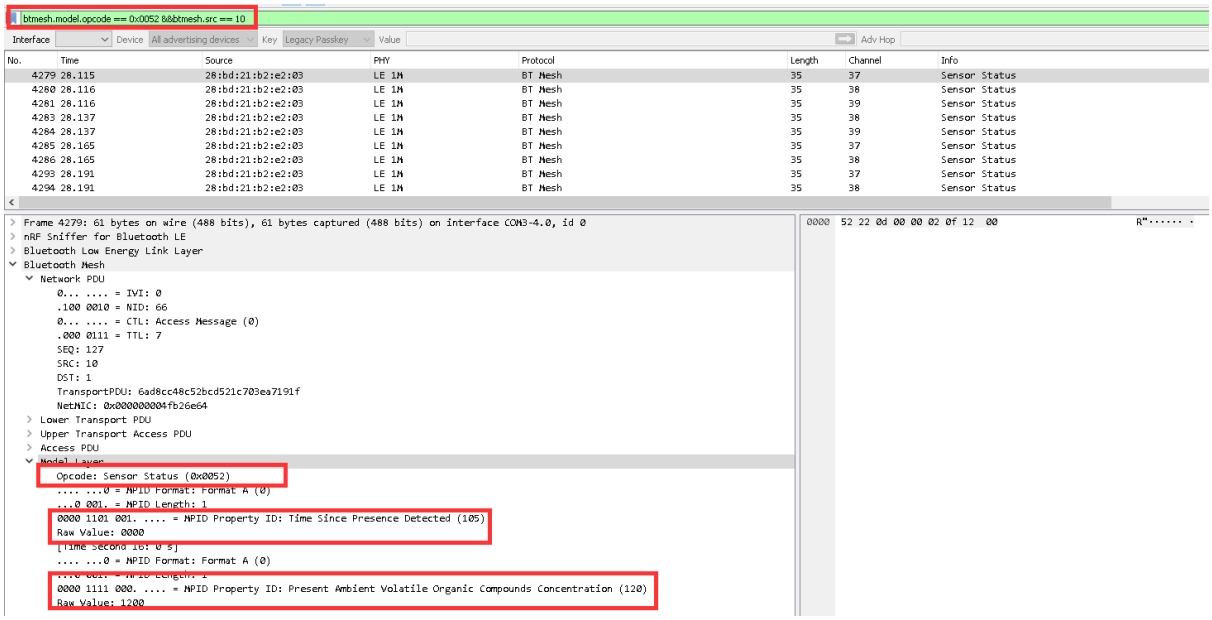
Para esta prueba se ha colocado al usuario directo cerca del nodo N3BLEMesh y se ha solicitado los datos de los sensores y se ha recibido una respuesta de todos los nodos ,en la Figura *151* se puede observar la ubicación del usuario directo respecto a los nodos .

Figura 151. Ubicación del usuario directo respecto a los nodos de la red de sensores

Fuente : Autoría .

Para la comprobación de la lectura de datos se realizará una captura de paquetes ,se utilizará los mismos parámetros de la sección de pruebas de RTT de la red .Solo que en este caso se deberá comprobar las direcciones de red de los sensores y los respectivos datos recibidos por los mismos .Las direcciones de red se mantienen ,por lo que se deberá utilizar el siguiente filtro de información en wireshark :.En la Figura *152* se puede observar los datos recogidos ,notar la presencia de los datos de sensores capturados así como las respuesta de cada uno de los nodos con dichos datos.

Figura 152. Captura de paquetes de los datos solicitados a la red durante la prueba de presentación al usuario directo.



Fuente : Autoría .

Por otro lado ,se puede observar en la Figura *153* al usuario directo usando la aplicación desarrollada y obteniendo los datos de los sensores ,ademas del uso de las funciones de accesibilidad de la aplicación para personas no videntes .Nótese que en la sección (a) de la imagen se realiza la solicitud de los datos a la red de sensores ,mientras en la sección (b) de la imagen se recibe los datos de sensores .

Figura 153 .Interacción del usuario directo con la aplicación móvil.

(a)

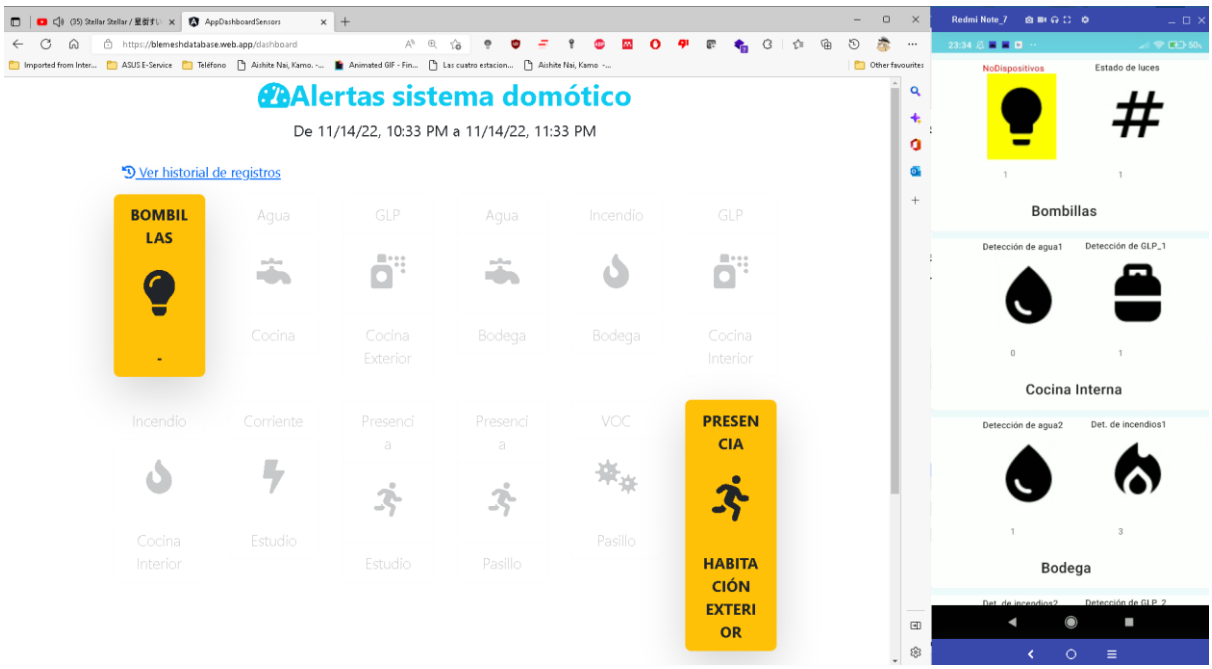
(b)

Fuente : Autoría .

4.9.2. Presentación de información a usuarios indirectos

La presentación de información para los usuarios indirectos está enfocada en dos ámbitos :la interfaz web y la pantalla de la aplicación móvil .Estas dos muestran información en base a iconos ,cada icono cambiará de color .Para la aplicación web se leerá los valores actuales de la base de datos ,mientras que la interfaz web obtendrá los datos de hace una hora antes del ingreso a la misma .En la Figura *154* se puede observar la comparativa entra las dos herramientas de presentación de información hacia los usuarios indirectos.

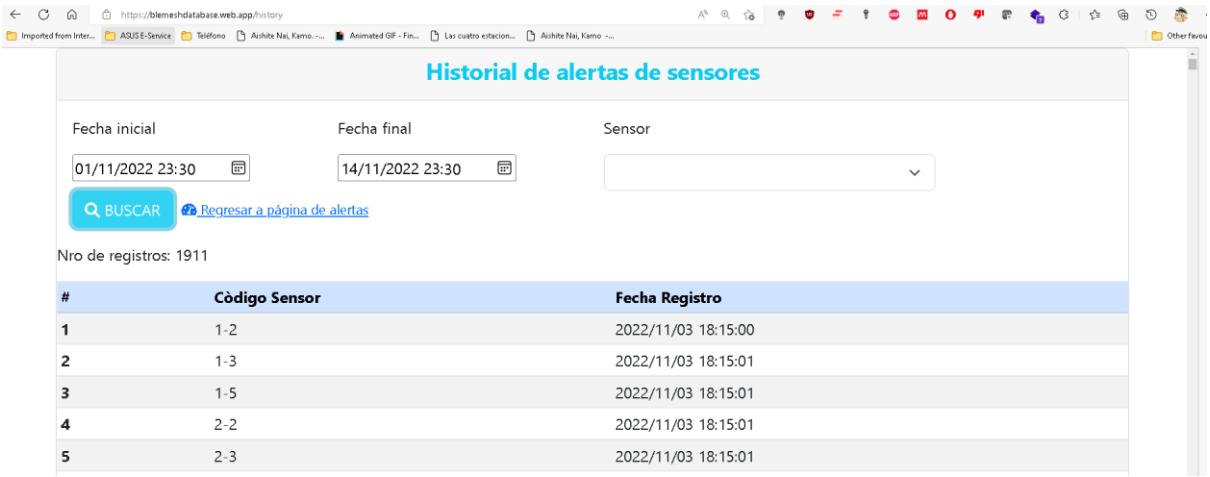
Figura 154. Presentación de información hacia los usuarios indirectos



Fuente : Autoría .

También será posible acceder a la base de datos desde la interfaz web con un link ,en este enlace se podrá escoger un intervalo de fechas y tipo de sensor a visualizar .Esta lista se observa en la Figura *155*.

Figura 155. Presentación de lista de alertas a los usuarios indirectos



Fuente : Autoría .

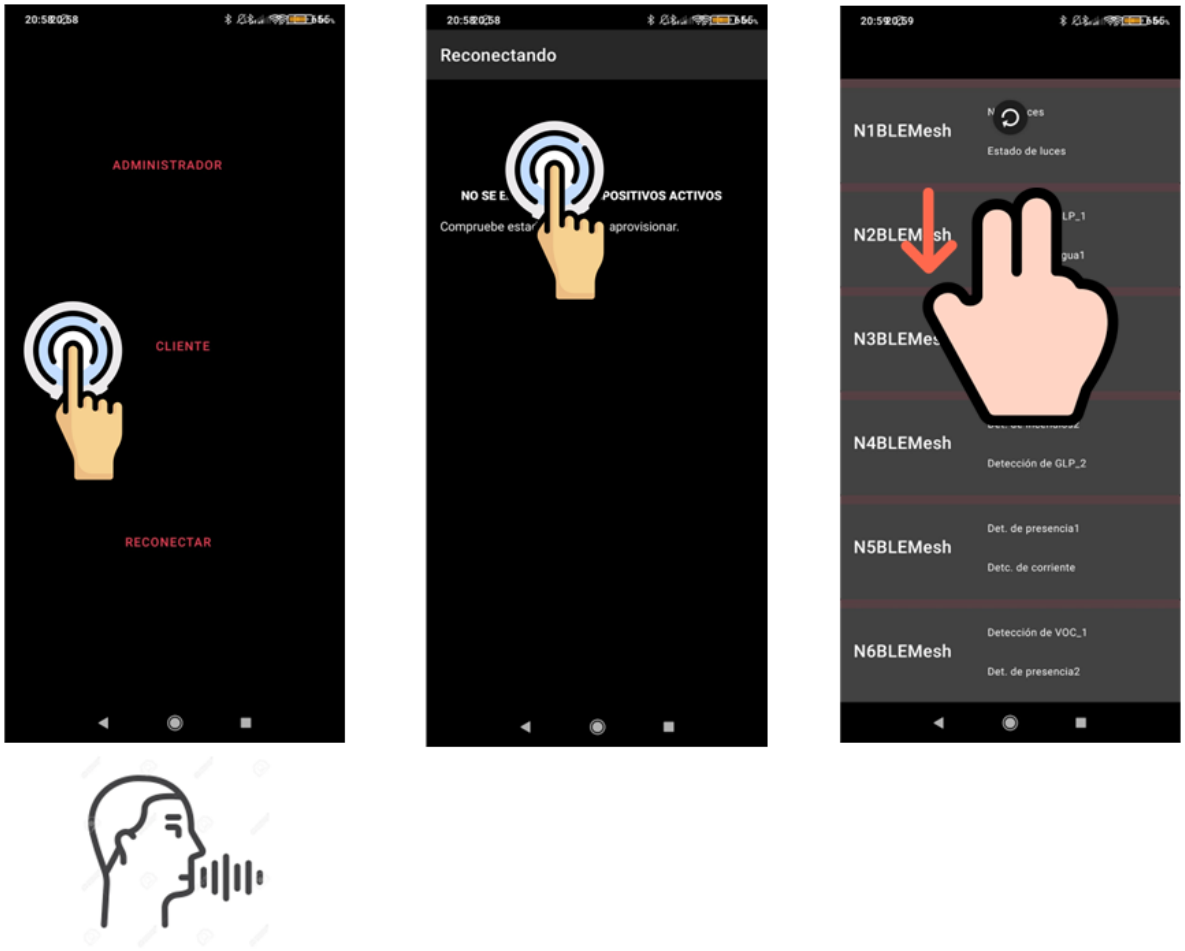
4.9.3. Accesibilidad al usuario directo

Las pantallas de la aplicación ,a la cuales el usuario directo debe acceder serán tres .Una pantalla para escoger hacía que funcionalidad ir, una pantalla para realizar la reconexión del smartphone a la red y una pantalla donde se alojaran los datos recogidos por los sensores de la red .Las pantallas desarrolladas reducen la cantidad de botones al mínimo así como la interacción de la pantalla con el usuario ,es decir : se utilizaron menos de 5 botones en todo la aplicación y los datos de los sensores se consiguen mediante la opción de refrescar la pantalla.

Ademas se ha añadido una función adicional para la navegación entre algunas pantallas de la aplicación móvil usando reconocimiento de palabras ,una vez se inicia la aplicación se podrá usar la palabra “reconectar “ para ingresar a la pantalla de reconexión a la red de sensores .De igual forma se podrá usar la palabra “usuario “ para acceder a la pantalla de datos recabados por los sensores.

Con todos los métodos de accesibilidad se asegura que la aplicación tenga características de accesibilidad y sea cómoda al usuario ,las acciones que se pueden realizar en la aplicación por el usuario se observan en la Figura 156.

Figura 156.Acciones realizadas en la aplicación por el usuario directo.



Fuente : Autoría .

4.9.4. Validación de requerimientos de stakeholders

En base a las pruebas realizadas en la presente sección es posible realizar la validación de requerimientos de skateholders ,estos validan la presentación del sistema hacia los usuarios directos e indirectos .Para realizar este análisis se usa la tabla de verificación de requerimientos de skateholders ,se puede observar en la Tabla 41.

Tabla 41. Validación de requerimientos de skateholders

StRS					
Requerimientos de Stakeholders					
No	**Requerimiento**	**Estado**			**Observación**
		SI	**NO**	**N/A**	
Requerimientos de Usuarios					
StRS1	Las alertas generadas por el sistema serán comunicadas al usuario por medio de sintetizadores de voz	X			
StRS2	Implementación de una aplicación móvil para el control centralizado del sistema	X			
StRS3	Función de navegación con voz dentro de la aplicación	X			
StRS4	La aplicación debe tener un indicativo perceptible para las personas con discapacidad visual cuando se navegue entre pantallas	X			
StRS5	Los costos de diseño y construcción serán bajos.	X			
StRS6	La ubicación y tamaño de los sensores no afectará la ejecución de actividades cotidianas de los usuarios dentro del domicilio.	X			
StRS7	El sistema debe permitir el acceso a usuarios indirectos y al administrador del sistema en forma continua e ininterrumpida	X			
Requerimientos Operacionales					
StRS8	El nodo central del sistema debe ser un smartphone con características de hardware para emitir alertas de voz	X			
StRS9	Todos los componentes de la red de sensores del sistema deben contar con soporte de la tecnología Bluetooth en su version 4.x o superior.	X			
StRS10	El nodo central del sistema debe tener conexión a internet estable	X			

StRS11	El smartphone que se va a usar en el sistema debe contar con sistema operativo Android 8.0 superior.	X
StRS12	La aplicación móvil que controlara el sistema debe desarrollarse usando un IDE que brinde documentación sólida y libre acceso	X
StRS13	El sistema debe tener alimentación eléctrica para cada uno de los nodos usando una fuente externa o una batería	X
StRS14	La aplicación móvil tardara máximo de 1 minuto en emitir las notificaciones de alerta	X

Fuente : Autoría .

4.10. Consumo de energía eléctrica del sistema

En esta sección se realizará el cálculo de consumo eléctrico estimado del sistema en base a cada uno de los sensores y módulos ,dicho cálculo servirá para estimar el costo de funcionamiento del sistema en un determinado periodo de tiempo .

Para realizar el cálculo se utilizará los valores máximos de potencia eléctrica de acuerdo con los datasheet de los elementos empleados en el sistema .Ademas se considera que los dispositivos van a funcionar mediante un cargador USB siempre .Se considera que la mayor parte del sistema funciona con 5 o 3.3 [v] en DC. Como primer paso se realiza el cálculo para todos los dispositivos como :tarjetas de desarrollos ,periféricos y otros más ,en la Tabla 42 se puede observar el consumo energético de los kits utilizados en el sistema, así como otros elementos .Como se alimenta a la placa ESP32 con 5 voltios ,se considerará este su valor de voltaje .

Tabla 42 .Potencia eléctrica de dispositivos sados para la construcción de la red

Descripción	N Elementos	Consumo unitario en Amperios	Voltaje de funcionamiento	Consumo unitario en watios
ESP32 DevKitC v4	4	260 mA	5 V DC	0.858 W
Thingy 52*	2	40 mA	3.3V DC	0.132 W
Sengled A19	1	--	120 V AC	8.5 W

Nota : * Funcionamiento con batería
Fuente : Autoría .

De igual forma se debe conocer los valores máximos de potencia para cada uno de los sensores dentro de la red ,considerando que dichos sensores se alimentaran eléctricamente de la placa de desarrollo utilizada ,por lo que la potencia se podrá sumar a la obtenida anteriormente .En la *Tabla 43* se puede observar la potencia de los sensores utilizados en el sistema.

Tabla 43. Potencia eléctrica de los sensores utilizados en el sistema

Descripción	N elementos	Consumo unitario en Amperios	Voltaje de funcionamiento	Consumo unitario en watios
Sensor MQ6	2	150 mA	5 V DC	0.75 W
Sensor GAOHOU A930	2	20 mA	3.3V DC	0.066 W

Sensor KY 026	2	15 mA	3.3V DC	0.0495 W
Sensor SCT 013	1	5 mA	3.3V DC	0.0165 W
Sensor HC SR501	1	65 mA	3.3V DC	0.2145 W

Fuente : Autoría .

No se ha realizado la suma total de la potencia eléctrica de los sensores dado que no se utilizará dos tipos de sensores en la misma placa de desarrollo ,por lo que es necesario realizar una suma de los sensores utilizados en cada placa de desarrollo .Estos cálculos se pueden observar en la .Considerar que el valor de potencia del Thingy 52 incluye el consumo de sus sensores.

Nodo	**Sensores utilizados**	**Potencia sensor más placa**
N2BLEMESH	Sensor GAOHOU A930 Sensor MQ6	1.674 W
N3BLEMESH	Sensor GAOHOU A930 Sensor KY 026	0.9735 W
N4BLEMESH	Sensor KY 026 Sensor MQ6	1.6575 W
N5BLEMESH	Sensor SCT 013 Sensor HC SR501	1.089 W
N6BLEMESH	Thingy 52	0.132 W
N7BLEMESH	Thingy 52	0.132 W
Total		5.658 W

Se procede a realizar la suma de estos valores para obtener la potencia total del sistema .El valor obtenido se puede observar en la Tabla *44*.

Tabla 44. Potencia total del sistema.

Descripción	**Valor**
Potencia total de nodos	5.68 W
Potencia total de dispositivos adicionales	8.5 W
Total	14.158 W

Fuente : Autoría .

El valor obtenido será utilizado para calcular el consumo de energía eléctrica que se mide en kilovatio hora ,para ello se utiliza la formula:

$$\text{kWh} = (\text{watts} \times \text{horas}) \div 1{,}000 \quad (5)$$

Para aplicar la formula se utiliza el caso extremo en donde los nodos van a funcionar las 24 horas sin ningún tipo de descanso .

$$\text{kWh} = (14.158\,\text{W} \times 24\text{horas}) \div 1{,}000 = 0.339792\,\text{kWh}$$

Ademas se considera que funcionara los 30 días de un mes, por lo que se debe multiplicar el valor obtenido con anterioridad .

$$\text{TotalkWh} = 0.339792\text{kWh} * 30\text{ dias} = 10.19376\text{kWh}$$

El valor de 20.99 kWh corresponde al consumo de energía eléctrica del sistema dentro de un mes ,ahora se realiza el cálculo del valor adicional de facturación ,correspondiente al servicio de energía eléctrica que el sistema va a adicionar .

Para lo cual se considera algunos factores como :costo del servicio eléctrico fijado por el CONELEC correspondiente a 9,20 cUSD/kWh(Ministerio de Energía y Minas, 2022) y una reducción de las tarifas de consumo eléctrico de un 50% para las personas con discapacidad(CONADIS, 2012).Con estos factores ,el costo del servicio eléctrico será de 4.60 cUSD/kWh para un cliente objetivo de este proyecto.

Finalmente se calcula el costo adicional del servicio eléctrico del sistema con la siguiente operación matemática.

$$Valor\ Adicional = \frac{0.0460\ USD * 10.19376\ kWh}{1kWh} = 0.4689\ USD$$

Es decir que :el sistema va a añadir el valor adicional de 47 centavos de dólares americanos al consumo mensual de energía eléctrica en el hogar ,recordando que la alimentación de las placas será mediante cargador principalmente .

4.11. Costo final del sistema

Con todas las pruebas del sistema realizadas se comprueba que el sistema funciona ,por lo que es necesario realizar el cálculo del costo final del sistema .para analizar si cumple con el valor de costo planteado en la planificación del sistema .Para estimar el costo final del

sistema se hace referencia al presupuesto económico usado adquirir componentes y servicios de hardware, software ,componentes físicos e ingeniería.

4.11.1. Costo de hardware

Los costos económicos de hardware comprenden todos los elementos adquiridos para construir el sistema ;ya sean placas de desarrollo ,sensores ,fuentes de alimentación eléctrica y equipos complementarios .En la Tabla 45 se detallan el listado de costos para hardware .

Tabla 45. Resumen de costos de hardware

Descripción	**Cantidad**	**Precio unitario**	**Subtotal**
ESP32-DevKitC V 4	3	$10	$30
ESP32-DevKitC V 2	1	$10	$10
Thingy 52	2	$40	$80
LED Bluetooth A19 E26	1	$25	$25
J-LINK EDU MINI	1	$20	$20
nRF 52840 dongle	1	$12.99	$12.99
Sensor MQ6	2	$4	$8
Sensor GAOHOU A930	2	$4.02	$8.04
Sensor KY-26	2	$2.68	$5.36
Sensor sct -013	1	$10	$10
Sensor PIR HC-SR501	1	$2.89	$2.89
Batería recargable	4	$11	$44
Cargador de Batería recargable	1	$20	$20
Cargador con entrada USB	4	$3	$12
Cable micro USB tipo B	4	$9	$36
Componentes electrónicos varios	1	$30	$30
		Total	$354.28

Fuente : Autoría

4.11.2. Costo de software

Los costos económicos de software comprenden todos programas ,IDEs o entornos de desarrollo usados para levantar el sistema ,según los requerimientos el software debía ser de

uso libre por lo que en consecuencia el valor será de $0 .En la se detallan el listado de costos para software.

Tabla 46 .Resumen de costos de software

Descripción	Cantidad	Precio unitario	Subtotal
Visual Studio Code	1	$0	$0
Firebase	1	$0	$0
Android Studio	1	$0	$0
ESP-IDF	1	$0	$0
nRF Connect for Desktop	1	$0	$0
SEGGER Embedded Studio	1	$0	$0
J-Link	1	$0	$0
		Total	$0

Fuente : Autoría

4.11.3. Costo de infraestructura

Los costos económicos de infraestructura corresponden a los gastos para construir los contendores de los nodos para su ubicación en el domicilio del cliente, todos los materiales usados se detallan en la Tabla 47.

Tabla 47. Resumen de costos de infraestructura

Descripción	Cantidad	Precio unitario	Subtotal
Plancha de tríplex (0.6 mm)de 1 m x 1 m	1	$10	$10
Pegamento blanco Blancola de 250 g	1	$1.25	$1.25
Componentes varios de carpintería	1	$5	$5
Pintura en spray	2	$2.50	$5
		Total	$21.25

Fuente : Autoría

4.11.4. Costo de ingeniería

Los costos de ingeniería corresponden a la remuneración económica al administrador del sistema de acuerdo a los estudios ,diseño ,implementación ,documentación y mantenimiento preventivo del sistema ,dado que este proyecto es una tesis se considera este costo como $0 .En la se puede observar estos parámetros

Tabla 48. Resumen de costos de ingeniería

Descripción	**Cantidad**	**Precio unitario**	**Subtotal**
Estudios del estándar de comunicación de red aplicado a domótica	1	$200	$200
Diseño del sistema	1	$300	$300
Implementación del sistema	1	$100	$100
Documentación del sistema	1	$30	$30
Mantenimiento preventivo	1	$20	$20
		Total	$650

Fuente : Autoría

4.11.5. Costo final del sistema

Con el análisis de todos los costos ,es posible obtener el costo final del sistema ,en la Tabla 49 se puede observar el costo total del sistema considerando :costo de hardware ,costo de software ,costo de ingeniería y costo de infraestructura.

El valor de Costo de Ingeniería será de $0 dado que este sistema es un proyecto de titulación.

Tabla 49 .Costo final del sistema

Descripción	**Subtotal**
Costo de hardware	$354.28
Costo de software	$0
Costo de Ingeniería	$0
Costo de Infraestructura	$21.25
Total	$375.53

Fuente : Autoría

Conclusiones y Recomendaciones

Conclusiones

Las redes de sensores que funcionan bajo el estándar de comunicación de red Bluetooth mesh son útiles para transmisión de bajas cantidades de datos en entornos con una alta cantidad o con una distribución especifica de nodos ,por lo que la ubicación de los sensores en el entorno de funcionamiento de la red tiene un gran impacto en el funcionamiento de esta.

Cuando un nodo externo desea comunicarse a la red se presenta una limitación importante :se debe usar la función proxy tanto en el nodo interno como en el nodo externo .El problema radica en que un nodo externo tiene las limitaciones correspondientes al uso de la tecnología Bluetooth Low Energy ,es decir solo pueden realizar conexiones esclavo -maestro ,piconet ,supernet y similares .En conclusión ,solo podrán conectarse a un número limitado de dispositivos a la vez en lugar de utilizar la conexión propia de una red Bluetooth Mesh. De igual forma, la mayor ventaja de la red viene de la mano con la mayor desventaja del estándar de comunicación de red .Al asegurar la comunicación de un nodo externo ,se considera que la mayoría de los dispositivos que cuenten con una interfaz Bluetooth puedan acceder a la red ,sin antes intercambiar las respectivas llaves de seguridad .

El uso de una red Bluetooth mesh no brinda un Throughtput extremadamente alto en comparación con otras tecnologías y protocolos de redes de sensores en malla, ya establecidas en el mercado como Thread ,Zigbee, Wi-Fi entre otros .El fuerte del uso del estándar radica en :extender el rango de la red, definir roles para cada nodo con un alto nivel de escalabilidad contando con dispositivos de bajo coste y mantenimiento ,sin la necesidad de poseer la última version de Bluetooth ;y poseyendo un alto nivel de seguridad .De tal forma que se podrá añadir,

quitar roles de un nodo, reparar nodos con una extrema facilidad .Siendo un estándar :fácil de mantener ,de fácil adquisición ,de alta escalabilidad y seguro .

El estándar de comunicación de red funciona bajo las versiones de Bluetooth 4.2. Por lo que no se aprovechan las ventajas incluidas en las nuevas versiones de Bluetooth como la version 5.0 y superiores .Debido a ello no puede alcanzar las características de otras tecnologías de comunicación inalámbrica típicas usadas para redes de sensores.

El costo neto del sistema se encuentra sobre los trescientos dólares americanos ,siendo este costo muy bajo considerando lo que normalmente cuesta un sistema domótico común a la vez posee un costo bastante bajo de escalabilidad si se utiliza el mismo esquema de nodos. Considerando los subsidios en consumo de energía eléctrica en el país, esto no representa una afectación a la economía del usuario.

Los frameworks usados para flashear las placas de desarrollo están todos basados en el RTOS Zephyr ,pero la presentación de datos varia al enviarse al medio físico ;por lo que los datos recibidos en el smartphone deben tratarse de acuerdo a dichos formatos antes de presentarlos al cliente ,considerando la accesibilidad que obligatoriamente debe tener la aplicación móvil.

Los datos que se manejan en este proyecto y que posteriormente circulan dentro de la red son pequeños demostrando que, bajo este escenario la red esta funcionado de una manera óptima .Se asegura dicho resultado en función del RTT de los escenarios experimentales presentados. También se comprueba que, sin importar la ubicación de los sensores ,el sistema funcionara de manera óptima .

El uso de Android Studio como SDK para desarrollar la aplicación móvil permitió que la aplicación móvil elaborada cuente con la mayoría de funcionalidades para poder obtener información de la red de sensores ,así como incluir funciones de accesibilidad para que el cliente puede manejar la aplicación de una forma sencilla y eficiente .También se incluye algunas características básicas para la configuración de red por medio del administrador .De esta forma ,se asegura que el cliente puede acceder fácilmente a la información de los sensores de la red desarrollada.

Recomendaciones

La ubicación de nodos en una red que funcione bajo el estándar de comunicación de red Bluetooth Mesh juega un papel vital en la reducción del número de salto y por ende una reducción del RTT. Por lo que la ubicación del cliente con respecto a los nodos de la red juega un papel importante al momento de comprobar el rendimiento del sistema.

Cada vez que se retira un nodo de la red de forma manual ,es decir sin desconectarlo lógicamente de la red ,se debe asegurar que las llaves almacenadas en el nodo sean eliminadas adecuadamente ,con el fin de evitar el robo de dichas credenciales de seguridad que pondrían en peligro de ataque a la red de sensores.

De ser posible siempre usar la opción de OOB al realizar el proceso de aprovisionamiento ,esta opción asegura que los datos intercambiados durante el aprovisionamiento no sean transparentes a dispositivos externos que no forman parte de la red .La mayoría de los dispositivos que soportan el estándar de red permiten habilitar esta opción con el fin de añadir un nivel extra de seguridad .

La cantidad de nodos en una red de sensores que utiliza el estándar de comunicación de red es un factor de confiabilidad de la red .Pero no se debe confiar totalmente en dicho factor ,en una red tan pequeña como la desarrollada .Por lo que se debe activar las funciones de

Heartbeat en los nodos ,con el fin de conocer cuando un nodo se apaga inesperadamente y actuar de forma inmediata.

Tomando en cuenta la naturaleza de los sensores ,se debe realizar un proceso de calibración de los mismos .La calibración deberá realizarse de acuerdo al tipo de sensor y la placa de desarrollo utilizada .Así como sumista el voltaje y corriente de alimentación de acuerdo a los datasheet de dichos sensores.

La elección del API, SDK o Framework que soporten el estándar Bluetooth mesh ,es una de las tareas más difíciles para trabajar con dicho estándar .Se debe escoger a aquel que presente la mayor parte de funcionalidades así como cumpla la mayor parte de los requisitos .Algunos SDK funcionan con la version del perfil 1.0 solamente ,otros son propietarios y funcionan específicamente en software propietario, otros están limitados por el hardware sobre el que funcionan .Por lo que de esta elección dependerá el éxito de la red de sensores construida.

Al momento de la realización de este proyecto ,la documentación del estándar en Raspberry pi es demasiado pobre para el desarrollo de aplicaciones utilizando sensores, así como también no existe un API tan poderosa como la utilizada en la aplicación móvil .La mayoría de la documentación presente no presenta una solución factible debido a la poca popularidad del estándar por lo que no fue posible utilizarla a la placa como Gateway. A futuro ,una gran actualización seria migrar las funcionalidades de la aplicación móvil a la Raspberry Pi.

Referencias bibliográficas

Abdullah, H., Rahman, M. S., Garcia, W., Warren, K., Yadav, A. S., Shrimpton, T., & Traynor, P. (2021). Hear “no Evil”, See “kenansville”*: Efficient and transferable black-box attacks on speech recognition and voice identification systems. *Proceedings - IEEE Symposium on Security and Privacy*, *2021-May*, 712–729. https://doi.org/10.1109/SP40001.2021.00009

Amazon. (2021). *Create Alexa Skills Kit | Amazon Alexa Voice Development*. https://developer.amazon.com/en-US/alexa/alexa-skills-kit

Apple Inc. (2021). *Speech | Apple Developer Documentation*. https://developer.apple.com/documentation/speech

Argenox Technologies LLC. (2020). *Introduction to Bluetooth Classic*. https://www.argenox.com/library/bluetooth-classic/introduction-to-bluetooth-classic/

Asamblea General de la Organización de las Naciones Unidas. (2008). Convención sobre los derechos de las personas con discapacidad. *Asamblea General de la Organización de las Naciones Unidas*, *20881*(September), 35.

Asamblea Nacional del Ecuador. (2008). *Constitución de la República del Ecuador*. *449*(449), 1–219. www.lexis.com.ec

Bhowmick, A., & Hazarika, S. M. (2017). An insight into assistive technology for the visually impaired and blind people: state-of-the-art and future trends. En *Journal on Multimodal User Interfaces* (Vol. 11, Número 2, pp. 149–172). Springer Verlag. https://doi.org/10.1007/s12193-016-0235-6

Bluetooth S.I.G. (2017). *Mesh Model Specifications*.

Bluetooth S.I.G. (2019a). *Bluetooth Mesh Profile Specification, Revision v 1.0.1*. https://www.bluetooth.com/specifications/specs/mesh-profile-1-0-1/

Bluetooth S.I.G. (2019b). *Qualified Bluetooth Mesh Products*. https://www.bluetooth.com/learn-about-bluetooth/recent-enhancements/mesh/mesh-qualified/

Bluetooth S.I.G. (2019c). *Using BlueZ v5.50 and theRaspberry Pi 4 Update toCreate a Bluetooth MeshProvisioner | Bluetooth® Technology Website*. https://www.bluetooth.com/blog/use-bluez-v5-50-andraspberry-pi-4-updateto-create-a-bluetoothmesh-provisioner/

Bluetooth S.I.G. (2020). *An Introduction to the Bluetooth Mesh Proxy Function*. Bluetooth SIG,Resources,Study Guides. https://www.bluetooth.com/bluetooth-resources/bluetooth-mesh-proxy-kit/?utm_campaign=mesh&utm_source=internal&utm_medium=blog&utm_content=bluetooth-mesh-developer-study-guide-v2.0

Bluetooth S.I.G. (2021). *Bluetooth Technology Overview | Bluetooth® Technology Website*. https://www.bluetooth.com/learn-about-bluetooth/tech-overview/

Brunes, A., & Heir, T. (2021). Serious life events in people with visual impairment versus the general population. *International Journal of Environmental Research and Public Health*, *18*(21). https://doi.org/10.3390/ijerph182111536

Cao, Y., Kandula, H., & Li, X. (2021). Measurement and Analysis of RSS Using Bluetooth Mesh Network for Localization Applications. *Network 2021, Vol. 1, Pages 315-334*, *1*(3), 315–334. https://doi.org/10.3390/NETWORK1030018

Clark, J. O. (2008). System of systems engineering and family of systems engineering from a standards perspective. *2008 IEEE International Conference on System of Systems Engineering, SoSE 2008*. https://doi.org/10.1109/SYSOSE.2008.4724201

Components101. (2018). *TP4056A Li-ion Battery Charging/Discharging Module Pinout, Uses*

& Datasheet. https://components101.com/modules/tp4056a-li-ion-battery-chargingdischarging-module

CONADIS. (2012). *Ley Orgánica de Discapacidades*. https://www.consejodiscapacidades.gob.ec/wp-content/uploads/downloads/2014/02/ley_organica_discapacidades.pdf

Consejo Nacional para la igualdad de Discapacidades. (2022). *Estadísticas de Discapacidad – Consejo Nacional para la Igualdad de Discapacidades*. Ministerio de Salud Pública. https://www.consejodiscapacidades.gob.ec/estadisticas-de-discapacidad/

Darroudi, S. M., Gomez, C., & Crowcroft, J. (2020). Bluetooth Low Energy Mesh Networks: A Standards Perspective. En *IEEE Communications Magazine* (Vol. 58, Número 4, pp. 95–101). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/MCOM.001.1900523

David, C. (2020). *KY-026 Flame Sensor Tutorial for Arduino, ESP8266 and ESP32*. https://diyi0t.com/flame-sensor-arduino-esp8266-esp32/

desensores.com. (2018). *High Sensitivity Water Sensor -Red Version*. https://desensores.com/datasheets/39-Datasheet-Sensor-de-Agua.pdf

Encargo, P. DE. (2018). *LIBRO BLANCO Y NEGRO*.

Espressif Systems. (2020a). *[Reference Design] ESP32-DevKitC-V4 Reference Design r2.1 - ESP32 Forum*. https://www.esp32.com/viewtopic.php?t=13885

Espressif Systems. (2020b). *ESP-BLE-MESH Architecture*. https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/esp-ble-mesh/ble-mesh-architecture.html

Espressif Systems. (2021a). *ESP32-DevKitC V4 Getting Started Guide - ESP32 - — ESP-IDF Programming Guide latest documentation*. https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html

Espressif Systems. (2021b). *ESP32-S3-WROOM-1 ESP32-S3-WROOM-1U Datasheet*. https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf

Espressif Systems. (2021c). *ESP32-WROOM-32D & ESP32-WROOM-32U Datasheet*. https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf

Espressiff Systems. (2021). *ESP-SDK | Espressif Systems*. https://www.espressif.com/en/products/sdks/esp-idf/esp-ble-mesh

Garg, S., & Baliyan, N. (2021). Comparative analysis of Android and iOS from security viewpoint. En *Computer Science Review* (Vol. 40, p. 100372). Elsevier. https://doi.org/10.1016/j.cosrev.2021.100372

Giudice, N. A. (2018). Navigating without vision: Principles of blind spatial cognition. En D.R. Montello (Ed.), *Handbook of Behavioral and Cognitive Geography* (pp. 260–288). MA: Edward Elgar Publishing. https://doi.org/10.4337/9781784717544.00024

Google Cloud Platform. (2022). *Descripción general de los servicios de Google Cloud Platform (GCP)*. https://cloud.google.com/load-balancing/docs/backend-service

Google Developers. (2019). *Platform Architecture | Android Developers*. Developers. https://developer.android.com/guide/platform

Google Developers. (2021). *Android Developers*. https://developer.android.com/reference/android/speech/package-summary

Graessler, I., Hentze, J., & Bruckmann, T. (2018). V-models for interdisciplinary systems engineering. *Proceedings of International Design Conference, DESIGN*, *2*, 747–756. https://doi.org/10.21278/idc.2018.0333

Haase, C. (2021). ANDROIDS (The Team That Built The Android Operating System). En *Angewandte Chemie International Edition, 6(11), 951–952.*

https://www.barnesandnoble.com/w/androids-chet-haase/1140075499
Hajjaji, Y., Boulila, W., Farah, I. R., Romdhani, I., & Hussain, A. (2021). Big data and IoT-based applications in smart environments: A systematic review. *Computer Science Review*, *39*, 100318. https://doi.org/10.1016/J.COSREV.2020.100318
ISO/IEC/IEEE. (2018). IEEE/ISO/IEC 29148-2018 - ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle processes -- Requirements engineering. *IEEE*, 1–104. https://doi.org/10.1109/IEEESTD.2018.8559686
JOY-IT. (2017). *KY-026 Flame-sensor module*. https://moviltronics.com/wp-content/uploads/2019/10/KY-026.pdf
Katz, W. F., & Assmann, P. F. (2019). The routledge handbook of phonetics. En *The Routledge Handbook of Phonetics*. Taylor and Francis. https://doi.org/10.4324/9780429056253
Kumar Pani, S., & Pandey, M. (Eds.). (2021). *Internet of Things: Enabling Technologies, Security and Social Implications*. Springer Singapore. https://doi.org/10.1007/978-981-15-8621-7
Kuriakose, B., Shrestha, R., & Sandnes, F. E. (2020). Tools and Technologies for Blind and Visually Impaired Navigation Support: A Review. En *IETE Technical Review (Institution of Electronics and Telecommunication Engineers, India)* (p. 73). Taylor and Francis Ltd. https://doi.org/10.1080/02564602.2020.1819893
Lee, S. Y., & Yoo, S. E. (2015). Effects of housing conditions and environmental factors on accidents and modification intention of the vision impaired. *Journal of Asian Architecture and Building Engineering*, *14*(2), 347–354. https://doi.org/10.3130/jaabe.14.347
Leporini, B., & Buzzi, M. (2018). Home automation for an independent living: Investigating the needs of visually impaired people. *Proceedings of the 15th Web for All Conference : Internet of Accessible Things, W4A 2018*, 1–9. https://doi.org/10.1145/3192714.3192823
Liu, Y., Tong, K. F., Qiu, X., Liu, Y., & Ding, X. (2017). Wireless Mesh Networks in IoT networks. *2017 International Workshop on Electromagnetics: Applications and Student Innovation Competition, iWEM 2017*, 183–185. https://doi.org/10.1109/iWEM.2017.7968828
Marlin P. Jones & Assoc. Inc. (2011). Hc-Sr501 Pir Motion Detector. En *Marlin P. Jones & Assoc. Inc.* (pp. 3–5). https://www.mpja.com/download/31227sc.pdf
MathWorks. (2021). *Bluetooth Mesh Networking - MATLAB & Simulink - MathWorks América Latina*. https://la.mathworks.com/help/comm/ug/bluetooth-mesh-networking.html
Meike, G. B., & Schiefer, L. (2022). *Inside the android OS : building, customizing, managing and operating android system services*. https://www.oreilly.com/library/view/inside-the-android/9780134096377/
Ministerio de Energía y Minas. (2022). *Las tarifas de energía eléctrica no se incrementarán en el 2022*. https://www.recursosyenergia.gob.ec/las-tarifas-de-energia-electrica-no-se-incrementaran-en-el-2022/
Ministerio del Interior. (2017). Plan Nacional del Buen Vivir. *Plan Nacional*, 1–78. www.gestionderiesgos.gob.ec
Miori, V., Russo, D., & Ferrucci, L. (2019). Interoperability of home automation systems as a critical challenge for IoT. *2019 4th International Conference on Computing, Communications and Security, ICCCS 2019*, 1–7. https://doi.org/10.1109/CCCS.2019.8888125
Mobile Applications Team Nordic Semiconductor ASA. (2022). *Android-nRF-Mesh-Library: The Bluetooth Mesh Provisioner and Configurator library*. https://github.com/NordicSemiconductor/Android-nRF-Mesh-Library
NanJing Top Power ASIC Corp. (2017). *TP4056 1A Standalone Linear Li-Ion Battery Charger with Thermal Regulation in SOP-8 DESCRIPTION*.
Nordic Semiconductor. (2018a). *Basic Bluetooth Mesh concepts*.

https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.meshsdk.v1.0.1%2Fmd_doc_introduction_basic_concepts.html
Nordic Semiconductor. (2018b). *Nordic Thingy:52 - Downloads - nordicsemi.com*. https://www.nordicsemi.com/Products/Development-hardware/Nordic-Thingy-52/Download?lang=en#infotabs
Nordic Semiconductor. (2018c). *Nordic Thingy:52 Product Brief Version 2.0*. https://www.nordicsemi.com/-/media/Software-and-other-downloads/Product-Briefs/Nordic-Thingy52-product-brief.pdf?la=en&hash=4976338D3BEF6549B2C9470834A2EF0094F785D4
Nordic Semiconductor. (2018d). *Nordic Thingy:52 User Guide*. https://infocenter.nordicsemi.com/pdf/Thingy_UG_v1.1.pdf
Nordic Semiconductor. (2020). *Introduction to Bluetooth Low Energy*. 2020-05-14. https://webinars.nordicsemi.com/introduction-to-bluetooth-low-6
Nordic Semiconductor. (2021a). *Bluetooth mesh stack architecture — nRF Connect SDK 1.7.1 documentation*. https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/ug_bt_mesh_architecture.html
Nordic Semiconductor. (2021b). *Welcome to the nRF Connect SDK!* https://developer.nordicsemi.com/nRF_Connect_SDK/doc/1.7.1/nrf/index.html
Nugroho, S., Waluyo, S. H., & Hakim, L. (2017). Comparative Analysis of Software Development Methods between Parallel, V-Shaped and Iterative. *International Journal of Computer Applications*, *169*(11), 7–11. https://doi.org/10.5120/ijca2017914605
Oosako, Y., Ishiura, N., Tomiyama, H., & Kanbara, H. (2019). Synthesis of Full Hardware Implementation of RTOS-Based Systems. *Proceedings - IEEE International Symposium on Rapid System Prototyping, RSP*, *2018-Octob*, 1–7. https://doi.org/10.1109/RSP.2018.8631993
Organización Mundial de la Salud. (2020). Informe mundial sobre la visión. *Organización Mundial de la Salud*, 388. https://apps.who.int/iris/bitstream/handle/10665/331423/9789240000346-spa.pdf
Organización Nacional de Ciegos Españoles. (2012). *Discapacidad visual y autonomía personal*. ONCE. https://sid.usal.es/idocs/F8/FDO26230/discap_visual.pdf
Pacheco, M. (2021). *8,4 puntos en el 2019 subió la penetración de Internet en Ecuador*. El Comercio. https://www.elcomercio.com/actualidad/negocios/internet-ecuador-teletrabajo-conectividad-pandemia.html
Ramadhan, A. J. (2018). Wearable smart system for visually impaired people. *Sensors (Switzerland)*, *18*(3). https://doi.org/10.3390/s18030843
Raspberry Pi Foundation. (2018). *Raspberry Pi 4 Model B (REDUCED)*. Raspberry Pi 4 Model B Schematics. https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-reduced-schematics.pdf
Sale, P. (2018). *Gerontechnology, Domotics, and Robotics* (pp. 161–169). Springer, Cham. https://doi.org/10.1007/978-3-319-57406-6_19
Sauter, M. (2017). From GSM to LTE-Advanced Pro and 5G. En *From GSM to LTE-Advanced Pro and 5G*. Wiley. https://doi.org/10.1002/9781119346913
Seminconductor, N. (2021). *Bluetooth mesh stack architecture — nRF Connect SDK 1.5.99 documentation*. https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/ug_bt_mesh_architecture.html
Sengled. (2021). *Sengled Smart Bluetooth LED Daylight A19 Bulb – Sengled USA*. https://us.sengled.com/products/sengled-smart-bluetooth-led-daylight-a19-bulb
Sharma, C., & Gondhi, N. K. (2018, noviembre 1). Communication Protocol Stack for

Constrained IoT Systems. *Proceedings - 2018 3rd International Conference On Internet of Things: Smart Innovation and Usages, IoT-SIU 2018*. https://doi.org/10.1109/IoT-SIU.2018.8519904

Sharma, S., & Umme Salma, M. (2021). *Social, Medical, and Educational Applications of IoT to Assist Visually Impaired People* (pp. 195–214). Springer, Singapore. https://doi.org/10.1007/978-981-15-4112-4_10

SiliconLabs. (2019). *AN1098: Understanding the Silicon Labs Bluetooth® Mesh Lighting Demonstration in SDK v1.x*. https://www.silabs.com/documents/public/application-notes/an1098-understanding-bluetooth-mesh-lighting-demo.pdf

SiliconLabs. (2021). *Bluetooth Mesh Software Development Kit*. https://www.silabs.com/developers/bluetooth-mesh#

SilvairGit. (2022). *python-bluetooth-mesh*. https://github.com/SilvairGit/python-bluetooth-mesh

Simón, C., Ochaíta, E., & Huertas, J. A. (1995). El sistema Braille: Bases para su enseñanza-aprendizaje. *Comunicación, Lenguaje y Educación*, *7*(4), 91–102. https://doi.org/10.1174/021470395763771891

Singh, R. (2019). Profiling Humans from their Voice. *Profiling Humans from their Voice*. https://doi.org/10.1007/978-981-13-8403-5

Soni, A., Upadhyay, R., & Jain, A. (2017). Internet of Things and Wireless Physical Layer Security: A Survey. En *Lecture Notes in Networks and Systems* (Vol. 5, pp. 115–123). Springer, Singapore. https://doi.org/10.1007/978-981-10-3226-4_11

Sparkfun. (2014). *HANWEI SENSORS MQ-6*. https://cdn.sparkfun.com/datasheets/Sensors/Biometric/MQ-6 Ver1.3 - Manual.pdf?__hstc=175869181.c80e8a76ede39d23c9f7282049f19df0.1635809304281.1635809304281.1635809304281.1&__hssc=175869181.2.1635809304281&__hsfp=2445552440

STMicroelectronics. (2021). *BlueMS - BlueMS Application for Android and iOS - STMicroelectronics*. https://www.st.com/en/embedded-software/stblemesh.html

TechnoElectronics44. (2021). *MQ6-GAS SENSOR*. https://www.technoelectronics44.com/2021/01/MQ6-Gas Sensor.html

Texas Instruments Incorporated. (2020). *Overview — SimpleLink™ CC13x2 / CC26x2 SDK BLE5-Stack User's Guide 2.02.00.00 documentation*. https://software-dl.ti.com/simplelink/esd/simplelink_cc13x2_26x2_sdk/4.40.00.44/exports/docs/ble5stack/ble_user_guide/html/ble-mesh/overview.html#ti-bluetooth-mesh-software-architecture

Universidad de Salamanca. (2019). *Los dispositivos activados por voz en España*. https://universoabierto.org/2019/12/11/los-dispositivos-activados-por-voz-en-espana/

Valenzuela-Pérez, A., García-Lozano, M., Valenzuela, J. L., Pérez-Díaz-de-Cerio, D., Hernández-Solana, & Valdovinos, A. (2022). On the use of sniffers for spectrum occupancy measurements of Bluetooth low energy primary channels. *Measurement: Journal of the International Measurement Confederation*, *199*, 111573. https://doi.org/10.1016/j.measurement.2022.111573

Vanishree, M. L., Sushmitha, S., & Roopa, B. K. (2017). Addressing the challenges of Visually Impaired using IoT. *International Journal on Recent and Innovation Trends in Computing and Communication*, *5*(1), 182–186. http://www.ijritcc.org

Ville, U. (2021). *Smartphones : Global Artificial Intelligence Technologies Forecast to 2025*. https://www.strategyanalytics.com/access-services/devices/mobile-phones/emerging-device-technologies/market-data/report-detail/smartphones-global-artificial-intelligence-technologies-forecast-to-2025

Woolley, M. (2019). *Bluetooth Mesh Models Technical Overview*. *March*, 1–41. https://www.bluetooth.com/wp-content/uploads/2019/04/1903_Mesh-Models-

Overview_FINAL.pdf
World Health Organization. (2013). Informe mundial sobre la discapacidad. *Convergencia Educativa*, 1–388. http://www.who.int/about/licensing/
XLSEMI. (2017). *Datasheet 5A 180KHz 36V Buck DC to DC Converter XL4015*. http://www.xlsemi.com/datasheet/xl4015 datasheet.pdf
YHDC. (2018). *SCT-013*. https://www.mcielectronics.cl/website_MCI/static/documents/Datasheet_SCT013.pdf
Yu, D., & Deng, L. (2014). Automatic Speech Recognition: A Deep Learning Approach. En *Springer*. https://github.com/weimeng23/Documents/blob/master/%5BDong Yu, Li Deng%5DAutomatic Speech Recognition A Deep Learning Approach(pdf)%7BZzzzz%7D.pdf
Zephyr Project. (2021). *Bluetooth Overview Documentation*. https://docs.zephyrproject.org/latest/guides/bluetooth/overview.html
Zumba, J. P., & Arreaga, C. A. L. (2018). Evolución de las metodologías y modelos utilizados en el desarrollo de software. *INNOVA Research Journal*, *3*(10), 20–33. https://doi.org/10.33890/INNOVA.V3.N10.2018.651

Glosario de términos

ADC : Convertidor analógico a digital por sus siglas en inglés Analog to Digital Converter. Presente en la mayoría de las placas de desarrollo actuales ,permiten la integración de sensores analógicos a las placas mencionadas.

BLE : Siglas que representan la tecnología de comunicación inalámbrica Bluetooth Low energy.

Bluetooth SIG : Grupo de interés especial por sus siglas en inglés Special Interest Group, se trata de la organización encargada de la aprobación de normativas para la tecnología de comunicación inalámbrica homónima .

ECDH : Protocolo de establecimiento de llaves de seguridad entre dos dispositivos a través de un canal no seguro. En el estándar de comunicación de red ,es utilizado al momento de aprovisionar un nuevo elemento en la red .

GFSK: Modulación por salto de frecuencia usando un filtro gaussiano ,es la modulación usada por la radio de Bluetooth desde las primeras versiones .Específicamente en la version 1.1.

SDK: Kit de desarrollo de software por sus siglas en inglés ,se trata de un grupo de herramientas que sirven para el desarrollo de aplicaciones que pueden ser instalados bajo un mismo paquete. Por lo general contienen un compilador ,un depurador y en algunos casos un framework para el desarrollo y Debug de aplicaciones .

CRC: Control de redundancia clínica por sus siglas en inglés ,es usado en Networking para el control de errores en capa de red .En el estándar de comunicación usado tiene un tamaño de 24 bits y es indexado por el transmisor .Se comprueba su valor en el receptor

CSRK: Parámetro utilizado para autenticar conexiones en Bluetooth Low energy ,generalmente es usado en las conexiones para evitar ataques de replay después de establecer la conexión.

FHSS: tipo de modulación utilizada en capa física en Bluetooth Low Energy ,se utiliza saltos de frecuencia en un espectro ensanchado con el fin de optimizar el uso del medio físico ,utilizando los 40 caneles definidos en el estándar de comunicación de red bluetooth mesh.

GAP: Parámetro que define el rol del dispositivo dentro de una topología de red ,que utilizan como tecnología de comunicación Bluetooth Low Energy. Se define dos tipos de dispositivos :Broadcasting y Connecting

GATT: Parámetro que define como se trata a la información que es transmitida por dispositivos que han establecido una conexión previa por medio de BLE. Define roles mucho más complejos como cliente y servidor en una topología de red.

HCI: Capa del stack de BLE que define la interacción entre las capas de los bloques host y controlador en un dispositivo que posee una interfaz bluetooth.

CONTROLLER :bloque del Bluetooth Stack encargado de controlar la capa física y enlace de un nodo bluetooth ,generalmente se trata de un chip dentro del dispositivo.

HOST: bloque del Bluetooth Stack encargado del funcionamiento del stack de bluetooth dentro de un dispositivo ,generalmente se hace en el cerebro del aparato.

IoT: Termino usado para definir las nuevas redes de dispositivos que interconectan todos los dispositivos conocidos dentro de un entorno que envían datos al internet y se comunican entre ellos.

L2CAP: Protocolo de bloque controlador en el stack de bluetooth low energy ,se encarga proveer servicios desde las capas del bloque de host a las capas del bloque de controlador .

LESC :característica de seguridad para conexiones BLE introducida en la version 4.2 de Bluetooth ,mejora la seguridad cuando se utiliza GAP.

LTK : Llave de seguridad que se guarda en un dispositivo bluetooth después del establecimiento de una conexión .Esta llave varia si se usa LESC.

MTU: Tamaño máximo de un paquete que se puede transmitir a través de una conexión de datos ,generalmente se define en bytes.

OOB -Out off band :comunicación a través de BLE usada para el intercambio de llaves de entre dos dispositivos BLE para establecer el canal de comunicación seguro .

PAYLOAD: Carga útil de un paquete de red ,en bluetooth mesh es el mensaje por enviar encapsulado en la capa de red .

APROVISIONADOR : Elemento de red que se encarga de gestionar y añadir nodos dentro de una red Bluetooth mesh .Guarda las llaves de red y aplicación a asignarse a cada uno de los nodos de la red .Debe soportar la tecnología de comunicación Bluetooth en la version 4.2 y superiores .

TTL: Tiempo de vida de un paquete dentro de la red Bluetooth mesh ,corresponde al número de retransmisiones máximas que un nodo Relay puede retransmitir un paquete recibido .Se resta uno cada vez que se realiza una retransmisión.

Anexos

Anexo 1 Formato de entrevista para obtención de requerimientos del sistema

Entrevista población objetivo del Sistema Domótico para personas con discapacidad visual usando una bluetooth mesh

Objetivo : Obtener datos de la persona con discapacidad visual acerca de interacción previa con sistemas IoT, así como expectativas ,gustos y preferencias de este tipo de sistemas .

Preguntas informativas

Buen día gracias por su tiempo soy Luis Farinango estudiante de la Universidad técnica del Norte Carrera Electrónica y Redes de Comunicación por favor ayúdeme con su nombre y su edad.

Vamos a proceder con la entrevista .

Preguntas acerca de las dificultades de las personas no videntes dentro del hogar

1. ¿Cuáles son las principales dificultades que enfrenta al realizar las actividades cotidianas dentro de su vivienda?
2. ¿Cuándo se presenta un problema con un dispositivo eléctrico o sistema eléctrico dentro de su vivienda realiza una inspección manual del aparato?
3. ¿Vive con familiares ?
4. ¿Como afecta a su movilidad las acciones de sus familiares?
5. ¿Ha sufrido lesiones a causa de estas actividades y de qué tipo de ser el caso ?

Preguntas acerca de la interacción previa con un sistema IoT

6. ¿Dispone de servicio de internet ?
7. ¿Utiliza algún sistema o herramienta tecnológica que le permita movilizarse sin temor sufrir lesiones por las causas expuestas anteriormente ?

8. ¿Estaría dispuesto a invertir en un sistema que mediante su teléfono celular le permita controlar los ambientes de su hogar para reducir significativamente los riesgos de su entorno ?

Gracias por su valiosa ayuda , la información proporcionada se utilizará en la tesis Sistema Domótico para personas con discapacidad visual usando una bluetooth mesh

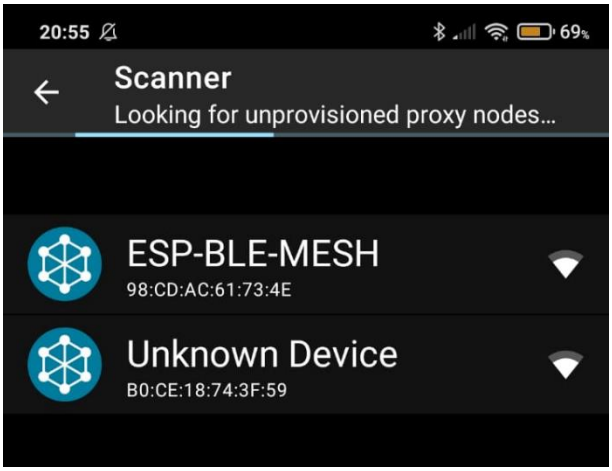
El audio de la entrevista realizada se encuentra en https://utneduec-my.sharepoint.com/:f:/g/personal/lffarinangot1_utn_edu_ec/EtVusnONCR1CoMH2hQMoaU8B6030xF7HbrQxxdSJBttJVQ?e=G6MHhg

Anexo 2 Aprovisionamiento usando la aplicación móvil

La aplicación permite realizar el aprovisionamiento de equipos que funcionen con la tecnología Bluetooth Low Energy usando el esquema de red de Bluetooth mesh .En la pantalla principal de la aplicación aparecerán los nodos que ya han sido aprovisionados .Para mostrar el procedimiento realizado se usara el SmartBuld

Para realizar el aprovisionamiento de un nuevo nodo se presiona el botón Add Node ,este botón realizara un scanner de los nodos que no están aprovisionados de acuerdo al alcance de la interfaz Bluetooth del smartphone utilizado .Se incluye el nombre del nodo ,en el caso de que este haya sido asignado y la direccion MAC .Se puede observar el listado de nodos no aprovisionados en la Figura 157

Figura 157. Listado de nodos sin aprovisionar en la aplicación nRF Mesh

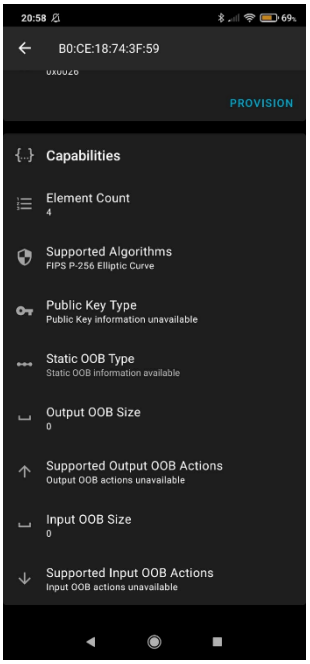


Fuente : Autoría

Si se selecciona el dispositivo que se va a aprovisionar entonces se mostrara la información de aprovisionamiento como :nombre del dispositivo ,llaves para encriptación de aplicación ,direccion unicast del dispositivo, número de elementos y otros parámetros más que se observan en la

Figura *158*.

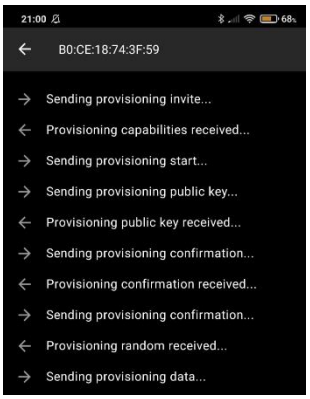
Figura 158. Información de un nodo sin aprovisionar en la aplicación nRF Mesh



Fuente : Autoría

Para comenzar con el aprovisionamiento se presiona la opción PROVISION ,el proceso se iniciará ,se solicita que tipo de OOB se va a utilizar. Con el fin de demostrar el aprovisionamiento no se utilizará ningún OBB Al terminar esta configuración entonces se muestra en pantalla todos los procesos realizados para realizar el aprovisionamiento como se observa en la Figura 159.

Figura 159 Finalización del proceso de aprovisionamiento de un nodo usando la aplicación nRF Mesh.

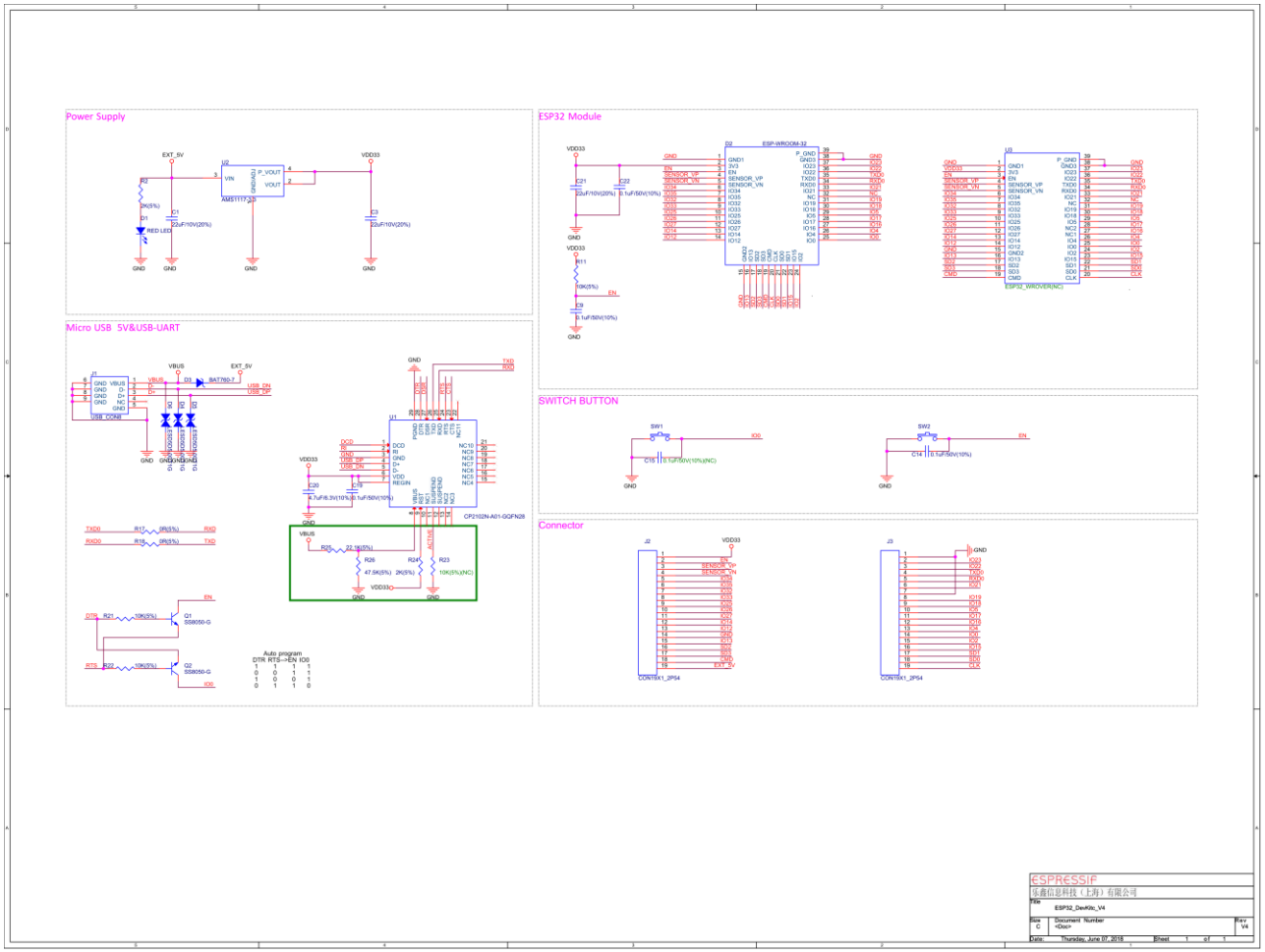


Fuente : Autoría

El nuevo nodo aparecerá en la lista de dispositivos previamente aprovisionados y se podrá acceder a las configuraciones para: obtención de datos, publicación-suscripción entre otros más.

Anexo 3 Diagrama esquemático de la placa de desarrollo ESP32 DevKitC V4

Figura 160. Grafica de diagrama esquemático de la placa ESP32 utilizada

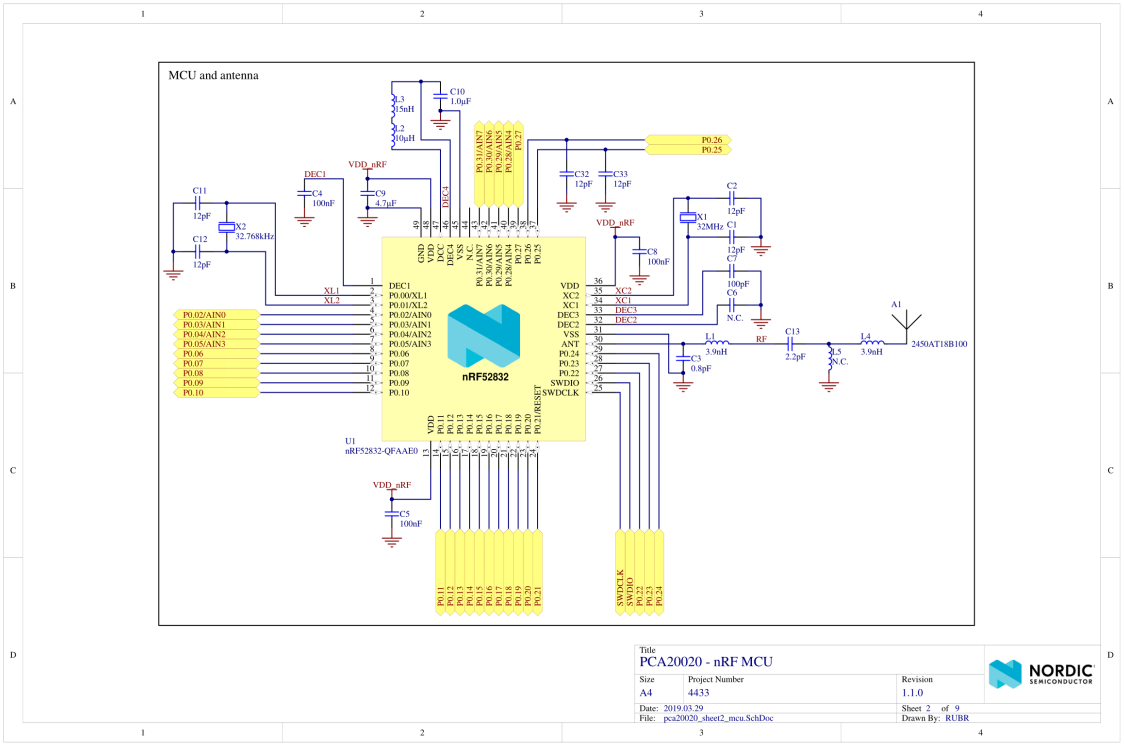


Fuente : (Espressif Systems, 2020a), [Reference Design] ESP32-DevKitC-V4 Reference Design r2.1 - ESP32 Forum. Recuperado el 31 de octubre de 2021, de https://www.esp32.com/viewtopic.php?t=13885

Anexo 4 : Diagrama esquemático de la placa de desarrollo Nordic Thingy :52

Se presenta solamente el diagrama esquemático del MCU la placa de desarrollo Nordic Thingy :52 ,la documentación oficial ofrece mayor información.

Figura 161. Grafica de diagrama esquemático del MCU de la placa de desarrollo Thingy 52

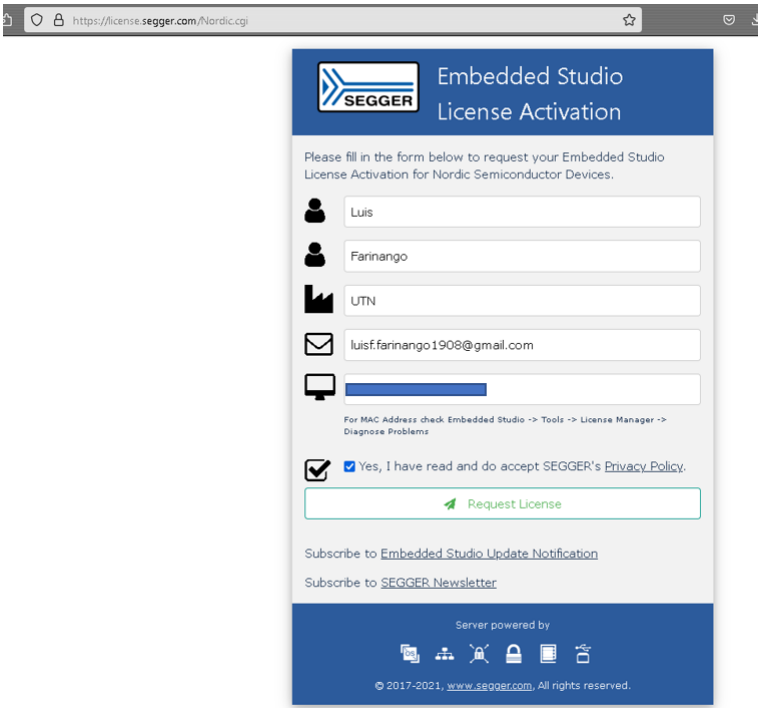


Fuente :(Nordic Semiconductor, 2018b), Nordic Thingy:52 - Downloads - nordicsemi.com. Recuperado el 17 de noviembre de 2021, de https://www.nordicsemi.com/Products/Development-hardware/Nordic-Thingy-52/Download?lang=en#infotabs

Anexo 5 : Configuraciones básicas de la placa de desarrollo Nordic Thingy :52 en SEGGER Embedded Studio

Las configuraciones por realizar se basaron en la implementación de la licencia que Nordic brinda para el software SEGGER Embedded Studio, como primer paso se solicita una licencia en la página oficial de SEGGER .La página solicita datos únicos del usuario que va a utilizar el dispositivo como se observa en la Figura 162.

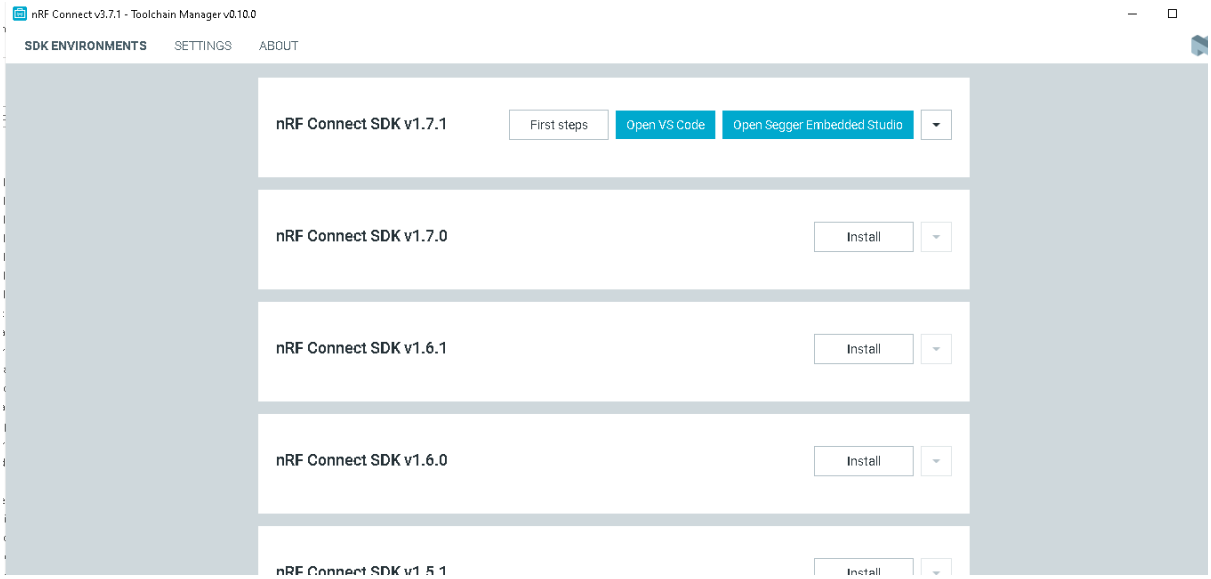
Figura 162. Solicitud de licencia de Nordic para el programa SEGGER Embedded Studio.



Fuente :Autoría

Un numero de licencia sera enviado al correo que fue colocado en la solicitud ,este código al ser ingresado permitira usar algunos firmware y codigos de ejemplos propios de equipos de la marca Nordic.Tambien es posible configurar al software Visual Studio Code para configurar los equipos .Se puede comprobar que las configuraciones han sido exitosas al abrir el programa nrfConnect >Toolchain,donde se podra configurar el equipo usando usando Visual Studio Code o Segger Embebed Studio.

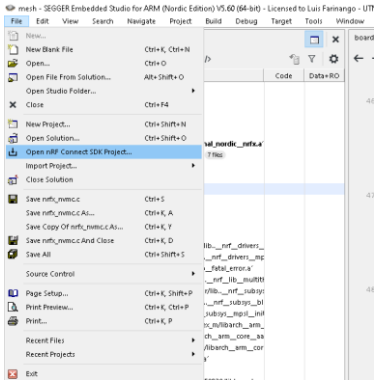
Figura 163. Opciones para configurar equipos de la marca Nordic desde la herramienta Toolchain.



Fuente :Autoría

La marca Nordic recomienda el uso de SEGGER Embedded Studio por lo que esta será la opción a utilizar ,como siguiente paso de configuración se procede a abrir el programa en cuestión. Se busca la opción Open nRF Connect SDK Project ,esta opción permite buscar ejemplos de configuración de los equipos con los que se va a trabajar. En la Figura 164 se puede observar donde acceder a la opción .

Figura 164. Opción nRF Connect SDK Project.

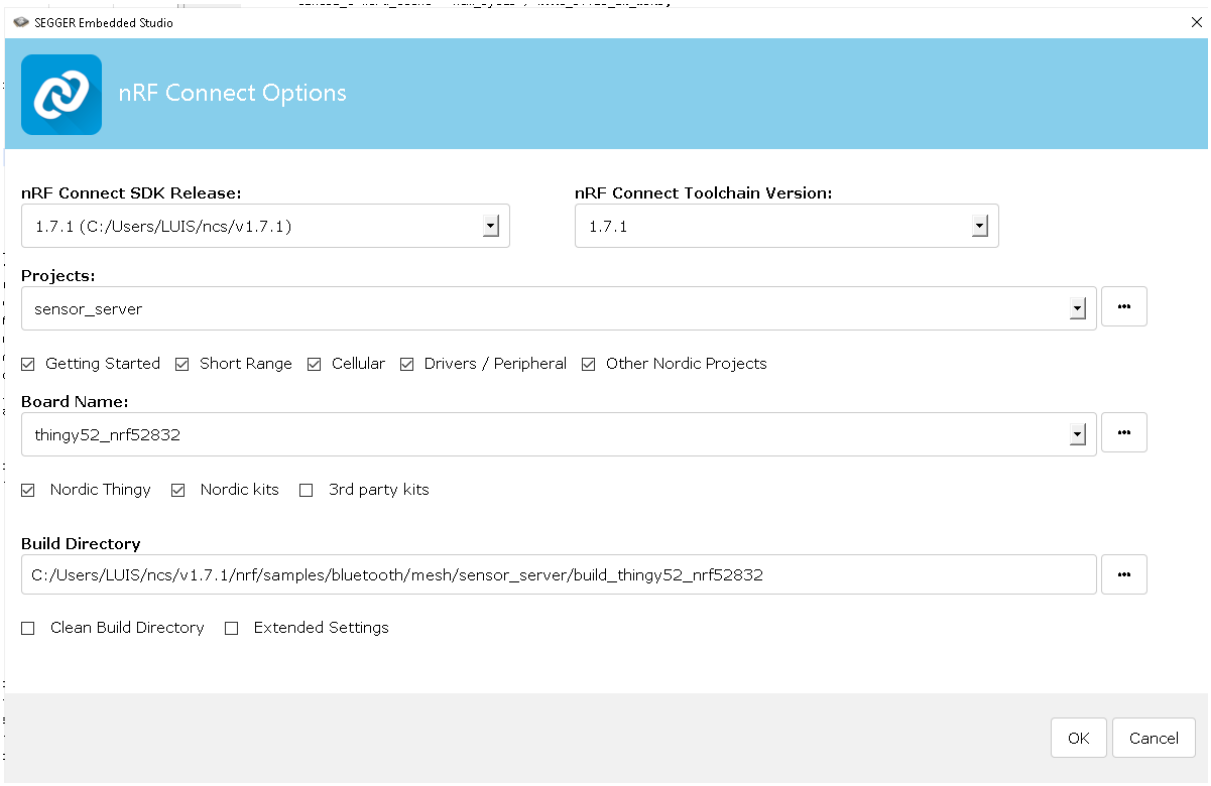


Fuente :Autoría

La nueva ventana permite elegir :la version del software nRF Connect a la cual comectarse ,la version de la herramienta ToolChain ,el proyecto de ejemplo a usar ,el nombre

de la tarjeta que se va a configura y otras opciones mas que se pueden observar en la Figura 165

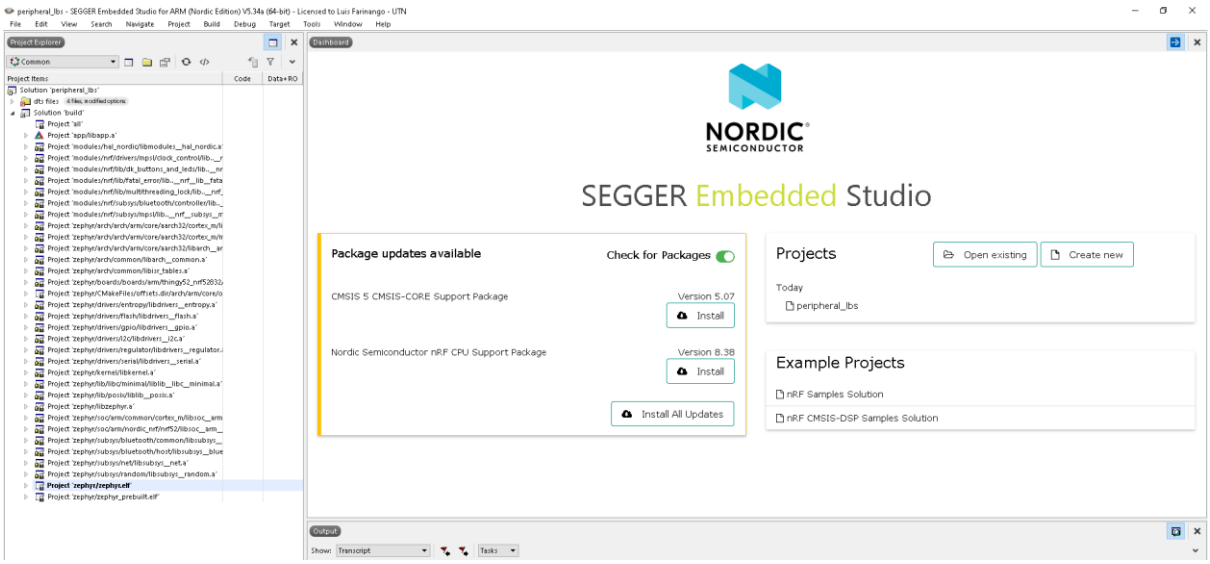
Figura 165. Integración del programa nRF Connect con SEGGER Embebed Studio.



Fuente :Autoría

Al finalizar este proceso se prodra acceder al los codigos de ejemplo y se podra iniciar con la configuracion del dispositivo una vez sea conectado al PC usando un modulo J-Link Debug Probes.La pantalla de inicio se puede observa en la Figura 166

Figura 166. Pantalla de inicio del software SEGGER Embedded Studio después de realizar las configuraciones .

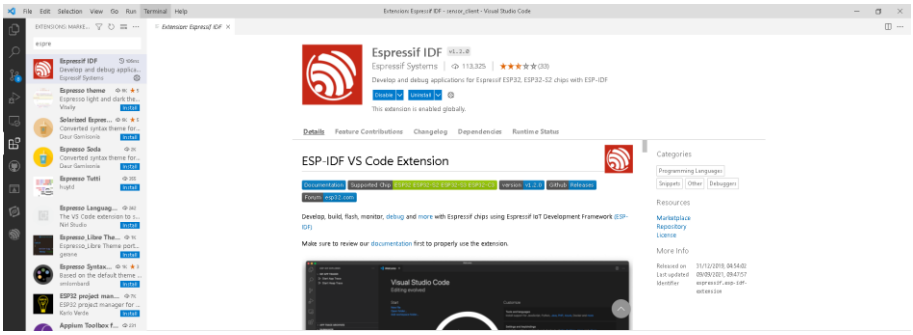


Fuente :Autoría

Anexo 6 Configuraciones básicas de la placa de desarrollo ESP32 en ESP-IDF usando Visual Studio Code

Espressif recomienda el uso del IDE de Eclipse o Visual Studio Code para la subida de código y configuración de placas de desarrollo ESP32 en todas las versiones .Debido a la facilidad de uso se ha elegido la opción de Visual Studio Code para configurar las placas ESP32. Para configurar el software se debe utilizar un complemento .Hay que recalcar que antes de realizar este proceso se ha instalado la herramienta ESP-IDF con éxito .El complemento está disponible en la tienda de extensiones de Visual Studio Code bajo el nombre Espressif IDF .En la Figura 167 se puede observar la extensión dentro de la tienda de aplicaciones.

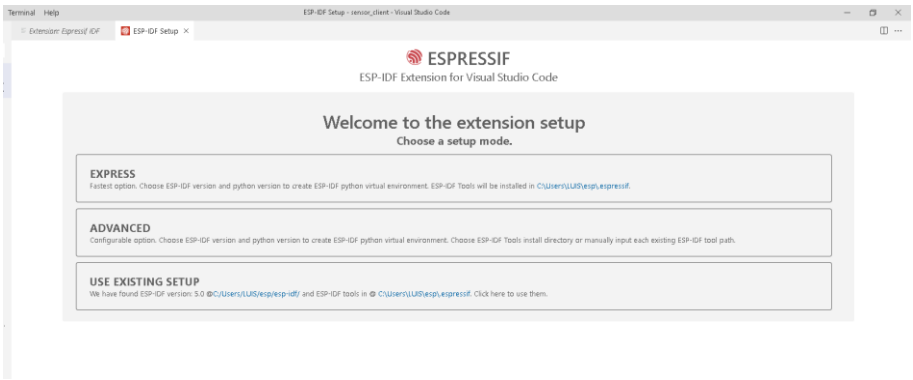
Figura 167. Extensión Espressif IDF en Visual Studio Code .



Fuente :Autoría

Cuando la extensión haya terminado el proceso de instalación se usa el comando ESP-IDF Configure-ESP-IDF extensión. Esta opción permite instalar el entorno ESP-IDF, así como realizar configuraciones si ya se ha instalado la aplicación .Considerando que ya se ha instalado se escoge la opción de usar la configuración existente .En la Figura 168 se puede observar la lista de opciones disponibles para configuración.

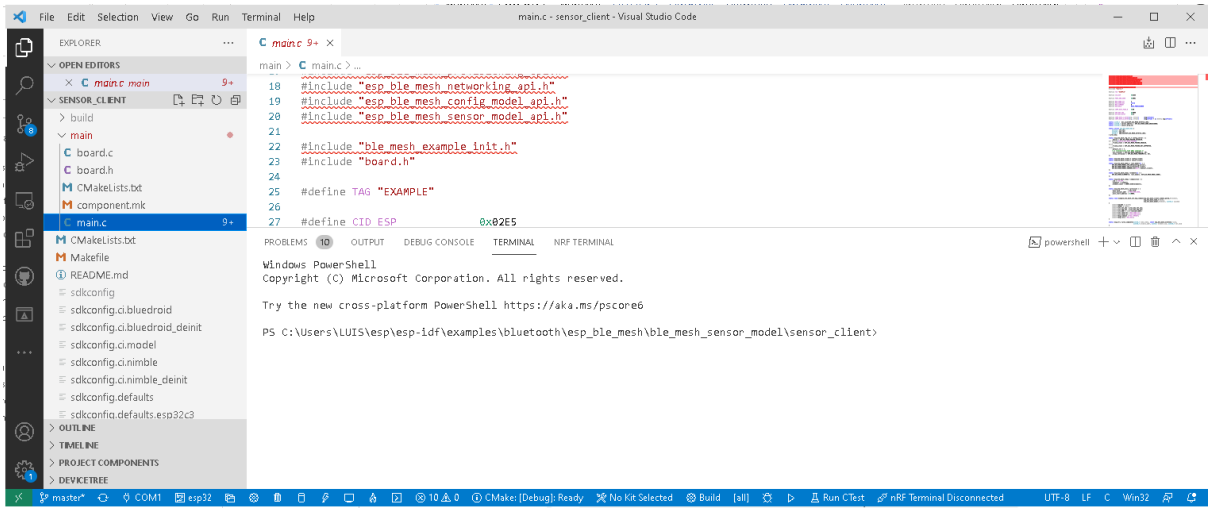
Figura 168. Opciones de configuración de la extensión Espressif IDF en Visual Studio Code .



Fuente :Autoría

Al terminar este proceso aparecerá en la zona inferior de la ventana todas las opciones que brinda ESP-IDF .La recomendación del fabricante es usar el entorno de Visual Studio como editor de código, flasheo de programas y el entorno propio para monitoreo de las placas de desarrollo . En Figura 169 la se puede observar la ventana de Visual Studio Code luego de instalar y configurar la extensión de Espressif.

Figura 169. Ejemplo Sensor Cliente de Espressif IDF en Visual Studio Code .



Fuente :Autoría

Como configuración final se debe indicar a Visual Studio Code donde buscar los ficheros y librerías que se usan en los firmwares .Para ello se debe generar un archivo de extensión .json .este archivo debe incluir cada una de las rutas de las librerías que el código necesita para ejecutarse correctamente, si las rutas no han sido añadidas se muestran advertencias como en la Figura 169 .En la se observa las rutas añadidas para evitar errores al momento de importación de librerías.

Figura 170. Agregación de rutas de las librerías usadas en los proyectos de ESP-IDF en Visual Studio Code .

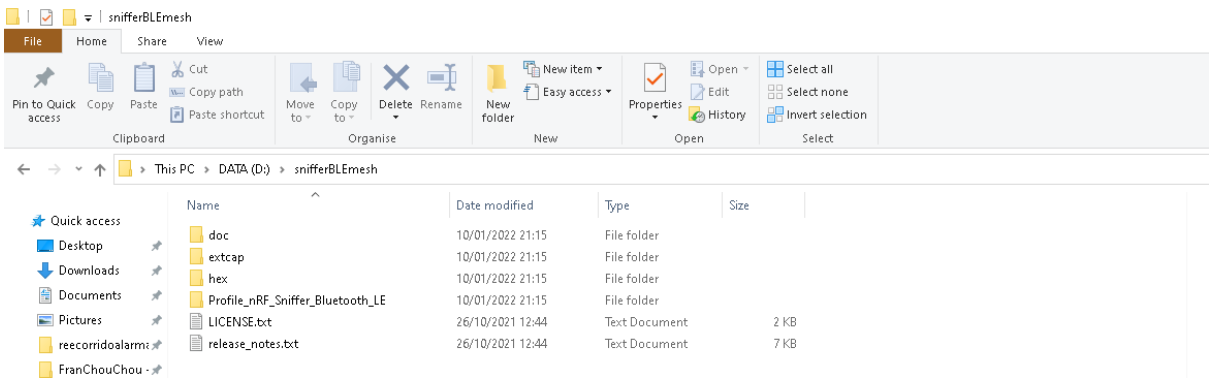
Fuente :Autoría

Anexo 7 Configuración de WireShark para realizar la captura de paquetes de la red.

Para realizar la captura de paquetes es necesario de ciertas configuraciones en wireshark ,también es necesario el uso de hardware adicional y algunos paquetes en el equipo donde se realizará este proceso .El dispositivo usado para la captura de paquetes es el USB Dongle nRF 25382/ nRF 52840 ,las herramientas necesarias en el equipo a donde se conectará el dispositivo USB son : Python y el software oficial de Nordic nRF Sniffer for Bluetooth LE. Esta sección tratara de las configuraciones realizadas para lograr realizar un sniffer de la red Bluetooth mesh.

Primeramente, se descarga la herramienta nRF Sniffer for Bluetooth LE desde la página oficial de Nordic Semiconductor y se descomprime el contenido en una carpeta ,la ruta de esta carpeta será necesaria posteriormente para la configuración en Wireshark. En la se puede observar los archivos que contiene la herramienta.

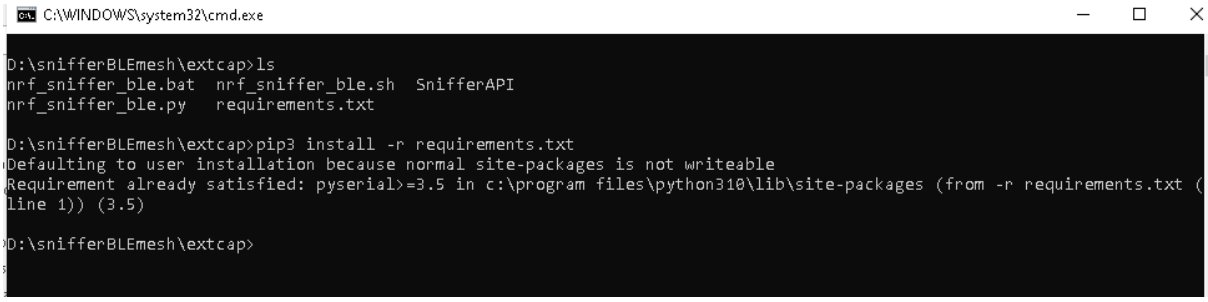
Figura 171.Archivos de la herramienta para utilizar el dongle nRF 52840 como sniffer de una red Bluetooth mesh.



Fuente :Autoría

Dentro del fichero descargado se abre una terminal en la carpeta extcap ,se usa el comando *pip3 install -r requirements.txt* para instalar todas las dependencias necesarias para el correcto funcionamiento del paquete .En este caso ya se cumplieron los requisitos por lo que se obtiene el mensaje de la

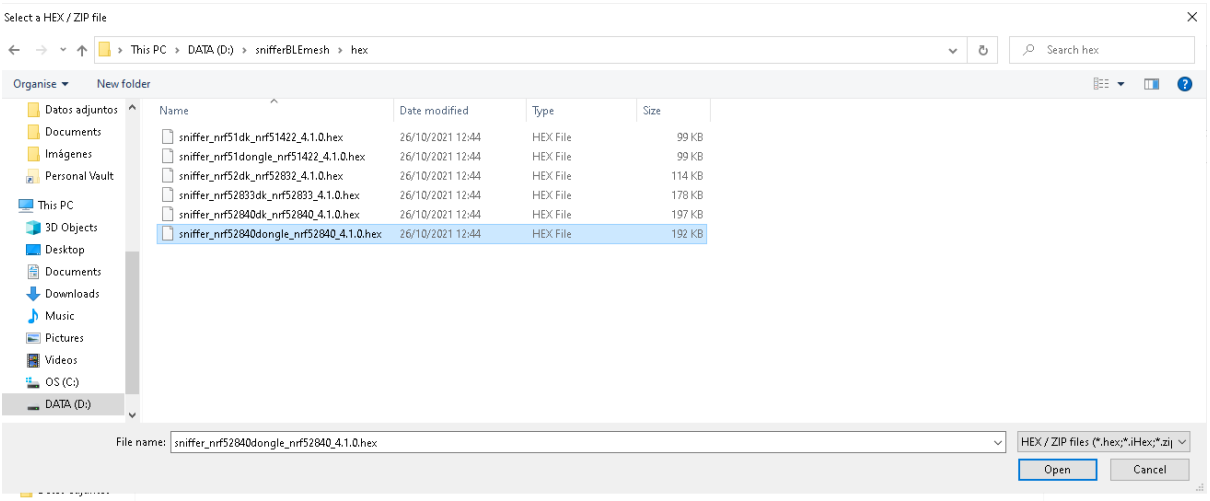
Figura 172.Mensaje que confirma el cumplimiento de los requerimientos de la herramienta nRF Sniffer for Bluetooth



Fuente :Autoría

A continuación, se debe subir el código necesario al hardware a utilizar ,se debe utilizar un archivo con extensión .hex que está incluido en la herramienta descargada de Nordic Semiconductor .Se debe utilizar el archivo correspondiente al modelo utilizado ,se muestra el archivo en la

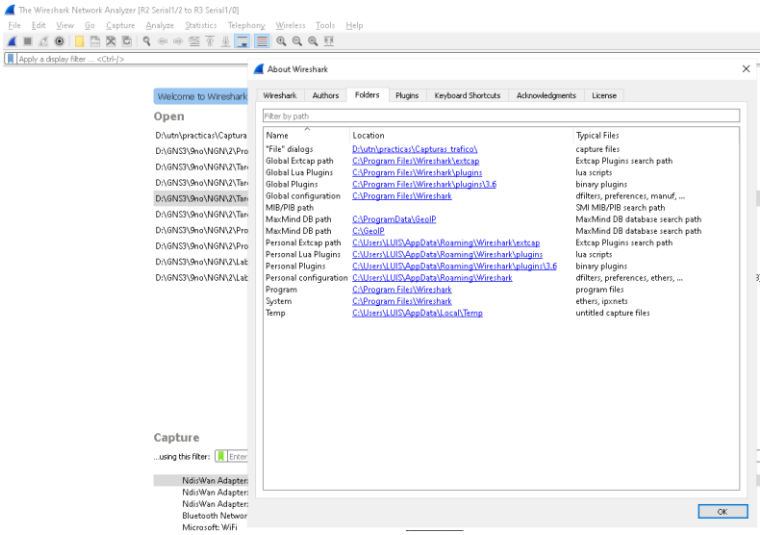
Figura 173.Archivo con extensión .hex que se subirá al dispositivo.



Fuente :Autoría

Con todas estas configuraciones realizadas se procede a abrir WireShark, se selecciona la pestaña Help > Folders .En la ruta de la sección Global Extcap path se realiza la copia de los archivos de la carpeta extacap de la herramienta nRF Sniffer for Bluetooth LE. Se puede observar el proceso en la

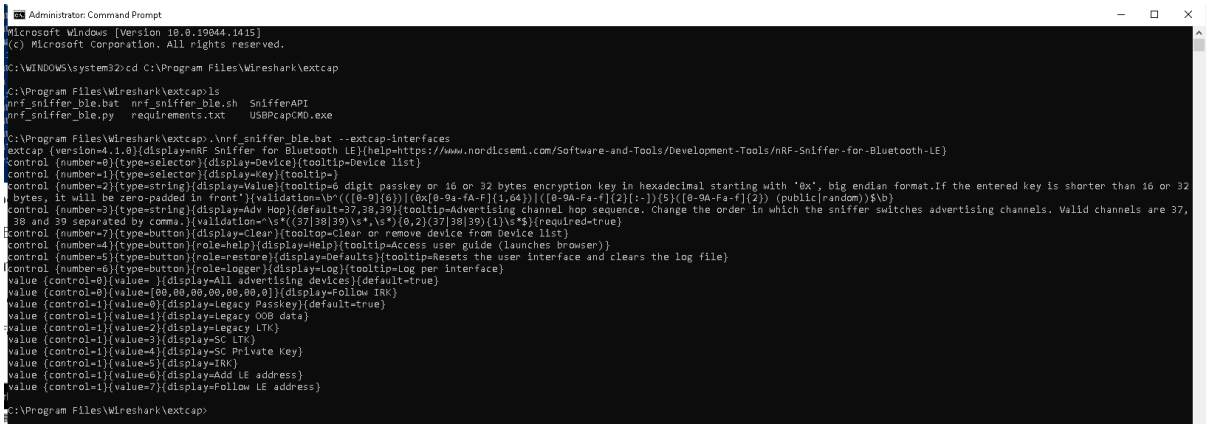
Figura 174. Adición de perfil para capturar paquetes del estándar de comunicación Bluetooth mesh



Fuente :Autoría

Para comprobar el funcionamiento de la herramienta se usa el comando *.\nrf_sniffer_ble.bat --extcap-interfaces* ,si el resultado obtenido es similar a la entonces las configuraciones se han realizado de forma correcta hasta el momento. El resultado del comando se puede observar

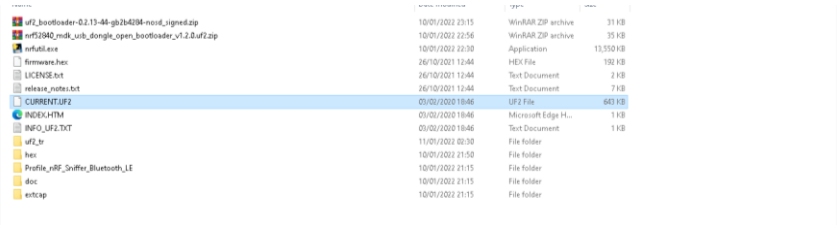
Figura 175.Resultado del comando utilizado para comprobar la correcta instalación de la herramienta.



Fuente :Autoría

Finalmente queda transformar este archivo a una extensión que sea posible subir al dongle utilizado ,para finalmente poder usarlo como sniffer de red .Este proceso puede realizarse por separado tomando en cuenta el sistema operativo en donde se va a utilizar la herramienta. En la se puede observar el archivo a subir al dongle USB.

Figura 176.Archivo subido al dongle USB.

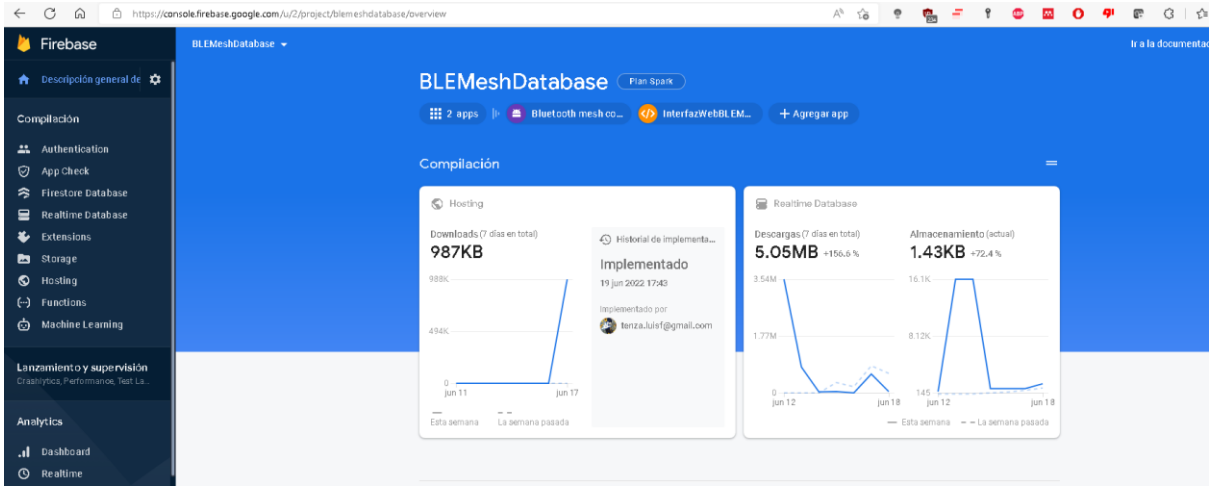


Fuente :Autoría

Anexo 8 Proyecto creado en Firebase

Para la creación del proyecto en Firebase ,se sigue los pasos del asistente .Después de la creación del proyecto y, la respectiva vinculación de la aplicación web y aplicación Android se podrá observar al proyecto de acuerdo a la Figura *177* .

Figura 177.Información del proyecto creado en Firebase.

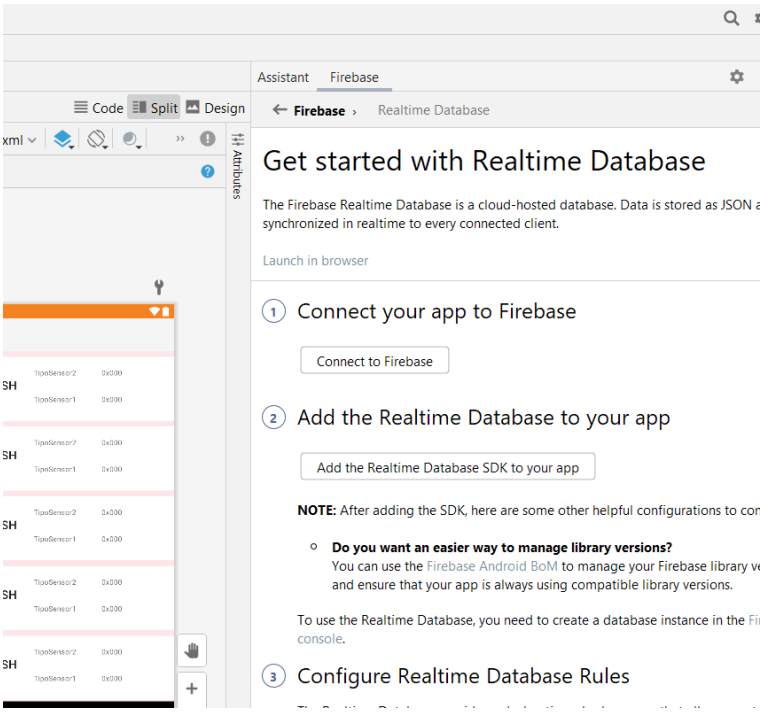


Fuente :Autoría

Anexo 9 Conexión de Firebase a una aplicación en Android Studio

Android Studio permite conectar una aplicación a un proyecto de Firebase de una forma muy sencilla ,se realiza clic en Tools >Firebase .Aparecerá un menú que permitirá realizar la conexión .Se busca la opción de Real Database seguido de la opción Get star with real time Database .Esta opción permitirá conectar la aplicación a Firebase ,al escoger esta opción se abrirá una ventana de un navegador donde se deberá escoger el proyecto a vincular .

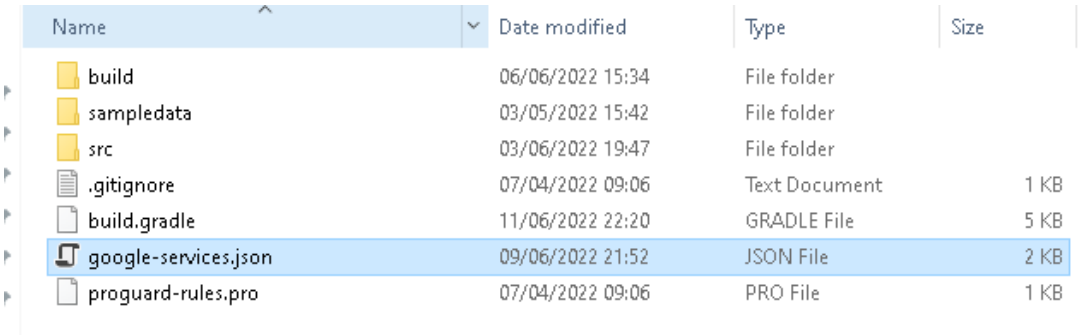
Figura 178. Pantalla de ayuda de Firebase presente en Android Studio



Fuente :Autoría

En esta ventana aparecerán las instrucciones a seguir ,como resultado se obtendrá un archivo llamado google-services.json que contendrá todos los datos necesarios para realizar la vinculación de aplicación móvil con el proyecto de Firebase .Este archivo debe colocarse en la carpeta del proyecto de la aplicación de acuerdo a la .

Figura 179. Archivo generado por Firebase que se debe utilizar para la integración con un proyecto en Android.



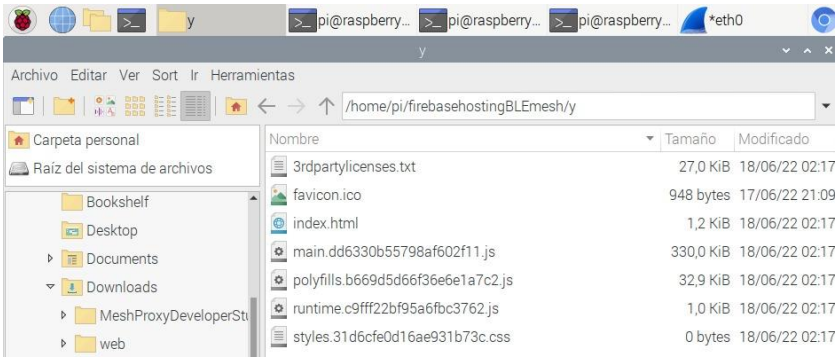
Fuente :Autoría

Anexo 11. Despliegue de Firebase Hosting en el Raspberry Pi

El uso de BaaS permite levantar aplicaciones web de una forma fácil y sencilla utilizando Raspberry Pi o cualquier computadora ,solo se debe crear una carpeta .Esta carpeta contendrá a futuro la aplicación móvil .Esta carpeta deberá vincularse con un proyecto en

Firebase .En la se puede observar el contenido de la carpeta asociada al proyecto de Firebase creado.

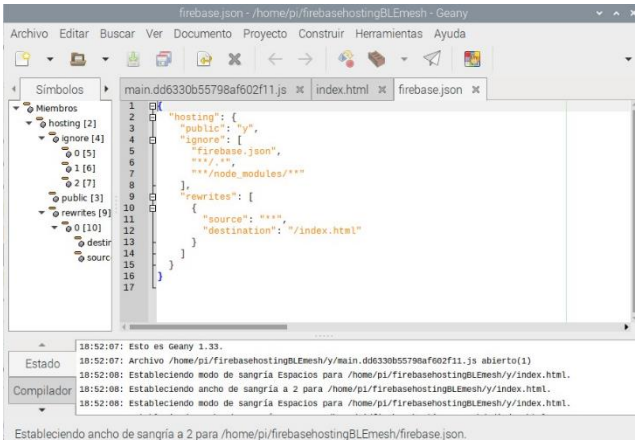
Figura 180.Contenido de la aplicación Web que se va a subir a Firebase.



Fuente :Autoría

Cuando se vincule la carpeta se realiza una pregunta para enrutar la aplicación a un archivo llamado index.html ,se debe escoger la opción afirmativa .Esta opción generará un archivo con extensión .json con las instrucciones de hosting y ruteo de la aplicación web .En la se puede observar el archivo generado en cuestión.

Figura 181 .Archivo de ruteo de la aplicación web .

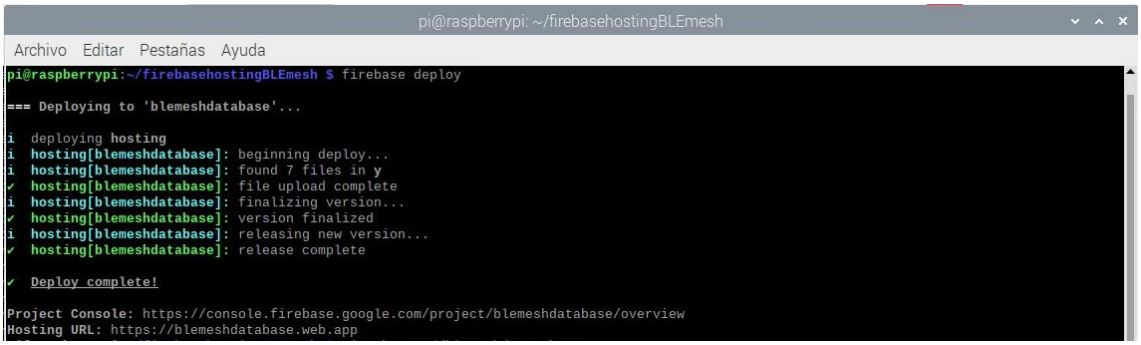


Fuente :Autoría

Con el proyecto creado y vinculado se procede a subir la aplicación web al hosting del proyecto ,se utiliza el comando *firebase deploy* .En caso de existir archivos modificados se subirán los cambios y se obtendrá la direccion del dominio ,al cual se podrá entrar para

comprobar el correcto funcionamiento de la aplicación .En la se puede comprobar el resultado del uso del comando en cuestión.

Figura 182.Despliegue de la aplicación web al hosting de Firebase.



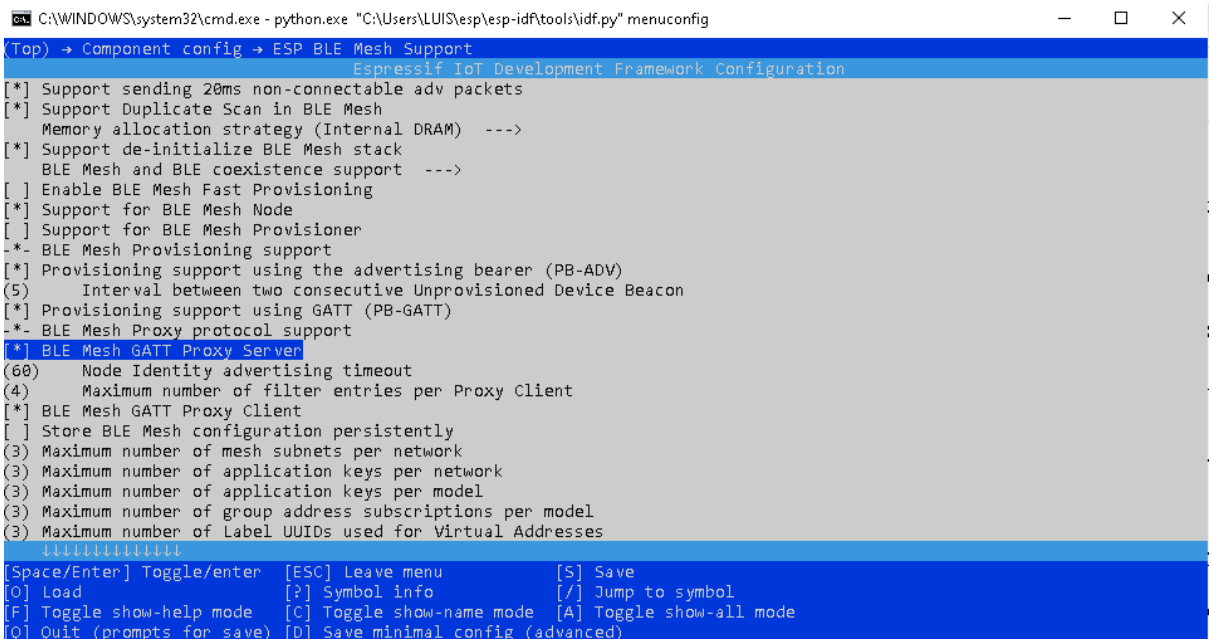
Fuente :Autoría

Se puede ingresar a la URL https://blemeshdatabase.web.app/ para observar la aplicación web desplegada .

Anexo 10 Configuración de nodo proxy en el entorno ESP-IDF

Un nodo proxy dentro de la red permite la comunicación de un dispositivo que no soporta el estándar Bluetooth mesh con una red que funciona baja dicho estándar. Un nodo puede cumplir varios roles por lo que solo es necesario activar esta funcionalidad cuando se realiza la configuración de los parámetros de aprovisionamiento. Se utiliza un protocolo llamado Mesh Proxy Protocol, el nodo tendrá el rol de servidor y el teléfono móvil con la aplicación móvil será el cliente. En la figura a continuación se puede observar la activación de soporte del protocolo en un equipo ESP32.

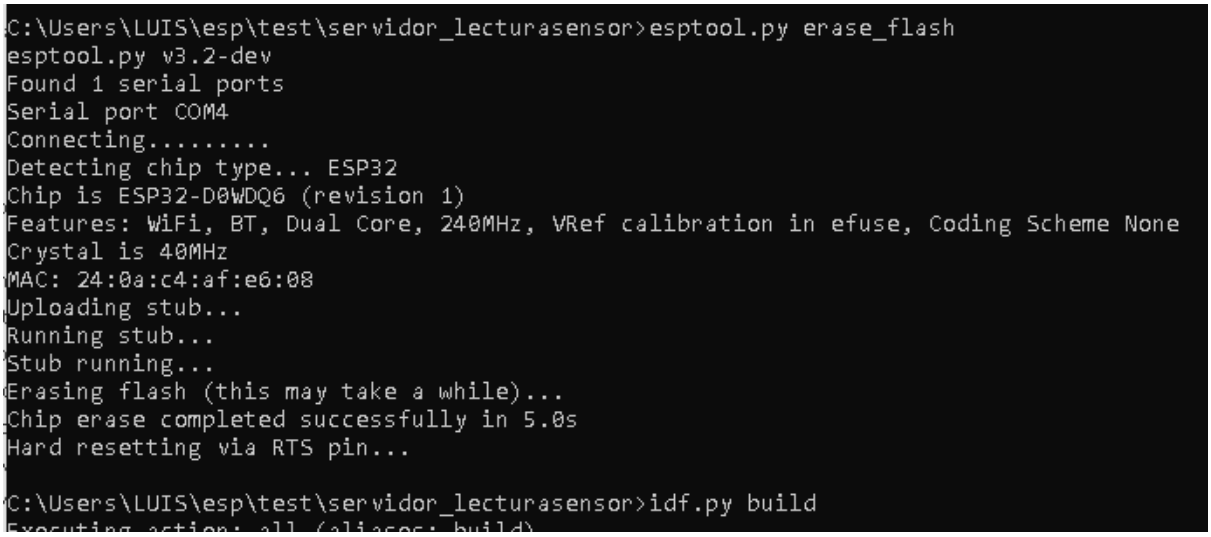
Figura 183. Activación de la función Proxy usando ESP-BLE mesh en una placa ESP32.



Fuente: Autoría

En algunos casos es necesario borrar la memoria flash de las placas de desarrollo utilizadas ,para lo cual se usa el comando :esptool.py erase_flash,este comando ha sido utilizado debido a que en algunos casos la información de red se quedaba guardada dentro de la memoria del dispositivo y no permitía realizar la configuración de red nuevamente ,el uso del comando se puede observar en la

Figura 184. Uso del comando para borrar memoria flash del dispositivo esp32.



Fuente: Autoría

Anexo 11 :Código utilizado en la placa de desarrollo ESP32

```
/* main.c - Application main entry point */
```

```c
/*
 * Copyright (c) 2017 Intel Corporation
 * Additional Copyright (c) 2018 Espressif Systems (Shanghai) PTE LTD
 *
 * SPDX-License-Identifier: Apache-2.0
 */

#include <stdio.h>
#include <string.h>


// herramientas para logs y flash de la tarjeta de desarrollo//
#include "esp_log.h"
#include "nvs_flash.h"
// APIs para el estándar Bluetooth mesh//
#include "esp_ble_mesh_defs.h"
#include "esp_ble_mesh_common_api.h"
#include "esp_ble_mesh_networking_api.h"
#include "esp_ble_mesh_provisioning_api.h"
#include "esp_ble_mesh_config_model_api.h"
#include "esp_ble_mesh_sensor_model_api.h"
#include "esp_bt.h"

#include "ble_mesh_example_init.h"
#include "ble_mesh_example_nvs.h"

#include "board.h"
// herramientas para incluir rtos y lectura de ADC//
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"
#include "driver/adc.h"
#include "esp_adc_cal.h"
#define TAG "Sensor model "
uint8_t voltage;
static esp_adc_cal_characteristics_t adc1_chars;
uint8_t voltage1;
static struct example_info_store {
    uint16_t net_idx;   /* NetKey Index */
    uint16_t app_idx;   /* AppKey Index */
} __attribute__((packed)) store = {
    .net_idx = ESP_BLE_MESH_KEY_UNUSED,
    .app_idx = ESP_BLE_MESH_KEY_UNUSED,
};

static nvs_handle_t NVS_HANDLE;
static const char * NVS_KEY = "sensor_server";
int8_t veal()
{
    //pin 36
    adc1_config_width(ADC_WIDTH_BIT_12);
    adc1_config_channel_atten(ADC1_CHANNEL_0, ADC_ATTEN_DB_11);
    int8_t vals = adc1_get_raw(ADC1_CHANNEL_0);
```

```
          voltage= (vals*(3.3))/250;
         //printf(" valorleido _%i\n", voltage);
         return voltage;
      }
      int8_t val1()
      {
         //pin 34
                esp_adc_cal_characterize(ADC_UNIT_1,     ADC_ATTEN_DB_11,
ADC_WIDTH_BIT_DEFAULT, 0, &adc1_chars);
         ESP_ERROR_CHECK(adc1_config_width(ADC_WIDTH_BIT_DEFAULT));
                ESP_ERROR_CHECK(adc1_config_channel_atten(ADC1_CHANNEL_6,
ADC_ATTEN_DB_11));
                                                                    int8_t
vals1=esp_adc_cal_raw_to_voltage(adc1_get_raw(ADC1_CHANNEL_6),
&adc1_chars);
         voltage1=(vals1*(3.3))/250;
         return voltage1;
      }
      //esp_bt_dev_set_device_name("test");

      #define CID_ESP 0x02E5
      /* Sensor Property ID */
      #define SENSOR_PROPERTY_ID_0 0x0059 //present input voltage pin 34
      //0x0099
      // promedio voltaje
      #define SENSOR_PROPERTY_ID_1 0x005D //present output voltage pin 36
      // Present output voltage 0x005D
      //0x00B4
      /*
       *                                  https://github.com/espressif/esp-
idf/blob/7bd93e3362e212b1e27fbd19a7a45cbe8e0b46ff/components/bt/esp_ble_me
sh/mesh_models/common/include/device_property.h
       */

      #define                                    SENSOR_POSITIVE_TOLERANCE
ESP_BLE_MESH_SENSOR_UNSPECIFIED_POS_TOLERANCE
      #define                                    SENSOR_NEGATIVE_TOLERANCE
ESP_BLE_MESH_SENSOR_UNSPECIFIED_NEG_TOLERANCE
      #define SENSOR_SAMPLE_FUNCTION ESP_BLE_MESH_SAMPLE_FUNC_UNSPECIFIED
      #define                                       SENSOR_MEASURE_PERIOD
ESP_BLE_MESH_SENSOR_NOT_APPL_MEASURE_PERIOD
      #define                                     SENSOR_UPDATE_INTERVAL
ESP_BLE_MESH_SENSOR_NOT_APPL_UPDATE_INTERVAL

      static uint8_t dev_uuid[ESP_BLE_MESH_OCTET16_LEN] = {0x32, 0x10};

      static esp_ble_mesh_cfg_srv_t config_server = {
         .relay = ESP_BLE_MESH_RELAY_ENABLED,
         .beacon = ESP_BLE_MESH_BEACON_ENABLED,
      #if defined(CONFIG_BLE_MESH_FRIEND)
        .friend_state = ESP_BLE_MESH_FRIEND_ENABLED,
      #else
         .friend_state = ESP_BLE_MESH_FRIEND_NOT_SUPPORTED,
      #endif
```

```
    #if defined(CONFIG_BLE_MESH_GATT_PROXY_SERVER)
        .gatt_proxy = ESP_BLE_MESH_GATT_PROXY_ENABLED,
    #else
        .gatt_proxy = ESP_BLE_MESH_GATT_PROXY_NOT_SUPPORTED,
    #endif
        .default_ttl = 7,
        /* 3 transmissions with 20ms interval */
        .net_transmit = ESP_BLE_MESH_TRANSMIT(2, 20),
        .relay_retransmit = ESP_BLE_MESH_TRANSMIT(2, 20),
    };

    NET_BUF_SIMPLE_DEFINE_STATIC(sensor_data_0, 1);
    NET_BUF_SIMPLE_DEFINE_STATIC(sensor_data_1, 1);
    //NET_BUF_SIMPLE_D

    static esp_ble_mesh_sensor_state_t sensor_states[2] = {
        /* Mesh Model Spec:
         * Multiple instances of the Sensor states may be present within
the same model,
         * provided that each instance has a unique value of the Sensor
Property ID to
         * allow the instances to be differentiated. Such sensors are
known as multisensors.
         * In this example, two instances of the Sensor states within the
same model are
         * provided.
         */
        [0] = {
            /* Mesh Model Spec:
             * Sensor Property ID is a 2-octet value referencing a device
property
             * that describes the meaning and format of data reported by
a sensor.
             * 0x0000 is prohibited.
             */
            .sensor_property_id = SENSOR_PROPERTY_ID_0,
            /* Mesh Model Spec:
            * Sensor Descriptor state represents the attributes describing
the sensor
             * data. This state does not change throughout the lifetime
of an element.
             */
            .descriptor.positive_tolerance = SENSOR_POSITIVE_TOLERANCE,
            .descriptor.negative_tolerance = SENSOR_NEGATIVE_TOLERANCE,
            .descriptor.sampling_function = SENSOR_SAMPLE_FUNCTION,
            .descriptor.measure_period = SENSOR_MEASURE_PERIOD,
            .descriptor.update_interval = SENSOR_UPDATE_INTERVAL,
            .sensor_data.format = ESP_BLE_MESH_SENSOR_DATA_FORMAT_A,
            .sensor_data.length = 1, /* 0 represents the length is 1 */
            .sensor_data.raw_value = &sensor_data_0,

        },
        [1] = {
            .sensor_property_id = SENSOR_PROPERTY_ID_1,
```

```
            .descriptor.positive_tolerance = SENSOR_POSITIVE_TOLERANCE,
            .descriptor.negative_tolerance = SENSOR_NEGATIVE_TOLERANCE,
            .descriptor.sampling_function = SENSOR_SAMPLE_FUNCTION,
            .descriptor.measure_period = SENSOR_MEASURE_PERIOD,
            .descriptor.update_interval = SENSOR_UPDATE_INTERVAL,
            .sensor_data.format = ESP_BLE_MESH_SENSOR_DATA_FORMAT_A,
            .sensor_data.length = 1, /* 0 represents the length is 1 */
            .sensor_data.raw_value = &sensor_data_1,
        },
    };

    /* 20 octets is large enough to hold two Sensor Descriptor state
values. */
    ESP_BLE_MESH_MODEL_PUB_DEFINE(sensor_pub, 20, ROLE_NODE);
    static esp_ble_mesh_sensor_srv_t sensor_server = {
        .rsp_ctrl.get_auto_rsp = ESP_BLE_MESH_SERVER_RSP_BY_APP,
        .rsp_ctrl.set_auto_rsp = ESP_BLE_MESH_SERVER_RSP_BY_APP,
        .state_count = ARRAY_SIZE(sensor_states),
        .states = sensor_states,
    };

    ESP_BLE_MESH_MODEL_PUB_DEFINE(sensor_setup_pub, 20, ROLE_NODE);
    static esp_ble_mesh_sensor_setup_srv_t sensor_setup_server = {
        .rsp_ctrl.get_auto_rsp = ESP_BLE_MESH_SERVER_RSP_BY_APP,
        .rsp_ctrl.set_auto_rsp = ESP_BLE_MESH_SERVER_RSP_BY_APP,
        .state_count = ARRAY_SIZE(sensor_states),
        .states = sensor_states,
    };

    static esp_ble_mesh_model_t root_models[] = {
        ESP_BLE_MESH_MODEL_CFG_SRV(&config_server),
        ESP_BLE_MESH_MODEL_SENSOR_SRV(&sensor_pub, &sensor_server),
                    ESP_BLE_MESH_MODEL_SENSOR_SETUP_SRV(&sensor_setup_pub,
&sensor_setup_server),
    };

    static esp_ble_mesh_elem_t elements[] = {
        ESP_BLE_MESH_ELEMENT(0, root_models, ESP_BLE_MESH_MODEL_NONE),
    };

    static esp_ble_mesh_comp_t composition = {
        .cid = CID_ESP,
        .elements = elements,
        .element_count = ARRAY_SIZE(elements),
    };

    static esp_ble_mesh_prov_t provision = {
        .uuid = dev_uuid,
    };
    static void mesh_example_info_store(void)
    {
        ble_mesh_nvs_store(NVS_HANDLE, NVS_KEY, &store, sizeof(store));
    }
```

```
      static void mesh_example_info_restore(void)
      {
          esp_err_t err = ESP_OK;
          bool exist = false;

             err   =   ble_mesh_nvs_restore(NVS_HANDLE,   NVS_KEY,   &store,
sizeof(store), &exist);
          if (err != ESP_OK) {
              return;
          }

          if (exist) {
              ESP_LOGI(TAG, "Restore, net_idx 0x%04x, app_idx 0x%04x",
                  store.net_idx, store.app_idx);

          }
      }
      static  void  prov_complete(uint16_t  net_idx,  uint16_t  addr,  uint8_t
flags, uint32_t iv_index)
      {
          ESP_LOGI(TAG, "net_idx 0x%03x, addr 0x%04x", net_idx, addr);
          ESP_LOGI(TAG, "flags 0x%02x, iv_index 0x%08x", flags, iv_index);
          board_led_operation(LED_G, LED_OFF);

           /*  Initialize  the  indoor  and  outdoor  temperatures  for  each
sensor.  */
          net_buf_simple_add_u8(&sensor_data_0,val());
          net_buf_simple_add_u8(&sensor_data_1,val1());
          store.net_idx = net_idx;
           ESP_LOGI(TAG, "Store, net_idx 0x%04x",
                  net_idx );
      }

      static                                                          void
example_ble_mesh_provisioning_cb(esp_ble_mesh_prov_cb_event_t event,
                                                esp_ble_mesh_prov_cb_param_t
*param)
      {
          switch (event)
          {
          case ESP_BLE_MESH_PROV_REGISTER_COMP_EVT:
               ESP_LOGI(TAG, "ESP_BLE_MESH_PROV_REGISTER_COMP_EVT, err_code
%d", param->prov_register_comp.err_code);
              mesh_example_info_restore();
              break;
          case ESP_BLE_MESH_NODE_PROV_ENABLE_COMP_EVT:
                    ESP_LOGI(TAG,   "ESP_BLE_MESH_NODE_PROV_ENABLE_COMP_EVT,
err_code %d", param->node_prov_enable_comp.err_code);
              break;
          case ESP_BLE_MESH_NODE_PROV_LINK_OPEN_EVT:
               ESP_LOGI(TAG, "ESP_BLE_MESH_NODE_PROV_LINK_OPEN_EVT, bearer
%s",
                                        param->node_prov_link_open.bearer   ==
ESP_BLE_MESH_PROV_ADV ? "PB-ADV" : "PB-GATT");
```

```
            break;
        case ESP_BLE_MESH_NODE_PROV_LINK_CLOSE_EVT:
            ESP_LOGI(TAG, "ESP_BLE_MESH_NODE_PROV_LINK_CLOSE_EVT, bearer %s",
                     param->node_prov_link_close.bearer == ESP_BLE_MESH_PROV_ADV ? "PB-ADV" : "PB-GATT");
            break;
        case ESP_BLE_MESH_NODE_PROV_COMPLETE_EVT:
            ESP_LOGI(TAG, "ESP_BLE_MESH_NODE_PROV_COMPLETE_EVT");
            prov_complete(param->node_prov_complete.net_idx, param->node_prov_complete.addr,
                          param->node_prov_complete.flags, param->node_prov_complete.iv_index);
            break;
        case ESP_BLE_MESH_NODE_PROV_RESET_EVT:
            ESP_LOGI(TAG, "ESP_BLE_MESH_NODE_PROV_RESET_EVT");
            break;
        case ESP_BLE_MESH_NODE_SET_UNPROV_DEV_NAME_COMP_EVT:
            ESP_LOGI(TAG, "ESP_BLE_MESH_NODE_SET_UNPROV_DEV_NAME_COMP_EVT, err_code %d", param->node_set_unprov_dev_name_comp.err_code);
            break;
        default:
            break;
        }
    }

    static void example_ble_mesh_config_server_cb(esp_ble_mesh_cfg_server_cb_event_t event,
                                                  esp_ble_mesh_cfg_server_cb_param_t *param)
    {
        if (event == ESP_BLE_MESH_CFG_SERVER_STATE_CHANGE_EVT)
        {
            switch (param->ctx.recv_op)
            {
            case ESP_BLE_MESH_MODEL_OP_APP_KEY_ADD:
                ESP_LOGI(TAG, "ESP_BLE_MESH_MODEL_OP_APP_KEY_ADD");
                ESP_LOGI(TAG, "net_idx 0x%04x, app_idx 0x%04x",
                         param->value.state_change.appkey_add.net_idx,
                         param->value.state_change.appkey_add.app_idx);
                ESP_LOG_BUFFER_HEX("AppKey", param->value.state_change.appkey_add.app_key, 16);
                break;
            case ESP_BLE_MESH_MODEL_OP_MODEL_APP_BIND:
                ESP_LOGI(TAG, "ESP_BLE_MESH_MODEL_OP_MODEL_APP_BIND");
                ESP_LOGI(TAG, "elem_addr 0x%04x, app_idx 0x%04x, cid 0x%04x, mod_id 0x%04x",
                         param->value.state_change.mod_app_bind.element_addr,
                         param->value.state_change.mod_app_bind.app_idx,
                         param->value.state_change.mod_app_bind.company_id,
                         param->value.state_change.mod_app_bind.model_id);
```

```
                                                    if     (param-
>value.state_change.mod_app_bind.model_id                                   ==
ESP_BLE_MESH_MODEL_ID_SENSOR_SRV)
                                {
                                                        store.app_idx   =   param-
>value.state_change.mod_app_bind.app_idx;
                                    mesh_example_info_store(); /* Store proper
mesh example info */
                                }
                    break;
                case ESP_BLE_MESH_MODEL_OP_MODEL_SUB_ADD:
                    ESP_LOGI(TAG, "ESP_BLE_MESH_MODEL_OP_MODEL_SUB_ADD");
                      ESP_LOGI(TAG, "elem_addr 0x%04x, sub_addr 0x%04x, cid
0x%04x, mod_id 0x%04x",
                                                                        param-
>value.state_change.mod_sub_add.element_addr,
                               param->value.state_change.mod_sub_add.sub_addr,
                              param->value.state_change.mod_sub_add.company_id,
                               param->value.state_change.mod_sub_add.model_id);
                    break;
                default:
                    break;
                }
            }
        }

        struct example_sensor_descriptor
        {
            uint16_t sensor_prop_id;
            uint32_t pos_tolerance : 12,
                neg_tolerance : 12,
                sample_func : 8;
            uint8_t measure_period;
            uint8_t update_interval;
        } __attribute__((packed));

        static                                                          void
example_ble_mesh_send_sensor_descriptor_status(esp_ble_mesh_sensor_server_
cb_param_t *param)
        {
            struct example_sensor_descriptor descriptor = {0};
            uint8_t *status = NULL;
            uint16_t length = 0;
            esp_err_t err;
            int i;

                    status    =    calloc(1,    ARRAY_SIZE(sensor_states)    *
ESP_BLE_MESH_SENSOR_DESCRIPTOR_LEN);
            if (!status)
            {
                ESP_LOGE(TAG, "No memory for sensor descriptor status!");
                return;
            }
```

```
        if (param->value.get.sensor_descriptor.op_en == false)
        {
            /* Mesh Model Spec:
             * Upon receiving a Sensor Descriptor Get message with the
Property ID field
             * omitted, the Sensor Server shall respond with a Sensor
Descriptor Status
             * message containing the Sensor Descriptor states for all
sensors within the
             * Sensor Server.
             */
            for (i = 0; i < ARRAY_SIZE(sensor_states); i++)
            {
                                                descriptor.sensor_prop_id     =
sensor_states[i].sensor_property_id;
                                                 descriptor.pos_tolerance     =
sensor_states[i].descriptor.positive_tolerance;
                                                 descriptor.neg_tolerance     =
sensor_states[i].descriptor.negative_tolerance;
                                                   descriptor.sample_func     =
sensor_states[i].descriptor.sampling_function;
                                                descriptor.measure_period     =
sensor_states[i].descriptor.measure_period;
                                               descriptor.update_interval     =
sensor_states[i].descriptor.update_interval;
                                  memcpy(status   +   length,   &descriptor,
ESP_BLE_MESH_SENSOR_DESCRIPTOR_LEN);
                length += ESP_BLE_MESH_SENSOR_DESCRIPTOR_LEN;
            }
            goto send;
        }

        for (i = 0; i < ARRAY_SIZE(sensor_states); i++)
        {
                 if  (param->value.get.sensor_descriptor.property_id  ==
sensor_states[i].sensor_property_id)
            {
                                                descriptor.sensor_prop_id     =
sensor_states[i].sensor_property_id;
                                                 descriptor.pos_tolerance     =
sensor_states[i].descriptor.positive_tolerance;
                                                 descriptor.neg_tolerance     =
sensor_states[i].descriptor.negative_tolerance;
                                                   descriptor.sample_func     =
sensor_states[i].descriptor.sampling_function;
                                                descriptor.measure_period     =
sensor_states[i].descriptor.measure_period;
                                               descriptor.update_interval     =
sensor_states[i].descriptor.update_interval;
                                                memcpy(status,     &descriptor,
ESP_BLE_MESH_SENSOR_DESCRIPTOR_LEN);
                length = ESP_BLE_MESH_SENSOR_DESCRIPTOR_LEN;
                goto send;
            }
```

```
            }

            /* Mesh Model Spec:
             * When a Sensor Descriptor Get message that identifies a sensor descriptor
             * property that does not exist on the element, the Descriptor field shall
             * contain the requested Property ID value and the other fields of the Sensor
             * Descriptor state shall be omitted.
             */
             memcpy(status, &param->value.get.sensor_descriptor.property_id, ESP_BLE_MESH_SENSOR_PROPERTY_ID_LEN);
            length = ESP_BLE_MESH_SENSOR_PROPERTY_ID_LEN;

        send:
            ESP_LOG_BUFFER_HEX("Sensor Descriptor", status, length);

             err = esp_ble_mesh_server_model_send_msg(param->model, &param->ctx,
                                                      ESP_BLE_MESH_MODEL_OP_SENSOR_DESCRIPTOR_STATUS, length, status);
            if (err != ESP_OK)
            {
                ESP_LOGE(TAG, "Failed to send Sensor Descriptor Status");
            }
            free(status);
        }

        static void example_ble_mesh_send_sensor_cadence_status(esp_ble_mesh_sensor_server_cb_param_t *param)
        {
            esp_err_t err;

            /* Sensor Cadence state is not supported currently. */
             err = esp_ble_mesh_server_model_send_msg(param->model, &param->ctx,
                                                      ESP_BLE_MESH_MODEL_OP_SENSOR_CADENCE_STATUS,
                                                      ESP_BLE_MESH_SENSOR_PROPERTY_ID_LEN,
                                                           (uint8_t *)&param->value.get.sensor_cadence.property_id);
            if (err != ESP_OK)
            {
                ESP_LOGE(TAG, "Failed to send Sensor Cadence Status");
            }
        }

        static void example_ble_mesh_send_sensor_settings_status(esp_ble_mesh_sensor_server_cb_param_t *param)
        {
```

```
            esp_err_t err;

            /* Sensor Setting state is not supported currently. */
             err  =  esp_ble_mesh_server_model_send_msg(param->model,  &param-
>ctx,
                                                       ESP_BLE_MESH_MODEL_OP_S
ENSOR_SETTINGS_STATUS,
                                                       ESP_BLE_MESH_SENSOR_PRO
PERTY_ID_LEN,
                                                            (uint8_t  *)&param-
>value.get.sensor_settings.property_id);
            if (err != ESP_OK)
            {
                ESP_LOGE(TAG, "Failed to send Sensor Settings Status");
            }
        }

        struct example_sensor_setting
        {
            uint16_t sensor_prop_id;
            uint16_t sensor_setting_prop_id;
        } __attribute__((packed));

        static                                                          void
example_ble_mesh_send_sensor_setting_status(esp_ble_mesh_sensor_server_cb_
param_t *param)
        {
            struct example_sensor_setting setting = {0};
            esp_err_t err;

            /* Mesh Model Spec:
             * If the message is sent as a response to the Sensor Setting Get
message or
             * a Sensor Setting Set message with an unknown Sensor Property
ID field or
             * an unknown Sensor Setting Property ID field, the Sensor Setting
Access
             * field and the Sensor Setting Raw field shall be omitted.
             */

                               setting.sensor_prop_id          =          param-
>value.get.sensor_setting.property_id;
                          setting.sensor_setting_prop_id        =        param-
>value.get.sensor_setting.setting_property_id;

             err  =  esp_ble_mesh_server_model_send_msg(param->model,  &param-
>ctx,
                                                       ESP_BLE_MESH_MODEL_OP_S
ENSOR_SETTING_STATUS,
                                                      sizeof(setting), (uint8_t
*)&setting);
            if (err != ESP_OK)
            {
                ESP_LOGE(TAG, "Failed to send Sensor Setting Status");
```

```
        }
    }

    static uint16_t example_ble_mesh_get_sensor_data(esp_ble_mesh_sensor_state_t *state, uint8_t *data)
    {
        uint8_t mpid_len = 0, data_len = 0;
        uint32_t mpid = 0;

        if (state == NULL || data == NULL)
        {
            ESP_LOGE(TAG, "%s, Invalid parameter", __func__);
            return 0;
        }

        if (state->sensor_data.length == ESP_BLE_MESH_SENSOR_DATA_ZERO_LEN)
        {
            /* For zero-length sensor data, the length is 0x7F, and the format is Format B. */
            mpid = ESP_BLE_MESH_SENSOR_DATA_FORMAT_B_MPID(state->sensor_data.length, state->sensor_property_id);
            mpid_len = ESP_BLE_MESH_SENSOR_DATA_FORMAT_B_MPID_LEN;
            data_len = 0;
        }
        else
        {
            if (state->sensor_data.format == ESP_BLE_MESH_SENSOR_DATA_FORMAT_A)
            {
                mpid = ESP_BLE_MESH_SENSOR_DATA_FORMAT_A_MPID(state->sensor_data.length, state->sensor_property_id);
                mpid_len = ESP_BLE_MESH_SENSOR_DATA_FORMAT_A_MPID_LEN;
            }
            else
            {
                mpid = ESP_BLE_MESH_SENSOR_DATA_FORMAT_B_MPID(state->sensor_data.length, state->sensor_property_id);
                mpid_len = ESP_BLE_MESH_SENSOR_DATA_FORMAT_B_MPID_LEN;
            }
            /* Use "state->sensor_data.length + 1" because the length of sensor data is zero-based. */
            data_len = state->sensor_data.length + 1;
        }

        memcpy(data, &mpid, mpid_len);
        memcpy(data + mpid_len, state->sensor_data.raw_value->data, data_len);

        return (mpid_len + data_len);
    }
```

```
    static                                                          void
example_ble_mesh_send_sensor_status(esp_ble_mesh_sensor_server_cb_param_t
*param)
    {

        uint8_t *status = NULL;
        uint16_t buf_size = 0;
        uint16_t length = 0;
        uint32_t mpid = 0;
        esp_err_t err;
        int i;

        /**
         * Sensor Data state from Mesh Model Spec
         * |--------Field--------|-Size (octets)-|----------------------
-Notes-------------------------|
         * |----Property ID 1----|-------2-------|--ID of the 1st device
property of the sensor---------|
         *  |-----Raw  Value  1-----|----variable---|--Raw  Value  field
defined by the 1st device property--|
         * |----Property ID 2----|-------2-------|--ID of the 2nd device
property of the sensor---------|
         *  |-----Raw  Value  2-----|----variable---|--Raw  Value  field
defined by the 2nd device property--|
         * | ...... |
         * |----Property ID n----|-------2-------|--ID of the nth device
property of the sensor---------|
         *  |-----Raw  Value  n-----|----variable---|--Raw  Value  field
defined by the nth device property--|
         */
        for (i = 0; i < ARRAY_SIZE(sensor_states); i++)
        {
            esp_ble_mesh_sensor_state_t *state = &sensor_states[i];
                            if      (state->sensor_data.length      ==
ESP_BLE_MESH_SENSOR_DATA_ZERO_LEN)
            {
                buf_size += ESP_BLE_MESH_SENSOR_DATA_FORMAT_B_MPID_LEN;
            }
            else
            {
                /* Use "state->sensor_data.length + 1" because the length
of sensor data is zero-based. */
                                    if    (state->sensor_data.format    ==
ESP_BLE_MESH_SENSOR_DATA_FORMAT_A)
                {
                    buf_size += ESP_BLE_MESH_SENSOR_DATA_FORMAT_A_MPID_LEN
+ state->sensor_data.length + 1;
                }
                else
                {
                    buf_size += ESP_BLE_MESH_SENSOR_DATA_FORMAT_B_MPID_LEN
+ state->sensor_data.length + 1;
                }
            }
```

```
            }

            status = calloc(1, buf_size);
            if (!status)
            {
                ESP_LOGE(TAG, "No memory for sensor status!");
                return;
            }

            if (param->value.get.sensor_data.op_en == false)
            {
                /* Mesh Model Spec:
                 * If the message is sent as a response to the Sensor Get
message, and if the
                 * Property ID field of the incoming message is omitted, the
Marshalled Sensor
                 * Data field shall contain data for all device properties
within a sensor.
                 */
                for (i = 0; i < ARRAY_SIZE(sensor_states); i++)
                {
                                                                length        +=
example_ble_mesh_get_sensor_data(&sensor_states[i], status + length);
                }
                goto send;
            }

            /* Mesh Model Spec:
             * Otherwise, the Marshalled Sensor Data field shall contain data
for the requested
             * device property only.
             */
            for (i = 0; i < ARRAY_SIZE(sensor_states); i++)
            {
                        if    (param->value.get.sensor_data.property_id    ==
sensor_states[i].sensor_property_id)
                {
                                                                length        =
example_ble_mesh_get_sensor_data(&sensor_states[i], status);
                    goto send;
                }
            }

            /* Mesh Model Spec:
             * Or the Length shall represent the value of zero and the Raw
Value field shall
             * contain only the Property ID if the requested device property
is not recognized
             * by the Sensor Server.
             */
                                                    mpid                    =
ESP_BLE_MESH_SENSOR_DATA_FORMAT_B_MPID(ESP_BLE_MESH_SENSOR_DATA_ZERO_LEN,
                                                                        param-
>value.get.sensor_data.property_id);
```

```
                                        memcpy(status,                &mpid,
ESP_BLE_MESH_SENSOR_DATA_FORMAT_B_MPID_LEN);
            length = ESP_BLE_MESH_SENSOR_DATA_FORMAT_B_MPID_LEN;

        send:
            ESP_LOG_BUFFER_HEX("Sensor Data", status, length);

             err  =  esp_ble_mesh_server_model_send_msg(param->model,  &param-
>ctx,
                                                       ESP_BLE_MESH_MODEL_OP_S
ENSOR_STATUS, length, status);
            if (err != ESP_OK)
            {
                ESP_LOGE(TAG, "Failed to send Sensor Status");
            }
            free(status);
        }

        static                                                          void
example_ble_mesh_send_sensor_column_status(esp_ble_mesh_sensor_server_cb_p
aram_t *param)
        {
            uint8_t *status = NULL;
            uint16_t length = 0;
            esp_err_t err;

                length   =   ESP_BLE_MESH_SENSOR_PROPERTY_ID_LEN   +   param-
>value.get.sensor_column.raw_value_x->len;

            status = calloc(1, length);
            if (!status)
            {
                ESP_LOGE(TAG, "No memory for sensor column status!");
                return;
            }

                memcpy(status,   &param->value.get.sensor_column.property_id,
ESP_BLE_MESH_SENSOR_PROPERTY_ID_LEN);
              memcpy(status  +  ESP_BLE_MESH_SENSOR_PROPERTY_ID_LEN,  param-
>value.get.sensor_column.raw_value_x->data,
                    param->value.get.sensor_column.raw_value_x->len);

             err  =  esp_ble_mesh_server_model_send_msg(param->model,  &param-
>ctx,
                                                       ESP_BLE_MESH_MODEL_OP_S
ENSOR_COLUMN_STATUS, length, status);
            if (err != ESP_OK)
            {
                ESP_LOGE(TAG, "Failed to send Sensor Column Status");
            }
            free(status);
        }
```

```
      static                                                                    void
example_ble_mesh_send_sensor_series_status(esp_ble_mesh_sensor_server_cb_p
aram_t *param)
      {
          esp_err_t err;

           err  =  esp_ble_mesh_server_model_send_msg(param->model,  &param-
>ctx,
                                                      ESP_BLE_MESH_MODEL_OP_S
ENSOR_SERIES_STATUS,
                                                      ESP_BLE_MESH_SENSOR_PRO
PERTY_ID_LEN,
                                                           (uint8_t *)&param-
>value.get.sensor_series.property_id);
          if (err != ESP_OK)
          {
              ESP_LOGE(TAG, "Failed to send Sensor Column Status");
          }
      }

      static                                                                    void
example_ble_mesh_sensor_server_cb(esp_ble_mesh_sensor_server_cb_event_t
event,
                                                       esp_ble_mesh_sensor_se
rver_cb_param_t *param)
      {
           ESP_LOGI(TAG, "Sensor server, event %d, src 0x%04x, dst 0x%04x,
model_id 0x%04x",
                    event, param->ctx.addr, param->ctx.recv_dst, param->model-
>model_id);

          switch (event)
          {
          case ESP_BLE_MESH_SENSOR_SERVER_RECV_GET_MSG_EVT:
              switch (param->ctx.recv_op)
              {
              case ESP_BLE_MESH_MODEL_OP_SENSOR_DESCRIPTOR_GET:
                                                                    ESP_LOGI(TAG,
"ESP_BLE_MESH_MODEL_OP_SENSOR_DESCRIPTOR_GET");
                  example_ble_mesh_send_sensor_descriptor_status(param);
                  break;
              case ESP_BLE_MESH_MODEL_OP_SENSOR_CADENCE_GET:
                 ESP_LOGI(TAG, "ESP_BLE_MESH_MODEL_OP_SENSOR_CADENCE_GET");
                  example_ble_mesh_send_sensor_cadence_status(param);
                  break;
              case ESP_BLE_MESH_MODEL_OP_SENSOR_SETTINGS_GET:
                                                                    ESP_LOGI(TAG,
"ESP_BLE_MESH_MODEL_OP_SENSOR_SETTINGS_GET");
                  example_ble_mesh_send_sensor_settings_status(param);
                  break;
              case ESP_BLE_MESH_MODEL_OP_SENSOR_SETTING_GET:
                                                                    ESP_LOGI(TAG,
"ESP_BLE_MESH_MODEL_OP_SENSOR_SETTINGS_GET");
                  example_ble_mesh_send_sensor_setting_status(param);
```

```
                break;
            case ESP_BLE_MESH_MODEL_OP_SENSOR_GET:

                ESP_LOGI(TAG, "ESP_BLE_MESH_MODEL_OP_SENSOR_GET");
                net_buf_simple_reset(&sensor_data_0);
                vTaskDelay(pdMS_TO_TICKS(10));
                net_buf_simple_reset(&sensor_data_1);
                vTaskDelay(pdMS_TO_TICKS(100));
                net_buf_simple_add_u8(&sensor_data_0, val());
                vTaskDelay(pdMS_TO_TICKS(10));
                net_buf_simple_add_u8(&sensor_data_1, val1());
                example_ble_mesh_send_sensor_status(param);
                break;
            case ESP_BLE_MESH_MODEL_OP_SENSOR_COLUMN_GET:
                ESP_LOGI(TAG, "ESP_BLE_MESH_MODEL_OP_SENSOR_COLUMN_GET");
                example_ble_mesh_send_sensor_column_status(param);
                break;
            case ESP_BLE_MESH_MODEL_OP_SENSOR_SERIES_GET:
                ESP_LOGI(TAG, "ESP_BLE_MESH_MODEL_OP_SENSOR_SERIES_GET");
                example_ble_mesh_send_sensor_series_status(param);
                break;
            default:
                ESP_LOGE(TAG, "Unknown Sensor Get opcode 0x%04x", param-
>ctx.recv_op);
                return;
            }
            break;
        case ESP_BLE_MESH_SENSOR_SERVER_RECV_SET_MSG_EVT:
            switch (param->ctx.recv_op)
            {
            case ESP_BLE_MESH_MODEL_OP_SENSOR_CADENCE_SET:
               ESP_LOGI(TAG, "ESP_BLE_MESH_MODEL_OP_SENSOR_CADENCE_SET");
                example_ble_mesh_send_sensor_cadence_status(param);
                break;
            case ESP_BLE_MESH_MODEL_OP_SENSOR_CADENCE_SET_UNACK:
                                                        ESP_LOGI(TAG,
"ESP_BLE_MESH_MODEL_OP_SENSOR_CADENCE_SET_UNACK");
                break;
            case ESP_BLE_MESH_MODEL_OP_SENSOR_SETTING_SET:
               ESP_LOGI(TAG, "ESP_BLE_MESH_MODEL_OP_SENSOR_SETTING_SET");
                example_ble_mesh_send_sensor_setting_status(param);
                break;
            case ESP_BLE_MESH_MODEL_OP_SENSOR_SETTING_SET_UNACK:
                                                        ESP_LOGI(TAG,
"ESP_BLE_MESH_MODEL_OP_SENSOR_SETTING_SET_UNACK");
                break;
            default:
                ESP_LOGE(TAG, "Unknown Sensor Set opcode 0x%04x", param-
>ctx.recv_op);
                break;
            }
            break;
        default:
            ESP_LOGE(TAG, "Unknown Sensor Server event %d", event);
```

```
            break;
        }
    }

    static esp_err_t ble_mesh_init(void)
    {
        esp_err_t err;

         esp_ble_mesh_register_prov_callback(example_ble_mesh_provisionin
g_cb);
         esp_ble_mesh_register_config_server_callback(example_ble_mesh_co
nfig_server_cb);
         esp_ble_mesh_register_sensor_server_callback(example_ble_mesh_se
nsor_server_cb);

        err = esp_ble_mesh_init(&provision, &composition);
        if (err != ESP_OK)
        {
            ESP_LOGE(TAG, "Failed to initialize mesh stack");
            return err;
        }
        esp_ble_mesh_set_unprovisioned_device_name("N5BLEMesh");
          err  =  esp_ble_mesh_node_prov_enable(ESP_BLE_MESH_PROV_ADV  |
ESP_BLE_MESH_PROV_GATT);
        esp_ble_tx_power_set(ESP_BLE_PWR_TYPE_DEFAULT, ESP_PWR_LVL_P9);
        if (err != ESP_OK)
        {
            ESP_LOGE(TAG, "Failed to enable mesh node");
            return err;
        }

        board_led_operation(LED_G, LED_ON);

        ESP_LOGI(TAG, "BLE Mesh sensor server initialized");

        return ESP_OK;
    }
    static void parse_buf(struct net_buf_simple *buf)
    {
        ESP_LOG_BUFFER_HEX("sensor_data", buf->data, buf->len);
    }

    void app_main(void)
    {
        esp_err_t err;

        ESP_LOGI(TAG, "Initializing...");
        err = nvs_flash_init();
        if (err == ESP_ERR_NVS_NO_FREE_PAGES)
        {
            ESP_ERROR_CHECK(nvs_flash_erase());
            err = nvs_flash_init();
        }
        ESP_ERROR_CHECK(err);
```

```
        board_init();

        err = bluetooth_init();
        if (err)
        {
            ESP_LOGE(TAG, "esp32_bluetooth_init failed (err %d)", err);
            return;
        }

        err = ble_mesh_nvs_open(&NVS_HANDLE);
        if (err) {
            return;
        }
        //esp_err_t esp_bt_dev_set_device_name(const char *N2BLEMesh);
        ble_mesh_get_dev_uuid(dev_uuid);
                    //esp_ble_tx_power_set(ESP_BLE_MESH_PROV_ADV       |
ESP_BLE_MESH_PROV_GATT,ESP_PWR_LVL_P9);

        /* Initialize the Bluetooth Mesh Subsystem */
        err = ble_mesh_init();
        if (err)
        {
            ESP_LOGE(TAG, "Bluetooth mesh init failed (err %d)", err);
        }

    }
```

Anexo 12 Código utilizado en la placa de desarrollo Thingy 52

```
    /*
     * Copyright (c) 2020 Nordic Semiconductor ASA
     *
     * SPDX-License-Identifier: LicenseRef-Nordic-5-Clause
     */

    #include <bluetooth/bluetooth.h>
    #include <zephyr.h>
    #include <bluetooth/mesh/models.h>
    #include <dk_buttons_and_leds.h>
    #include <settings/settings.h>
    #include <drivers/gpio.h>
    #include <drivers/sensor.h>
    #include <stdio.h>
    #include <bluetooth/mesh.h>
    #include <random/rand32.h>
    #include <device.h>
    #include <sys/util.h>
    #include <drivers/sensor/ccs811.h>
    #include "model_handler.h"

    #if DT_NODE_EXISTS(DT_ALIAS(bme680))
```

```c
/** Thingy53 */
#define SENSOR_INST DT_PROP(DT_ALIAS(bme680), label)
#define SENSOR_DATA_TYPE SENSOR_CHAN_AMBIENT_TEMP
#elif DT_NODE_EXISTS(DT_NODELABEL(temp))
/** nRF52 DK */
#define SENSOR_INST DT_PROP(DT_NODELABEL(temp), label)
#define SENSOR_DATA_TYPE SENSOR_CHAN_DIE_TEMP
/** Thingy52 */
#else
#define SENSOR_INST NULL
#define SENSOR_DATA_TYPE NULL
#error "Unsupported board!"
#endif
struct k_delayed_work temperature_timer;
static const struct device *dev;
int val_co2=01;
struct sensor_value co2, tvoc, voltage, current;
/*
*
*Sensor de Co2
*/
static bool app_fw_2;

static const char *now_str(void)
{
    static char buf[16]; /* ...HH:MM:SS.MMM */
    uint32_t now = k_uptime_get_32();
    unsigned int ms = now % MSEC_PER_SEC;
    unsigned int s;
    unsigned int min;
    unsigned int h;

    now /= MSEC_PER_SEC;
    s = now % 60U;
    now /= 60U;
    min = now % 60U;
    now /= 60U;
    h = now;

    snprintf(buf, sizeof(buf), "%u:%02u:%02u.%03u",
         h, min, s, ms);
    return buf;
}

static int do_fetch(const struct device *dev)
{

    int rc = 0;
    int baseline = -1;

#ifdef CONFIG_APP_MONITOR_BASELINE
    rc = ccs811_baseline_fetch(dev);
    if (rc >= 0) {
        baseline = rc;
```

```
        rc = 0;
    }
#endif
    if (rc == 0) {
        rc = sensor_sample_fetch(dev);
    }
    if (rc == 0) {
        const struct ccs811_result_type *rp = ccs811_result(dev);

        sensor_channel_get(dev, SENSOR_CHAN_CO2, &co2);
        sensor_channel_get(dev, SENSOR_CHAN_VOC, &tvoc);
        sensor_channel_get(dev, SENSOR_CHAN_VOLTAGE, &voltage);
        sensor_channel_get(dev, SENSOR_CHAN_CURRENT, &current);
                val_co2=co2.val1;
        printk("\n[%s]: CCS811: %u ppm eCO2; %u ppb eTVOC\n",
               now_str(), co2.val1, tvoc.val1);
       printk("Voltage: %d.%06dV; Current: %d.%06dA\n", voltage.val1,
               voltage.val2, current.val1, current.val2);
#ifdef CONFIG_APP_MONITOR_BASELINE
        printk("BASELINE %04x\n", baseline);
#endif

    }
    return rc;
}



 static int co2_voc_get(struct bt_mesh_sensor *sensor,
                        struct bt_mesh_msg_ctx *ctx,
                        struct sensor_value *rsp)
{
int rc = do_fetch(dev);
return rc;
}
struct bt_mesh_sensor co2_voc_sensor = {
    .type = &bt_mesh_sensor_present_amb_co2_concentration,
    .get = co2_voc_get,
};
/*
* sensor de temperatura exterior HTS221
*/
static int temperatura_htss1_get(struct bt_mesh_sensor *sensor,
             struct bt_mesh_msg_ctx *ctx, struct sensor_value *rsp)
{
        int err;
        const struct device *temp = device_get_binding("HTS221");
        while (true) {

        //device_get_binding(HTS221);
    if (sensor_sample_fetch(temp)<0)
        {
        break;
        }
```

```
        err = sensor_channel_get(temp, SENSOR_CHAN_AMBIENT_TEMP,rsp);

        if (err<0) {
            printk("Error getting temperature sensor data (%d)\n", err);
                    break;
        }
            return err;

            k_sleep(K_MSEC(1500));
            }
            k_sleep(K_MSEC(100));

    }

    struct bt_mesh_sensor temperatura_htss1_sensor = {
        .type = &bt_mesh_sensor_present_amb_temp,
        .get = temperatura_htss1_get,
    };


    /* The columns (temperature ranges) for relative
     * runtime in a chip temperature
     */
    static const struct bt_mesh_sensor_column columns[] = {//valores
prestablecidos para temperatura interna
        { { 0 }, { 20 } },
        { { 20 }, { 25 } },
        { { 25 }, { 30 } },
        { { 30 }, { 100 } },
    };


    static uint32_t tot_temp_samps;//variable para guardar temperatura de
chip
    static uint32_t col_samps[ARRAY_SIZE(columns)];

    static int32_t prev_pres;//boton de presencia

    static int chip_temp_get(struct bt_mesh_sensor *sensor,
                 struct bt_mesh_msg_ctx *ctx, struct sensor_value *rsp)
    {
        int err;

        sensor_sample_fetch(dev);

        err = sensor_channel_get(dev, SENSOR_DATA_TYPE, rsp);

        if (err) {
            printk("Error getting temperature sensor data (%d)\n", err);
        }

        for (int i = 0; i < ARRAY_SIZE(columns); ++i) {
            if (bt_mesh_sensor_value_in_column(rsp, &columns[i])) {
```

```
                col_samps[i]++;
                break;
            }
        }

        tot_temp_samps++;

        return err;
    }

    static struct bt_mesh_sensor chip_temp = {
        .type = &bt_mesh_sensor_present_dev_op_temp,
        .get = chip_temp_get,
    };

    static int relative_runtime_in_chip_temp_get(
        struct bt_mesh_sensor *sensor, struct bt_mesh_msg_ctx *ctx,
         const struct bt_mesh_sensor_column *column, struct sensor_value
*value)
    {
        if (tot_temp_samps) {
            int32_t index = column - &columns[0];
            uint8_t percent_steps =
                (200 * col_samps[index]) / tot_temp_samps;

            value[0].val1 = percent_steps / 2;
            value[0].val2 = (percent_steps % 2) * 500000;
        } else {
            value[0].val1 = 0;
            value[0].val2 = 0;
        }

        value[1] = column->start;
        value[2] = column->end;

        return 0;
    }

    static struct bt_mesh_sensor rel_chip_temp_runtime = {
        .type = &bt_mesh_sensor_rel_runtime_in_a_dev_op_temp_range,
        .series = {
            columns,
            ARRAY_SIZE(columns),
            relative_runtime_in_chip_temp_get,
            },
    };

    static struct bt_mesh_sensor presence_sensor = {
        .type = &bt_mesh_sensor_presence_detected,
    };

    static  int  time_since_presence_detected_get(struct  bt_mesh_sensor
*sensor,
                              struct bt_mesh_msg_ctx *ctx,
```

```
                                struct sensor_value *rsp)
{
    if (prev_pres) {
        rsp->val1 = (k_uptime_get_32() - prev_pres) / MSEC_PER_SEC;
    } else {
        rsp->val1 = 0;
    }

    return 0;
}

static struct bt_mesh_sensor time_since_presence_detected = {
    .type = &bt_mesh_sensor_time_since_presence_detected,
    .get = time_since_presence_detected_get,
};

static struct bt_mesh_sensor *const sensors[] = {
    //&chip_temp,
    //&rel_chip_temp_runtime,
    //&presence_sensor,
        &co2_voc_sensor,
    &time_since_presence_detected,
        &temperatura_htss1_sensor,
};

static struct bt_mesh_sensor_srv sensor_srv =
    BT_MESH_SENSOR_SRV_INIT(sensors, ARRAY_SIZE(sensors));

static struct k_work_delayable end_of_presence_work;

static void end_of_presence(struct k_work *work)
{
    int err;

    /* This sensor value must be boolean -
     * .val1 can only be '0' or '1'
     */
    struct sensor_value val = {
        .val1 = 0,
    };

    err = bt_mesh_sensor_srv_pub(&sensor_srv, NULL, &presence_sensor,
&val);

    if (err) {
        printk("Error publishing end of presence (%d)\n", err);
    }
}

static void button_handler_cb(uint32_t pressed, uint32_t changed)
{
    if ((pressed & BIT(0))) {
        int err;
```

```
            /* This sensor value must be boolean -
             * .val1 can only be '0' or '1'
             */
            struct sensor_value val = {
                .val1 = 1,
            };

            err = bt_mesh_sensor_srv_pub(&sensor_srv, NULL,
                                         &presence_sensor, &val);

            if (err) {
                printk("Error publishing presence (%d)\n", err);
            }

            prev_pres = k_uptime_get_32();

            k_work_reschedule(&end_of_presence_work, K_MSEC(2000));
        }
    }

    static struct button_handler = {
        .cb = button_handler_cb,
    };

    /* Set up a repeating delayed work to blink the DK's LEDs when attention
is
     * requested.
     */
    static struct k_work_delayable attention_blink_work;
    static bool attention;

    static void attention_blink(struct k_work *work)
    {
        static int idx;
        const uint8_t pattern[] = {
    #if DT_NODE_EXISTS(DT_ALIAS(led0))
            BIT(0),
    #endif
    #if DT_NODE_EXISTS(DT_ALIAS(led1))
            BIT(1),
    #endif
    #if DT_NODE_EXISTS(DT_ALIAS(led2))
            BIT(2),
    #endif
    #if DT_NODE_EXISTS(DT_ALIAS(led3))
            BIT(3),
    #endif
        };

        if (attention) {
            dk_set_leds(pattern[idx++ % ARRAY_SIZE(pattern)]);
            k_work_reschedule(&attention_blink_work, K_MSEC(30));
        } else {
            dk_set_leds(DK_NO_LEDS_MSK);
```

```
    }
}

static void attention_on(struct bt_mesh_model *mod)
{
    attention = true;
    k_work_reschedule(&attention_blink_work, K_NO_WAIT);
}

static void attention_off(struct bt_mesh_model *mod)
{
    /* Will stop rescheduling blink timer */
    attention = false;
}

static const struct bt_mesh_health_srv_cb health_srv_cb = {
    .attn_on = attention_on,
    .attn_off = attention_off,
};

static struct bt_mesh_health_srv health_srv = {
    .cb = &health_srv_cb,
};

BT_MESH_HEALTH_PUB_DEFINE(health_pub, 0);

static struct bt_mesh_elem elements[] = {
    BT_MESH_ELEM(1,
             BT_MESH_MODEL_LIST(BT_MESH_MODEL_CFG_SRV,
                    BT_MESH_MODEL_HEALTH_SRV(&health_srv,
                                 &health_pub),
                    BT_MESH_MODEL_SENSOR_SRV(&sensor_srv)),
             BT_MESH_MODEL_NONE),
};

static const struct bt_mesh_comp comp = {
    .cid = CONFIG_BT_COMPANY_ID,
    .elem = elements,
    .elem_count = ARRAY_SIZE(elements),
};

const struct bt_mesh_comp *model_handler_init(void)
{
    k_work_init_delayable(&attention_blink_work, attention_blink);
    k_work_init_delayable(&end_of_presence_work, end_of_presence);

    dev = device_get_binding(SENSOR_INST);

    if (dev == NULL) {
        printk("Could not initiate temperature sensor\n");
    } else {
        printk("Temperature sensor (%s) initiated\n", dev->name);
    }
```

```
    dk_button_handler_add(&button_handler);

    return &comp;
}
```