UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN MECATRÓNICA

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN MECATRÓNICA

TEMA:

"PLATAFORMA PARA EXPERIMENTACIÓN NATURALÍSTICA EN BICICLETAS: SISTEMA EMBEBIDO"

AUTOR: IVÁN ANDRÉS QUISHPE QUIROZ

DIRECTOR: CARLOS XAVIER ROSERO CHANDI

IBARRA

2019

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

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

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DEL AUTOR			
CÉDULA DE IDENTIDAD	0401850268		
APELLIDOS Y NOMBRES	QUISHPE QUIROZ IVÁN ANDRÉS		
DIRECCIÓN	Los Olivos		
EMAIL	iaquishpeq@utn.edu.ec - andrz9352@gmail.com		
TELÉFONO FIJO	06 – 2510 – 565	TELÉFONO MÓVIL	0967501828
DATOS DE LA OBRA			
TÍTULO	“PLATAFORMA PARA EXPERIMENTACIÓN NATURALÍSTICA EN BICICLETAS: SISTEMA EMBEBIDO”		
AUTOR	IVÁN ANDRÉS QUISHPE QUIROZ		
FECHA	11-07-2019		
PROGRAMA	PREGRADO		
TÍTULO POR EL QUE OPTA	INGENIERO EN MECATRÓNICA		
DIRECTOR	CARLOS XAVIER ROSERO CHANDI		

2. CONSTANCIA

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 el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 11 días del mes de julio de 2019

Iván Andrés Quishpe Quiroz
C.I.: 0401850268

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN

En calidad de director del trabajo de grado “PLATAFORMA PARA EXPERIMENTACIÓN NATURALÍSTICA EN BICICLETAS: SISTEMA EMBEBIDO”, presentado por el egresado IVÁN ANDRÉS QUISHPE QUIROZ, para optar por el título de Ingeniero en Mecatrónica, certifico que el mencionado proyecto fue realizado bajo mi dirección.

Ibarra, Julio de 2019

Carlos Xavier Rosero Chandi
DIRECTOR DE TESIS

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

DECLARACIÓN

Yo, Iván Andrés Quishpe Quiroz con cédula de identidad Nro. 0401850268, declaro bajo juramento que el trabajo aquí escrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Universidad Técnica del Norte - Ibarra, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

Ibarra, Julio de 2019

Iván Andrés Quishpe Quiroz
C.I.: 0401850268

Agradecimiento

A mi madre Esthela.

Que siempre creyó en mí y ha estado conmigo en mis triunfos y fracasos brindándome todo su amor, por ser el pilar fundamental para la culminación de mi carrera.

A mi padre Iván.

Por guiarme a lo largo de toda mi vida, por inculcarme enseñanzas que me han convertido en la persona que soy actualmente.

A mis abuelitos Luciano y Teresa.

Por las palabras de aliento, consejos; por apoyarme, cuidarme y quererme siempre.

A mis hermanas.

A mi hermana Gaby por el amor y consejos de hermana mayor, a mi hermana Emily por estar conmigo y pasar momentos inolvidables siempre.

A mi tutor Xavier Rosero.

Por su tiempo, confianza, dedicación, por su gran apoyo, motivación, por su valiosa guía y asesoramiento en el desarrollo de este trabajo.

A todos aquellos familiares que me apoyaron directa o indirectamente en mi trajinar estudiantil.

A mis amigos por haber hecho de mi etapa estudiantil un trayecto con vivencias inolvidables, además, por la gran calidad humana que siempre me han demostrado.

Dedicatoria

Este trabajo esta dedicado a mis padres Esthela e Iván por haberme dado la vida, por enseñarme que todo se puede lograr con mucho trabajo y dedicación. Por el sacrificio mostrado para salir adelante y sobre todo por su amor.

A mis abuelitos Luciano y Teresa por formar parte en mi desarrollo personal, brindandome todo su amor y apoyo incondicional.

Andrés Q.

Resumen

Actualmente, el método más acertado para comprender el comportamiento de ciclistas en un entorno urbano es recopilar y analizar datos naturalísticos. Este documento presenta la primera parte del desarrollo de una plataforma de estudio naturalístico en bicicletas, para ello se desarrolla un sistema embebido implementado en una bicicleta de montaña.

El sistema está conformado por un mini computador Raspberry Pi 3 modelo B como unidad central destinada a obtener y almacenar velocidad, aceleración, tiempo, orientación y datos de posicionamiento global del ciclista. La orientación expresada en ángulos de Euler y la aceleración se obtienen por medio de una unidad de medición inercial MPU-9250. La información de posicionamiento global y velocidad sobre la superficie terrestre se obtiene desde un receptor GY-GPS6MV2. Adicionalmente, se lee el estado de la batería con un microcontrolador Attiny85. El mini computador comanda el sistema mediante algoritmos de programación desarrollados en el lenguaje Python.

Además, el sistema consta de un menú representado en una pantalla OLED de 128x64 pixeles con 4 modos de funcionamiento : 1) configuración de la hora interna del dispositivo, 2) calibración de la unidad de medición inercial, 3) almacenamiento de tramas de información en una base de datos desarrollada en SQlite, 4) modo seguro de apagado para protección del sistema ya que se lee el estado de la batería con un microcontrolador Attiny85.

Las pruebas realizadas en el sistema establecen que las medidas de las variables físicas se encuentran dentro de un margen de error aceptable, cumpliendo el objetivo como dispositivo de medición. La base de datos desarrollada en SQLite permite de manera sencilla manipular, analizar y compartir la información.

Abstract

Currently, the most accurate method to understand the behavior of cyclists in an urban environment is to collect and analyze naturalistic data. This document presents the first part of the development of a naturalistic study platform on bicycles, for which an embedded system is developed, implemented on a mountain bike.

The system is conformed by a Raspberry Pi 3 model B mini-computer as the central unit destined to obtain and store speed, acceleration, time, orientation and global positioning data of the cyclist. The orientation expressed in Euler angles and the acceleration are obtained by means of an inertial measurement unit MPU-9250. The global positioning and speed information on the Earth's surface is obtained from a GY-GPS6MV2 receiver. Additionally, the state of the battery is read with an Attiny85 microcontroller. The mini computer commands the system through programming algorithms developed in the Python language.

In addition, the system consists of a menu represented on a 128x64-pixel OLED screen with 4 operating modes: 1) configuration of the internal time of the device, 2) calibration of the inertial measurement unit, 3) storage of information frames in a database developed in SQlite, 4) safe shutdown mode for system protection since the battery status is read with an Attiny85 microcontroller.

The tests realized on the system establish that the measurements of the physical variables are within an acceptable margin of error, fulfilling the objective as a measuring device. The database developed in SQLite allows in a simple way to manipulate, analyze and share information.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Este trabajo de grado ha sido realizado con el *Grupo de Investigación en Sistemas Inteligentes de la Universidad Técnica del Norte (GISI-UTN).*

1.1. Problema

La bicicleta es una alternativa de movilidad menos contaminante, menos costosa, más eficaz y más amigable [1]. En Ecuador tres de cada diez hogares tienen al menos una bicicleta y su uso se incrementa cada año. Según [2] el 49,83 % de las personas usan bicicleta al menos una vez a la semana, el 34,09 % la usa a diario.

La seguridad del ciclista es una preocupación constante debido al alto índice de accidentes de tránsito a causa de impericia o imprudencia de los conductores de vehículos (51,9 %), seguida del irrespeto a las señales de tránsito (13,4 %) y en tercer lugar debido al exceso de velocidad (12,4 %)[2]. La bicicleta surge como alternativa viable tomando en cuenta los problemas que ocasiona un parque motor abultado [3].

Según cifras de la Comisión de Tránsito del Ecuador (CTE), en los tres primeros meses del año, los accidentes registrados con bicicletas y triciclos han aumentado en un 16.67 % con

respecto a todo el 2016 [4]. Esto es debido a que el ciclista desacata el sentido de las vías, realiza maniobras peligrosas al adelantar a los autos, invade las aceras y no respeta las señales de tránsito [5].

No existen maneras formales para el análisis del comportamiento del ciclista en las vías, que permitan determinar las causas precisas de los accidentes de tránsito involucrando bicicletas. Además, se carece de estudios previos y herramientas de monitoreo de las maniobras realizadas por el ciclista [6].

1.2. Motivación

El incremento de ciclistas genera importantes problemas de seguridad tales como: si su uso es compatible con la infraestructura de las carreteras y las regulaciones de tránsito existentes, además del impacto con el resto del tráfico [7].

En casos de accidentes los datos generalmente se obtienen mediante declaraciones del ciclista involucrado o de las versiones de testigos, ambos sujetos a sesgos y errores de informes; también, se investiga la escena del choque. Sin embargo, no pueden proporcionar detalles sobre todas las acciones previas al evento [8].

Desarrollar un estudio naturalístico destinado a bicicletas parece ser un enfoque prometedor e ideal para abordar estos problemas ya que registra de manera objetiva las experiencias de los ciclistas en la carretera y se analiza la información sin los posibles sesgos que se generan con otros métodos.

En este trabajo se diseña un sistema embebido implementado en una bicicleta para proporcionar la mayor cantidad de información naturalística.

1.3. Objetivos

1.3.1. Objetivo Principal

- Desarrollar un sistema embebido para la adquisición de variables naturalísticas en bicicletas.

1.3.2. Objetivos Específicos

- Determinar los requerimientos necesarios para el funcionamiento del sistema.
- Establecer las estrategias de muestreo, tratamiento y almacenamiento local considerando la naturaleza de las variables físicas.
- Implementar el sistema embebido para la recolección de datos proporcionados por los sensores en un tiempo constante de muestreo.

1.4. Justificación

La información obtenida servirá como plataforma para realizar estudios de conducción de ciclistas. Podría también usarse para el análisis de la siniestralidad que involucra el uso inadecuado de la bicicleta a través de información capturada en el momento del accidente, de la misma forma que una caja negra o registrador de vuelo en los aviones.

El mismo sistema podría ser replicado para compartir inalámbricamente estados de conducción en una comunidad de ciclistas, permitiendo la exploración de las condiciones de las rutas. Además, con la información obtenida a través del sistema se podría incrementar la eficiencia en el manejo deportivo de bicicletas e incluso aumentar la seguridad del ciclista.

1.5. Alcance

El dispositivo es el primer escalón por superar para conseguir el objetivo global, el cual se centra en el desarrollo de una plataforma que comprende hardware de adquisición y software de procesamiento de variables desde bicicletas.

El sistema constará de lo siguiente:

- Lectura, tratamiento y almacenamiento de variables tales como:
  - Velocidad
  - Aceleración
  - Posicionamiento global.
  - Orientación y Rumbo.
  - Tiempo
- Piezas mecánicas para el acople de sensores en bicicletas.
- Control manual para adquirir y enviar datos.
- Pantalla para visualización de estados de funcionamiento del sistema.
- Envío de la información hacia una base de datos en computador.

Capítulo 2

Sustento Teórico

2.1. Estudio naturalístico

Es un método de investigación establecido en la recopilación y análisis de información captada de participantes mientras realizan actividades cotidianas.

Enfocado al estudio de la conducción, este método se desarrolló primeramente en vehículos con la finalidad de generar información previa al accidente y apoyar al desarrollo de medidas para prevención de choques. En este estudio se analiza el comportamiento del conductor, el entorno y otras circunstancias relacionadas con incidentes críticos [9].

2.1.1. Estudio naturalístico en bicicletas

Basados en el estudio de conducción naturalístico en vehículos descrito en [9], se han desarrollado varios trabajos enfocados al ciclista para analizar: factores de riesgo a los que está expuesto [10][8], seguridad [11], comportamiento del ciclista [7], uso de la infraestructura [12] e infracciones cometidas [13].

La recopilación de información se la realiza a través de dos métodos:

- Incorporar cámaras de vídeo.
- Utilizar bicicletas instrumentadas.

2.1.2. Cámaras de vídeo implementadas

Las cámaras de video son utilizadas para obtener información visual y determinar eventos críticos que experimenta el ciclista en un trayecto normal. Las cámaras se instalan discretamente y no se proporciona información específica a los participantes con la finalidad de capturar el comportamiento real [14]. La ubicación de las cámaras depende de las variables que se necesiten analizar.

En [15] se utiliza una cámara de video oculta en una caja y fijada a un poste de señalización para observar el comportamiento de ciclistas en 10 intersecciones de Melbourne. Este estudio analiza el comportamiento del ciclista al cruzar la intersección y la tasa de infracción de la luz roja. Un estudio similar llevado a cabo por [16] utiliza cámaras de vídeo en 3 intersecciones de China, se captura información sobre 229 ciclistas convencionales y 222 ciclistas e-bike (ver Figura 2.1).

Las cámaras de video se incorporan también en los cascos de ciclistas y en la parte frontal de la bicicleta para estudiar desde la perspectiva del ciclista la interacción con el tráfico e identificar factores de riesgo y las causas previas a accidentes y casi accidentes[8].

El estudio naturalístico realizado en el territorio de la capital australiana por [17] utiliza una cámara de vídeo en el casco del ciclista con un registrador de datos sobre posicionamiento global para analizar a 36 participantes en interacción con conductores vehiculares y así determinar alternativas para mejorar la seguridad del ciclista.

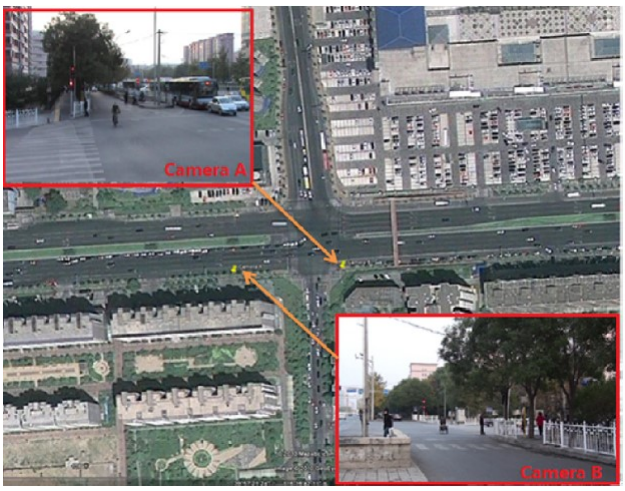


Figura 2.1: Cámaras implementadas en China [16]

Sin embargo, una investigación limitada a video es más lenta e imprecisa a causa de que la información es más difícil de clasificar y analizar, por requerir un proceso manual y sistemático a todo el conjunto de datos generando mayor consumo de tiempo [18].

2.1.3. Bicicletas instrumentadas

Se utilizan bicicletas equipadas con diferentes sensores con el fin de obtener mayor cantidad de información objetivamente. En [18] (ver Figura 2.2) se utiliza una bicicleta equipada con un GPS[1], sensores de presión para el freno, sensores de velocidad, una unidad de medición inercial conformada por un acelerómetro, un giroscopio y un magnetómetro de 3 ejes cada uno,una interfaz hombre-máquina en el manubrio para que los participantes registren los eventos críticos por medio de un pulsador. Además, dispone de dos cámaras de vídeo: una en dirección al ciclista para analizar la reacción del conductor y otra al frente de la bicicleta para supervisar el entorno.

[1] Sistema de posicionamiento global (global positioning system)

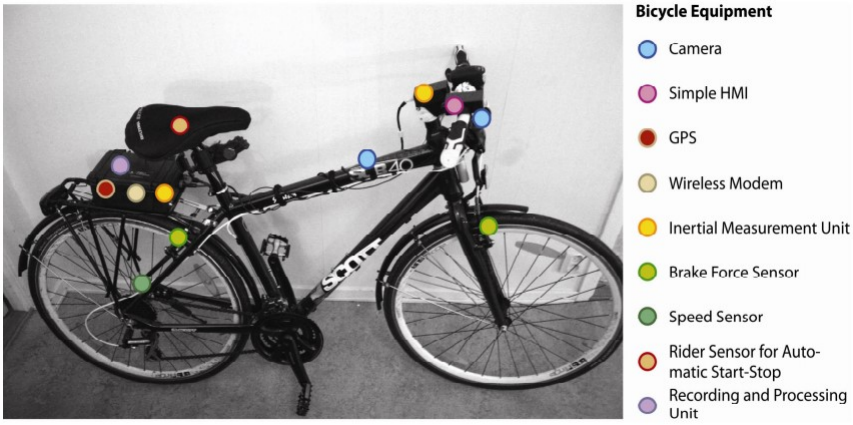


Figura 2.2: Bicicleta instrumentada utilizada en [18]

Se instrumentan 5 bicicletas en [19] con los sensores detallados en la Tabla 2.1 para recolectar información de 20 participantes e identificar la dinámica de un ciclo normal en bicicleta y el comportamiento del ciclista (ver Figura 2.3).

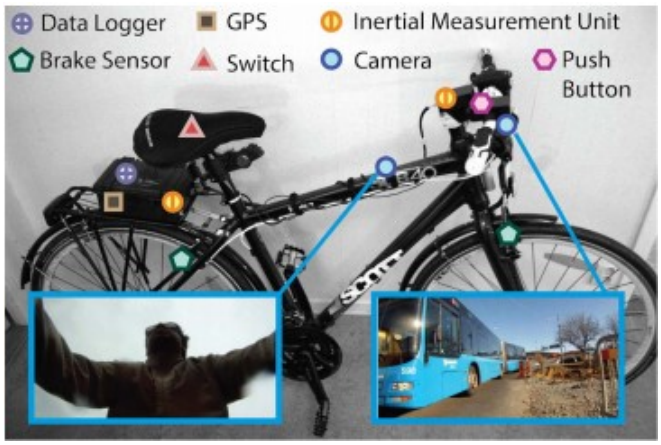


Figura 2.3: Bicicleta implementada en [19]

Tabla 2.1: Sensores y variables utilizados en [19]

Tipo	**Nombre**	**Datos obtenidos**	**Resolución**
Cámara	GoPro Hero, Hero2	Tramas de video	1920x1080 pixel
Acelerómetro	Phidgets IMU 1056	3D Aceleración	22 μg
Brújula		3D Vector direccional	400 μG
Giroscopio		Velocidad Angular	0.02 ^{o}/seg
GPS	Phidgets GPS 1040	Latitud y Longitud	2.5m error circular mínimo
		Rumbo	Varía con el error circular
		Velocidad	100 mm/s
Sensor de fuerza de frenado	Sensor de fuerza resistivo	Presión	0.01 N

En [14] se utiliza una bicicleta equipada con un sistema de adquisición de datos llamado *Mini-DAS* (ver Figura 2.4) desarrollado por el *VTTI*[2]. El *Mini-DAS* incluye sensores tales como: acelerómetro, giroscopio y un GPS. Además, implementa 2 cámaras: una ubicada hacia adelante para capturar el entorno y otra posicionada en dirección a la cara y al cuerpo del ciclista.

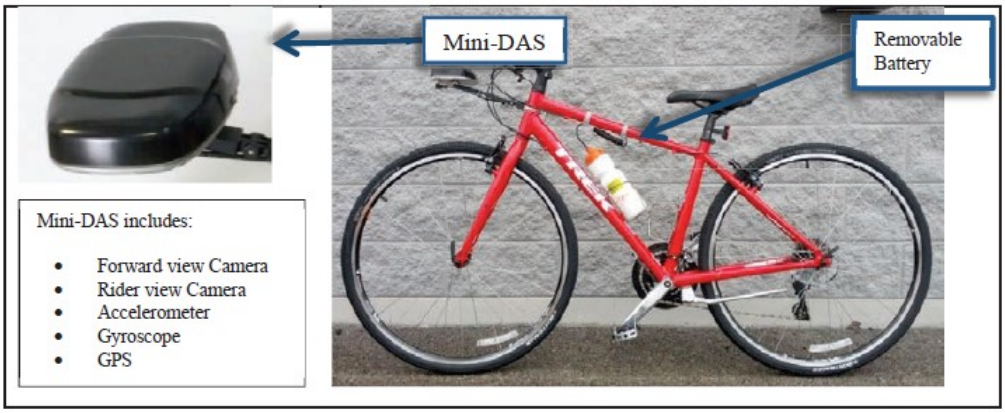


Figura 2.4: Sistema de adquisición Mini-Das [14]

De los trabajos citados anteriormente se destaca la utilización de sensores inerciales para capturar la orientación y rumbo del ciclista, GPS para obtener datos de geolocalización, sensores de velocidad y de frenado.

[2]Instituto de Transporte de Virginia Tech (Virginia Tech Transportation Institute)

2.2. Dispositivos de posicionamiento global

Los dispositivos de posicionamiento global forman parte de un sistema de radionavegación satelital desarrollado y gestionado por el departamento de Defensa de Estados Unidos, los cuales proporcionan al usuario estimaciones precisas de posición, velocidad y tiempo de un objeto sobre la superficie terrestre a través del protocolo NMEA[3] [20].

2.2.1. Protocolo NMEA

Este protocolo creado por la *Asociación Nacional de Electrónica Marina* de los Estados Unidos hace posible la comunicación entre dispositivos de navegación marítima y actualmente es usado también por receptores GPS. Está compuesto por un conjunto de mensajes que no tienen longitud fija, sin embargo, el número máximo de caracteres ASCII[4] permitidos es de 79 excluyendo el comienzo y fin de trama [20].

Los datos son emitidos en forma de sentencias bajo la siguiente estructura: empieza con "$"seguido de la identificación del receptor (dos letras), identificación del mensaje (tres letras), número de campos separados por comas, checksum optativo (consta de "$*$" y dos dígitos hexadecimales) y termina con "\r\n"[21].

El protocolo NMEA permite interpretar varias sentencias debido a que los fabricantes de receptores GPS pueden definir sentencias propias. Las sentencias más representativas son: GGA, GLL, GSA, GSV, RMC, VTG y ZDA.

2.2.1.1. Sentencia GGA

Esta sentencia describe al sistema de posicionamiento global de datos fijos. Es un mensaje que posee datos relacionados con el tiempo, posición y el arreglo para un receptor GPS. La

[3] Asociación Nacional de Electrónica Marina (National Marine Electronics Association)

[4] Código estándar estadounidense para el intercambio de información (American standard code for information interchange)

estructura de este mensaje se detalla en la Tabla 2.2 [20].

Estructura: $ GPGGA,a,b,c,d,e,f,g,h,i,j,k,l,m,n*xx

Tabla 2.2: Descripción de la sentencia GGA

Campo	**Descripción**
a	Tiempo UTC: horas, minutos, segundos
b	Latitud en grados y decimas de minutos
c	Dirección de la latitud: N=norte, S=sur
d	Longitud en grados y decimas de minutos
e	Dirección de la longitud: E=este, W=oeste
f	Tipo de posición: 1=absoluto, 2= RTCM
g	Número de satélites activos
h	Dilución de la posición horizontal (HDOP)
i	Altitud en metros sobre el elipsoide de referencia
j	Unidad en que se mide la altitud de la antena: M=metros
k	Separación geoidal en metros
l	Unidad en que se mide la separación geoidal: M=metros
m	Edad de las correcciones diferenciales (en segundos)
n	Identificación de la estación de referencia (modo RTCM)
*xx	Checksum

2.2.1.2. Sentencia GLL

Esta sentencia contiene la posición geográfica *latitud-longitud*. Es un mensaje que contiene información sobre el tiempo y posición. La estructura de este mensaje se detalla en la Tabla 2.3 [20].

Estructura: $GPGLL,a,b,c,d,e,f*xx

Tabla 2.3: Descripción de la sentencia GLL

Campo	Descripción
a	Latitud en grados y décimas de minutos
b	Dirección de la latitud: N=norte, S=sur
c	Longitud en grados y décimas de minutos
d	Dirección de la longitud: E=este, W=oeste
e	Tiempo UTC: horas, minutos, segundos
f	Estado: A=Válido, V=Inválido
*xx	Checksum

2.2.1.3. Sentencia GSA

Esta sentencia abarca información acerca de los satélites empleados para el cálculo de la posición, los valores del HDOP *(precisión horizontal)*, VDOP *(precisión vertical)* y PDOP *(precisión de posición)*. La estructura de este mensaje se detalla en la Tabla 2.4 [20].

Estructura: $GPGSA,a,b,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,d,e,f*xx

2.2.1.4. Sentencia GSV

Este mensaje incluye información de los satélites empleados en el cálculo de la posición, número de satélites, elevación, azimut y relación señal/ruido. La Tabla 2.5 detalla la estructura de este mensaje [20].

Estructura: $GPGSV,a,b,c,d,e,f,.....,g*cc

Tabla 2.4: Descripción de la sentencia GSA

Campo	Descripción
a	Modo: M=manual, A=automático
b	Modo: 1=Inicializándose, 2=2D, 3=3D
c1-c12	Satélites empleados en los cálculos de posición (nulos para los campos no utilizados)
d	PDOP
e	HDPO
f	VDPO
*xx	Checksum

Tabla 2.5: Descripción de la sentencia GSV

Campo	Descripción
a	Número total de mensaje
b	Número de mensaje
c	Número total de satélites a la vista
d	Satélite
e	Ángulo de elevación en grados
f	Azimut, grados del norte verdadero
g	Relación señal/ruido (dB/Hz)
*xx	Checksum

2.2.1.5. Sentencia VTG

Este mensaje aporta con información sobre la orientación y velocidad de movimiento sobre el suelo. Su estructura se presenta a continuación en la Tabla 2.6 [20].

Estructura: $GPVTG,a,b,c,d,e,f,g,h*xx

2.2.1.6. Sentencia RMC

Esta sentencia muestra información sobre la posición y otros parámetros de navegación. En la Tabla 2.7 se explica la estructura de este mensaje [20].

Estructura: $GPRMC,a,b,c,d,e,f,g,h,i,j,k*xx

Tabla 2.6: Descripción de la sentencia VTG

Campo	Descripción
a	Orientación respecto al norte, en grados)
b	Norte
c	Orientación respecto al norte magnético, en grados
d	Norte Magnético
e	Velocidad en nudos
f	Unidades: N=nudos
g	Velocidad en Km/h
h	Unidades: K=km/h
*xx	Checksum

Tabla 2.7: Descripción de la sentencia RMC

Campo	Descripción
a	Tiempo UTC: horas, minutos, segundos
b	Estado: A=válido, V=inválido
c	Latitud en grados y decimas de minutos
d	Dirección de la latitud: N=norte, S=sur
e	Longitud en grados y decimas de minutos
f	Dirección de la longitud: E=este, W=oeste
g	Velocidad en nudos
h	Orientación de la trayectoria en grados
i	Fecha en mes, día y año
j	Variación magnética en grados
k	Dirección de la variación: E=este, W=oeste
*xx	Checksum hexadecimal

2.2.1.7. Sentencia ZDA

Este mensaje abarca la información referente a la fecha y zona horaria. La estructura de esta sentencia se muestra en la Tabla 2.8 [20].

Estructura: $GPZDA,a,b,c,d,e,f*xx

Tabla 2.8: Descripción de la sentencia ZDA

Campo	Descripción
a	Tiempo UTM en horas, minutos y segundos
b	Día
c	Mes
d	Año
e	Zona horaria: desfase respecto al UTC, s=signo, h=hora
f	Zona horaria: desfase respecto al UTC, s=signo, m=minutos
*xx	Checksum

2.3. Sistemas de navegación y orientación

Un sistema de navegación inercial *(INS-Inertial Navigation System)* se compone de sensores de movimiento y un computador para determinar la posición, orientación y velocidad de un objeto en relación a un punto de inicio conocido [22].

Los sistemas de referencia de actitud y rumbo *(AHRS-Attitude and Heading Reference System)* proporcionan información sobre el rumbo y orientación de un objeto a través de sensores como: giroscopios, acelerómetros, magnetómetros y algunos integran receptores GPS.

Existen dos tipos de sistemas de navegación inercial: gimbaled y strapdown.

Sistema gimbaled: En este sistema los sensores inerciales se montan en una plataforma aislada de los movimientos de rotación externos. La plataforma se sostiene a un marco rígido que rota, permitiendo cancelar cualquier rotación [22].

Sistema strapdown: En este sistema los sensores inerciales tienen los ejes alineados a los ejes del objeto en el que se encuentran y se mueven conjuntamente [22]. Los sistemas Strapdown son usados actualmente debido a la gran cantidad de información de navegación que ofrecen, ya que dispone de sensores inerciales (acelerómetros y giroscopios) [23].

2.3.1. Sistema de coordenadas

Son un conjunto de vectores o coordenadas que permiten especificar la posición y orientación de un punto respecto a una referencia conocida. Existen varios sistemas de coordenadas aplicados en el estudio de movimiento [24]:

Inercial I-frame: Este sistema es idealmente inercial ya que no rota ni se mueve con respecto a nada. Muy difícil de conseguir en la práctica.

ECI i-frame: Se considera como un sistema inercial, su origen es en el centro de masa de la Tierra. El eje Z coincide con el eje polar y el plano vertical al eje Z coincide con el Ecuador. Los ejes "x" y "y" no rotan con la Tierra.

ECEF e-frame (Sistemas de coordenadas geocéntrico) : El origen está en el centro de masa de la Tierra, el eje Z apunta al norte a lo largo del eje polar. Los ejes "x" y "y" rotan con la Tierra y se encuentran en el plano ecuatorial.

De Navegación n-frame: Se conoce como NED (North, East, Down) debido a que los ejes señalan estas direcciones. El eje "x" apunta al Norte, el "y" al Este y "z" hacia abajo. El origen es en la localización del sistema inercial.

Body b-frame: Es usado normalmente en plataformas Strap-down. Su origen coincide con el centro de masa del objeto.

2.4. Sensores inerciales

Estos dispositivos permiten calcular la aceleración y velocidad angular de un objeto en un sistema de coordenadas tridimensional ortogonal propio [25].

Una *IMU*[5] se caracteriza por utilizar giroscopios para estimar la variación de rotación (velocidad angular), acelerómetros para medir aceleración y magnetómetros para obtener la inten-

[5]Unidad de medición inercial (inertial measurement unit)

sidad del campo magnético de la Tierra y así determinar el rumbo.

El principal uso de estos sensores es en la navegación de automóviles, naves (comerciales o militares) y vehículos aéreos no tripulados. Además, son empleados como sensores de captura movimiento en personas y robots.

2.4.1. Giroscopios

Son dispositivos que miden velocidad angular sobre un eje, esta se expresa en grados por segundo (^{o}/s) o revoluciones por segundo (RPS). Su funcionamiento se basa en el desplazamiento de una pequeña masa resonante a medida que se gira el sensor [26].

Las características más significativas de un giroscopio son el rango y la sensibilidad. El rango es el valor máximo de velocidad angular que puede detectar y la sensibilidad es el valor mínimo y se expresa en milivoltios por grado por segundo (mV/^{o}/s) [26].

2.4.2. Acelerómetros

Estos dispositivos permiten detectar variaciones en las fuerzas estáticas *(gravedad)* y/o dinámicas *(vibraciones o movimientos)* [27]. Su funcionamiento se basa en determinar la diferencia entre la aceleración total y la aceleración gravitacional alrededor de sus ejes en los que se genera el movimiento [28].

Los parámetros de desempeño de un acelerómetro se muestran a continuación:

- Unidad de medida: ”g”dónde $1g = 9{,}8m/s^2$.
- Sensibilidad
- Rango
- Resolución

Con la finalidad de reducir el tamaño de los sensores y obtener la aceleración con otros métodos surgen nuevas micro tecnologías (MEMS)[6]. Con estas tecnologías se crean sensores capacitivos los cuales integran un muelle de silicón de masa definida que al vibrar produce una variación en la capacitancia de dos electrodos uno fijo y otro que se mueve junto con la masa, esta variación es proporcional a la posición de la masa [29].

2.4.3. Magnetómetros

Son dispositivos capaces de medir la intensidad del campo magnético de la tierra. El funcionamiento de un magnetómetro es a través de sensores de efecto Hall en miniatura, el voltaje resultante es proporcional a la intensidad y polaridad del campo magnético.

2.5. Métodos para obtener la orientación y rumbo

La orientación o actitud es el ángulo de inclinación respecto de los ejes de referencia determinados. El rumbo es el angulo de dirección en el cual se realiza la navegación respecto a una referencia establecida, la referencia generalmente es el meridiano terrestre o el rumbo magnético dado por una brújula.

Existen varios métodos para obtener la actitud y rumbo a partir de sensores.

- Cálculo de la actitud usando giroscopios.
- Cálculo de la actitud usando acelerómetros.
- Cálculo del rumbo usando magnetómetros.

[6] Sistemas microelectromecánicos (microelectromechanical systems)

2.5.1. Orientación usando giroscopios

Para calcular la posición angular se realiza una integración de la velocidad angular en el tiempo. Este cálculo está dado por la Ecuación 2.1 [23].

$$d\theta = w * dt \tag{2.1}$$

Dónde:

$d\theta$ =*ángulo medido*

w =*velocidad angular medida*

dt =*tiempo de muestreo (tiempo transcurrido entre cada medida)*

Sin embargo, con la Ecuación 2.1 no se determina el ángulo de giro total en un intervalo de tiempo y respecto a un ángulo inicial establecido; por lo tanto, se dispone de un algoritmo acumulativo expresado en la Ecuación 2.2 para determinar el ángulo total en un rango de tiempo.

$$\theta_k = \theta_{k-1} + w_k * dt \qquad k = 0,1,2,3,4,...,n \tag{2.2}$$

Dónde:

θ_k =*ángulo actual*

θ_{k-1} =*angulo anterior*

w_k =*velocidad angular actual*

dt =*tiempo de muestreo (tiempo transcurrido entre cada medida*

Utilizar un giroscopio para determinar el ángulo de rotación presenta dos desventajas que son: la deriva, generada por el incremento en el tiempo del error en la medida del ángulo, el segundo inconveniente es que no se tiene inicialmente un valor para θ_{k-1} [23].

2.5.2. Orientación usando acelerómetros

El acelerómetro cuando se encuentra paralelo a la superficie terrestre se usa para medir ángulos de Euler; de esta manera los ejes de medición se encuentran perpendiculares a la gravedad, cuando el sensor se inclina aparecen componentes de gravedad en los ejes "x" y "y" como se ilustra en la Figura 2.5 [23].

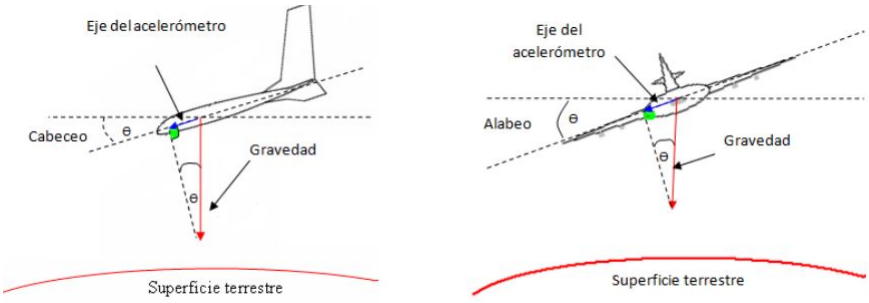


Figura 2.5: Cabeceo y Alabeo [23]

Para obtener el valor de la inclinación sobre cada eje mostrado en la Figura 2.6 se hace uso de algoritmos de trigonometría. El ángulo *"cabeceo"* es el ángulo formado por las componentes "x" y "z", el ángulo *"alabeo"* por las componentes "y" y "z", el tercer eje se considera igual a la gravedad y es el eje sobre el cual se mide las rotaciones. El cálculo de estos ángulos se muestra en las Ecuaciones 2.3 y 2.4.

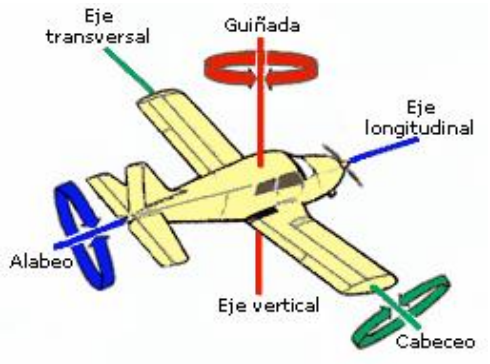


Figura 2.6: Ángulos: cabeceo (pitch), alabeo (roll), guiñada (yaw) [30].

$$Roll = \arctan \frac{Ay}{\sqrt{Az^2 + Ax^2}} \tag{2.3}$$

$$Pitch = \arctan \frac{Ax}{\sqrt{Az^2 + Ay^2}} \tag{2.4}$$

Dónde:

Ax =*Component de la gravedad sobre el eje x.*

Ay =*Componente de la gravedad sobre en eje y.*

Az =*Componente de la gravedad sobre en eje z.*

El inconveniente de realizar este cálculo es que el sensor al detectar aceleración lineal, además de vibraciones; puede alterar la medida de los ángulos de Euler.

2.5.3. Rumbo usando magnetómetros

Con este sensor se obtiene el *"heading"* usando la Ecuación 2.5. El *heading* es el ángulo calculado por medio de las componentes del campo magnético de la Tierra y es igual a la diferencia entre el heading actual y la declinación magnética [31].

La declinación magnética es el ángulo entre el norte magnético (ángulo indicado por una brújula) y el norte verdadero (norte geográfico), el valor de la declinación magnética varía dependiendo del lugar geográfico en el cual se encuentra el sensor.

$$heading = \arctan \frac{By}{Bx} \tag{2.5}$$

Dónde:

By =*Componente del campo magnético terrestre sobre el eje y.*

Bx =*Componente del campo magnético terrestre sobre el eje x.*

La representación de la rotación del sensor alrededor de 360 grados, debería ser una circunferencia perfecta con el centro en el origen. Sin embargo, por las alteraciones del campo magnético de la tierra la representación no es la indicada y se debe realizar un proceso de calibración denominado “hard iron” y “soft iron”. Este proceso de calibración se detalla en [31].

2.5.4. Formas de representar la actitud

Existen varias formas de representar matemáticamente la orientación de un cuerpo con respecto a una referencia conocida [32], las más usuales son :

Matriz de cosenos de dirección(DCM) “Direction Cosine Matrix”: Cada elemento de la matriz es uno de los cosenos de los ángulos entre los ejes del sistema de coordenadas.

Cuaternios (Quatnions): Se representa como un vector que posee cuatro elementos que indican la orientación y la magnitud de rotación.

Ángulos de Euler: Esta representación comprende los ángulos de roll - alabeo (rotación sobre el eje “x”), pitch - cabeceo (rotación sobre el eje “y”) y yaw - rumbo o guiñada (rotación sobre el eje “z”).

Estas formas se relacionan entre sí, ya que es posible representar una en función de otra, mediante relaciones matemáticas detalladas en [24].

2.6. Display de visualización

En este apartado se detalla los tipos y características de pantallas de visualización más populares en el desarrollo de proyectos.

2.6.1. Display LCD de líneas

Los dispositivos LCD[7] son más comunes en proyectos ya que permiten visualizar información de manera muy sencilla y su precio es relativamente bajo.

Estas pantallas comúnmente llamadas de cristal líquido son delgadas y están conformadas por un arreglo de pixeles monocromos que se sitúan delante de un haz de luz. Estas pantallas permiten mostrar caracteres que se forman por una matriz de 8x5 pixeles.

Dependiendo de la cantidad de líneas que posee la pantalla se tienen varios tipos:

- Display 2x16 (2 líneas y 16 caracteres por línea).
- Display 4x20 (4 líneas y 20 caracteres por línea).

2.6.1.1. Conexionado

La mayoría de estas pantallas poseen dieciséis pines (catorce si no posee retroiluminación). Estos pines se describen en la Tabla 2.9.

Tabla 2.9: Descripción de pines LCD de líneas

Número	**Pin**	**Descripción**
1	V_{SS}	Tierra
2	V_{DD}	Alimentación
3	V_0	Contraste de pantalla
4	RS	Registro de selección
5	R/W	Modo: 1=lectura / 0=escritura
6	E	Habilitar pantalla
7-10	DB0-DB3	Bus de comunicación (8bits)
11-14	DB4-DB7	Bus de comunicación (4bits)
15	LED+	Alimentación retroiluminación
16	LED-	Tierra retroiluminación

[7]Pantalla de cristal líquido (liquid crystal display)

2.6.1.2. Conector I2C

Las pantallas se comunican de manera muy sencilla, pero al utilizar muchos pines de conexión surge un inconveniente para utilizar con un microcontrolador especialmente de gama baja.

Utilizando un conector I2C se logra solucionar este inconveniente de conexión ya que sólo se utilizan cuatro pines, dos para alimentación y dos para comunicación.

2.6.2. Display LCD en matriz de puntos

Toda la pantalla se conforma por una matriz de pixeles que pueden ser activados para representar gráficos debido a que presenta mejor resolución.

Se usan normalmente en impresoras 3D y máquinas CNC como menús ya que permiten una interfaz más amigable para el usuario.

2.6.3. Display OLED

Los displays OLED[8] son pantallas de tamaño reducido conformadas por leds elaborados con materiales orgánicos. Utilizadas principalmente en proyectos con limitaciones de tamaño, son más luminosas con mayor contraste, tienen gran resolución ya que se conforman por una matriz de puntos y tienen menor consumo de energía.

2.6.3.1. Clasificación

La clasificación de estas pantallas se detalla en la Tabla 2.10.

[8]Diodo orgánico emisor de luz (organic light emitting diode)

Tabla 2.10: Clasificación Display OLED

Campo	**Clasificación**
Por el color	Monocromática.
	Color.
Comunicación	I2C
	3-SPI
	4-SPI
Número de pines	4 pines (I2C)
	7 pines (I2C, 3-SPI, 4-SPI)
Por el controlador IC	SSD1306
	SSD1331
Por el tamaño	(128x32) = 0.91”
	(128x64) = 0.96”

Capítulo 3

Diseño del sistema

En este capítulo se determinan las características y funcionalidades de hardware y software necesarias para satisfacer las demandas que se han propuesto para este trabajo.

3.1. Descripción general

El sistema se diseña para cumplir la función de instrumento de medida implementando una unidad central encargada de adquirir, procesar y almacenar información recopilada desde sensores destinados a obtener las variables relacionadas en el comportamiento de la conducción del ciclista.

3.1.1. Requerimientos del sistema

El sistema debe cumplir con las siguientes características:

- Estar desarrollado en hardware y software libre.
- Proporcionar datos de geolocalización del ciclista.
- Calcular la orientación y rumbo.
- Obtener velocidad y aceleración.

- Poseer una pantalla para la visualización de estados de funcionamiento del sistema.
- Lectura del estado de la batería de alimentación.
- Permitir un manejo simple e interactivo al usuario.
- Piezas para la correcta sujeción a la bicicleta.
- Almacenamiento de tramas de información en una base de datos.

3.1.2. Diagrama de bloques

El sistema presenta varios módulos representados en la Figura 3.1. Estos módulos son:

- Unidad central: es la plataforma de hardware libre que comanda el funcionamiento del sistema.
- Recetor GPS: encargado de enviar a la unidad central la información de latitud y longitud para determinar la ubicación exacta del ciclista sobre la superficie terrestre.
- Sensor de velocidad: la unidad central recibe información y realiza el cálculo de velocidad lineal del ciclista.
- Sensor inercial: envía información a la unidad central para el cálculo de la orientación sobre cada uno de los ejes del sistema de coordenadas.
- Pantalla: permite representar visualmente el estado de funcionamiento del sistema.
- Sensor de voltaje: determina si la alimentación del sistema es suficiente para el correcto funcionamiento del sistema.
- Base de datos: almacena las tramas de información.

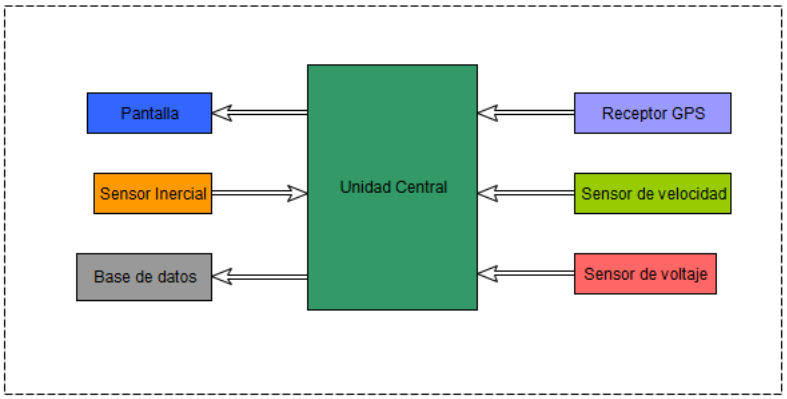


Figura 3.1: Diagrama de bloques

3.2. Caracterización del sistema

Se determinan los componentes que deben implementarse en el sistema para asegurar un correcto desempeño.

3.2.1. Selección de la unidad central

Dentro de las plataformas de hardware libre más reconocidas se encuentran: Raspberry Pi, Arduino, Beaglebone Black e Intel Galileo. Estas plataformas permiten el desarrollo de múltiples aplicaciones relacionadas con la electrónica, robótica, domótica, procesamiento de señales, telecomunicaciones, entre otras más; con la finalidad de crear e innovar. Además de contribuir a la generación de ciencia con menos recursos debido a su relativo bajo costo y reducido tamaño [33].

Cada tarjeta cuenta con fortalezas y debilidades para una determinada aplicación por lo cual se analizan las características más significativas de cada una. La Tabla 3.1 resume el estudio comparativo sobre las cuatro plataformas realizado por [34].

Para seleccionar la plataforma a implementarse, se consideran las siguientes características:

Tabla 3.1: Plataformas de hardware libre

Criterio \Placa	**Arduino Yun**	**BeagleBone Black**	**Intel Galileo**	**Raspberry Pi 3B**
Consumo de potencia	1.5w	2.3w	3w	5w
Memoria Ram	64 MB DDR2	512 MB DDR3	256 MB DRAM	1 GB
CPU	400 MHz	720 MHz	400 MHz	1.2 GHz
Alimentación	5V, 2A	5V, 2A	5V, 3A	5V, 2.5A
Sistema Operativo	Linux	Linux	Linux	Linux 64 bits
Conectividad	Puerto Ethernet 10/100 Base T Módulo Wi-Fi integrado	Puerto Ethernet 10/100 Base T	Puerto Ethernet 10/100 Base T	Puerto Ethernet 10/100 Base T Módulos Wi-Fi 802.11 y Bluetooth 4.1
Puertos USB	1	1	1	4
Salidas de video	No	Micro HDMI	No	HDMI
Entradas/Salidas	32 pines	2x46 pines	28 pines	40 pines
Ranura de tarjeta	Micro SD	Micro SD	Micro SD	Micro SD

capacidad para almacenar gran cantidad de información recopilada desde sensores, alta velocidad de procesamiento, admitir la conexión de dispositivos tales como GPS y sensores. Ya que el sistema se ubicará en la estructura de la bicicleta es necesario poder acceder remotamente a él.

La tarjeta que mejor se acopla a las demandas establecidas es Raspberry Pi modelo 3B debido a que posee memoria RAM de 1Gb, velocidad de procesamiento de 1.2GHz ideal para manejar gran cantidad de información, es posible conectarse a una red WI-FI y acceder remotamente a ella. Además, posee protocolos de comunicación como son: UART[1], SPI[2] e I2C[3] utilizados para la conexión con diferentes sensores y otros periféricos.

[1]Receptor-transmisor asíncrono universal (universal asynchronous receiver-transmitter)
[2]Interfaz periférica en serie (serial peripheral interface)
[3]Circuito inter-integrado (inter-integrated circuit)

3.2.2. Selección del receptor GPS

Para el desarrollo del sistema se ha escogido el receptor GPS modelo GY-GPS6MV2 debido a la facilidad de conexión y comunicación con la placa Raspberry Pi 3B seleccionada.

3.2.2.1. Receptor GY-GPS6MV2

Este modelo es de la serie NEO-6, posee un motor de posicionamiento de alto rendimiento u-blox 6. Su tamaño reducido lo hace ideal para proyectos con limitaciones de espacio y energía. Además, posee una antena de cerámica externa que suprime las interferencias y permite encontrar satélites al instante, mejorando la navegación incluso en entornos difíciles.

3.2.2.1.1. Características. Las especificaciones del receptor GPS GY-GPS6MV2 se detallan en la Tabla 3.2 [35].

Tabla 3.2: Características GY-GPS6MV2

Ítem	**Descripción**
Alimentación	2.7-3.6 V
Interfaces	UART, USB,SPI, DDC
Tiempo de conexión	27 s (arranque en frío) y 1 s (arranque en caliente)
Sensibilidad	-147 dBm (arranque en frío) y -156 dBm (arranque en caliente)
Precisión de posición horizontal	2.5m
Exactitud de velocidad	0.1 m/s
Altitud máxima	50000 m
Protocolos	NMEA, UBX, RTCM
Velocidad de comunicación	9600 Baudrate
Sentencias NMEA	GGA, GLL, GSA, GSV, RMC, VTG, TXT

3.2.2.1.2. Descripción de pines. El modelo GY-GPS6MV2 (Figura 3.2) dispone de cuatro pines, dos pines para alimentación y dos pines que permiten la comunicación serial. Dicha distribución se detalla en la Tabla 3.3.

Tabla 3.3: Descripción de pines GY-GPS6M

Número	Pin	Descripción
1	VCC	Alimentación
2	RX	Recepción
3	TX	Transmisión
4	GND	Tierra

Figura 3.2: Receptor GY-GPS6MV2

3.2.3. Selección de sensores inerciales

Se pueden encontrar estos dispositivos de manera separada o incluidos en un integrado como es el caso de una IMU. Un estudio de comparación realizado en [27] muestra que el uso de una IMU frente a los sensores: ADXL345, ITG3200 y HMGC5883L es mucho más conveniente para la reducción de tamaño, consumo energético, costo y optimización de procesamiento.

Considerando todas las ventajas que presenta el uso de una IMU, para el desarrollo del sistema se opta por utilizar la IMU MPU9250 ya que presenta las mismas características de otras IMU y su costo es relativamente bajo.

3.2.3.1. MPU9250

Este sensor se ha seleccionado debido a que integra en un solo circuito un acelerómetro, un giroscopio y un magnetómetro (AK8963) de tres ejes cada uno, los cuales poseen un ADC

de 16 bits; además, el reducido tamaño de este sensor lo hace ideal para el desarrollo del sistema.

El sensor MPU-9250 incorpora un DMP [4] el cual obtiene y procesa los datos provenientes del acelerómetro, giroscopio, magnetómetro y otros sensores. Estos datos se pueden leer desde los registros DMP o pueden ser almacenados en un buffer FIFO de 512 bytes [36].

3.2.3.1.1. Características MPU9250. La Tabla 3.4 detalla las especificaciones técnicas más significativas del sensor [36].

Tabla 3.4: Características MPU-9250

Ítem	**Descripción**
Rango	Giroscopio:$\pm$ 250,$\pm$ 500,$\pm$ 1000,$\pm$2000 deg/sec Acelerómetro: $\pm$ 2g,$\pm$ 4g,$\pm$ 8g,$\pm$ 16g Magnetómetro: $\pm$ 4800 μT
Corriente de operación	3.5 mA
Alimentación	2.4V - 3.6V
Comunicación	I2C: 400kHz SPI: 1MHz-20MHz
Dirección i2c	AD0=0: 68 hex AD0=1: 69 hex Magnetómetro: 0C hex

3.2.3.1.2. Descripción de pines. Esta IMU dispone de diez pines como indica la figura 3.3. La descripción de cada pin se muestra en la Tabla 3.5 [36].

[4] Procesador digital de movimiento (digital motion processor)

Tabla 3.5: Descripción de pines MPU-9250

Número	Pin	Descripción
1	VCC	Alimentación
2	GND	Tierra
3	SCL	Serial Clock
4	SDA	Datos Seriales
5	EDA	Datos Seriales Auxiliares
6	ECL	Auxiliar Serial Clock
7	AD0	Selección dirección i2c
8	INT	Interrupción
9	NCS	Selección de chip (SPI)
10	FSYNC	Sincronización de trama de la entrada digital

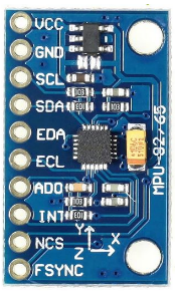


Figura 3.3: IMU MPU-9250

3.2.4. Selección de la pantalla

Se utiliza una pantalla OLED de 128x64 pixeles ya que en comparación con los displays LCD presenta menor tamaño, menor consumo de energía y mayor contraste permitiendo mejor visibilidad de información al usuario. Además, presenta la facilidad de comunicación con la tarjeta Raspberry Pi seleccionada ya que incorpora protocolos de comunicación como I2C y SPI reduciendo considerablemente el número de pines para la conexión.

3.2.4.1. Display OLED de 128x64 pixeles

Esta pantalla de tamaño reducido posee 4 pines de conexión y un controlador SSD1306 que posibilita la visualización de texto, formas e imágenes usando los protocolos I2C y SPI.

3.2.4.1.1. Características. La Tabla 3.6 detalla las especificaciones más representativas del display [37].

Tabla 3.6: Características OLED de 128x64 pixeles

Campo	**Descripción**
Alimentación	2.8V - 3.3V
Color	Azul
Temperatura de funcionamiento	(-40^{o}C) - (+80^{o}C)
Dimensión	(26.7x 19.26 x 1.65) mm
Comunicación	I2C
Controlador IC	SSD1306BZ

3.2.4.1.2. Descripción de pines. La pantalla OLED de 128x64 pixeles (ver Figura 3.4) consta de cuatro pines que se detallan en la Tabla 3.7

Tabla 3.7: Descripción de pines OLED 128x64

Número	**Pin**	**Descripción**
1	GND	Tierra
2	VCC	Alimentación
1	SCL	Reloj
1	SDA	Bus de datos

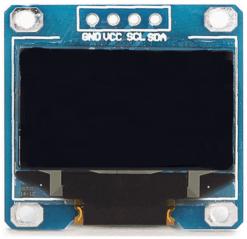


Figura 3.4: OLED de 128x64 pixeles

3.2.5. Alimentación del sistema

El sistema por estar ubicado en la estructura de la bicicleta necesita de una fuente de alimentación portable y recargable para energizar cada uno de los elementos involucrados.

Para la alimentación del sistema se incorpora la batería (Cager B030-3) representada en la Figura 3.5.

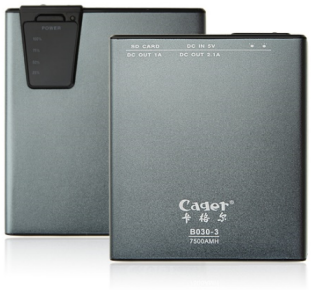


Figura 3.5: Batería recargable Cager B030-3

3.2.5.1. Características

Las características de esta batería se detallan a continuación:

- Batería de litio.
- Protección contra cortocircuito y sobrecarga.
- Capacidad de 7500mAh.
- Tipo de conexión: Doble puerto USB.
- Indicador Led.
- Entrada: 5V DC - 1A.
- Salida: 5V DC 2.1 A.
- Peso: 203 g.

3.2.6. Nivel de la batería

Raspberry Pi 3B no permite la lectura de valores analógicos a través de los pines GPIO[5] ya que no dispone de ningún convertidor analógico digital. Una solución es utilizar un elemento externo como es el caso de un chip convetidor ADC o un microcontrolador.

Las placas de desarrollo Attiny son una buena alternativa ya que son menos costosas, de tamaño reducido, ofrecen gran rendimiento y bajo consumo energético. Para la lectura del estado de la batería se hace uso del microcontrolador Attiny 85.

3.2.6.1. Attiny 85

Es un pequeño microcontrolador AVR[6] que tiene en su encapsulado un total de ocho pines.

3.2.6.1.1. Características. Las características de este microcontrolador se detallan en la Tabla 3.8 [38].

Tabla 3.8: Características Attiny84

Campo	Descripción
Alimentación	2.7V - 5.5V
Arquitectura	RISC
Registros de trabajo de propósito general	32x8
Memoria Flash	2 Kbytes - 8 Kbytes
Resolución ADC	10 bits
Memoria EEPROM	128 bytes - 512 bytes
Timers	Timer0, Timer1 de 8 bits
Velocidad de reloj interno	0 - 10 MHz

[5]Entrada / salida de uso general (general purpose input/output)
[6]Familia de microcontroladores desarrollado por Atmel

3.2.6.1.2. Descripción de pines. El integrado cuenta con ocho pines detallados en la Figura 3.6.

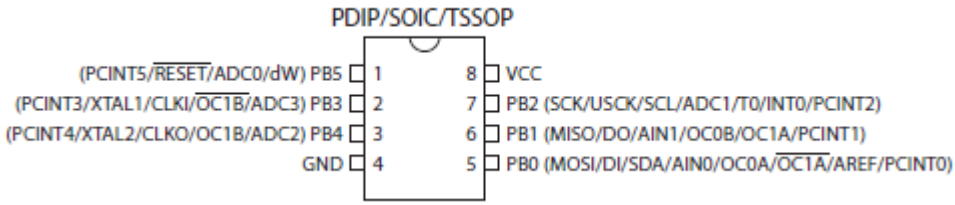


Figura 3.6: Distribución de pines Attiny85 [38]

3.2.7. Software

En este apartado se describen los programas utilizados para el desarrollo del sistema.

3.2.7.1. Selección del sistema operativo

Raspberry Pi 3B al ser un mini computador permite cargar en ella algunos sistemas operativos, sin embargo los desarrolladores de Raspberry Pi en su página oficial recomiendan el uso de una versión de Debian Linux llamado Raspbian.

Se dispone de dos versiones de Raspbian en el sitio oficial:

- **Raspbian Stretch:** Es la versión completa y cuenta con un entorno gráfico de escritorio con ventanas e íconos.
- **Raspbian Stretch Lite:** Es una versión reducida que sólo posee un entorno de consola.

Para este trabajo se opta por instalar la versión reducida ya que se requiere del mayor rendimiento de la placa que no se lograría utilizando un entorno gráfico.

3.2.7.2. Python

Es un lenguaje de programación considerado de alto nivel por ser legible con una sintaxis clara y no requiere tiempo de compilación por ser un lenguaje interpretado. Python es utilizado en diversos sistemas operativos como son: Windows, Mac OS y Linux ya que es "open source", de este modo el intérprete es distribuido libremente para cada sistema operativo [39].

3.2.7.2.1. Características. Las principales características de Python son:

- Posee un intérprete encargado de leer y ejecutar el fichero fuente, haciendo posible la ejecución del mismo código en diferentes sistemas operativos.
- La sintaxis es sencilla y ordenada permite escribir programas que sean fáciles de interpretar.
- Es un lenguaje con tipado dinámico, es decir que no es necesario declarar una variable con un tipo específico.
- Puede usarse como lenguaje orientado a objetos.
- Incluye varias estructuras de datos como son: diccionarios, listas, tuplas y strings para facilitar la programación.

3.2.7.3. SQLite

Es una biblioteca de gestión de base de datos basada en archivos que posee herramientas para manejar cualquier tipo de datos. SQLite se centra en el almacenamiento local de datos con eficiencia, confiabilidad y simplicidad [40].

SQLite es una buena alternativa para implementarse en sistemas embebidos ya que no requiere de administración ni soporte humano. Para el manejo y análisis de datos la información se puede exportar en archivos csv[7] o a través de scripts escritos en Python para un análisis

[7] Valores separados por comas (comma-separated values)

más complejo [40]. La base de datos generada en SQLite es un solo archivo compacto multiplataforma que puede almacenarse localmente y/o ser enviado como un paquete de información.

Los tipos de datos que son compatibles con SQLite son:

- Null: Valor nulo
- Entero
- Real
- Texto
- Booleano: Se almacena como entero.
- BLOB: Sin especificación del tipo de datos.

Capítulo 4

Sistema Embebido

En este capítulo se detalla la configuración, acondicionamiento y programación necesaria de cada uno de los dispositivos seleccionados.

4.1. Configuración de Raspberry Pi 3B

El minicomputador es la unidad central del sistema ya que se encarga de adquirir los datos provenientes de los diferentes sensores implementados, para ello se realiza las siguientes configuraciones.

4.1.1. Instalación del sistema operativo

Se instala Raspbian mediante la imagen ISO[1] previamente descargada de la página web, esta imagen se encuentra en formato .zip que al descomprimirse tiene un formato .img.

Para cargar la imagen ISO en la tarjeta SD[2] se utiliza una herramienta llamada Etcher disponible también en la página oficial de Raspberry, esta herramienta permite la escritura de imágenes tanto en formato .zip como .img.

[1] Archivo que almacena una copia o imagen exacta de un sistema de archivos
[2] Seguro digital (secure digital)

4.2. Configuración para el acceso remoto

Debido a que el sistema se encontrará instalado en la estructura de la bicicleta es necesario poder acceder remotamente a él. Raspberry permite el control remoto mediante el protocolo SSH (Secure Shell).

Secure Shell (SSH) Este protocolo permite el inicio de sesión remoto entre máquinas de forma segura dentro de una red insegura. Está integrado por tres componentes [41]:

- **Capa de transporte (SSH-TRANS):** Determina la autenticación, confidencialidad e integridad entre el servidor y cliente.
- **Autenticación de usuario (SSH-USERAUTH):** Se efectúa sobre la capa de transporte en la cual se autentica el usuario mediante el intercambio de claves compartidas.
- **Conexión (SSH-CONNECT):** Se ejecuta sobre el protocolo de autenticación del usuario y es el encargado de multiplexar el túnel cifrado en varios canales lógicos (cifrado de datos a transmitirse).

4.2.1. Habilitación de interfaces

Para habilitar o inhabilitar interfaces en el minicomputador se siguen los siguientes pasos:

- Ingresar al menú de configuración.
- Seleccionar la quinta opción “Interfacing Options”.
- La Figura 4.1 indica las opciones disponibles.
- Escoger la interfaz y habilitarla o inhabilitarla.

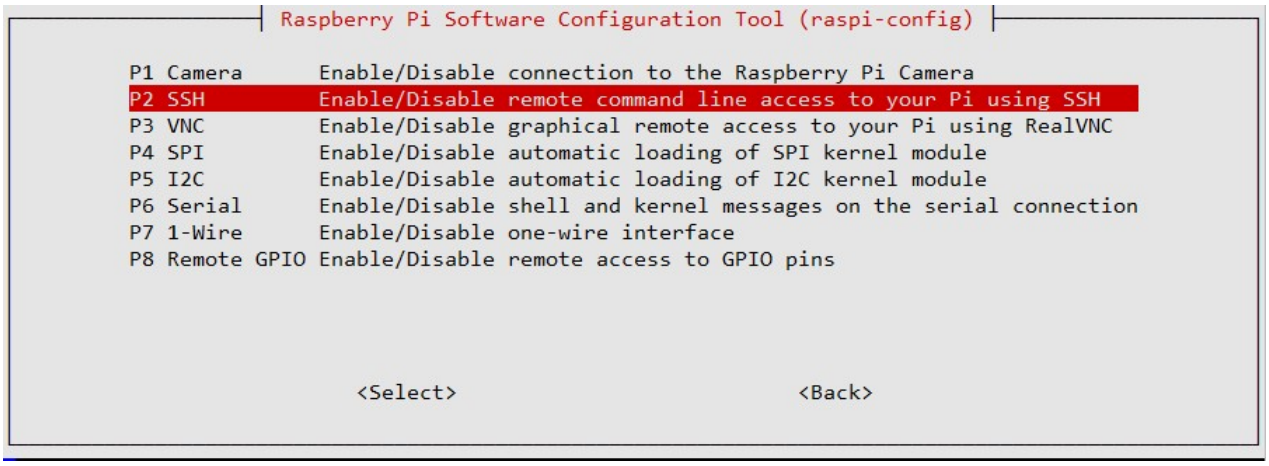


Figura 4.1: Interfacing Options raspi-config

4.2.2. Acceso remoto

Para acceder remotamente a Raspberry Pi desde un ordenador se necesita conocer la dirección IP[3] que adquiere la tarjeta cuando se conecta a internet, el usuario y contraseña. Para conocer la dirección IP que tiene el minicomputador se escribe en la terminal el comando "ifconfig", la respuesta obtenida se muestra en la Figura 4.2.

La dirección IP adquirida varía cada vez que el minicomputador se conecta a una red ya que es asignada por el DHCP[4], motivo por el cual es indispensable asignar una IP estática para el correcto funcionamiento del sistema.

[3]Protocolo de internet (internet protocol)
[4]Protocolo de configuración dinámica de host (dynamic host configuration protocol)

```
pi@raspberrypi:~ $ ifconfig
enxb827eb49dc49: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether b8:27:eb:49:dc:49  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.50  netmask 255.255.255.0  broadcast 192.168.1.255
        ether b8:27:eb:1c:89:1c  txqueuelen 1000  (Ethernet)
        RX packets 14814  bytes 1254162 (1.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1106  bytes 203807 (199.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figura 4.2: Dirección IP adquirida

4.2.3. Configuración de IP estática

Para configurar una dirección IP estática en Raspberry Pi se puede realizar mediante la configuración del archivo "/etc/dhcpcd.conf" pero realizar esta configuración presenta algunas desventajas como son: el tipo de cifrado y la reconexión automática. Como solución a esto se instala una herramienta llamada *wicd-curses*. Esta herramienta permite visualizar una interfaz simple para la administración de redes.

Una vez instalada se ejecuta utilizando el comando "sudo wicd-curses", La Figura 4.3 muestra la pantalla de la herramienta en la que se visualizan todas las redes disponibles.

```
                                                                    Interfaz Wicd Curses
Redes inalámbricas
  STR                    ESSID   ENCRYPT             BSSID   MODE CHNL
  100%               MECHATRONIC      WPA2 E8:39:DF:18:23:34 Master   11
  60%              marxesmipastor      WPA2 46:DB:30:D0:17:F4 Master   11
  47%                        JABO      WPA2 90:94:E4:B1:B1:18 Master   10
  31%                  Plus_kony.      WPA2 98:DE:D0:5B:2D:86 Master    7
  31%    NETLIFE-FAMILIA CARRERA      WPA2 B4:75:0E:DF:25:B3 Master   11
  24%                        Mela      WPA2 60:E3:27:32:1B:36 Master    6
```

Figura 4.3: Administrador de redes wicd-curses

A la red seleccionada se le realiza la siguiente configuración.

- Marcar la casilla “Usar direcciones IP estáticas”.
- Asignar dirección IP: 192.168.1.50.
- Máscara de red: 255.255.255.0.
- Puerta de enlace: 192.168.1.1.
- Marcar la casilla “Conectarse automáticamente a esta red”.
- Seleccionar casilla “Usar cifrado”: WPA 1/2
- Ingresar contraseña de la red.

La Figura 4.4 muestra la configuración realizada para la red “MECHATRONIC”, finalmente se guarda la configuración.

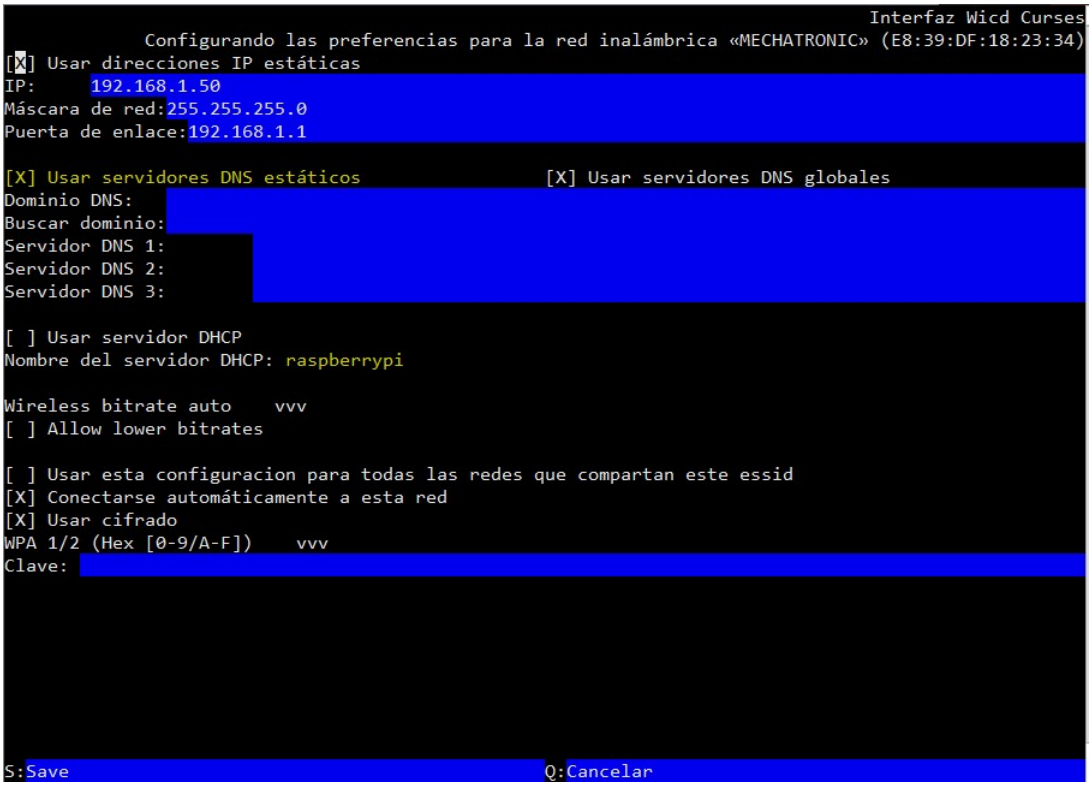


Figura 4.4: Configuración de IP estática

4.3. Configuración del receptor GY-GPS6MV2

El minicomputador Raspberry Pi se equipa con un módulo Bluetooth de forma predeterminada por medio del UART PL011, la comunicación del dispositivo GPS y Raspberry Pi es a través de una mini UART que se conecta a la interfaz serial.

Para comunicarse a la mini UART se desactiva la interfaz serial, ingresando al menú de configuración de Raspbian y se siguen los pasos detallados en el apartado 4.2.1.

4.3.1. Conexión

La conexión del GPS y el minicomputador utiliza cuatro pines de conexiones, dos para alimentación y dos para la transmisión y recepción de información, la conexión se ilustra en la Figura 4.5.

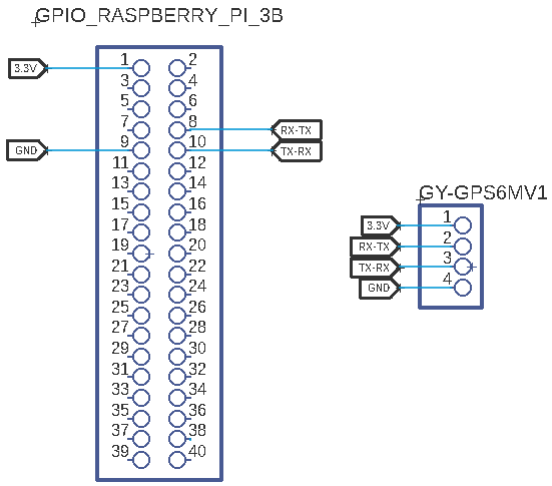


Figura 4.5: Conexión GY-GPS6MV2 y Raspberry Pi 3B

4.3.2. Lectura de datos en consola

Para realizar un test de funcionamiento del dispositivo se utiliza la terminal de Raspbian para visualizar las sentencias NMEA que proporciona el receptor GPS. Se ingresar el comando:

"sudo cat /dev/ttyS0". La Figura 4.6 muestra un ejemplo de la lectura de las sentencias NMEA que proporciona el sensor.

```
pi@raspberrypi:~ $ cat /dev/ttyS0
$GPRMC,234948.00,A,0021.79856,N,07806.80834,W,0.403,,240219,,,D*64

$GPVTG,,T,,M,0.403,N,0.746,K,D*24

$GPGGA,234948.00,0021.79856,N,07806.80834,W,2,09,0.88,2228.5,M,13.4,M,,0000*75

$GPGSA,A,3,03,11,17,18,01,30,07,09,19,,,,1.82,0.88,1.59*0C

$GPGSV,5,1,17,01,44,037,25,03,27,118,14,04,35,214,,06,18,213,14*7D

$GPGSV,5,2,17,07,87,043,21,09,32,194,19,11,29,024,17,17,33,293,32*7C

$GPGSV,5,3,17,18,16,035,17,19,21,264,37,22,21,094,20,23,17,161,13*77

$GPGSV,5,4,17,28,15,339,16,30,52,330,29,46,32,270,33,48,27,270,*7B

$GPGSV,5,5,17,51,56,269,30*46

$GPGLL,0021.79856,N,07806.80834,W,234948.00,A,D*76
```

Figura 4.6: Test GY-GPS6MV2

4.3.3. Programación

Del receptor GPS se obtienen datos de geolocalización *(latitud y longitud)*, asimismo se obtiene la velocidad sobre la superficie terrestre.

La lectura de datos en Python se detalla en el algoritmo representado en la Figura 4.7.

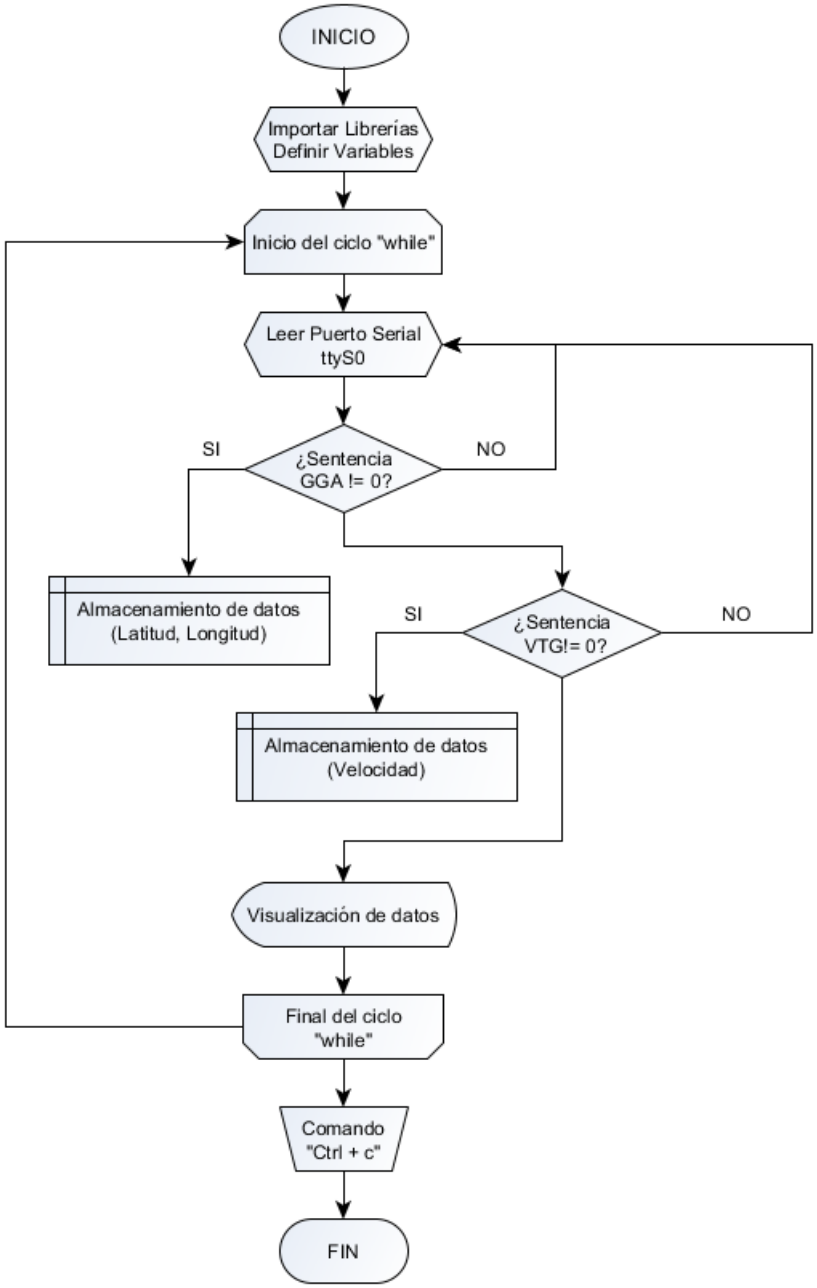


Figura 4.7: Pseudocódigo para el receptor GY-GPS6MV2

4.4. Configuración MPU9250

El sensor usa el protocolo I2C para la comunicación con el minicomputador, se habilita esta interfaz desde el menú de configuración de Raspbian como se indica en el apartado 4.2.1.

4.4.1. Conexión

La conexión entre los dos dispositivos se muestra en la Figura 4.8.

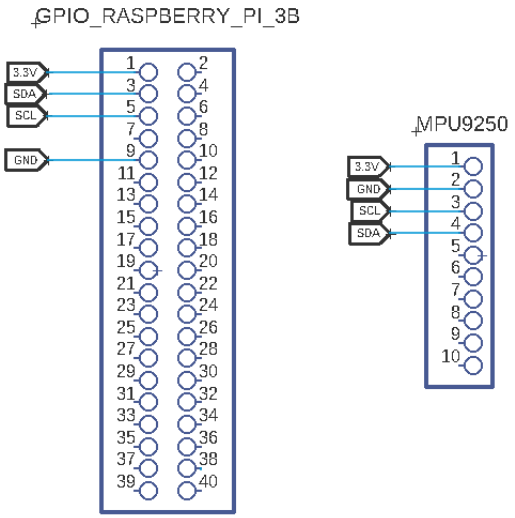


Figura 4.8: Conexión MPU9250 y Raspberry pi 3B

4.4.2. Calibración del magnetómetro

La representación de la rotación del sensor alrededor de 360 grados y paralelo a la superficie terrestre debe ser una circunferencia perfecta con su centro en el origen, sin embargo, al rotar el sensor se aprecia que el centro de la circunferencia no está en el origen (ver Figura 4.9).

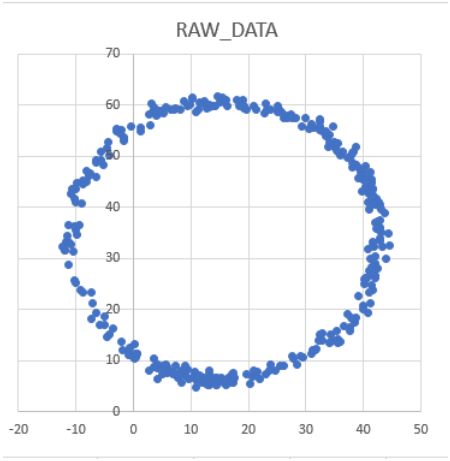


Figura 4.9: Gráfica del magnetómetro sin calibración

Para trasladar el centro de la circunferencia al origen se realiza la denominada calibración *(hard-iron)*, para lo cual se determinan dos factores de corrección llamados (*alpha* y *beta*). El parámetro *alpha* se calcula a partir de los valores máximos y mínimos del campo magnético en la componente del eje X (Ecuación 4.1) y beta a partir de los valores del campo magnético en el eje Y (Ecuación 4.2).

$$alpha = \frac{(X_{max} + X_{min})}{2} \tag{4.1}$$

$$beta = \frac{(Y_{max} + Y_{min})}{2} \tag{4.2}$$

El parámetro alpha se resta a cada medición del campo magnético en la componente X, de igual manera sucede con el parámetro beta en la componente Y. El resultado de esta corrección es una circunferencia con centro en el origen (ver Figura 4.10).

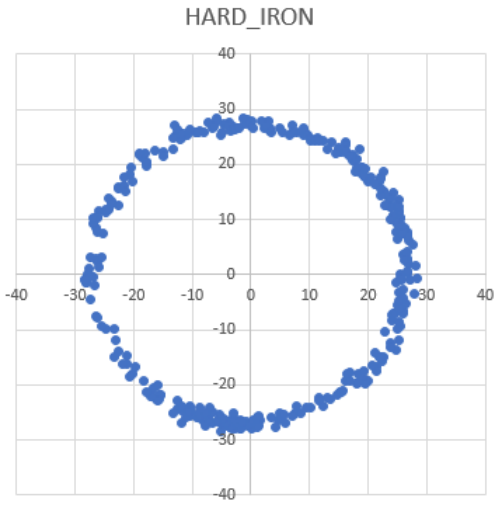


Figura 4.10: Calibración Hard-Iron

La calibración *soft-iron* se utiliza cuando la gráfica resultante es una elipse y no una circunferencia, para esta calibración se determina otra factor denominado *sigma*, sin embargo, dado que la gráfica si presenta una forma circular perfecta la calibración soft-iron no se aplica.

4.4.3. Programación

Con este sensor se obtiene la aceleración, orientación y rumbo del ciclista a través del acelerómetro y el magnetómetro como se detalla en el apartado 2.5. Para la lectura de datos se emplea la librería ”smbus”de python necesaria para la comunicación I2C.

El esquema de la figura 4.11 ilustra la programación realizada en Python.

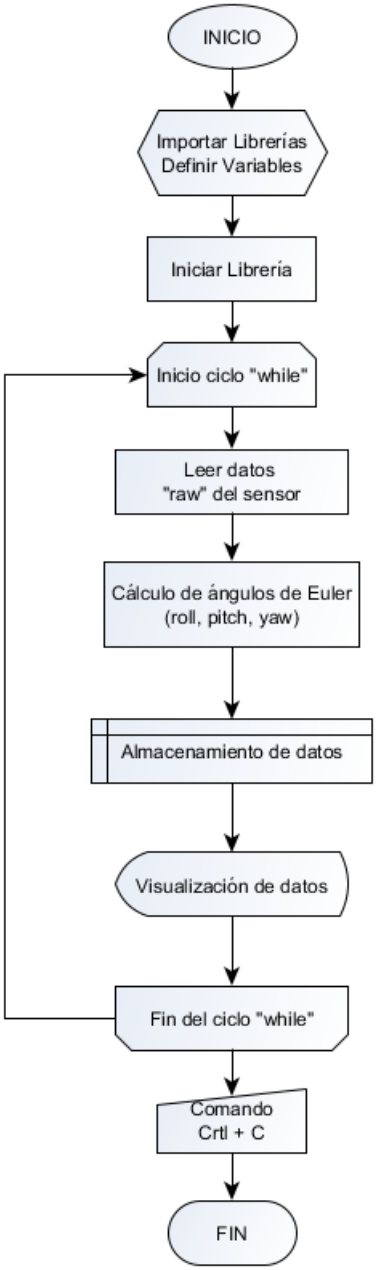


Figura 4.11: Pseudocódigo para la IMU MPU9250

4.5. Configuración pantalla OLED

La comunicación entre la pantalla OLED y Raspberry Pi 3B se realiza mediante el protocolo I2C y el controlador SSD1306.

4.5.1. Conexión

El siguiente esquema indica los pines de conexión entre los dispositivos. Ver Figura 4.12.

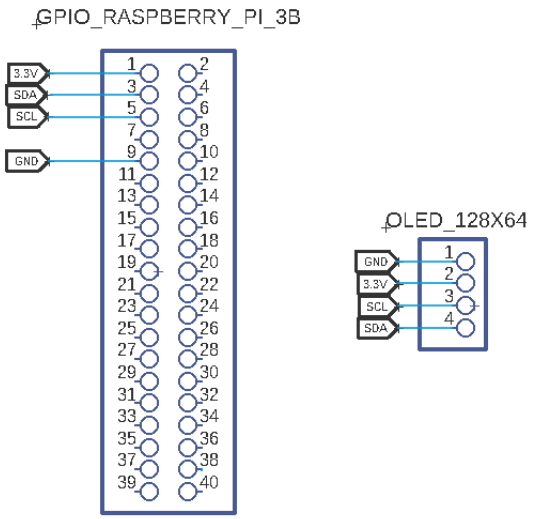


Figura 4.12: Conexión OLED 128x64 y Raspberry Pi 3B

4.5.2. Programación

Se utiliza la librería de Adrafuit Python por ser compatible con todas las pantallas basadas en el controlador SSD1306. Esta librería permite representar texto, formas e imágenes de manera sencilla. La programación se ilustra en el diagrama de la Figura 4.13.

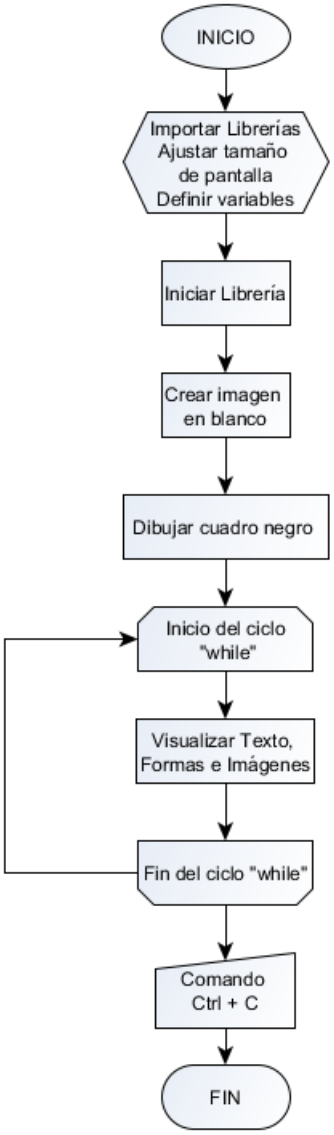


Figura 4.13: Pseudocódigo para el display OLED 128x64

4.6. Estado de la batería

Este proyecto al ser un dispositivo portátil, es vital determinar el estado de la batería de alimentación para permitir que el sistema se apague de manera segura y así proteger la información y al minicomputador.

4.6.1. Configuración Attiny 85

El microcontrolador es el encargado de realizar el cálculo del estado de la batería. La programación se la realiza en el IDE[5] de Arduino y para cargar el programa al microcontrolador se utiliza la tarjeta Arduino UNO como programador ISP como se detalla en [42].

4.6.2. Conexión

Se utiliza un divisor de tensión resistivo (ver Figura 4.14) para asegurar que el voltaje máximo que ingrese a la tarjeta Raspberry no supere los 3.3 V. Los valores de las resistencias R_1 y R_2 están dados por la Ecuación 4.3.

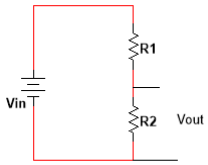


Figura 4.14: Divisor de voltaje resistivo

$$V_{out} = \frac{V_{in} * R_2}{R_1 + R_2} \tag{4.3}$$

Dónde:

$V_{out} = 3{,}3V$

$V_{in} = 5V$

Asumiento $R_1 = 15k\Omega$

El valor de la resistencia (R_2) está dado por la Ecuación 4.4.

$$R_2 = \frac{V_{out} * R_1}{V_{in} - V_{out}} = 29117{,}65\Omega \approx 30k\Omega \tag{4.4}$$

Los pines de conexión entre los dispositivos se representan en la Figura 4.15.

[5]Entorno de desarrollo integrado (integrated development environment)

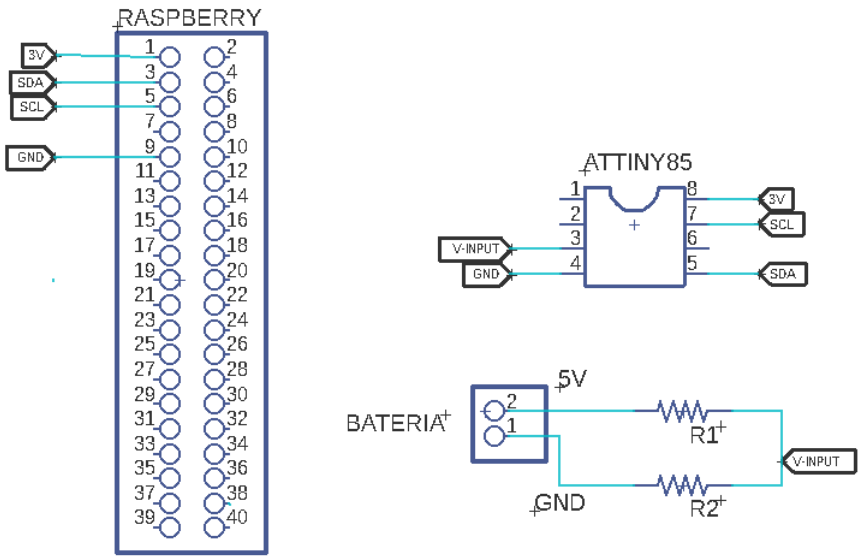


Figura 4.15: Conexión Attiny85 y Raspberry pi 3B

4.6.3. Programación

La programación del microcontrolador se realiza bajo la configuración maestro-esclavo a través del protocolo i2c, el microcontrolador es el esclavo encargado de calcular y enviar el estado de la batería al maestro (Raspberry Pi 3B) cuando este haga un llamado. El pseudocódigo utilizado se detalla en la Figura 4.16.

El microcontrolador utiliza la librería Wire y la librería SMBus en Raspberry Pi 3B.

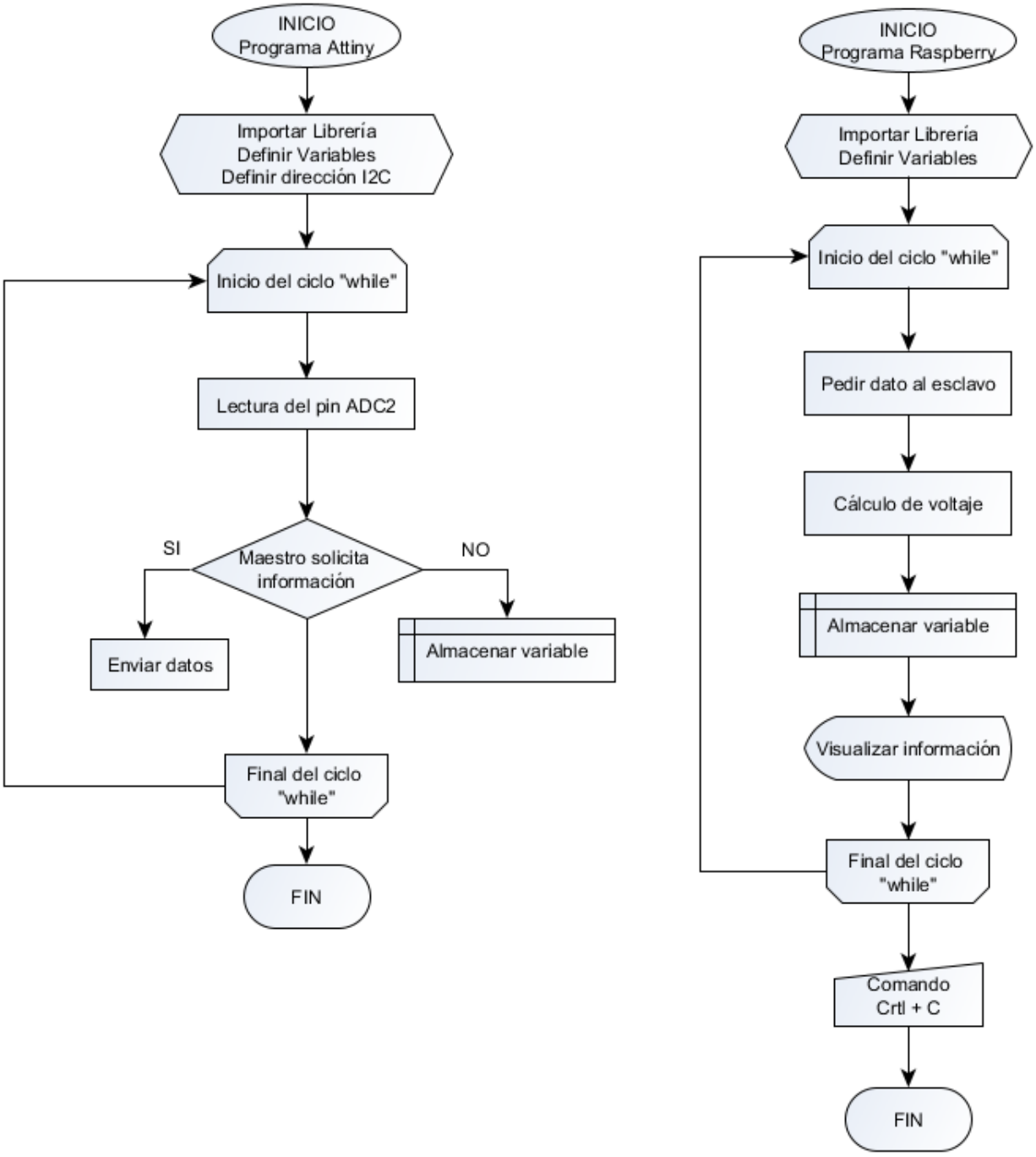


Figura 4.16: Pseudocódigo Attiny85-Raspberry Pi 3B

4.7. Base de datos

La base de datos desarrollada en SQLite 3 se utiliza para almacenar toda la información recopilada por los sensores y el receptor GPS. Para instalar SQLite en Raspbian se ingresa el siguiente comando en la terminal ”sudo apt-get install sqlite3”.

4.7.1. Programación

La programación de la base de datos se representa en diagrama de la Figura 4.17.

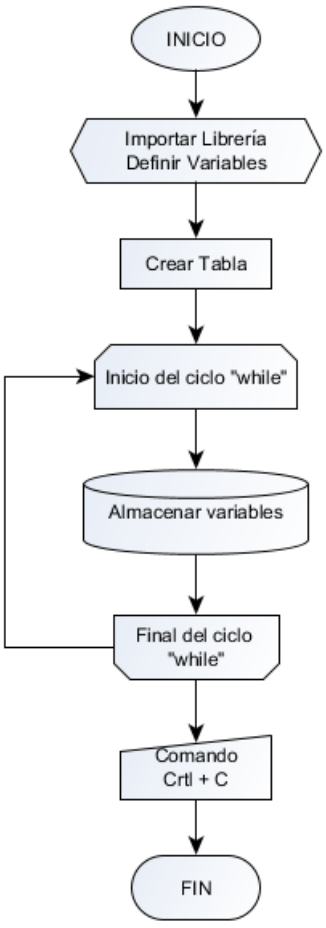


Figura 4.17: Pseudocódigo para la Base de Datos en SQLite

Capítulo 5

Implementación

5.1. Diagrama de bloques del sistema completo

El esquema de la Figura 5.1 detalla la forma como los dispositivos se encuentran conectados (alimentación y comunicación). Raspberry Pi es la parte central del sistema pues se encarga de recolectar, procesar y almacenar en una base de datos la información proporcionada por sensores y otros dispositivos.

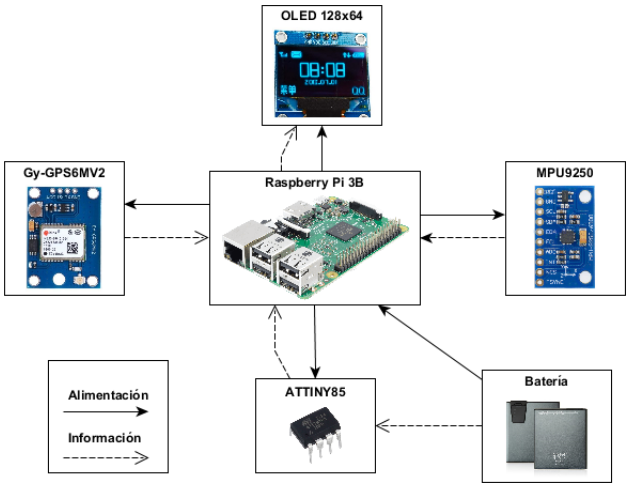


Figura 5.1: Diagrama de bloques de hardware

5.2. Diagrama de bloques de software

El dispositivo se compone por cinco subsistemas (ver Figura 5.2) visualizados en un menú en una pantalla OLED, para la navegación a través del menú el sistema consta de cuatro botones, dos para el desplazamiento y dos botones de selección. Los subsistemas son:

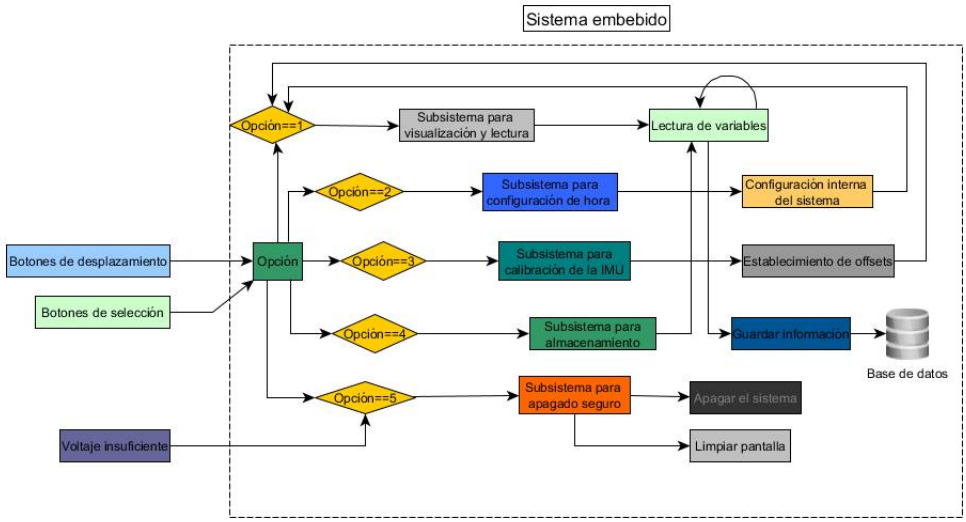


Figura 5.2: Diagrama de bloques de software

- Subsistema para visualización y lectura.
- Subsistema para configuración de hora.
- Subsistema para calibración de la IMU.
- Subsistema para almacenamiento.
- Subsistama para apagado seguro.

5.2.1. Descripción de los subsistemas

- **Subsistema para visualización y lectura:** Representa el menú y el estado de funcionamiento en el que se encuentra el dispositivo. En este modo el sistema calcula todos los

valores de las variables relacionadas a la conducción del ciclista.

- **Subsistema para la configuración de hora interna:** Adquiere la hora UTC del receptor GPS leyendo la sentencia NMEA GGA, posteriormente se establece la configuracion para la hora de Ecuador (UTC-5) y se ajusta como nueva hora interna del sistema.
- **Subsistema para la calibración de la IMU:** Realiza el promedio entre 50 lecturas y se determinan los valores de offset necesarios para establecer correctamente el sistema de coordenadas.
- **Subsistema para almacenamiento:** Las tramas de información se almacenan en una nueva fila, dentro de la base de datos creada.
- **Subsistema para el apagado seguro del sistema:** Este subsistema se complementa con la lectura del estado de la batería y se encarga de apagar el sistema de forma segura cuando esta opción es seleccionada o cuando el voltaje de la batería es insuficiente para que el sistema funcione de manera correcta, además proporciona protección de la información.

5.3. Placa de conexión

Se diseña una placa para la conexión entre los dispositivos con la finalidad de reducir el tamaño del dispositivo y disminuir el riesgo de errores por malas conexiones del cableado. Así,el sistema es más funcional y compacto, el diseño final se muestra en la Figura 5.3. El esquema y construcción se detallan en el Anexo A.

5.4. Piezas para el acople del sistema

Se diseñan y construyen piezas para la sujeción y protección del sistema utilizando impresión 3D. Se selecciona el PLA (Ácido poliláctico) como el material a utilizarse debido a que la temperatura de fusión es menor al ABS y no necesita de una base caliente lo cual reduce el

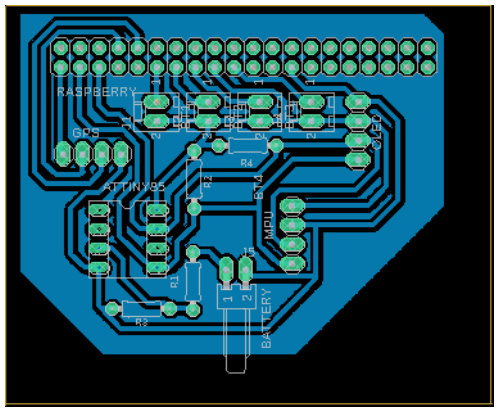


Figura 5.3: Diseño de placa de conexión

tiempo de impresión; además, por ser un plástico biodegradable y reciclable. Dos piezas son utilizadas para contener todo el circuito eléctrico y dos piezas para la sujeción a la estructura de la bicicleta.

5.4.1. Sujeción de la parte eléctrica

Se diseñan dos piezas (Figura 5.4) en las cuales se alojan las dos tarjetas electrónicas, las piezas se diseñan teniendo en consideración las medidas del minicomputador y la placa de conexión construida.

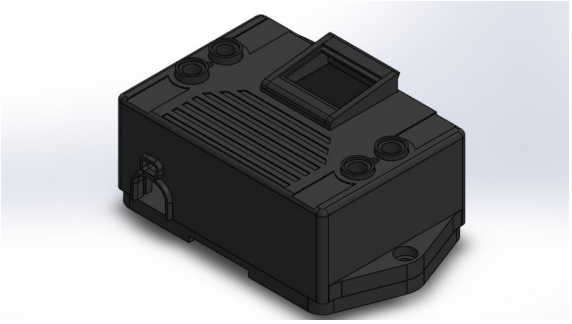


Figura 5.4: Piezas de protección

La pieza 1 (Figura 5.5) alberga al minicomputador y a la tarjeta de conexión, la pieza pre-

senta soportes para dar fijación y orificios que permiten el ingreso del cable para energizar al minicomputador. La pieza 2 (Figura 5.6) es la protección superior del sistema, los agujeros son para incorporar pulsadores que sirven para la navegación a través del menú, una pantalla OLED de 128x64 pixeles para visualizar el estado de funcionamiento del sistema; y la entrada de un cable para la lectura del estado de la batería.

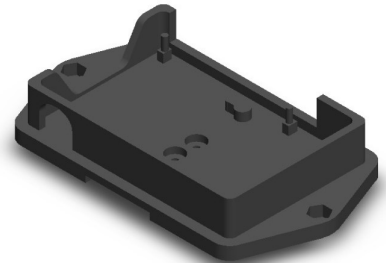


Figura 5.5: Pieza inferior

Figura 5.6: Pieza superior

5.4.2. Sujeción a la bicicleta

Se diseñan dos piezas en forma de abrazadera para fijarse al tubo del manubrio. La pieza 3 (Figura 5.7) se ubica en la parte superior del tubo y se encarga de mantener firme la pieza 1 a la estructura de la bicicleta. La pieza 4 (figura 5.8) es el complemento de la pieza 3 que al unirse por medio de tornillos mantienen firme el sistema al tubo del manubrio.

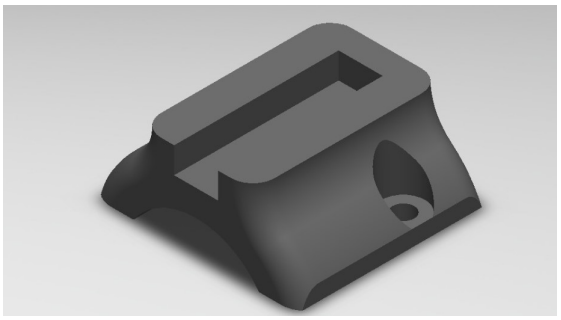


Figura 5.7: Acople superior

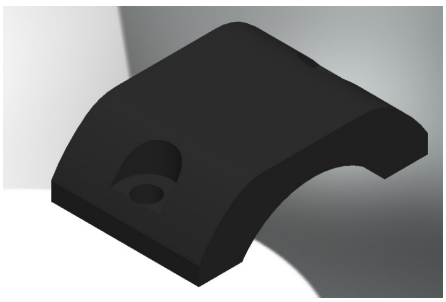


Figura 5.8: Acople inferior

5.5. Montaje y funcionamiento

5.5.1. Montaje

Para el montaje del sistema a la estructura de la bicicleta, primero se fijan las piezas de soporte en forma utilizando tornillos (Figura 5.9-1), a continuación se conectan las dos tarjetas electrónicas (Figura 5.9-2), se aseguran las placas a las piezas fijadas en el tubo del manubrio (Figura 5.9-3) y finalmente se coloca la pieza de protección (Figura 5.9-4).

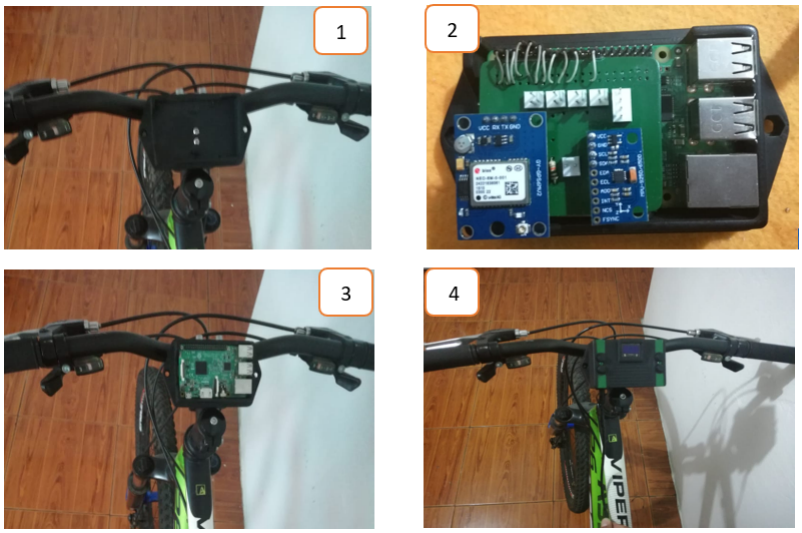


Figura 5.9: Montaje del sistema

5.5.2. Indicaciones del funcionamiento

El sistema muestra un menú con cuatro opciones (ver Figura 5.10), para la navegación a través del menú se utilizan cuatro botones (Figura 5.11); los dos botones de la parte izquierda son usados para desplazarse entre cada opción y los botones ubicados a la derecha son usados para seleccionar una opción.

Figura 5.10: Visualización del menú

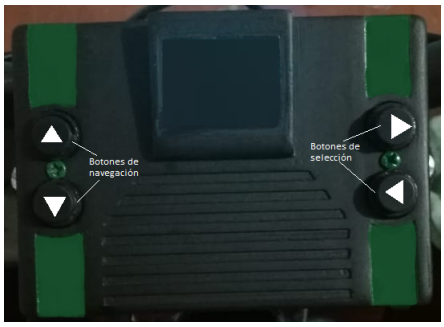


Figura 5.11: Descripción de los botones del dispositivo

5.6. Pruebas

En este apartado se detallan las pruebas realizadas para comprobar el correcto funcionamiento del sistema.

5.6.1. Evaluación de la orientación y rumbo

Para esta prueba se toman 200 datos de los ángulos roll, pitch y yaw, se determina el promedio y se calcula el error. Para los ángulos de roll y pitch se ubica el sensor a 30,-30, 45 y -45 grados utilizando unas escuadras. Para el ángulo yaw se gira el sensor en intervalos de 90 grados y para comparar los resultados se hace uso de una brújula. Los resultados se muestran en las Tablas 5.1, 5.2.

Tabla 5.1: Error en los ángulos roll y pitch

Ángulo roll (grados)	**Error (grados)**	**Àngulo pitch (grados)**	**Error (grados)**
30	0.2219	30	0.1963
-30	0.1973	-30	0.1956
45	0.2201	45	0.2203
-45	0.2092	-45	0.2066

Tabla 5.2: Error del ángulo yaw

Ángulo (grados)	Error (grados)
90	0.4807
180	0.320
270	0.4761

5.6.2. Evaluación del offset de la unidad inercial

Dado que el sistema de coordenadas del sensor no coincide con el sistema de referencia establecido para el estudio (ver Figura 5.12), se deben determinar los valores de corrección para que los sistemas de coordenadas concuerden.

Figura 5.12: Sistemas de coordenadas

Se toma una muestra de 200 datos (Figuras 5.13 y 5.14) cuando el dispostivo se coloca en posición vertical al suelo, se calculan los promedios de los ángulos pitch y roll detallados en la Tabla 5.3 que serán los valores de corrección para alinear los sistemas de coordenadas.

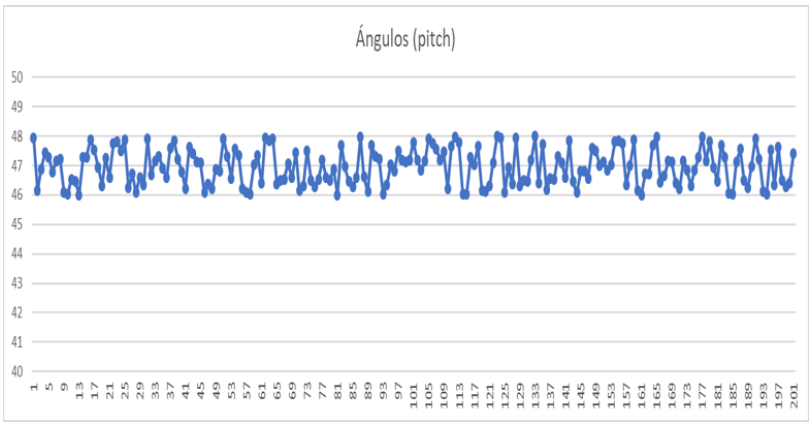


Figura 5.13: Gráfica de las medidas del ángulo pitch

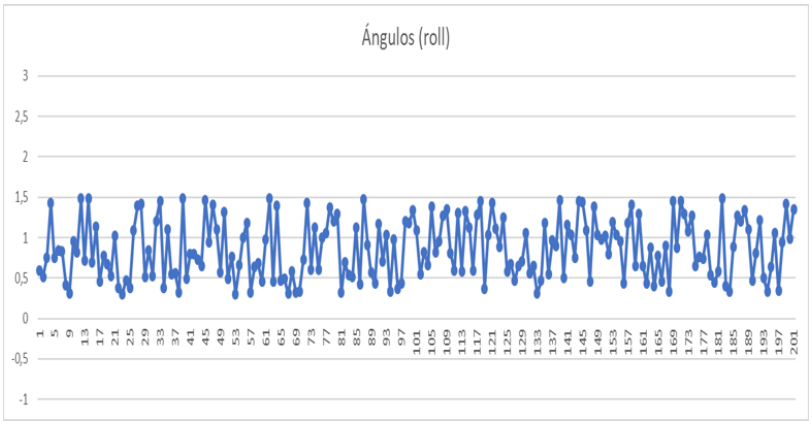


Figura 5.14: Gráfica de las medidas del ángulo roll

Tabla 5.3: Offsets de los ángulos pitch y roll

Ángulo	**Offset**	**Desviación estándar (grados)**
pitch	46.32	0.60
roll	0.84	0.36

5.6.3. Evaluación de datos de posicionamiento global

Para observar los datos de posicionamiento global almacenados, se utiliza un mapa creado en google maps en el cual se importan los datos de latitud y longitud.

La representación de estos datos se observa en la Figura 5.15.

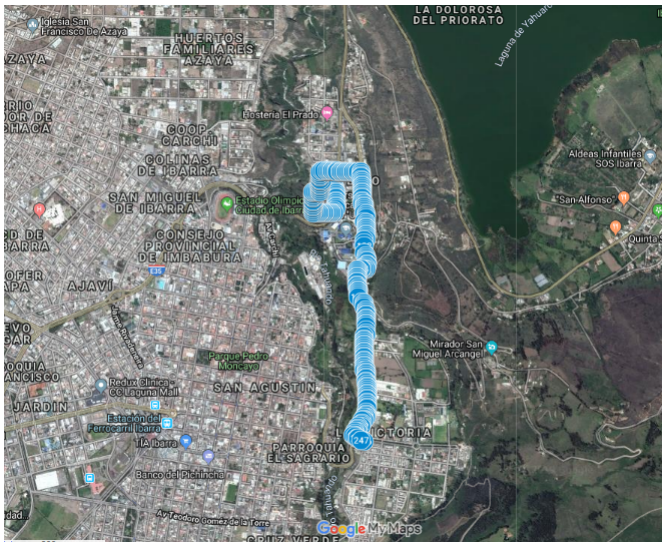


Figura 5.15: Datos de posicionamiento global

5.6.4. Tiempo de muestreo

Para esta prueba se analiza el tiempo de muestreo para 200 capturas de información (ver Figura 5.16). La gráfica de color azul representar los valores del tiempo, mientras que la gráfica de color gris muestra el valor promedio igual a 0.3797 milisegundos. Se observa que el tiempo de muestreo no es constante, asimismo se calcula una desviación estándar igual a 0.1713 milisegundos.

5.6.5. Evaluación de la base de datos

La base de datos resultante se observa en la Figura 5.17, para visualizar los campos y los datos almacenados se utiliza la herramienta *Sqlite Browser*.

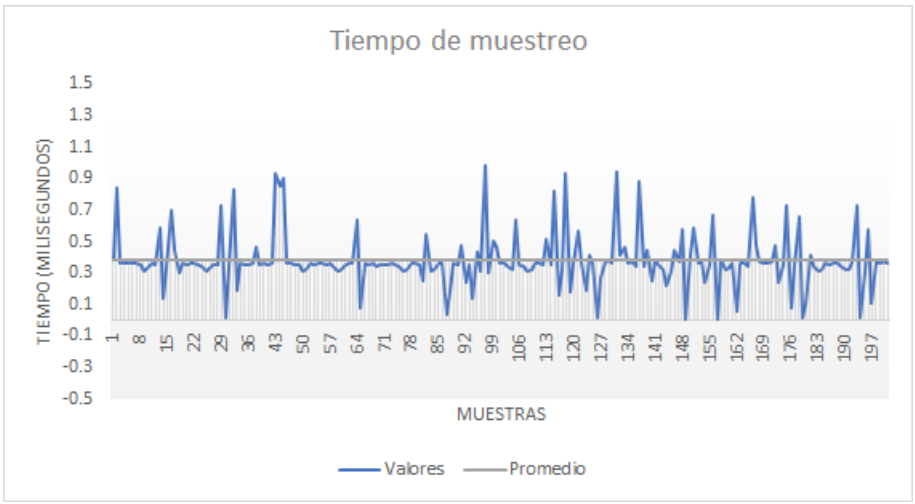


Figura 5.16: Tiempo de muestreo

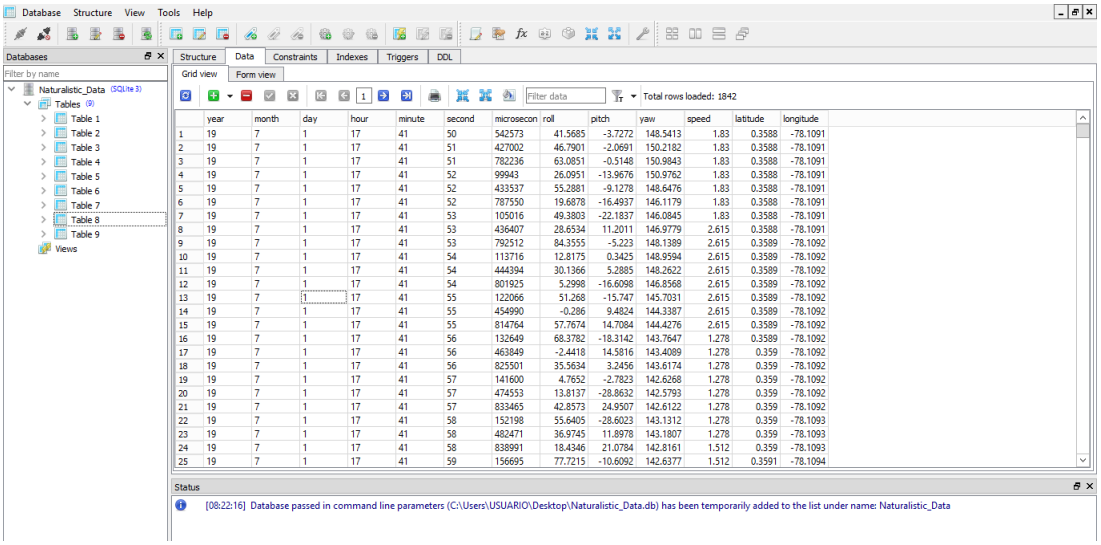


Figura 5.17: Base de datos

Capítulo 6

Conclusiones y recomendaciones

Este capítulo muestra las conclusiones del presente proyecto y manifiesta algunas posibles sugerencias para su aplicación en trabajos futuros.

6.1. Conclusiones

- Las pruebas realizadas al sistema establecen que el diseño cumple con la finalidad como instrumento de medida, siendo capaz de recopilar, procesar y almacenar variables naturalísticas en bicicletas.
- Con base en el funcionamiento del dispositivo se determina que satisface todos los requerimientos propuestos para el desarrollo del proyecto.
- El tratamiento de las señales provenientes de los sensores aseguran la obtención de variables físicas involucradas al estudio de la conducción de ciclistas con un error considerado aceptable. El almacenamiento de la información en una base de datos desarrollada en SQLite asegura que la información se analice y comparta de manera muy sencilla.
- No es posible establecer un tiempo constante de muestreo debido a la latencia del minicomputador, sin embargo, el tiempo de muestreo almacenado sirve para analizar de manera correcta cada variable obtenida y para posteriores aplicaciones.

6.2. Recomendaciones

- Asegurar la correcta conexión de cada componente del sistema embebido.
- Realizar la configuración de la hora interna y calibración de la unidad de medición inercial antes de almacenar la información, para que los valores obtenidos sean de utilidad para el estudio.
- El dispositivo no cuenta con gran protección contra el clima, por tal razón se debe utilizar en ambientes ”normales”.

6.3. Trabajo futuro

Con la información obtenida se realizarán estudios naturalísticos enfocados al análisis sobre la conducción de ciclistas.

Resolver problemas de clasificación en el reconocimiento del comportamiento del ciclista aplicando algoritmos de aprendizaje de máquina.

El sistema puede ser replicado para la exploración de rutas para clubes de ciclistas.

Bibliografía

[1] M. Proaño, "Cultura ciclera en Quito, políticas de movilidad: estudio de caso ciclópolis y al sur en Bici", trabajo de fin de máster, Universidad Andina Simón Bolívar. Quito,2012.

[2] J.García y L.Naranjo, "Anuario de estadísticas de transporte", Instituto Nacional de Estadísticas y Censos,Quito,Ecuador,2016.

[3] N. Pinto, F.Fuentes y D. Alcivar, "La situación de la bicicleta en Ecuador: avances, retos y perspectivas", *Friedrich Ebert Stiftung*, Quito,Ecuador,ISBN: 978-9978-94-147-8, mar.2015.

[4] A. García, "Los accidentes con bicicletas aumentaron 16 % en primer trimestre del 2017", J. El Comercio. May., 2017 [En línea]. Disponible en: https://www.elcomercio.com/actualidad/accidentes–bicicletas–guayas–aumentaron–ciclopaseo.html. [Accedido: 15-may-2018]

[5] El Comercio, "Imágenes de las faltas más comunes en las ciclovías de Quito", J. El Comercio, 2014. [En línea]. Disponible en: https://www.elcomercio.com/actualidad/quito/imagenes–de–faltas–mas–comunes.html. [Accedido: 15-may-2018]

[6] F. Molina, G. Alvarez y J. Torres, "Determinacion del ciclo típico de conducción de una bicicleta en las ciclovías de la ciudad de Cuenca", trabajo de fin de grado, Universidad del Azuay, Cuenca, 2016.

[7] M. Dozza, G. Bianchi y J. Werneke, “Using naturalistic data to assess e-cyclist behavior,” ScienceDirect, vol. 41, pp. 217-226, 2016.

[8] M. Johnson, J. Charlton, J. Oxley y S. Newstead, “Naturalistic Cycling Study: Identifying Risk Factors for On-Road Commuter Cyclists,” PMC, vol. 54, pp. 275-283, 2010.

[9] V. Neale, T. Dingus, S. Klauer, J. Sudweeks, R. Knipling, G. Holbrook y A. Petersen, “The 100 Car Naturalistic Driving Study,” Virginia Tech Transportation Institute, Tech. Rep. DOT HS 809 536 , 2002.

[10] M. Dozza y J. Werneke, “Introducing naturalistic cycling data: What factors influence bicyclists’ safety in the real world?,” ScienceDirect, vol. 24, pp. 83-91, 2014.

[11] K. Schleinitz, T. Petzoldt, L. Franke-Bartholdt, J. Krems y T. Gehlert, “Conflict partners and infrastructure use in safety critical events in cycling – Results from a naturalistic cycling study,” ScienceDirect, vol. 31, pp. 99-111, 2015

[12] K. Schleinitz, T. Petzoldt, L. Franke-Bartholdt, J. Krems y T. Gehlert, “Conflict partners and infrastructure use in safety critical events in cycling – Results from a naturalistic cycling study,” ScienceDirect, vol. 31, pp. 99-111, 2015.

[13] A. Jahangiri, M. Elhenawy, H. Rakha y T. Dingus, “Investigating Cyclist Violations at Signal-Controlled Intersections using Naturalistic Cycling Data,” de IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 2016

[14] A. Jahangiri y T. D. H Rakha, “Developing a system architecture for cyclist violation prediction,” de 6th International Conference on Applied Human Factors and Ergonomics (AHFE 2015) and the Affiliated Conferences, USA, 2015.

[15] M. Johnson, J. Charlton y J. Oxley, “Cyclists and red lights – a study of behaviour of commuter cyclists in Melbourne,” de 2008 Australasian Road Safety Research, Policing and Education Conference, South Australia, 2008

[16] C. Wu, L. Yao y K. Zhang, “The red-light running behavior of electric bike riders and cyclists at urban intersections in China: An observational study,” ScienceDirect, vol. 49, pp. 186-192, 2012.

[17] M. Johnson, D. Chong, J. Carroll, R. Katz, J. Oxley y J. Charlton, “Naturalistic Cycling Study: Identifying Risk Factors for Cyclists in the Australian Capital Territory,” Monash University Accident Research Centre, Tech. Rep. ISBN 0-7326-2392-8, Australia, 2014.

[18] M. Dozza, J. Werneke y A. Fernandez, “Piloting the Naturalistic Methodology on Bicycles,” de Proceedings, International Cycling Safety Conference 2012, Helmond, The Netherlands, 2012.

[19] M. Dozza y A. Fernandez, “Understanding Bicycle Dynamics and Cyclist Behavior From Naturalistic Field Data,” IEEE Transactions on Intelligent Transportation Systems, vol. 15, nº 1, pp. 376-384, 2012.

[20] A. Pozo, “Sistema sensorial para control y localización de vehículos en exteriores,” Tesis doctoral,Universidad de Málaga, Málaga, 2001.

[21] J. Raldúa, “Prototipo de sistema de localización por GPS para salvamento marítimo,” Trabajo de fin de grado, Universidad Carlos III de MAdrid, Madrid, 2007.

[22] O. Woodman, “An introduction to inertial navigation,” University of Cambridge Comuputer Laboratory,Tech. Rep. ISSN 1476-2986, United Kingdom, 2007.

[23] D. Pozo, “Diseño y Construcción de una Plataforma Didáctica para medir Ángulos de Inclinación usando Sensores Inerciales como Acelerómetro y Giroscopio,” Trabajo de fin de grado,Escuela Politécnica Nacional, Quito, 2014.

[24] G. Ferrer, “Integración Kalman de sensores inerciales INS con,” Trabajo de fin de grado, Universitat Politècnica de Catalunya, Barcelona, 2009.

[25] T. Seel, J. Raisch y T. Schauder, “IMU-Based Joint Angle Measurement for Gait Analysis,” Sensors,Tech. Rep. ISSN 1424-8220 Berlin, 2014.

[26] R. Delgado, “Identificación de personas por su forma de andar usando sensores inerciales,” Trabajo de fin de grado, Universidad de Málaga, Málaga, 2017.

[27] L. Cuenca, “Implementación de un sistema de navegación inercial, para mejorar la precisión de posicionamiento de un prototipo GPS en una trayectoria dentro de la ESPOCH,” Trabajo de fin de grado, Escuela Superior Politécnica de Chimborazo, Riobamba-Ecuador, 2017.

[28] R. Curey, M. Ash, L. Thielman y C. Barker, “Proposed IEEE Inertial Systems Terminology Standard and Other Inertial Sensor Standards,” IEEE Transactions on Intelligent Transportation Systems, pp. 83-90, 2004

[29] C. Larco, “Diseño e implementación de una plataforma Strap- Down para “Tracking Position” en sistemas indoor usando visión artificial y fusión sensorial a través de filtro de Kalman extendido,” Trabajo de fin de grado, Escuela Politécnica Nacional, Quito-Ecuador, 2015.

[30] F. Candelas y J. Ramón, “Giroscopios en el sistema GypsyGyro-18,” Universidad de Alicante, Pub. lnt.4, 2007.

[31] P. Martínez, “Unidad de medición inercial: estudio de fiabilidad y precisión,” Trabajo de fin de grado, Escuela Universitaria Politécnica de Teruel, Zaragoza, 2016.

[32] R. Quesada, “Acondicionamiento de las mediciones de sensores inerciales de bajo costo con fines de navegación,” Trabajo de fin de grado, Universidad Central “Marta Abreu” de Las Villas, Santa Clara, 2014.

[33] S. Martín, “Raspberry Pi, Arduino y Beaglebone Black Comparación y Aplicaciones,” Universidad Católica Nuestra Señora de la Asunción, Asunción, Paraguay, 2014.

[34] R. Pucha, “Diseño de un sistema prototipo para transmisión de imágenes de estaciones meteorológicas a través de la red celular para brindar soporte al personal técnico de meteo-

logía del INAMHI," Trabajo de fin de grado, Escuela Politécnica Nacional, Quito-Ecuador, 2016.

[35] "NEO-6 u-blox 6 GPS Modules," ublox, Technical data sheet, GPS.G6-HW-09005-E.

[36] "MPU-9250," InvenSense, Technical data sheet, PS-MPU-9250A-01.

[37] "128x64 Graphic OLED," VISHAY, Technical data sheet, OLED-128O064D-BPP3N00000.

[38] "ATtiny24/44/84," AVR Microcontroller, Technical Data Sheet.

[39] A. Fernández, Python al descubierto, Madrid: RC libros, 2012.

[40] "SQLite," [En línea]. Disponible en: https://www.sqlite.org/whentouse.html. [Accedido: 15-julio-2018].

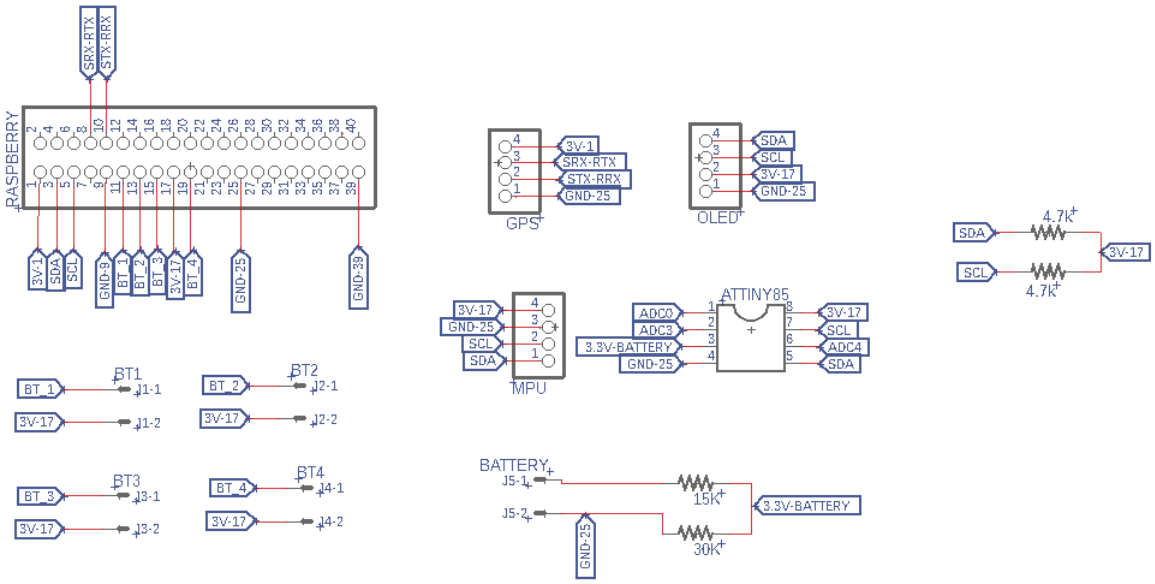
[41] I. Gonzales, F. Gomez y S. Lopez, "Implementación de SSH sobre un Sistema Auto-Reconfigurable," Universidad Autónoma de Madrid, Madrid.

[42] J. P. Alonso, "Diseño e implementación de un sistema de adquisición y actuación inalámbrico para vehículos aéreos no tripulados," Trabajo de fin de grado, Universitat Politécnica de Catalunya,2013.

Anexos

Anexo A

Anexo I: Esquema eléctrico de la placa de conexión



Anexo B

Anexo II: Programa implementado

Programa B.1: Script realizado en python

```
#Import libraries needed
import sqlite3
import smbus
import math
import time
from time import sleep
from datetime import date, datetime, timedelta
import Adafruit_GPIO.SPI as SPI
import Adafruit_SSD1306
from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont
import serial
import FaBo9Axis_MPU9250
import pynmea2
import RPi.GPIO as GPIO
import os,sys
import socket
from datetime import datetime
#================================
#GPIO Configuration
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
```

```
def map(x,in_min,in_max,out_min,out_max):
        return (x-in_min)*(out_max-out_min)/(in_max-in_min)+out_min

#========================================
#Configuration GPS

gps_port="/dev/ttyS0"
ser_gps=serial.Serial(gps_port,baudrate=9600, timeout=0.5)
latitude=0.0
longitude=0.0
speed=0.0
time_EC=" "

def read_gps_geo(option):
        global latitude
        global longitude
        data_gps=ser_gps.readline()
        if data_gps[0:6] == '$GPGGA':
                #print data_gps
                newmsg=pynmea2.parse(data_gps)
                latitude=newmsg.latitude
                longitude=newmsg.longitude
                if str(type(latitude))=="<type 'NoneType'>": latitude=0.0
                if str(type(longitude))=="<type 'NoneType'>": longitude=0.0

        if option=="latitude": data=latitude
        if option=="longitude": data=longitude
        return data

def read_gps_speed():
        data_gps=ser_gps.readline()
        global speed
        if data_gps[0:6]== '$GPVTG':
                msg=pynmea2.parse(data_gps)
                speed=msg.spd_over_grnd_kmph
                if str(type(speed))=="<type 'NoneType'>": speed=0.0
        return speed

def read_gps_time():
        data_gps=ser_gps.readline()
        global time_EC
        if data_gps[0:6] == '$GPGGA':
                newmsg=pynmea2.parse(data_gps)
```

```
                time_gps=newmsg.timestamp
                if str(type(time_gps))=="<type 'NoneType'>": time_gps=datetime.time(datetime.
                    now())
                date_gps=date.today()
                time_gps2=datetime.combine(date_gps,time_gps)
                time_EC=time_gps2-timedelta(hours=5)
                time_EC=time_EC.strftime("%c")
        return time_EC
#=======================================
#Configuration Buttons
btt_up=17
btt_down=27
btt_ok=22
btt_back=10
ok=0  # auxilar value
pointer=15

#Pines Configuration
GPIO.setup(btt_up   ,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(btt_down ,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(btt_ok   ,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(btt_back ,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)

#================================
#Read Voltage from Attiny85
bus=smbus.SMBus(1)
tiny_address=0x08
voltage=0

def read_Voltage():
        value=bus.read_byte(tiny_address)
        voltage=value*3.3/255
        if voltage < 0: voltage=0
        return voltage

#================================
#Configuration IMU
mpu9250 = FaBo9Axis_MPU9250.MPU9250()
alpha=16.0685
beta=33.243
sigma=0.9967
declination_mg=-3.617
```

```
def read_word(adr):
    high = bus.read_byte_data(address, adr)
    low = bus.read_byte_data(address, adr+1)
    val = (high << 8) + low
    return val

def read_word_2c(adr):
    val = read_word(adr)
    if (val >= 0x8000):
        return -((65535 - val) + 1)
    else:
        return val

def dist(a,b):
    return math.sqrt((a*a)+(b*b))

def get_y_rotation(x,y,z):
    radians = math.atan2(x, dist(y,z))
    return -math.degrees(radians)

def get_x_rotation(x,y,z):
    radians = math.atan2(y, dist(x,z))
    return math.degrees(radians)

def read_Yaw():
        mag=mpu9250.readMagnet()
        mx=mag['x']-alpha
        my=(mag['y']-beta)/sigma
        yaw=get_Mg_angle(mx,my) - declination_mg
        if(yaw<0): yaw=yaw+360
        return yaw

def get_Mg_angle(x,y):
        heading=-math.degrees(math.atan2(y,x))
        return (heading)

ax=0
ay=0
az=0
def read_Accel():
        global ax
        global ay
        global az
```

```
        accel=mpu9250.readAccel()
        ax=accel['x']
        ay=accel['y']
        az=accel['z']
#====================================================================
#IMU Calibration
offset_X=0
offset_Y=0

def imu_calibration(state,pointer):
        global offset_X
        global offset_Y
        global ok
        rotx=0
        roty=0
        if(state==1 and pointer==23):
                for i in range(20):
                        read_Accel()
                        rotx+=get_x_rotation(ax,ay,az)
                        roty+=get_y_rotation(ax,ay,az)
                        if i==20: break
                draw.text((20,54), 'IMU calibrated', font=font, fill=255)
                offset_X=rotx/20
                offset_Y=roty/20
                ok=0
                sleep(5)
#====================================================================
#Configuration OLED
# Raspberry Pi pin configuration
RST = 24
#128x64 display with  hardware I2C:
disp = Adafruit_SSD1306.SSD1306_128_64(rst=RST, i2c_address=0x3C)

#Initialize library oled
disp.begin()

#Clear display
disp.clear()
disp.display()

#Create blank image for drawing
#Make sure to create image with mode 1 for 1-bit color
width = disp.width
```

```
height = disp.height
#Print Initial Image
image1 = Image.open('mecatronica3.png').resize((disp.width, disp.height), Image.ANTIALIAS).
    convert('1')
disp.image(image1)
disp.display()
time.sleep(3)

disp.clear()
disp.display()
image = Image.new ('1',(width,height))

#Get drawing object to draw on image
draw=ImageDraw.Draw(image)

#Load default font
font=ImageFont.load_default()
font2=ImageFont.truetype("baby_blocks.ttf",8)
#============================================================
#Menu Configuration

def screen():
    draw.text((6,15), 'Conf. Date/Time ', font=font, fill=255)
    draw.text((6,15+8), 'IMU calibration ',font=font, fill=255)
    draw.text((6,15+16), 'Save Data ', font=font, fill=255)
    draw.text((6,15+24), 'Shutdown ', font=font, fill=255)

def pointer_dynamic(valor):
    # draw.rectangle((96,valor+2,104,valor+8), outline=255, fill=1)
    #draw.ellipse((96,valor+2,104,valor+8), outline=255, fill=1)
    draw.text((105,valor),'O', font=font2, fill=1)
    disp.display()

#============================================================
#Configuration Date/time Option

def conf_date_time(state,pointer):
    global ok
    if((state==1) and (pointer==15)):
        time_new=read_gps_time()
        if time_new >=' ':
                os.system(("sudo date -s '{}' ").format(time_new))
                draw.text((20,54), 'Tiempo cambiado', font=font, fill=255)
```

```
                sleep(3)
                ok=0


#============================================================
#Shutdown Option
def shutdown(state,pointer):
    if ((state==1) and (pointer==39)):
        a=0
        for i in range (5):
                a=5-i
                draw.rectangle((102,54,123,62),outline=0, fill=0)
                draw.text((3,54), 'Shutting Down in %r s' %a, font=font, fill=255)
                sleep(0.3)
                draw.rectangle((3,51,width-3,62),outline=0, fill=0)
                disp.image(image)
                disp.display()
                sleep(0.1)
                draw.text((3,54), 'Shutting Down in %r s' %a, font=font, fill=255)
                sleep(0.6)
                disp.image(image)
                disp.display()
                if i==5: break
        #Shutdown
        sleep(1)
        disp.clear()
        disp.display()
        disp.image(image1)
        disp.display()
        sleep(1)
        disp.clear()
        disp.display()
        os.system("sudo poweroff")
#================================================================
#Read Date / Time from System
def read_date(option):
        if (option=='year'):date=(datetime.now()).strftime("%y")
        if(option=='month'):date=(datetime.now()).strftime("%m")
        if(option=='day'):date=(datetime.now()).strftime("%d")
        return date
def read_time(option):
        if (option=='hour'): time=(datetime.now()).strftime("%H")
        if(option=='minute'): time=(datetime.now()).strftime("%M")
```

```
        if(option=='second'): time=(datetime.now()).strftime("%S")
        if(option=='microsecond'): time=(datetime.now()).strftime("%f")
        if(option=='time'): time=(datetime.now()).strftime("%H:%M:%S")
        return time
#============================================================
#Batery ICON
def draw_icon(voltag):
    percentage=int(map(voltag,0,3.3,0,100))
    draw.rectangle((3,4,23,10), outline=255, fill=0)
    draw.rectangle((23,6,25,8), outline=255, fill=1)

    if (percentage>0 and percentage <20):
       draw.rectangle((3,4,7,10), outline=255, fill=1)
    elif (percentage>=20 and percentage <40):
       draw.rectangle((3,4,7,10), outline=255, fill=1)
       draw.rectangle((7,4,11,10), outline=255, fill=1)
    elif (percentage>=40 and percentage <60):
       draw.rectangle((3,4,7,10), outline=255, fill=1)
       draw.rectangle((7,4,11,10), outline=255, fill=1)
       draw.rectangle((11,4,15,10), outline=255, fill=1)
    elif (percentage>=60 and percentage <80):
       draw.rectangle((3,4,7,10), outline=255, fill=1)
       draw.rectangle((7,4,11,10), outline=255, fill=1)
       draw.rectangle((11,4,15,10), outline=255, fill=1)
       draw.rectangle((15,4,19,10), outline=255, fill=1)
    elif (percentage>=80 and percentage <=100):
       draw.rectangle((3,4,7,10), outline=255, fill=1)
       draw.rectangle((7,4,11,10), outline=255, fill=1)
       draw.rectangle((11,4,15,10), outline=255, fill=1)
       draw.rectangle((15,4,19,10), outline=255, fill=1)
       draw.rectangle((19,4,23,10), outline=255, fill=1)
    return percentage
#=========================================================
#Clock ICON
def draw_clock():
        draw.ellipse((68-3,1+2,76-3,9+2), outline=255, fill=0)
        draw.line((72-3,5+2,72-3,9+2),fill=1)
        draw.line((72-3,5+2,76-3,5+2),fill=1)

#=========================================================
#Mensaje Animation
velocity=-15
startpos=width
```

```
pos=startpos

def animation(text,y):
        global pos
        maxwidth, unused = draw.textsize(text,font=font)
        x=pos
        for i, c in enumerate(text):
                if x > width: break
                if x < -10:
                        char_width, char_height =draw.textsize(c, font=font)
                        x += char_width
                        continue
                draw.text((x,y), c, font=font,fill=255)
                char_width, char_height =draw.textsize(c, font=font)
                x += char_width
        pos += velocity
        if pos < -maxwidth: pos=startpos

#============================================================
#DATABASE
conn= sqlite3.connect("Naturalistic_Data.db")
c=conn.cursor()
cont=1
format="Table %s" % str(cont)
db_aux=True

while (db_aux==True):
        try:
                c.execute('''CREATE TABLE '%s' (year text,month text,day text,hour text,minute
                     text,second text,microsecond text,roll real,pitch real, yaw real,speed
                    real,latitude real,longitude real)''' %format)
                db_aux=False

        except sqlite3.OperationalError:
                cont+=1
                format="Table %s" % str(cont)


def save_database(state,pointer,year,month,day,hour,minute,second,microsecond,roll,pitch,yaw,
    speed,latitude,longitude):
    if((state==1) and (pointer==31)):
        #draw.text((25,50), 'Saving Data', font=font, fill=255)
        animation("Saving Data",54)
```

```
        #draw.text((20,52), 'RotX ' + str(roll),font=font, fill=255)
        c.execute("INSERT INTO '%s' VALUES (%s,%s,%s,%s,%s,%s,%s,%.4f, %.4f, %.4f, %.4f, %.4f,
              %.4f)" % (format,year,month,day,hour,minute,second,microsecond,roll,pitch,yaw,
             speed,latitude,longitude))
        conn.commit()
        sleep(1)

#===============================================================
# Main Program
pointer_accum=8

try:
    while True:
        #draw.rectangle((0,0,width-2,height-2), outline=0, fill=0)
        draw.rectangle((0,0,width,height), outline=0, fill=255)
        draw.rectangle((2,2,width-2,height-2), outline=0, fill=0)
#Read Buttons
        if (GPIO.input(btt_up)==GPIO.HIGH): pointer-=pointer_accum
        if (GPIO.input(btt_down)==GPIO.HIGH): pointer+=pointer_accum
        if (GPIO.input(btt_ok)==GPIO.HIGH): ok=1
        if (GPIO.input(btt_back)==GPIO.HIGH): ok=0
        if(ok==0): pointer_accum=8
        else: pointer_accum=0
        if (pointer>=39): pointer=39
        if (pointer<=15): pointer=15
#Set voltage value
        voltage=read_Voltage()
        draw_icon(voltage)
        perc=str(draw_icon(voltage))
        draw_clock()
        screen()
        pointer_dynamic(pointer)

#Get Time
        time=str(read_time('time'))
        hour=str(read_time('hour'))
        minute=str(read_time('minute'))
        second=str(read_time('second'))
        microsecond=str(read_time('microsecond'))
#Get Date
        year=str(read_date('year'))
        month=str(read_date('month'))
        day=str(read_date('day'))
```

```
#Get Latitude, Longitude
        latitude=float(read_gps_geo('latitude'))
        longitude=float(read_gps_geo('longitude'))
#Get Speed
        speed=float(read_gps_speed())
#IMU calibration
        imu_calibration(ok,pointer)
#Get Roll, Pitch, Yaw
        read_Accel()
        yaw=read_Yaw()
        roll=get_x_rotation(ax,ay,az)-offset_X
        pitch=get_y_rotation(ax,ay,az)-offset_Y

#Print percent of battery
        draw.text((28,2),perc+"%", font=font, fill=255)
        draw.text((76,2),time ,font=font, fill=255)

#Date/Time configuration
        conf_date_time(ok,pointer)

#Shutdown Option
        shutdown(ok,pointer)

#DATABASE
        save_database(ok,pointer,year,month,day,hour,minute,second,microsecond,roll,pitch,yaw,
            speed,latitude,longitude)
        disp.image(image)
        disp.display()

except KeyboardInterrupt:
        disp.clear()
        disp.display()
        disp.image(image1)
        disp.display()
        sleep(2)
        disp.clear()
        disp.display()
        conn.close()
        GPIO.cleanup()
        pass
```