



## **UNIVERSIDAD TÉCNICA DEL NORTE**

**FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS**

**CARRERA DE INGENIERÍA EN ELECTRÓNICA  
Y REDES DE COMUNICACIÓN**

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

**TEMA**

**PLACA DE ENTRENAMIENTO PARA APLICACIONES  
ELECTRÓNICAS CON TERMINALES MÓVILES BASADOS  
EN SISTEMA OPERATIVO ANDROID.**

**AUTOR: BYRON R. VALENZUELA M.**

**DIRECTOR: ING. EDGAR MAYA**

**Ibarra- Ecuador**

**2014**



## UNIVERSIDAD TÉCNICA DEL NORTE

### BIBLIOTECA UNIVERSITARIA

#### AUTORIZACIÓN DE USO Y PUBLICACIÓN

#### A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

#### 1. IDENTIFICACIÓN DE LA OBRA

La Universidad Técnica del Norte dentro del proyecto Repositorio Digital Institucional, determinó la necesidad de disponer de textos completos en formato digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la Universidad.

Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual pongo a disposición la siguiente información.

<b>DATOS DEL CONTACTO</b>	
Cédula de identidad	100287050-7
Apellidos y Nombres	Valenzuela Maila Byron Ramiro
Dirección	Río Yasuní 1-62 y Río Tahuando
E-mail	<a href="mailto:byron_101990@hotmail.com">byron_101990@hotmail.com</a>
Teléfono fijo	062-604-664
Teléfono móvil	0999185504
<b>DATOS DE LA OBRA</b>	
Título	PLACA DE ENTRENAMIENTO PARA APLICACIONES ELECTRÓNICAS CON TERMINALES MÓVILES BASADOS EN SISTEMA OPERATIVO ANDROID
Autor	Valenzuela Maila Byron Ramiro
Fecha	Octubre del 2014
Programa	Pregrado
Título	Ingeniero en Electrónica y Redes de Comunicación
Director	Ing. Edgar Maya



## 2. AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD

Yo, Valenzuela Maila Byron Ramiro, con cédula de identidad Nro. 100287050-7, en calidad de autor y titular de los derechos patrimoniales de la obra o trabajo de grado descrito anteriormente, hago entrega del ejemplar respectivo en forma digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y uso del archivo digital en la Biblioteca de la Universidad con fines académicos, para ampliar la disponibilidad de material y como apoyo a la educación, investigación y extensión, en concordancia con la ley de Educación Superior Artículo 144.

Firma:.....

Nombre: Valenzuela Byron

Cédula: 100287050-7

Ibarra, Octubre del 2014



**UNIVERSIDAD TÉCNICA DEL NORTE**

**FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS**

**CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE GRADO A FAVOR DE LA  
UNIVERSIDAD TÉCNICA DEL NORTE**

Yo Valenzuela Maila Byron Ramiro, con cédula de identidad número 100287050-7, manifiesto mi voluntad de ceder a la Universidad Técnica del Norte los derechos patrimoniales consagrados en la Ley de Propiedad Intelectual del Ecuador artículos 4, 5 y 6, en calidad de la autor del trabajo de grado con el tema: PLACA DE ENTRENAMIENTO PARA APLICACIONES ELECTRÓNICAS CON TERMINALES MÓVILES BASADOS EN SISTEMA OPERATIVO ANDROID. Que ha sido desarrollado con el propósito de obtener el título de Ingeniero en Electrónica y Redes de Comunicación de la Universidad Técnica del Norte, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En mi condición de autor me reservo los derechos morales de la obra antes citada. En concordancia suscribo en el momento que hago entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Técnica del Norte.

Valenzuela Byron

100287050-7

Ibarra, Octubre del 2014



**UNIVERSIDAD TÉCNICA DEL NORTE**  
**FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS**

**CERTIFICACIÓN**

INGENIERO EDGAR MAYA, DIRECTOR DEL PRESENTE TRABAJO DE TITULACIÓN  
CERTIFICA

Que, el presente Trabajo de Titulación "PLACA DE ENTRENAMIENTO PARA APLICACIONES ELECTRÓNICAS CON TERMINALES MÓVILES BASADOS EN SISTEMA OPERATIVO ANDROID." Ha sido desarrollado por el señor Valenzuela Maila Byron Ramiro bajo mi supervisión.

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

.....  
Ing. Edgar Maya  
DIRECTOR



**UNIVERSIDAD TÉCNICA DEL NORTE**  
**FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS**

**CONSTANCIAS**

Yo, VALENZUELA MAILA BYRON RAMIRO declaro bajo juramento que el trabajo aquí escrito es de mi autoría; y que este no ha sido previamente presentado para ningún grado o calificación profesional y que he consultado las referencias bibliográficas que se presentan en este documento.

A través de la presente declaración cedo los derechos de propiedad intelectual correspondientes a este trabajo, a la Universidad Técnica del Norte, según lo establecido por las leyes de propiedad intelectual, reglamentos y normatividad vigente de la Universidad Técnica del Norte.

En la ciudad de Ibarra, Octubre del 2014

EL AUTOR

Valenzuela Maila Byron Ramiro.

CI: 100287050-7



**UNIVERSIDAD TÉCNICA DEL NORTE**  
**FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS**

**DEDICATORIA**

La vida universitaria está llena de grandes retos, metas, anhelos, triunfos, risas y lágrimas, pero todo sacrificio es bueno por cumplir una meta personal y culminar un peldaño más en la formación académica.

Dedico este trabajo a mi familia, profesores, amigos y a todas las personas que de una u otra manera confiaron en mí y me apoyaron con sus consejos, palabras de aliento y enseñanzas en este proceso de formación profesional.

A mi mamá, papá y hermanos les dedico este proyecto de titulación ya que ellos fueron el sostén y la razón por la cual luche siempre con un solo objetivo ser Ingeniero. Una vez más gracias por su tiempo, apoyo y dedicación en las buenas y malas.

BYRON



**UNIVERSIDAD TÉCNICA DEL NORTE**  
**FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS**

**AGRADECIMIENTO**

A Dios, por colmarme de bendiciones y permitirme llegar hasta este punto que es cumplir mi meta estudiantil universitaria.

Un gran agradecimiento a la Universidad Técnica del Norte por haberme acogido y brindado los conocimientos necesarios para forjarme como profesional, a todos los profesores por su paciencia y entrega en las aulas, en especial al Ingeniero Edgar Maya por haberme brindado sus conocimientos, consejos y predisposición para salir adelante con este proyecto de titulación

A mis padres, por todo el cariño, la entrega y apoyo incondicional que me han entregado, por todo el sacrificio que han puesto hacia mí con el objetivo de formar un ente responsable solidario y buen profesional, que ha cumplido su meta y ha logrado hacerles sentir orgullosos del hombre que formaron.

BYRON

## RESUMEN

El presente trabajo de titulación, consiste en el diseño e implementación de un entrenador electrónico para dispositivos móviles basados en el sistema operativo Android, este sistema incluye varios dispositivos como led's, pulsadores, switch, buzzer, releé, display's, sensores de temperatura y módulos de comunicación como GPS, Wi-Fi, Xbee, I2C, con los que el móvil y el entrenador interactuarán para desarrollar aplicaciones.

El trabajo inicia con una revisión de la placa IOIO que es con la junta que se trabaja en este proyecto, se analiza sus características, estructura física, estructura interna, fuente de alimentación y los diferentes módulos que componen este dispositivo. A demás se exhibe varios proyectos realizados a nivel mundial con este equipo.

El segundo capítulo presenta un estudio del sistema operativo Android con el propósito de conocer detalladamente sus características, la estructura de un proyecto, el SDK para el desarrollo de aplicaciones, el lenguaje de programación, el IDE, y por último las librerías y recursos que trae consigo la placa electrónica IOIO con las que permite manejar los distintos periféricos y módulos de la misma.

En la tercera parte se realiza el diseño e implementación del entrenador y los módulos que forman parte de él como son: fuente de alimentación, módulo de entrada y salida digital, conversor análogo digital, comunicación serial, ZigBee, Bluetooth, GPS, Wi-fi, y módulo I2C.

La última parte del proyecto consiste en presentar varias propuestas de prácticas con cada uno de los módulos que componen el sistema y así verificar el funcionamiento tanto de la aplicación como del entrenador Android. En este capítulo se presenta todo el proceso que conlleva realizar una práctica de laboratorio.

## **ABSTRACT**

This graduation work consist in the desing and implementation of an electronic trainer for movil devices based on the Android operative system, this system include several devices such us: led's, switches, buzzer, relay, display's, temperature sensors and communication modules as GPS, Wi-Fi, Xbee, I2C, with this devices the trainer and the mobile will interact to develop aplications.

The work begin with a review of IOIO board that this board is wornking in this project, its characteristics, physical structure, internal structure, power supply and the different modules that comprise this device are analyzed. Also a few projects made worldwide with this device are presented.

The second chapter present a study of Android operative system in order of know the OS in detail the characteristics, the structure of project, the SDK for application development, the proگرامing language, IDE, the libraries and resources IOIO board that permit to handle the differents modules and periferics

In the third part desing and implementation the trainer and the differents its differents modules such us: power supply, Digital Input Output, temperature sensors serial communication Xbee, Bluetooth, GPS, Wi-Fi and I2C.

The last part of this Project consist in to present a few proposal of practices with Each of the modules that make up the system and verify the operation of both the application and the Android trainer. This chapter describes the process to make the laboratory practice.



## ÍNDICE DE CONTENIDO

AUTORIZACIÓN DE USO Y PUBLICACIÓN .....	II
CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE GRADO A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE .....	IV
CERTIFICACIÓN .....	V
CONSTANCIAS .....	VI
DEDICATORIA .....	VII
AGRADECIMIENTO .....	VIII
RESUMEN .....	IX
ABSTRACT .....	X
ÍNDICE DE CONTENIDO .....	XI
ÍNDICE DE FIGURAS .....	XV
ÍNDICE DE TABLAS .....	XIX
<b>CAPÍTULO I</b> .....	<b>1</b>
1 ESTUDIO DE LA PLACA IOIO .....	1
1.1 INTRODUCCIÓN .....	1
1.2 VERSIONES DE LA PLACA IOIO .....	2
1.3 CARACTERÍSTICAS .....	3
1.3.1 ESTRUCTURA DE LA PLACA IOIO .....	3
1.3.2 PINES DE LA PLACA IOIO .....	4
1.4 DESCRIPCIÓN .....	7
1.4.1 FUENTE DE ALIMENTACIÓN .....	7
1.4.1.1 CARACTERÍSTICAS DE POTENCIA .....	7
1.4.2 MÓDULO DE ENTRADA Y SALIDA .....	8
1.4.2.1 SALIDA DIGITAL .....	8
1.4.2.2 ENTRADA DIGITAL .....	9
1.4.3 CONVERTOR ANÁLOGO DIGITAL .....	9

1.4.4 MÓDULO PWM.....	10
1.4.5 MÓDULO UART .....	10
1.4.6 MÓDULO I2C.....	10
1.5 CONEXIÓN DE LA PLACA IOIO .....	11
1.5.1 CONEXIÓN MEDIANTE CABLE USB .....	11
1.5.2 CONEXIÓN MEDIANTE BLUETOOTH.....	12
1.6 APLICACIONES REALIZADAS CON LA PLACA IOIO .....	13
<b>CAPÍTULO II .....</b>	<b>15</b>
<b>2 ESTUDIO DEL SDK DE ANDROID .....</b>	<b>15</b>
<b>2.1 SISTEMA OPERATIVO ANDROID .....</b>	<b>15</b>
2.1.1 CARACTERÍSTICAS DEL SISTEMA OPERATIVO ANDROID .....	15
2.1.2 FUNDAMENTOS PARA EL DESARROLLO DE APLICACIONES EN EL SISTEMA OPERATIVO ANDROID .....	16
2.2 SDK DE ANDROID .....	17
2.2.2 HERRAMIENTAS SDK .....	19
2.3 SDK MANAGER .....	20
2.4 ANDROID VIRTUAL DEVICE.....	21
2.5 ANDROID DEVELOPER TOOLS.....	22
2.5.1 EDITORES DE CÓDIGO.....	23
2.5.2 EDITOR DE DISEÑO GRÁFICO.....	24
2.6 ESTRUCTURA DE UN PROYECTO ANDROID .....	25
2.7 LIBRERÍAS DE LA PLACA IOIO.....	27
2.7.1 MÓDULO DE ENTRADA Y SALIDA.....	29
2.7.1.1 SALIDA DIGITAL.....	29
2.7.1.2 ENTRADA DIGITAL.....	30
2.7.2 CONVERTOR ANÁLOGO DIGITAL .....	31
2.7.3 MÓDULO PWM.....	33
2.7.4 MÓDULO UART .....	33

2.7.5 MÓDULO I2C.....	34
<b>CAPÍTULO III</b> .....	<b>36</b>
<b>3 DISEÑO Y CONSTRUCCIÓN DEL ENTRENADOR</b> .....	<b>36</b>
3.1 DESCRIPCIÓN GENERAL DEL ENTRENADOR .....	36
3.2 MÓDULO DE ENTRADA Y SALIDA DIGITAL.....	37
3.2.1 SALIDA DIGITAL.....	37
3.2.2 EENTRADA DIGITAL .....	46
3.3 MÓDULO DE CONVERSIÓN ANÁLOGO DIGITAL.....	49
3.4 MÓDULO DE COMUNICACIÓN SERIAL .....	51
3.5 MÓDULO ZIGBEE .....	52
3.6 MÓDULO BLUETTOH.....	53
3.7 MÓDULO GPS.....	55
3.8 MÓDULO WI-FI.....	59
3.9 MÓDULO I2C.....	61
3.10 FUENTE DE ALIMENTACIÓN.....	62
3.11 RESUPUESTO.....	65
<b>CAPÍTULO IV</b> .....	<b>68</b>
<b>4 PROPUESTA DE GUÍAS DE PRÁCTICAS</b> .....	<b>68</b>
4.1 PRÁCTICAS CON EL MÓDULO ENTRADA Y SALIDA DIGITAL.....	68
4.1.1 JUEGO DE LUCES.....	68
4.1.2 CONTADOR DE PULSOS .....	87
4.1.3 CONTADOR.....	96
4.1.4 LETRERO DINÁMICO .....	120
4.2 PRÁCTICA CON EL MÓDULO CONVERSOR ANÁLOGO DIGITAL.....	136
4.2.1 MEDIDOR DE VALORES ANALÓGICOS .....	136
4.3 PRÁCTICA PWM .....	147
4.4 PRÁCTICA CON EL MÓDULO I2C .....	156

4.5 PRÁCTICAS CON EL MÓDULO DE COMUNICACIÓN SERIAL .....	168
4.5.1 MÓDULO BLUETOOTH .....	168
4.5.3 MÓDULO GPS.....	203
4.5.4 MÓDULO WI-FI.....	230
<b>CAPÍTULO V.....</b>	<b>249</b>
<b>5 CONCLUSIONES Y RECOMENDACIONES .....</b>	<b>249</b>
5.1 CONCLUSIONES .....	249
<b>ANEXOS.....</b>	<b>255</b>
ANEXO A PASOS PARA LA CONEXIÓN MEDIANTE BLUETOOTH.....	255
ANEXO B INSTALACIÓN DEL ENTORNO DE DESARROLLO PARA APLICACIONES ANDROID.....	256
ANEXO C IMPORTAR LAS LIBRERIAS Y LOS EJEMPLOS IOIO AL ENTORNO DE DESARROLLO ANDROID. ....	270
ANEXO D PROPUESTA DE PRACTICAS DIRIGAS A LOS ESTUDIANTES. ....	272
ANEXO E ENTRENADOR PARA APLICACIONES ELECTRÓNICAS CON TERMINALES MÓVILES BASADOS EN SISTEMA OPERATIVO ANDROID .....	273

## ÍNDICE DE FIGURAS

FIGURA 1. PLACA IOIO.....	1
FIGURA 2. COMPONENTES DE LA PLACA IOIO.....	3
FIGURA 3. MÓDULO DE ENTRADA Y SALIDA.....	8
FIGURA 4. CONFIGURACIÓN DE UN PIN OPEN DRAIN EN MODO PULL UP.....	9
FIGURA 5. CONEXIÓN DE LA PLACA IOIO CON EL DISPOSITIVO ANDROID MEDIANTE EL CABLE USB.....	11
FIGURA 6. DISPOSITIVO BLUETOOTH USB.....	12
FIGURA 7. CONEXIÓN DE LA IOIO MEDIANTE BLUETOOTH.....	12
FIGURA 8. IOIO ROVER.....	13
FIGURA 9. ALARMA DE INTRUSOS.....	14
FIGURA 10. PANTALLA PRINCIPAL ANDROID SDK MANAGER.....	20
FIGURA 11. INTERFAZ DE UN DISPOSITIVO VIRTUAL ANDROID (AVD).....	21
FIGURA 12. INTERFAZ DE LA APLICACIÓN ANDROID VIRTUAL DEVICE MANAGER.....	22
FIGURA 13. INTERFAZ DEL EDITOR DE DISEÑO GRAFICO.....	24
FIGURA 14. CONTENIDO DE UN PROYECTO ANDROID.....	25
FIGURA 15. DESCRIPCIÓN GENERAL DEL ENTRENADOR.....	36
FIGURA 16. CONECTOR MICKORBUS(I), CONECTOR XBEE EXPLORER (D).....	37
FIGURA 17. LED'S.....	37
FIGURA 18. ESQUEMA DE CONEXIÓN LED'S.....	39
FIGURA 19. BUZZER ELECTRÓNICO.....	39
FIGURA 20. CIRCUITO DE POLARIZACIÓN.....	40
FIGURA 21. ESQUEMA DE CONEXIÓN BUZZER.....	41
FIGURA 22. DISPLAY 7 SEGMENTOS.....	41
FIGURA 23. DECODIFICADOR 74LS47.....	42
FIGURA 24. ESQUEMA DE CONEXIÓN IOIO-DPISPLAYS.....	42
FIGURA 25. PANTALLA LCD 16X2.....	43
FIGURA 26. DIAGRAMA DE CONEXIÓN LCD.....	43
FIGURA 27. CONEXIÓN RELÉ.....	45
FIGURA 28. ESQUEMA DE CONEXIÓN RESISTENCIA PULL-UP.....	45
FIGURA 29. DIP-SWITCH 8 POSICIONES.....	46

FIGURA 30. ESQUEMA CONEXIÓN DIP-SWITCH .....	47
FIGURA 31. PULSADORES .....	47
FIGURA 32. CONEXIÓN PULSADORES CON LA PLACA IOIO .....	49
FIGURA 33. SENSOR DE TEMPERATURA LM 35 .....	50
FIGURA 34. DIAGRAMA DE CONEXIÓN LM35 .....	50
FIGURA 35. DIAGRAMA DE CONEXIÓN POTENCIOMETRO .....	50
FIGURA 36. NIVELES DE CONVERSIÓN RS 232 A TTL Y VICEVERSA .....	51
FIGURA 37. MAX232 BOARD .....	51
FIGURA 38. MÓDULO XBEE® .....	53
FIGURA 39. CONEXIÓN XBEE® CON LA IOIO .....	53
FIGURA 40. MÓDULO BLUETOOTH2 CLICK .....	54
FIGURA 41. CONEXIÓN BLUETOOTH2 CLICK CON LA IOIO .....	55
FIGURA 42. GPS2 CLICK .....	55
FIGURA 43. ANTENA ACTIVA DLGPS-E2 .....	56
FIGURA 44. CONEXIÓN GPS2 CLICK CON LA IOIO .....	58
FIGURA 45. CONFIGURACIÓN DEL AMPLIFICADOR NO INVERSOR .....	58
FIGURA 46. WIFLY RN-XV .....	60
FIGURA 47. CONEXIÓN WI-FLY CON LA IOIO .....	61
FIGURA 48. MÓDULO I2C TMP 102 .....	61
FIGURA 49. CONEXIÓN DEL MÓDULO I2C CON LA IOIO .....	61
FIGURA 50. DIAGRAMA DE LA FUENTE DE ALIMENTACIÓN DEL ENTRENADOR .....	64
FIGURA 51. FUENTE DE ALIMENTACIÓN VARIBALE .....	65
FIGURA 52. A) ESQUEMA DIODO LED B) SÍMBOLO DIODO LED .....	69
FIGURA 53. DIAGRAMA DE CONEXIÓN DE LA APLICACIÓN JUEGO DE LUCES .....	72
FIGURA 54. CIRCUITO EQUIVALENTE PARA EL ESTADO 1 LÓGICO .....	73
FIGURA 55. CREACIÓN DE UNA NUEVA APLICACIÓN ANDROID .....	77
FIGURA 56. INFORMACIÓN CORRESPONDIENTE A LA APLICACIÓN ANDROID A CREAR .....	77
FIGURA 57. CONFIGURACIÓN DEL NUEVO PROYECTO ANDROID .....	78
FIGURA 58. CONFIGURACIÓN DEL ICONO DEL PROYECTO .....	78
FIGURA 59. SELECCIÓN DEL TIPO DE FORMULARIO .....	79
FIGURA 60. PROPIEDADES ANDROID DEL PROYECTO .....	79

FIGURA 61. LIBRERÍAS IOIO A INCLUIR EN EL PROYECTO .....	80
FIGURA 62. DIAGRAMA DE CONEXIÓN DE LA APLICACIÓN CONTADOR DE PULSOS.....	89
FIGURA 63. RELACIÓN ENTRE UN TRANSISTOR Y UN INTERRUPTOR .....	97
FIGURA 64. CIRCUITO DE POLARIZACIÓN, ZONA DE CORTE Y SATURACIÓN DEL TRANSISTOR .....	97
FIGURA 65. DIAGRAMA DE CONEXIÓN DE LA APLICACIÓN CONTADOR .....	100
FIGURA 66. DIAGRAMA DE CONEXIÓN DE UNA MATRIZ DE LEDS DE 8X8 .....	120
FIGURA 67. DIAGRAMA DE CONEXIÓN DE LA IOIO CON LA MATRIZ DE LED´S.....	123
FIGURA 68. CUANTIFICACIÓN Y CODIFICACIÓN DE UNA SEÑAL ANALÓGICA.....	137
FIGURA 69. ESQUEMA DE CONEXIÓN POTENCIÓMETRO Y SENSOR DE TEMPERATURA CON LA IOIO.....	139
FIGURA 70. REPRESENTACIÓN DE PWM.....	148
FIGURA 71. CONEXIÓN DE LA APLICACIÓN PWM.....	150
FIGURA 72. CONDICIÓN DE INICIO Y PARADA EN I2C .....	157
FIGURA 73. CONEXIÓN SENSOR DE TEMPERATURA I2C Y LA IOIO .....	160
FIGURA 74. TOPOLOGIAS BLUETOOTH .....	169
FIGURA 75. ESQUEMA DE CONEXIÓN DE LA IOIO CON EL MÓDULO BLUETOOTH2 CLICK	173
FIGURA 76. DIAGRAMA GENERAL DE LA APLICACIÓN MEDIDOR DE TEMPERATURA INALÁMBRICO.....	173
FIGURA 77. CONEXIÓN DEL MÓDULO BLUETOOTH CON EL COMPUTADOR .....	178
FIGURA 78. IMAGEN DE PRESENTACIÓN DEL MÓDULO .....	178
FIGURA 79. DISPOSITIVOS BLUETOOTH DISPONIBLES .....	179
FIGURA 80. SERVICIOS DEL MÓDULO BLUETOOTH .....	180
FIGURA 81. ESTABLECIENDO COMUNICACIÓN CON EL MÓDULO BLUETOOTH .....	180
FIGURA 82. CONFIGURACIÓN DEL PUERTO COM.....	181
FIGURA 83. RECEPCIÓN DE LA TEMPERATURA EN EL PUERTO COM.....	181
FIGURA 84. TOPOLOGÍAS ZIGBEE .....	186
FIGURA 85. CANALES DE TRANSMISIÓN DE LA BANDA 2.4GHZ PARA IEEE 802.15.4 .....	187
FIGURA 86. REPRESENTACIÓN GRÁFICA DE LA MANERA EN QUE ESTÁ ORGANIZADA LA APLICACIÓN.....	192
FIGURA 87. SISTEMA TRANSMISOR DE LA APLICACIÓN.....	193
FIGURA 88. CONEXIÓN DE LA IOIO CON EL MÓDULO XBEE S1 .....	194

FIGURA 89. CONEXIÓN TOTAL DE LA APLICACIÓN CON MÓDULOS ZIGBEE .....	194
FIGURA 90. CONEXIÓN DEL XBEE AL PC.....	196
FIGURA 91. VENTANA DE VERIFICACIÓN DE CONECTIVIDAD CON EL MÓDULO .....	196
FIGURA 92. PESTAÑA DE CONFIGURACIÓN DEL MÓDULO XBEE .....	197
FIGURA 93. SENTENCIAS NMEA.....	204
FIGURA 94. CONEXIÓN DEL MÓDULO GPS CON LA IOIO .....	209
FIGURA 95. VENTANA PRINCIPAL ANDROID SDK MANAGER .....	214
FIGURA 96. VENTANA DE IMPORTACIÓN DE PROYECTOS EN ECLIPSE .....	215
FIGURA 97. INTERFAZ PARA BUSCAR LA DIRECCIÓN DE UN PROYECTO A IMPORTAR....	215
FIGURA 98. VENTANA IMPORTAR PROYECTOS EN ECLIPSE .....	216
FIGURA 99. VENTANA PARA AÑADIR LIBRERÍAS AL PROYECTO ANDROID .....	216
FIGURA 100. VENTANA PREFERENCES DE ECLIPSE.....	217
FIGURA 101. INGRESO A LA DIRECCIÓN MENCIONADA MEDIANTE EL CMD .....	217
FIGURA 102. EJECUCIÓN DEL COMANDO PARA VERIFICAR LA CLAVE SHA1 .....	218
FIGURA 103. CLAVE SAH1.....	218
FIGURA 104. PESTAÑA DE CREACIÓN DE UN NUEVO PROYECTO.....	219
FIGURA 105. ACTIVACIÓN DEL SERVICIO DE MAPAS PARA ANDROID .....	219
FIGURA 106. VENTANA API ACCESS .....	220
FIGURA 107. VENTANA DE CONFIGURACIÓN PARA OBTENER LA CLAVE ANDROID PARA EL USO DE LOS MAPAS.....	220
FIGURA 108. VENTANA DONDE SE OBSERVA LA CLAVE GENERADA PARA HACER USO DE LOS MAPAS DE GOOGLE .....	221
FIGURA 109. RED AD-HOC .....	231
FIGURA 110. RED DE INFRAESTRUCTURA.....	232
FIGURA 111. RED EXTENDIDA.....	232
FIGURA 112. ESQUE DE CONEXIÓN IOIO, SENSOR DE HUMEDAD DE SUELO Y WIFLY.....	238
FIGURA 113. ESQUEMA GENERAL DE LA APLICACIÓN CON EL MÓDULO WI-FLY.....	238



## ÍNDICE DE TABLAS

TABLA 1. VERSIONES DE LA PLACA IOIO .....	2
TABLA 2. PINES DE LA IOIO .....	4
TABLA 3. REQUERIMIENTOS DE ESPACIO EN DISCO DURO PARA EL ANDROID SDK.....	16
TABLA 4. HERRAMIENTAS SDK .....	19
TABLA 5. EDITORES DE CÓDIGOS XML .....	23
TABLA 6. SUBCARPETAS DE LA CARPETA RES .....	26
TABLA 7. ESPECIFICACIONES XBEE® Y XBEE-PRO®.....	52
TABLA 8. ESPECIFICACIONES BLUETOOTH2 CLICK .....	54
TABLA 9. ESPECIFICACIONES GPS2 CLICK.....	56
TABLA 10. ESPECIFICACIONES ANTENA DLGPS-E2 .....	57
TABLA 11. ESPECIFICACIONES WIFLY RN-XV.....	60
TABLA 12. REQUERIMIENTOS DE VOLTAJE DE LO MÓDULOS DEL ENTRENADOR.....	62
TABLA 13. COSTOS DE LOS MÓDULOS Y MATERIALES PARA LA CONSTRUCCIÓN DEL ENTRENADOR .....	65
TABLA 14. ESPECIFICACIONES DIODO LED ROJO .....	69
TABLA 15. REQUERIMIENTOS DE LA APLICACIÓN JUEGO DE LUCES .....	70
TABLA 16. REQUERIMIENTOS PARA LA APLICACIÓN CONTEO DE PULSOS .....	87
TABLA 17. DESCRIPCIÓN DE LOS PINES Y MÓDULOS USADOS EN LA APLICACIÓN.....	89
TABLA 18. TABLA DE RESULTADOS CONTADOR DE PULSOS .....	90
TABLA 19. REQUERIMIENTOS PARA LA APLICACIÓN CONTADOR .....	98
TABLA 20. DESCRIPCIÓN DE LOS PINES USADOS EN LA APLICACIÓN CONTADOR.....	100
TABLA 21. TABLA DE RESULTADOS DE LA APLICACIÓN CONTADOR .....	101
TABLA 22. REQUERIMIENTOS PARA LA APLICACIÓN LETRERO DINÁMICO .....	121
TABLA 23. PINES Y MATERIALES USADOS EN LA APLICACIÓN LETRERO DINÁMICO .....	123
TABLA 24. REQUERIMIENTOS DE LA APLICACIÓN CON EL MÓDULO CONVERSIONOR ANALÓGICO-DIGITAL.....	137
TABLA 25. DESCRIPCIÓN DE LOS PINES Y MATERIALES USADOS EN LA APLICACIÓN DE CONVERSIÓN ANÁLOGA DIGITAL.....	139

TABLA 26. COMPARACIÓN DE RESULTADOS .....	140
TABLA 27. REQUERIMEINTOS DE LA APLICACIÓN PWM .....	148
TABLA 28. MATERIALES NECESARIOS PARA LA APLICACIÓN I2C.....	160
TABLA 29. CAPTURAS DE TEMPERATURA OBTENIDAS. ....	161
TABLA 30. REQUERIMIENTOS DE LA APLICACIÓN CON EL MÓDULO BLUETOOTH .....	170
TABLA 31. MATERIALES NECESARIOS PARA LA APLICACIÓN BLUETOOTH .....	172
TABLA 32. REQUERIMIENTOS DE LA APLICACIÓN CON EL MÓDULO ZIGBEE .....	188
TABLA 33. DESCRIPCIÓN DE MATERIALES USADOS EN LA APLICACIÓN MEDIDOR DE DISTANCIA MEDIANTE MÓDULOS ZIGBEE .....	191
TABLA 34. PARÁMETROS DE CONFIGURACIÓN DE LOS MÓDULOS XBEE TRANSMISOR Y RECEPTOR.....	192
TABLA 35. RESULTADOS DE LAS MEDICIONES ENTRE EN TRANSMISOR Y RECEPTOR .....	195
TABLA 36: DESCRIPCIÓN DE LA TRAMA "\$GPGGA" .....	205
TABLA 37. REQUERIMIENTOS PARA LA APLICACIÓN DE UBICACIÓN GPS .....	206
TABLA 38. DESCRIPCIÓN DE LOS PINES Y REQUERIMIENTOS DE LA APLICACIÓN .....	209
TABLA 39. TABLA DE RESULTADOS DE MEDICIONES REALIZADAS CON EL GPS.....	210
TABLA 40. VENTAJAS Y DESVENTAJAS DE LAS REDES WI-FI.....	234
TABLA 41. MATERIALES NECESARIOS PARA LA APLICACIÓN WI-FI.....	237

# CAPÍTULO I

## 1 ESTUDIO DE LA PLACA IOIO

### 1.1 INTRODUCCIÓN

IOIO es una placa diseñada y fabricada especialmente para trabajar con dispositivos móviles basados en el sistema operativo Android, desde la versión 1.5 (Donout) hasta la versión actual de Android que es la versión 4.1 (Jelly Beam) (IOIO for Android, s.f.).

La conectividad con el dispositivo móvil se la puede realizar mediante la conexión del cable USB o mediante Bluetooth, IOIO contiene un microcontrolador PIC que interpreta el código escrito en la aplicación Android, y permite interactuar con los distintos módulos con los que cuenta como son: Entrada / Salida Digital, PWM<sup>1</sup>, conversores análogo-digitales, I2C<sup>2</sup>, SPI<sup>3</sup>, UART<sup>4</sup>.

Gracias a la facilidad de la interconexión con el móvil Android se puede sacar el máximo provecho a un sin número de sensores y periféricos que los dispositivos móviles poseen tales como: acelerómetro, pantalla táctil. GPS<sup>5</sup>, cámara digital, display etc. Y realizar aplicaciones con dichos sensores y la placa IOIO.

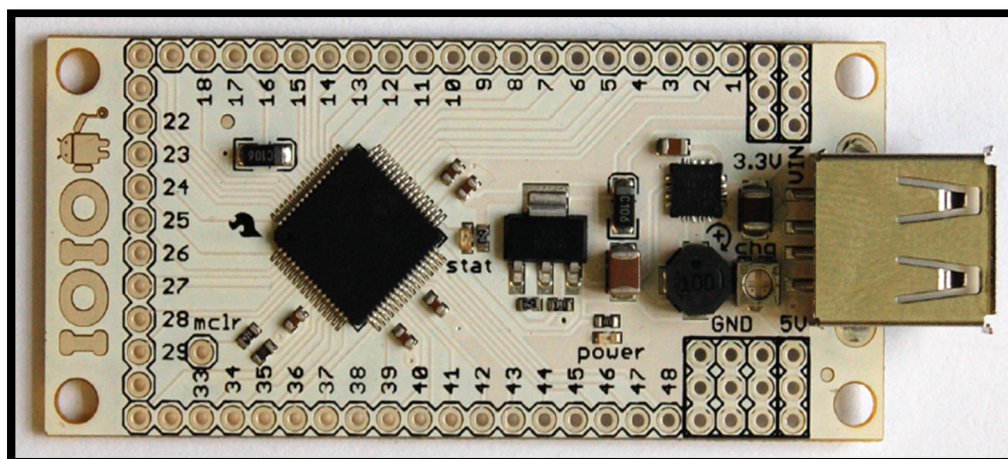


Figura 1. Placa IOIO

Referencia: Monk Simon, (2012). Making Android Accessories with IOIO

<sup>1</sup> PWM: Pulse Width Modulation.

<sup>2</sup> I<sup>2</sup>C: Inter-Integrated Circuit.

<sup>3</sup> SPI: Serial Peripheral Interface.

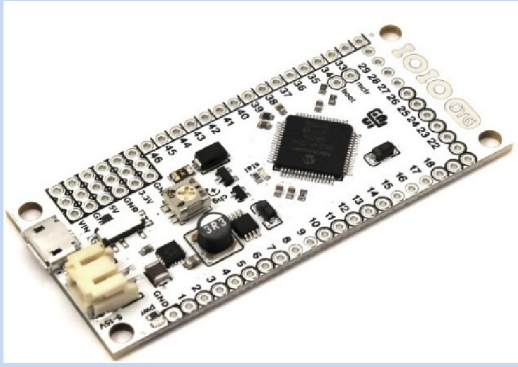

<sup>4</sup> UART: Universal Asynchronous Receiver-Transmitter.

<sup>5</sup> GPS: Global Positioning System.

## 1.2 VERSIONES DE LA PLACA IOIO

Existen dos versiones de la IOIO que son la versión OTG y la versión V1, las dos versiones cuentan con los mismos módulos, las mismas especificaciones de alimentación y corriente; la diferencia está en su estructura física, la distribución de sus pines, y los conectores que cuenta cada una de ellas.

**Tabla 1.** Versiones de la placa IOIO

IOIO OTG	IOIO V1
Posee un conector micro usb tipo A hembra	Cuenta con un conector usb tipo A hembra
Puede trabajar como un dispositivo conectado al pc, o como un host conectado al móvil Android	Trabaja como un host conectado al dispositivo Android
Posee un conector específico de alimentación	La alimentación se la realiza a través de sus pines.
	

**Referencia:** Elaborado por Byron Valenzuela, basado en el libro Making Android Accessories with IOIO

Para el desarrollo de este trabajo, se escogerá la placa IOIO OTG ya que permite trabajar como un host conectado al móvil Android o como un dispositivo conectado al PC y además la versión V1 de la IOIO actualmente se ha dejado de fabricar.



- Pines GND (9 pines).- Pines de conexión a tierra.
- Pines Vin (3 pines).- Pines de alimentación de la IOIO, el voltaje suministrado va desde 5V a 15V.
- Pines 5V (3 pines).- Se usa como salida de 5V; también puede ser usado como alimentación de 5V.
- Pines 3.3V (3 pines).- Son pines de salida de 3.3V.
- Pin Mclr <sup>8</sup>(1 pin).- Este pin solo se utiliza para la actualización de firmware.
- Power led.- Led que se enciende cuando la IOIO es alimentada.
- Stat led.- Led que se enciende cuando la IOIO se conecta con el dispositivo móvil.
- Charge current trimer (CHG).- Sirve para ajustar la cantidad de corriente que suministra a la línea VBUS del USB.
- Pines de entrada y salida.- Los pines de entrada y salida de la IOIO, son usados para conectar circuitos externos a las distintas interfaces de la placa, todos los pines de la placa IOIO pueden ser usados como entradas y salidas digitales de 3.3V
  - Pines rodeados por un cuadrado, pueden ser usados como entradas analógicas de 3.3V
  - Pines rodeados con un círculo, pueden ser usados como entradas y salidas digitales tolerantes a 5V.

### 1.3.2 PINES DE LA PLACA IOIO

A continuación se detalla los pines de la placa IOIO OTG (GitHub, 2008).

**Tabla 2.** Pines de la IOIO

IOIO Pin	A/D	I2C	PPSi	PPSo	5v	Comp.	Prog	Pic Pin	Función
1		DA1	yes	yes	yes			31	SDA2/RP10/GD4/CN17/RF4
2		CL1	yes	yes	yes			32	SCL2/RP17/GD5/CN18/RF5
3			yes	yes	yes			42	RTCC/DMLN/RP2/CN53/RD8
4		DA0	yes	yes	yes			43	DPLN/SDA1/RP4/GD8/CN54/RD9
5		CL0	yes	yes	yes			44	SCL1/RP3/GD6/CN55/RD10
6			yes	yes	yes			45	RP12/GD7/CN56/RD11
7			yes	yes	yes			46	DMH/RP11/INT0/CN49/RD0
8						3D		47	SOSCI/C3IND/CN1/RC13

<sup>8</sup> Mclr: Master Clear

IOIO Pin	A/D	I2C	PPSi	PPSo	5v	Comp.	Prog	Pic Pin	Función
9			yes			3C		48	SOSCO/SCLKI/T1CK/C3INC/RPI37/CN0/RC14
10			yes	yes	yes			49	VCPCON/RP24/GD9/VBUSCHG/CN50/RD1
11			yes	yes	yes			50	DPH/RP23/CN51/RD2
12			yes	yes	yes			51	RP22/GEN/CN52/RD3
13			yes	yes	yes			52	RP25/GCLK/CN13/RD4
14			yes	yes	yes			53	RP20/GPWR/CN14/RD5
15						3B		54	C3INB/CN15/RD6
16						3A		55	C3INA/SESEND/CN16/RD7
17								58	GD10/VBUSST/VCMPST1/VBUSVLD/CN68/RF0
18					yes			59	GD11/VCMPST2/SESSVLD/CN69/RF1
19					yes			60	GD0/CN58/RE0
20					yes			61	GD1/CN59/RE1
21					yes			62	GD2/CN60/RE2
22					yes			63	GD3/CN61/RE3
23					yes			64	HSYNC/CN62/RE4
24					yes			1	VSYNC/CN63/RE5
25		CL2			yes			2	GD12/SCL3/CN64/RE6
26		DA2			yes			3	GD13/SDA3/CN65/RE7
27			yes	yes		1D		4	C1IND/RP21/CN8/RG6
28			yes	yes		1C		5	C1INC/RP26/CN9/RG7
29			yes	yes		2D		6	C2IND/RP19/GD14/CN10/RG8
30			yes	yes		2C		8	C2INC/RP27/GD15/CN11/RG9
31	yes		yes	yes		1A	C3	11	PGEC3/AN5/C1INA/VBUSON/RP18/CN7/RB5

IOIO Pin	A/D	I2C	PPSi	PPSo	5v	Comp.	Prog	Pic Pin	Función
32	yes		yes	yes		1B	D3	12	PGED3/AN4/C1INB/USBOEN/RP28/CN6/RB4
33	yes					2A		13	AN3/C2INA/VPIO/CN5/RB3
34	yes		yes	yes		2B		14	AN2/C2INB/VMIO/RP13/CN4/RB2
35	ref(+)		yes	yes			C1	15	PGEC1/AN1/VREF-/RP1/CN3/RB1
36	ref(-)		yes	yes			D1	16	PGED1/AN0/VREF+/RP0/CN2/RB0
37	yes		yes	yes			C2	17	PGEC2/AN6/RP6/CN24/RB6
38	yes		yes	yes			D2	18	PGED2/AN7/RP7/RCV/CN25/RB7
39	yes		yes	yes				21	AN8/RP8/CN26/RB8
40	yes		yes	yes				22	AN9/RP9/CN27/RB9
41	yes							23	TMS/CVREF/AN10/CN28/RB10
42	yes							24	TDO/AN11/CN29/RB11
43	yes							27	TCK/AN12/CTEDG2/CN30/RB12
44	yes							28	TDI/AN13/CTEDG1/CN31/RB13
45	yes		yes	yes				29	AN14/CTPLS/RP14/CN32/RB14
46	yes		yes	yes				30	AN15/RP29/REFO/CN12/RB15
stat led			yes	yes	yes			39	RP16/USBID/CN71/RF3
Mclr							Vpp	7	MCLR

Referencia: GitHub, (2012). Getting To Know The Board. Recuperado de <https://github.com/ytai/ioio/wiki/Getting-To-Know-The-Board>

Leyenda.

- A/D.- Pines que pueden ser usados como entadas analógicas.
- I2C.- Pines que forman parte del módulo I2C. *DAx* Pin de Datos para el módulo x; *CLx* pin de reloj para el módulo x.



- PPSi (Peripheral Pin Select Input). - Pines que pueden ser usados como pines de entrada para los módulos UART, PWM y SPI.
- PPSo (Peripheral Pin Select Output). - Pines que pueden ser usados como pines de salida para para los módulos UART, PWM y SPI.
- 5V.- Pines que son tolerantes a 5V tanto como pines de entrada como de salida.
- Comp.- Pines que pueden ser utilizados como comparadores.
- Prog.- Pines a ser utilizados para motivos de programación del microcontrolador interno ICSP<sup>9</sup>.

## 1.4 DESCRIPCIÓN

### 1.4.1 FUENTE DE ALIMENTACIÓN

(GitHub, 2008) Menciona que la forma de alimentar a la IOIO, es suministrar voltaje entre 5V y 15V en cualquiera de los pines Vin y GND, al instante en que se alimenta la placa, se enciende el led power que indica que la misma fue alimentada. Además la IOIO es capaz de suministrar voltaje a través de los pines 5V, 3.3V y GND para alimentar cualquier circuito externo.

#### 1.4.1.1 Características de potencia

La IOIO cuenta con dos reguladores de voltaje (GitHub, 2008).

- Un regulador de conmutación que puede soportar entre 5V y 15V de entrada y de salida de hasta 1.5A a 5V estable.
- Un regulador lineal que se alimenta de la línea de 5V y brinda salidas de hasta 800mA a 3.3V estables.

Al momento de utilizar la IOIO como una fuente de alimentación de 5V el regulador de conmutación deja de funcionar; con esta fuente se puede obtener desde 800mA hasta 1A que puede servir para alimentar cualquier circuito o periférico externo.

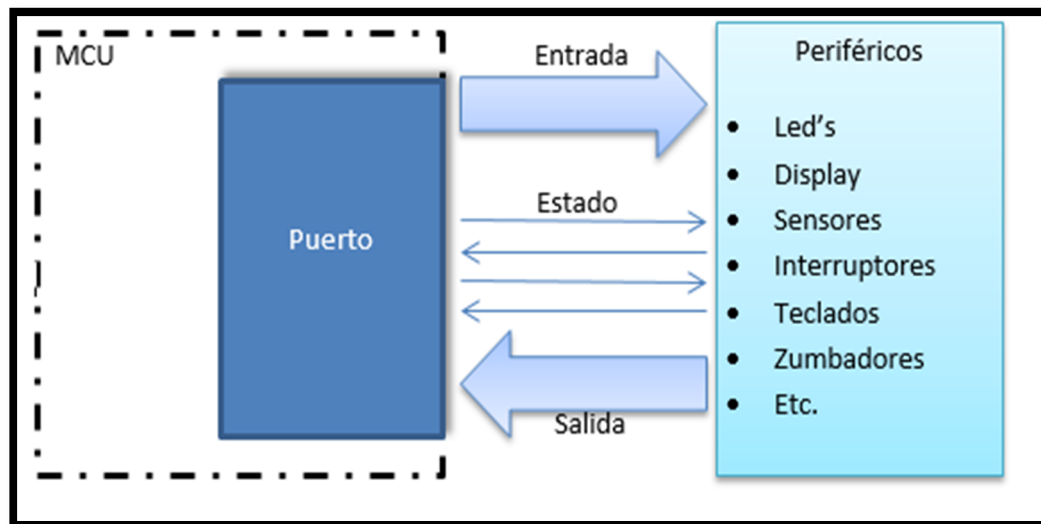
Para la alimentación del MCU<sup>10</sup> interno de la placa son necesarios 3.3V con una corriente de 30mA - 40mA dejando un poco más de 700mA para los pines externos de 3.3V.

---

<sup>9</sup> ICSP: In-Circuit Serial Programming

## 1.4.2 MÓDULO DE ENTRADA Y SALIDA

Los puertos de entrada y salida digitales son los más comunes con los que cualquier dispositivo electrónico cuenta, son un arreglo de circuitos lógicos que forma parte del microcontrolador, y permiten realizar transferencias de información e interactuar con cualquier periférico externo al mismo (Collaguazo, 2009).



**Figura 3.** Módulo de entrada y salida

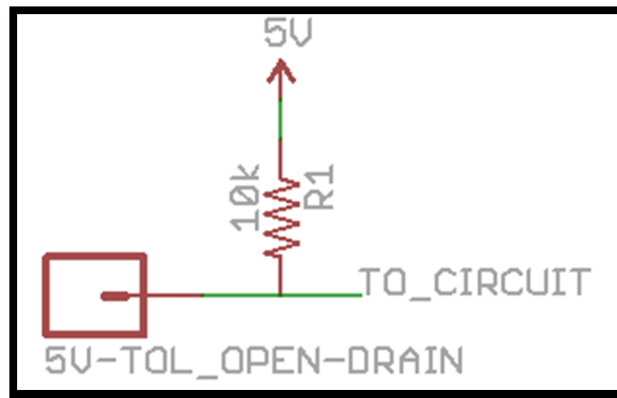
**Referencia:** Elaborado por Byron Valenzuela, basado en el libro Sistemas basados en microprocesadores.

### 1.4.2.1 Salida digital

Cuando se utiliza los pines de la IOIO como salida digital, estos tienen su propia manera de representar los niveles lógicos de salida, los representa de la siguiente manera (GitHub, 2008).

- Un cero lógico es decir un bajo lógico, es representado por medio de una tensión de 0V
- Un uno lógico es decir un alto lógico, es representado por medio una tensión de 3.3V
- La corriente máxima de salida recomendada es de 20mA.

<sup>10</sup> MCU: Microcontroller



**Figura 4.** Configuración de un pin open drain en modo pull up

**Referencia:** GitHub, (2012). Digital IO. Recuperado de <https://github.com/ytaioio/wiki/Digital-IO>

Si se necesita trabajar con la lógica digital TTL<sup>11</sup>, es decir para un uno lógico con un nivel de 5V y para un cero lógico con un nivel de 0V, se debe escoger uno de los pines tolerantes a 5V trabajar con el modo de operación open-drain y configurarlo con una resistencia pull up, así se está trabando en la lógica mencionada. Con esto un cero lógico será representado por una tensión de 0V y un uno lógico será representado por la fuente que esté conectado la resistencia pull up en este caso 5V.

#### 1.4.2.2 Entrada Digital

Cuando se va a utilizar los pines como entrada digital, no forzar a ningún voltaje desconocido en los mismos sino más bien se debe conocer el valor del voltaje con el que se está alimentando.

- La corriente máxima de entrada recomendada es de 20mA.
- Cuando se detecta 3.3V, el pin lo lee como un uno lógico. De igual manera los pines tolerantes a 5V, cuando detectan 5V lo leen como un uno lógico.
- Cuando no se está alimentado bajo ninguno valor el pin no influirá en el comportamiento de la placa.

#### 1.4.3 CONVERTOR ANÁLOGO DIGITAL

El módulo conversor análogo digital de la placa IOIO tiene las siguientes características (Microchip, 2010); (GitHub, 2008).

- 16 pines que pueden ser usados como entradas analógicas.

<sup>11</sup> TTL: Transistor-Transistor Logic

- Posee un conversor de 10 bits.
- Posee una velocidad de conversión superior a 500 ksp<sup>12</sup>.
- Utilizan el método de aproximación sucesiva para la conversión.
- Son capaces de medir voltaje entre 0V y 3.3V con una precisión de 3mV.
- Tiene pines externos para establecer un voltaje de referencia.

Se debe asegurar de no suministrar niveles de tensión que estén fuera del rango de lectura esto podría ocasionar que la placa IOIO pueda sufrir averías. Cuando los voltajes que se desea medir sobrepasan los niveles permitidos, se deberá utilizar circuitos reductores o amplificadores según sea el caso necesario.

#### 1.4.4 MÓDULO PWM

(GitHub, 2008) Asegura que la placa IOIO puede usar los pines que forman parte de los pines “PPSo”, para ser utilizados como salidas PWM. Además permite generar hasta 9 salidas simultáneas *PWM* con un nivel de 0V a 3.3V.

#### 1.4.5 MÓDULO UART

La placa IOIO cuenta con 4 módulos Universal Asynchronous Receiver Transmitter (UART) que tienen las siguientes características (GitHub, 2008; Microchip, 2010).

- Los niveles de voltaje que maneja son de 0V para un cero lógico y 3.3V para un uno lógico.
- Transmisor y receptor asíncrono.
- Modo de transmisión full-dúplex asíncrona.
- Los 4 módulos pueden hacer uso de cualquiera de los pines que forman parte de PPSi y PPSo para transmisión y recepción respectivamente.

#### 1.4.6 MÓDULO I2C

El módulo Inter Integrated Circuit (I2C), es una interfaz de dos hilos (TWI<sup>13</sup>) que sirve para la comunicación serial half-duplex entre circuitos integrados, el módulo I2C de la IOIO cuenta con las siguientes características (Fernández, 2004; GitHub, 2008).

- Cuenta con tres módulos I2C.
- Cuenta con pines específicos para hacer uso del módulo.

---

<sup>12</sup> KSPS: Kilo Samples Per Second

<sup>13</sup> TWI: Two Wired Interface

- El módulo I2C utiliza dos líneas que son SDA (datos) y SCL (reloj), maneja la lógica de 3.3V.
- Permite direcciones de 7 y 10 bits.
- Admite frecuencias de 100Khz, 400Khz o 1Mhz.
- Trabaja con dos niveles de voltaje SmBus <sup>14</sup> (0.8 y 2.1 V) y I2C (admite hasta 5V).

## 1.5 CONEXIÓN DE LA PLACA IOIO

La IOIO integra un firmware que permite la comunicación entre el dispositivo móvil android y la placa, este firmware se divide en dos programas los dos son grabados en la memoria flash del microcontrolador, los programas son “aplicación” y “gestor de arranque (bootloader)”.

El gestor de arranque es el primer programa que se ejecuta cuando la IOIO se reinicia este firmware establece la comunicación entre la placa y el móvil Android, comprueba si existe el código escrito de la aplicación si es así esta lo ejecuta caso contrario permanece en espera. Aplicación este firmware se comunica con las diferentes librerías que contiene IOIOLib para el control de los pines y sus módulos internos.

Tanto la comunicación por cable como por Bluetooth crea un socket para la transferencia de información y comandos entre el dispositivo Android y el tablero IOIO.

### 1.5.1 CONEXIÓN MEDIANTE CABLE USB

La placa IOIO suministra 5V al dispositivo Android a través de la conexión USB, esto permite que el móvil sea reconocido como un host y que posibilite la comunicación con la IOIO.



**Figura 5.** Conexión de la placa IOIO con el dispositivo Android mediante el cable USB

**Referencia:** Monk Simon, (2012). Making Android Accessories with IOIO

<sup>14</sup> SMBus: System Management Bus (Sistema de gestión de bus)

Para conseguir que el Smartphone Android sea reconocido al momento de conectarlo, se debe ajustar con un desarmador pequeño la cantidad de corriente en CHG (charge current trimer) que la IOIO suministrará al equipo móvil, hasta que en el dispositivo móvil reconozca que está conectado; en ese momento se logró vincular con éxito la IOIO con el Smartphone Android (GitHub, 2008).

### 1.5.2 CONEXIÓN MEDIANTE BLUETOOTH

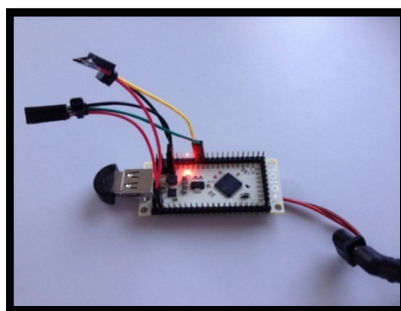
(GitHub, 2008) Establece que la IOIO se puede comunicar con el dispositivo móvil Android mediante la conexión de un dispositivo Bluetooth USB (figura 6). Esto se consigue fácilmente conectando el Bluetooth a la placa y la comunicación con el móvil se lo realizará mediante el Bluetooth del mismo. (Véase ANEXO 1).



**Figura 6.** Dispositivo Bluetooth USB

**Referencia:** Dispositivo Bluetooth recuperado de <http://www.maxmax.es/puerto-bluetooth>

La comunicación con el dispositivo Bluetooth y la placa es posible a partir del Bootloader v3.0 y un firmware v3.10, si la IOIO cuenta con un firmware de versión anterior a esta, no podrá comunicarse por medio del Bluetooth, para esto la solución es actualizar la versión de firmware a una versión que sí soporte la comunicación Bluetooth.



**Figura 7.** Conexión de la IOIO mediante Bluetooth

**Referencia:** Conexión de la IOIO mediante Bluetooth recuperado de <http://magdaaproject.org/2012/09/>

La conexión mediante Bluetooth solo admite que un dispositivo móvil controle una IOIO al mismo tiempo; el soporte Multi-IOIO aún está en desarrollo. Otros aspectos muy importantes en este tipo de conexión son: la latencia que aumenta considerablemente alrededor de los 10ms y el ancho de banda que baja notablemente a los 10KB/s frente a una conexión vía cable USB; dos inconvenientes que pueden ser notables al ejecutar una aplicación en la cual se haga uso del módulo de conversión analógica.

## 1.6 APLICACIONES REALIZADAS CON LA PLACA IOIO

Gracias a la aceptación que el sistema operativo móvil Android ha tenido en los últimos años, la placa IOIO ha tenido una gran acogida en desarrolladores que requieren un dispositivo que permita interactuar un celular con sistema operativo Android con una variedad de periféricos electrónicos.

Es así que existen un sin número de aplicaciones con esta placa, sacando el mejor provecho al hacer uso de todos los periféricos con los que cuenta, a continuación se enumeran algunas de las varias aplicaciones que se puede encontrar en la red.

- **IOIO Rover** (RobotFreak, 2008).



**Figura 8.** IOIO Rover

**Referencia:** RobotFreak. (2008). IOIO-Rover. Recuperado de <http://letsmakerobots.com/node/33968>

El IOIO Rover es un robot espía ganador del tercer lugar en el Campus Party en la sección "Best of Show" de software en Berlín, navega de forma autónoma, utiliza varios sensores y dispositivos con los que el teléfono cuenta, tiene cuatro modos de funcionamiento que son:

- Modo libre.- Este modo permite mostrar la distancia que un objeto se encuentra del robot y lo muestra en la pantalla.
- Modo camaleón.- En este modo el robot detecta un obstáculo y cambia el color del robot con el color del objeto que se ha detectado.
- Modo paparazzi.- IOIO Rover toma fotografías automáticamente cuando detecta un rostro.
- Modo acosador.- Este modo hace uso de los sensores ultrasonido con los que cuenta para perseguir a un objeto y mantenerse siempre a una distancia constante.
- **Alarma de intrusos** (Monk, 2012).



**Figura 9.** Alarma de Intrusos

**Referencia:** Monk Simon, (2012). Making Android Accessories with IOIO.

En este proyecto se hace uso de un sensor de movimiento, el cual activará una alarma, esto hará que la aplicación envíe un mensaje de texto a un número que el usuario desee, alertando que se activó la misma.



## **CAPÍTULO II**

### **2 ESTUDIO DEL SDK DE ANDROID**

#### **2.1 SISTEMA OPERATIVO ANDROID**

(Xatakandroid, 2005) Define a Android como un sistema operativo para dispositivos móviles al igual que iOS, Symbian, o Blackberry OS, desarrollado inicialmente por Android Inc. en 2003 y comprada posteriormente por Google en 2005, basado en el kernel de Linux 2.6.0, que es el encargado de gestionar aspectos como seguridad, manejo de memoria, procesos, networking y modelo de controladores (Drivers).

Android permite desarrollar aplicaciones mediante el SDK (Software Development Kit), aplicaciones que puedan sacar provecho a las diversas funciones del terminal móvil tales como: GPS, cámara, contactos, acelerómetro, llamadas, etc.

El lenguaje de programación utilizado para la creación de aplicaciones Android es Java, junto al kernel de Linux trabaja Dalvik, es una máquina virtual Java que se ejecuta por encima del núcleo del sistema y permite la ejecución las aplicaciones.

##### **2.1.1 CARACTERÍSTICAS DEL SISTEMA OPERATIVO ANDROID**

Android cuenta con las siguientes características (Zambrano, 2011).

- Navegador integrado: Basado en el motor de open Source Webkit.
- SQLite: Para almacenamiento de datos estructurado
- Multimedia: Soporta varios formatos de archivos multimedia tales como MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF.
- Máquina virtual Dalvik.
- Bluetooth, EDGE<sup>15</sup>, 3G, Wi-Fi (depende del hardware).
- Cámara, GPS, brújula, acelerómetro, táctil (depende del hardware).

---

<sup>15</sup> EDGE: Enhanced Data Rates for GSM Evolution (Estandar de telecomunicaciones).

## 2.1.2 FUNDAMENTOS PARA EL DESARROLLO DE APLICACIONES EN EL SISTEMA OPERATIVO ANDROID

Las aplicaciones Android están escritas en el lenguaje de programación Java. El código compilado de Java, junto con todos los archivos que forman parte de la aplicación desarrollada luego de ser compilada, forman un paquete con la extensión .apk <sup>16</sup> que es la que se instalará en el terminal móvil Android.

Los requerimientos necesarios tanto en hardware y software para comenzar a desarrollar aplicaciones son los siguientes (Pérchon, 2012; Argüello, 2012).

### Hardware

- Se requiere un total de 946 Mbytes de espacio en el disco duro para la instalación del entorno de desarrollo Android, distribuido de la siguiente manera (Pérchon, 2012; Argüello, 2012).

**Tabla 3.** Requerimientos de espacio en disco duro para el Android SDK

Componente	Tamaño aproximado
SDK Android	560 Mbytes
Eclipse IDE <sup>17</sup>	300Mbytes
JDK	85Mbytes

**Referencia:** (Argüello, 2012)

- 1Gb de Memoria RAM<sup>18</sup>, procesador doble núcleo, y puertos USB. Estos requerimientos son necesarios para la ejecución del AVD (Android Virtual Device) que emula un dispositivo Android y necesita un mínimo de 256 MBytes de memoria RAM, procesador doble núcleo para la ejecución simultánea de Eclipse IDE y AVD, y puertos USB para la comunicación con el dispositivo Android (Pérchon, 2012; Argüello, 2012).

<sup>16</sup> APK: Application Package File (Paquete que contiene una aplicación Android).

<sup>17</sup> IDE: Integrated Development Environment

<sup>18</sup> RAM: Random Access Memory (Memoria de acceso aleatorio).

- **Software.**

- Sistema operativo Windows XP (32) bits, Windows Vista, Windows 7 ó Windows 8 de 32 o 64 bits.
- Sistema operativo Mac OS X 10.5.8 o superior (x86 únicamente).
- Sistema operativo Linux en sus diferentes distribuciones (Ubuntu, Dapper y Drake), las distribuciones de 64 bits deben ser capaces de ejecutar aplicaciones de 32 bits.
- JDK (Java Development Kit) de Java 5 o superior.
- Se puede utilizar el entorno de desarrollo a conveniencia entre Eclipse, Netbeans y Apache. Se escogió Eclipse por ser el entorno recomendado por la página oficial de Android Developers.

## 2.2 SDK DE ANDROID

El contenido de esta sección 2.3, se basa en el trabajo creado y compartido por the Android Open Source Project y usado de acuerdo a los términos descritos en Creative Commons 2.5 Attribution License (<http://developer.android.com/intl/es/tools/help/sdk-manager.html>).

Android SDK provee un sin número de herramientas y bibliotecas API<sup>19</sup> necesarias para la creación y depuración de las aplicaciones Android; se compone de diversos paquetes, a continuación se describe cada uno ellos.

- **SDK Tools.**

Este paquete contiene las herramientas necesarias para la depuración y pruebas de la aplicación Android, es muy importante tener este paquete siempre actualizado, está ubicado en <SDK>/tools/.

- **Documentation.**

Una copia de la última documentación para la plataforma Android APIs, se encuentra en <SDK>/docs/.

- **SDK platform tools.**

Son herramientas dependientes de la plataforma, necesarias para el desarrollo y depuración de la aplicación, se actualiza cuando aparece una nueva plataforma, está ubicado en <SDK>/platform-tools/.

---

<sup>19</sup> API: Application Programming Interface (Interfaz de programación de aplicaciones; servicios que el sistema operativo ofrece al programador).

- **SDK Platform.**

Existe una plataforma disponible para cada versión de Android, incluye un paquete android.jar <sup>20</sup> que contiene todas las librerías para la creación de la aplicación. Para construir la aplicación Android se debe declarar para que plataforma se va a crear la aplicación. Se encuentra en <SDK>/platforms/<android-version>/.

- **Sources for Android SDK.**

Es una copia del código fuente de la plataforma Android, es útil para recorrer el código mientras depura la aplicación. Se encuentra en <SDK> /sources/.

- **Samples for SDK.**

Aplicaciones de ejemplo que pueden servir de guía en la construcción de la aplicación Android. Ubicado en <SDK>/platforms/<android-version>/samples/.

- **Google API<sup>21</sup>s.**

Un SDK adicional que proporciona una plataforma que se puede utilizar para desarrollar una aplicación utilizando los APIs de Google, se utiliza los APIs de Google para crear aplicaciones que necesiten del servicio de localización.

- **Android Support.**

Es una biblioteca que se puede añadir a la aplicación y permite usar APIs que no estén en la plataforma estándar. <SDK>/extras/android/support/.

- **Google play billing.**

Proporciona las librerías que permiten integrar los servicios de facturación de la aplicación con Google play. <SDK>/extras/google/

- **Google play licencing.**

Proporciona las librerías y ejemplos que permiten llevar a cabo la verificación de la licencia de la aplicación, cuando se distribuya en Google play. <SDK>/extras/google/.

La herramienta de desarrollo advertirá si es necesario incluir algún paquete adicional mediante mensajes de depuración.

---

<sup>20</sup> JAR: Java Archive (Paquete que contiene una aplicación en el lenguaje Java)

<sup>21</sup> API: Application Programming Interface

## 2.2.2 HERRAMIENTAS SDK

Las herramientas SDK están incluidas en la instalación del paquete SDK, se actualizan automáticamente y son necesarias para desarrolladores de aplicaciones Android, a continuación se presenta un resumen de las herramientas más usadas.

**Tabla 4.** Herramientas SDK

Herramienta	Función
android	Permite administrar AVDs, proyectos, y componentes instalados del SDK
Dalvik Debug Monitor Server (ddms)	Permite depurar las aplicaciones Android
Android emulator	Emula un dispositivo Android basado en QEMU <sup>22</sup> , permite depurar, probar y ejecutar aplicaciones en tiempo real.
Hierarchy Viewer	Permite depurar y organizar la interfaz de usuario de la aplicación Android.
layoutopt	Permite analizar los diseños de la aplicación con el fin de optimizarlos para la eficiencia.
mksdcard	Permite crear una imagen de disco que puede ser utilizada con el emulador Android, para simular la presencia de una tarjeta de almacenamiento externa.
ProGuard	Optimiza el código escrito mediante la eliminación del código no utilizado.
sqlite3	Permite acceder a los archivos SQLite creados y utilizados por la aplicación Android.

**Referencia:** Tabla basada en el trabajo creado y compartido por Android Open Source Project y usado de acuerdo a los términos descritos en Creative Commons 2.5 Attribution License.

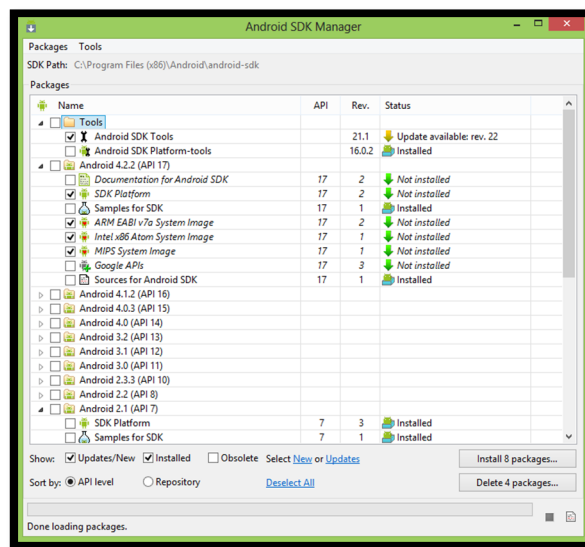
Recuperado de <http://developer.android.com/intl/es/tools/help/index.html#>.

<sup>22</sup> QEMU: Abreviatura de emulador rápido.

## 2.3 SDK MANAGER

El contenido de esta sección 2.3, se basa en el trabajo creado y compartido por the Android Open Source Project y usado de acuerdo a los términos descritos en Creative Commons 2.5 Attribution License (<http://developer.android.com/intl/es/tools/help/sdk-manager.html>).

El SDK manager muestra los paquetes SDK que necesitan actualización, que están disponibles, o que ya se encuentran instalados los ordena y los separa en herramientas, plataformas y otros componentes para facilitar la descarga de los mismos.



**Figura 10.** Pantalla principal Android SDK Manager

**Referencia:** Elaborado por Byron Valenzuela. Basado en el SDK Manager

Los paquetes que contiene el SDK manager son los siguientes:

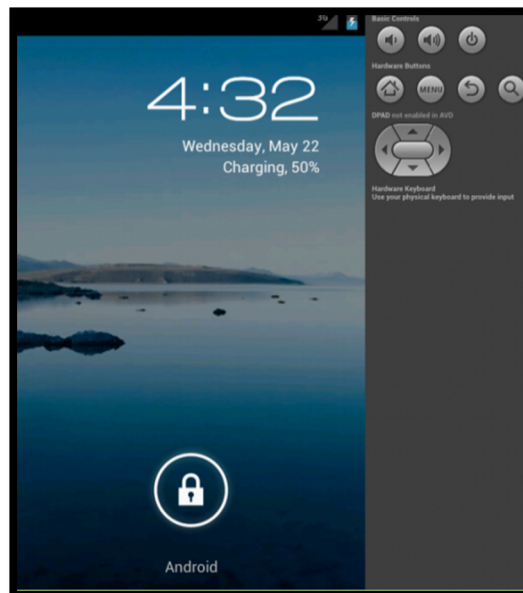
- Android SDK tools.- Paquete obligatorio, indica que existe una nueva versión SDK, se debe mantener actualizado.
- Android SDK Platform-tools.- Paquete obligatorio, este paquete se instala la primera vez que se instala el SDK.
- SDK Platform.- Paquete requerido, de acuerdo a las versiones de Android, existen las plataformas SDK, estas se debe descargar por lo menos una para compilar la aplicación, se recomienda utilizar la última plataforma para la construcción de la aplicación, ya que sí hay la posibilidad que una plataforma mayor pueda ejecutarse en una plataforma de menor versión.

- System Image.- Paquete recomendado, son archivos que contienen la imagen del sistema de un dispositivo Android, es poco probable que exista una imagen para cada versión de Android.
- Android Support.- Paquete recomendado, ofrece una biblioteca que permite utilizar APIs de Android en dispositivos que ejecutan versiones obsoletas de la plataforma Android.
- SDK Samples.- Paquete recomendado, brinda ejemplos con los que los desarrolladores Android pueden guiarse y crear aplicaciones, existen diferentes ejemplos para cada versión de Android.

## 2.4 ANDROID VIRTUAL DEVICE.

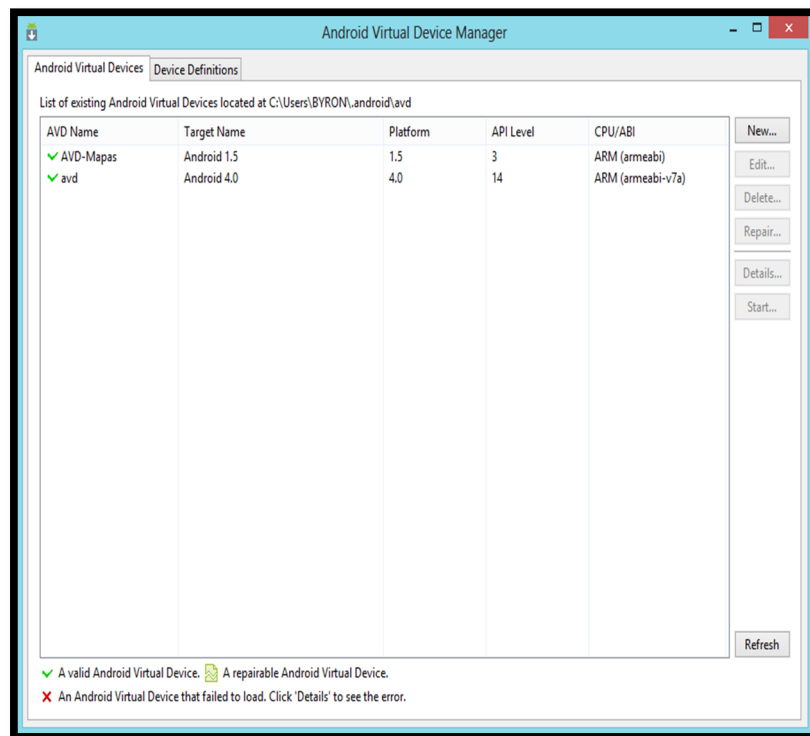
El contenido de esta sección 2.3, se basa en el trabajo creado y compartido por the Android Open Source Project y usado de acuerdo a los términos descritos en Creative Commons 2.5 Attribution License (<http://developer.android.com/intl/es/tools/help/sdk-manager.html>).

El SDK de Android incluye un emulador de dispositivo móvil virtual llamado Android Virtual Device (AVD), este imita todas las características de hardware y software de un dispositivo móvil real y permite comprobar el funcionamiento de las aplicaciones Android sin necesidad de contar con un equipo real.



**Figura 11.** Interfaz de un Dispositivo virtual Android (AVD)  
**Referencia:** Elaborado por Byron Valenzuela. Basado en el AVD.

La aplicación que administra las configuraciones de los dispositivos virtuales es el Android Virtual Device Manager, en esta aplicación se puede crear y editar AVD's, con características propias de un terminal real como: versión de Android, teclas, cámara, sonido, acelerómetro, tamaño de pantalla, sensores, etc.



**Figura 12.** Interfaz de la aplicación Android virtual device manager

**Referencia:** Elaborado por Byron Valenzuela. Basado en el AVD.

## 2.5 ANDROID DEVELOPER TOOLS

El contenido de esta sección 2.3, se basa en el trabajo creado y compartido por the Android Open Source Project y usado de acuerdo a los términos descritos en Creative Commons 2.5 Attribution License (<http://developer.android.com/intl/es/tools/help/sdk-manager.html>).

Android developer tools (ADT) es un plugin para Eclipse que brinda un conjunto de herramientas y ofrece acceso a características que ayudan a la creación de las aplicaciones Android. ADT brinda acceso a la Interfaz Gráfica de Usuario (GUI), como una herramienta de diseño y construcción de la interfaz de usuario que tendrá la aplicación Android, a continuación se tiene las características del ADT.



- ADT se integra a Eclipse para la creación, construcción, instalación y depuración del proyecto Android.
- ADT posee muchas de las herramientas propias de SDK Android.
- Lenguaje de programación JAVA, que incluye características como comprobación de sintaxis en tiempo de compilación, y autocompletado.
- Permite crear y editar archivos XML en una interfaz basada en formularios.

Muchas de las herramientas de Android SDK están incluidas en el ADT; tales como: Traceview, android, Dalvik Debug Monitor Server (ddms), Hierarchy Viewer, Pixel Perfect, ProGuard, etc.

### 2.5.1 EDITORES DE CÓDIGO

ADT ofrece opciones de edición de archivos XML<sup>23</sup>, para facilitar la creación y edición de archivos Manifest de Android, recursos y menús dentro del proyecto.

**Tabla 5.** Editores de códigos XML

Editor	Función
Graphical Layout Editor	Edita los archivos de diseño XML de la interfaz.
Android Manifest Editor	Edita el archivo AndroidManifest.xml
Menu Editor	Permite crear y editar un menú para la aplicación Android.
Resources Editor	Android admite recursos como imágenes, animaciones, etc.
XML Resources Editor	Permite editar el archivo XML de los recursos.

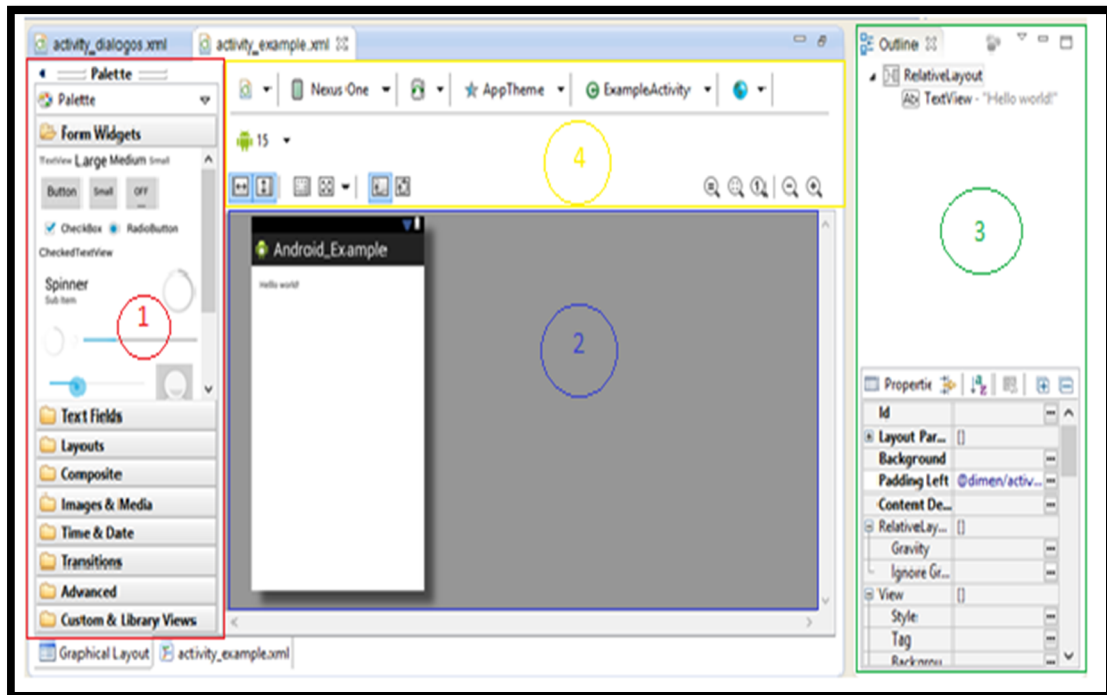
**Referencia:** Tabla basada en el trabajo creado y compartido por Android Open Source Project y usado de acuerdo a los términos descritos en Creative Commons 2.5 Attribution License.

Recuperado de <http://developer.android.com/intl/es/tools/help/adt.html>.

<sup>23</sup> XML: Extensible Markup Language

## 2.5.2 EDITOR DE DISEÑO GRÁFICO

Gracias a las características que ADT presenta, se puede crear y diseñar fácilmente la interfaz que la aplicación Android tendrá. El editor de diseño gráfico cuenta con las siguientes partes.



**Figura 13.** Interfaz del editor de diseño gráfico

**Referencia:** Elaborado por Byron Valenzuela. Basado en la interfaz de diseño gráfico de Eclipse.

1. En la parte izquierda del editor encontramos Palette (Paleta), contiene widgets que se pueden añadir al formulario de la aplicación Android, están ordenados de acuerdo a la función que realizan.
2. En la parte central del editor se encuentra la lona o lienzo de creación de la interfaz Android, aquí se pueden arrastrar y colar los widgets de la paleta que sean necesarios.
3. En la parte derecha del editor se encuentra la vista Outline (Esquema), que permite tener una visión de los widgets que forman parte del lienzo.
4. En la parte superior del editor se encuentra el selector de configuración, que ofrece opciones para modificar el lienzo según la pantalla del dispositivo.

## 2.6 ESTRUCTURA DE UN PROYECTO ANDROID

(Universidad Politécnica de Valencia, 2013) Indica que un proyecto Android se compone básicamente por un descriptor (AndroidManifest.xml), el código fuente en Java y una serie de ficheros con recursos. Cada elemento se almacena en una carpeta específica, como se puede observar en la figura 14

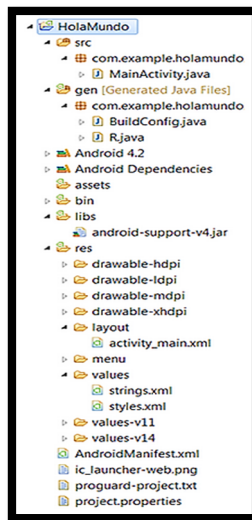


Figura 14. Contenido de un proyecto Android

**Referencia:** Recuperado de <http://www.androidcurso.com/index.php/recursos-didacticos/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/148-elementos-de-un-proyecto-android>

- **src**

(Universidad Politécnica de Valencia, 2013) Menciona que esta carpeta contiene todo el código de la aplicación, así como el paquete que acoge a las diferentes clases que componen el mismo. Eclipse crea automáticamente la primera clase correspondiente al primer formulario que tendrá el proyecto.

- **gen**

Esta carpeta contiene las clases generadas automáticamente por el SDK. No se debe modificar el código de estas clases (Universidad Politécnica de Valencia, 2013).

- BuildConfig.java.- Define la constante DEBUG utilizado para ejecutar el código en modo depuración.
- R.java.- Contiene la clase que hace referencia al ID de todos los recursos de la aplicación.

- **Android x.x.x**

Contiene el código .jar y el API de Android según la versión.

- **Android Dependencias**

Aquí se encuentran las librerías asociadas con el proyecto.

- **assets**

(Universidad Politécnica de Valencia, 2013) Se encuentra vacía por defecto, en esta carpeta se pueden colocar archivos o carpetas como: registros de datos, fuentes, ficheros .jar externos que serán usados en la aplicación.

- **bin**

(Universidad Politécnica de Valencia, 2013) Afirma que esta carpeta se compila la aplicación y se genera el .apk que se instalará en el dispositivo móvil Android.

- **libs**

Aquí encontramos las librerías .jar que se usa en el proyecto.

- **res**

Contiene los ficheros de recursos de la aplicación como: imágenes, animaciones, sonidos, texto, etc. Según el tipo de fichero se clasifican en subcarpetas. (Véase tabla 6).

**Tabla 6.** Subcarpetas de la carpeta res

Subcarpeta	Descripción
<b>/res/drawable/</b>	Se encuentran ficheros de imágenes e iconos según la densidad y resolución se dividen en subcarpetas <ul style="list-style-type: none"> <li>• /drawable-ldpi<sup>24</sup>.- Densidad baja.</li> <li>• /drawable-mdpi.- Densidad media.</li> <li>• /drawable-hdpi.- Densidad alta.</li> <li>• /drawable-xhdpi.- Densidad muy alta.</li> </ul>
<b>/res/layout/</b>	Contiene archivos XML de los formularios de la aplicación.
<b>/res/menú/</b>	Contiene ficheros XML de los menús de la aplicación.
<b>/res/values/</b>	Contiene ficheros XML de los recursos de la aplicación como: strings, colores y estilos.

**Referencia:** sgoliver.net.for. (n.d.). Estructura de un proyecto Android recuperado de <http://www.sgoliver.net/blog/?p=1278>

<sup>24</sup> DPI: Dot Per Inch (Puntos por pulgada).

- **AndroidManifest.xml**

(sgoliver.net foro, n.d.) Este fichero contiene información esencial acerca de la aplicación para el sistema Android, información necesaria para la ejecución del proyecto. El AndroidManifest contiene: el nombre de la aplicación, descripción de los componentes del proyecto, activity's, permisos necesarios que la aplicación utiliza, librerías requeridas, versión Android de ejecución, etc.

- **ic\_launcher-web.png**

Representa el icono del proyecto que es usado en aplicaciones web, el nombre puede variar dependiendo de la aplicación (Universidad Politécnica de Valencia, 2013).

- **proguard-project.txt**

Fichero de configuración de la herramienta ProGuard, permite optimizar el código generado (Universidad Politécnica de Valencia, 2013).

- **default.properties**

Este fichero es generado automáticamente por el SDK, es utilizado para comprobar la versión del API y otras características cuando se instala la aplicación en el dispositivo móvil. Este archivo no debe ser modificado (Universidad Politécnica de Valencia, 2013).

## 2.7 LIBRERÍAS DE LA PLACA IOIO

La placa IOIO tiene un acumulado de librerías que ayudarán a realizar las aplicaciones, estas librerías se denomina IOIOLib, IOIOLibPC, IOIOLibBT y IOIOLibAccessory, estas a su vez contienen un conjunto de clases y métodos que permiten controlar todas las características y funciones de la IOIO. Basta con descomprimir el archivo .jar que contiene las librerías e incluir en el proyecto a realizar y ya estará en la capacidad de utilizar esta librería. (GitHub, 2008).

- IOIOLib contiene las librerías que se utiliza para construir una aplicación que será ejecutada en el dispositivo móvil Android.
- IOIOLibPC acoge las librerías a utilizar para montar un proyecto que será ejecutada desde la PC.
- IOIOLibBT este paquete que contiene las librerías que posibilitarán la utilización del Bluetooth en la conexión con la IOIO. Esta librería se la puede utilizar desde la versión Android 2.x en adelante.

IOIOLibAccesory paquete donde se encuentran las librerías para conectar la IOIO en modo OpenAccesory. Disponible en versiones Android 2.3.4 o posteriores.

- Las librerías están organizadas en paquetes Java .jar que son:
- ioio.lib.api.- Este es el paquete que el proyecto va a utilizar, contiene las clases con los métodos que controlan la IOIO y que se usa en la aplicación.
- ioio.lib.api.exception.- Contiene algunas excepciones producidas por la IOIO.
- The ioio.lib.impl.- contiene la implementación de las clases y métodos de IOIOLib y IOIOPc.
- ioio.lib.util.- Este paquete acoge el framework de la IOIOLib, que ayudarán al desarrollo de los proyectos IOIO.
  - ioio.lib.util.android.- Contiene utilidades específicas, que posibilitarán el desarrollo de aplicaciones para dispositivos Android.
  - ioio.lib.util.pc.- De igual manera, contiene utilidades específicas, que posibilitan el desarrollo de aplicaciones para el computador.

El corazón de todas las aplicaciones IOIO se basa en realizar una instancia hacia el objeto **ioio** de la librería IOIOLib, que representa una interfaz entre la aplicación y la IOIO permitiendo el control de todas las funciones y módulos de la misma.

Todas las aplicaciones IOIO para Android se componen de las siguientes características y métodos (GitHub, 2008).

- La clase actual del proyecto se debe extender de la clase AbstractIOIOActivity o de la clase IOIOActivity, que permite hacer uso de los métodos que se encuentran en las librerías de la IOIO.
- Una clase propia extendida de la clase AbstractIOIOActivity.IOIOThread o de la clase BaseIOIOLooper en caso de usar IOIOActivity; en esta clase se aloja los métodos de la aplicación Android que controlan la conexión y uso de las diferentes funciones. Dentro de esta clase se tiene los siguientes métodos.
  - Método setup(); en este método se inicializa y configura las características que van a ser usadas en la aplicación. Ej. Se configura el pin a ser usado y el estado que tendrá inicialmente, se configura los pines que actuarán con el módulo de comunicación serial, la velocidad de transmisión, paridad, etc. Para crear una instancia hacia la clase IOIO se realiza mediante **ioio\_**.

- Método `loop()`; dentro de este método se desarrolla la aplicación Android una vez establecida la conexión, es un bucle infinito donde se controla las funciones y características de la placa.

Tanto el método `setup()` y el método `loop()` permiten lanzar excepciones `ConnectionLostException` e `InterruptedException`.

Cada vez que se ejecuta una aplicación IOIO esta determina automáticamente la forma de conexión ya sea por cable USB o por Bluetooth, si la conexión desafortunadamente falla inmediatamente la aplicación llama a un método `incompatible()` que verifica la causa y la solución del problema, en caso de no encontrar el posible error, la aplicación llama a un método `disconnected()` que hace que la aplicación IOIO sea abortada.

Las aplicaciones IOIO Android necesitan de permisos para su posible ejecución, estos permisos se incluyen dentro del desarrollo de la Aplicación en el `AndroidManifest` del proyecto y son los siguientes: Bluetooth (`android.permission.BLUETOOTH`) e Internet (`android.permission.INTERNET`) (GitHub, 2008).

Adicionalmente el paquete `IOIOlib` contiene una aplicación que servirá de ejemplo para el desarrollo de los nuevos proyectos.

## **2.7.1 MÓDULO DE ENTRADA Y SALIDA**

Como se mencionó en el capítulo anterior, los pines de la placa IOIO pueden ser usados como entrada y salida digital.

### **2.7.1.1 Salida digital**

Para utilizar un pin de la IOIO como salida digital, se debe usar la clase `DigitalOutput` del paquete `ioio.lib.api`. Para hacer uso de esta clase, se accede mediante un objeto de la clase `DigitalOutput` declarado en el proyecto a crear (GitHub, 2008).

El primer paso es declarar que pin va funcionar como salida digital, mediante el método `openDigitalOutput`; Esto hace que el pin funcione como salida digital a 3.3V. Para tener una salida a 5V se debe seleccionar un pin compatible y de igual forma declararlo como salida en modo colector abierto

```
DigitalOutput out = ioio.openDigitalOutput(pinNum) // declaración de un pin a 3.3V,  
pinNum indica el número del pin a ser usado como salida.
```

```
DigitalOutput out =  
ioio.openDigitalOutput(pinNum, DigitalOutput.Spec.Mode.OPEN_DRAIN, StartValue) //  
declaración de un pin a 5V, startValue indica el estado inicial del pin, HIGH 5V, LOW 0V.
```

Para hacer uso del Led stat que tiene la placa IOIO, se lo hace mediante la variable LED\_PIN.

```
DigitalOutput led = ioio.openDigitalOutput(IOIO.LED_PIN); // declaración del led stat de la  
IOIO como salida.
```

Una vez declarado el pin que se va usar como salida digital, se puede determinar el valor lógico que tendrá el pin entre alto y bajo; mediante el método write.

```
out.write (False) // pin con salida en nivel bajo 0V.  
out.write (True) // pin con salida a nivel alto 3.3V o 5V.
```

Para cerrar la conexión con el pin IOIO luego de hacer uso, se lo realiza mediante el método close.

```
out.close() // cerrar la conexión con el pin.
```

### **2.7.1.2 Entrada digital**

A través de la clase DigitalInput del paquete ioio.lib.api, se puede usar un pin IOIO como entrada digital, de igual manera se debe acceder a la clase mediante un objeto de la clase DigitalInput (GitHub, 2008).

El proceso es semejante a salida digital, el primer paso es indicar que pin de la IOIO se va usar como entrada digital e indicar si va funcionar en modo normal compatible con 3.3V o en modo pull-up compatible con 5V.

```
DigitalInput in = ioio.openDigitalInput(pinNum) // declaración de un pin compatible con  
3.3V de entrada, pinNum indica el número del pin a ser usado como entrada.
```



```
DigitalInput in = ioio.openDigitalInput(pinNum, DigitalInput.Spec.Mode.PULL_UP) //
declaración de un pin compatible con 5V de entrada.
```

Luego de haber declarado que pin va a funcionar como entrada digital, se puede leer el valor lógico que tiene dicho pin, mediante una variable de tipo booleana, el método read retornara un true o un false; true para un nivel alto y false para un nivel de voltaje bajo.

```
boolean value = in.read() // lectura de un pin IOIO
```

Existe otro método que permite sondear el pin en espera de un valor determinado valor alto (true) o bajo (false).

```
in.waitForValue(true) // lectura de un pin en espera de un valor alto
```

De igual manera después de utilizar el pin se debe cerrar la conexión.

```
out.close() // cerrar la conexión con el pin.
```

## **2.7.2 CONVERSION ANÁLOGO DIGITAL**

Para hacer uso del lector análogo de la IOIO se lo hace a través de la clase AnalogInput del paquete ioio.lib.api, declarando un objeto de la clase AnalogInput.

Primeramente declarar el pin que va a ser usado como entrada analógica. Para esto se debe usar un pin que acepte ser usado como entrada analógica (GitHub, 2008).

```
AnalogInput in = ioio.openAnalogInput(pinNum) // declaración del pin que va a ser usado
como entrada analógica.
```

Una vez declarado el pin a usar, se puede dar lectura al nivel de voltaje que tiene el pin mediante una variable tipo flotante y el método read que retornará un valor entre 0 y 1, que cubre todo el rango de voltaje admisible por el módulo análogo de la IOIO

```
float value = in.read() // lee el valor analógico del pin.
```

Existe otro método que permite adquirir el valor más exacto del pin analógico, de igual manera se necesita una variable tipo flotante.

```
float volts = in.getVoltage() // recupera el valor de voltaje en el pin analógico.
```

Para dar lectura a datos que son muestreados en un intervalo de tiempo se debe declarar un espacio de memoria en el cual se almacenarán las muestras obtenidas. Para esto se usa el método `setbuffer`.

```
in.setBuffer(capacity) // crea un espacio de memoria para almacenar las muestras,
capacity determina el número de muestras que el buffer será capaz de almacenar.
```

Y el método que permite leer el buffer, es el método `readBuffered` o `getVoltageBuffered`, que retornará un valor entre 0 y 1. Además este método bloqueará la IOIO hasta que exista por lo menos una muestra en el buffer.

```
float sample = in.readBuffered()
float volts = in.getVoltageBuffered() // métodos que permiten dar lectura de las muestras
almacenadas en el buffer.
```

Si se desea conocer el número de muestras que están almacenadas en el buffer, se usa el método `available`.

```
int samples = in.available() // permite conocer cuantas muestras se encuentran
almacenadas en el buffer.
```

Un desbordamiento hará que el buffer se desborde, existe un método que permite conocer el número de muestras que se perdieron luego del desborde.

```
in.getOverflowCount() // retorna el número de muestras que se perdieron luego de un
desborde.
```

Para conocer la frecuencia de muestro se usa el método `SampleRate()`.

```
in.getSampleRate() // retorna la frecuencia de muestreo en Hz.
```

Después de utilizar el módulo análogo digital, se debe cerrar la conexión.

```
in.close() // cierra la conexión del pin analógico.
```

### 2.7.3 MÓDULO PWM

La clase PwmOutput del paquete ioio.lib.api ayuda a usar los pines de la IOIO como salida PWM. El primer paso es declarar que pin va ser usado como salida PWM, para esto se usa el método openPwmOutput.

```
PwmOutput pwm = ioio.openPwmOutput(pinNum, freq) // declaración del pin que va a ser usado como salida Pwm, pinNum define el número del pin PWM, freq asigna la frecuencia de trabajo PWM al pin.
```

```
PwmOutput pwm = ioio.openPwmOutput(new DigitalOutput.Spec(pinNum, Mode.OPEN_DRAIN), freq) // declaración de un pin que va a ser usado como salida PWM en modo colector abierto.
```

El ciclo de trabajo puede ser controlado por el método setDutyCycle, que recibe una variable de tipo flotante entre 0 y 1.

```
pwm.setDutyCycle(dc) // controla el ciclo de trabajo de acuerdo al valor que tenga la variable flotante dc.
```

El método setPulseWidth permite controlar el ancho del pulso directamente, este método recibe como parametro el valor del ancho de pulso en microsegundos.

```
pwm.setPulseWidth(pw) // define el valor del ancho de pulso de acuerdo a la variable pw.
```

Para cerrar la conexión Pwm se usa el método close.

```
pwm.close() // Cierra la conexión Pwm.
```

### 2.7.4 MÓDULO UART

Si se va usar el módulo Uart de la IOIO, se usa la clase Uart del paquete ioio.lib.api y para acceder a este se debe declarar un objeto de la clase Uart que permita hacer uso de sus métodos.

El primer paso para usar el Uart IOIO es declarar las características de la comunicación Uart, tales como pin transmisor, pin receptor, velocidad de transmisión, bits de paridad y bits de parada (GitHub, 2008).

`Uart uart = ioio.openUart(rxPin, txPin, baud, parity, stopBits) // declaración de las características de la comunicación Uart de la IOIO, rxPin define el pin receptor, txPin define el pin transmisor, baud define la velocidad de transmisión en baudios, parity define los bits de paridad, define los bits de parada.`

Posteriormente se puede acceder a los métodos que permitan recibir y transmitir datos, declarando un objeto de la clase `InputStream` y `OutputStream` respectivamente. A través de los cuales se accederá a los métodos de lectura y escritura de datos estándar de Java.

`InputStream in = uart.getInputStream() // declaración de un objeto de la clase InputStream.`

`OutputStream out = uart.getOutputStream() // declaración de un objeto de la clase OutputStream.`

Por último, al terminar de hacer uso del módulo `Uart` de la `IOIO` se debe cerrar la conexión.

`uart.close() // Cierra la conexión del Uart IOIO.`

### **2.7.5 MÓDULO I2C**

Para manipular el módulo `I2C` de la `IOIO`, se debe usar la clase `TwiMaster` del paquete `ioio.lib.api`. Para hacer uso de esta clase, se acced mediante un objeto de la clase `TwiMaster` declarado en el proyecto a crear (GitHub, 2008).

La forma de usar este módulo es idéntico al módulo `I2C`, el primer paso es abrir la conexión con el módulo.

`TwiMaster twi = ioio.openTwiMaster(twiNum, rate, smbus) // declaración de un objeto de la clase TwiMaster, twiNum define el pin a ser SDA, rate impone la frecuencia de reloj, smbus define los niveles de voltaje a manejar cuando se coloca true se está escogiendo los niveles SMBus cuando es false los niveles I2C.`

Se usa el método `writeread` de la clase `TwiMaster` para la transferencia de información.

`twi.writeRead(address, tenBitAddr, writeData, writeSize, readData, readSize) // envia información al dispositivo esclavo, address define la dirección del dispositivo esclavo, tenBitAddr utilizado para especificar una dirección de 10 bits, writeData información a enviar , writeSize define el tamaño de la información que se está enviando, readData define la variable tipo byte donde se alojará la respuesta, readSize tamaño del byte donde se alojará la información.`

Este método bloqueará la IOIO hasta conseguir una respuesta del esclavo, si no se desea que la IOIO se bloquee hasta recibir la información existe el método `writeReadAsync`.

`twi.writeReadAsync(address, tenBitAddr, writeData, writeSize, readData, readSize) // lee la información del enviada por el esclavo de forma asíncrona.`

Al terminar la sesión con I2C IOIO se debe cerrar la conexión.

`twi.close(); // Cierra la conexión del I2C IOIO.`

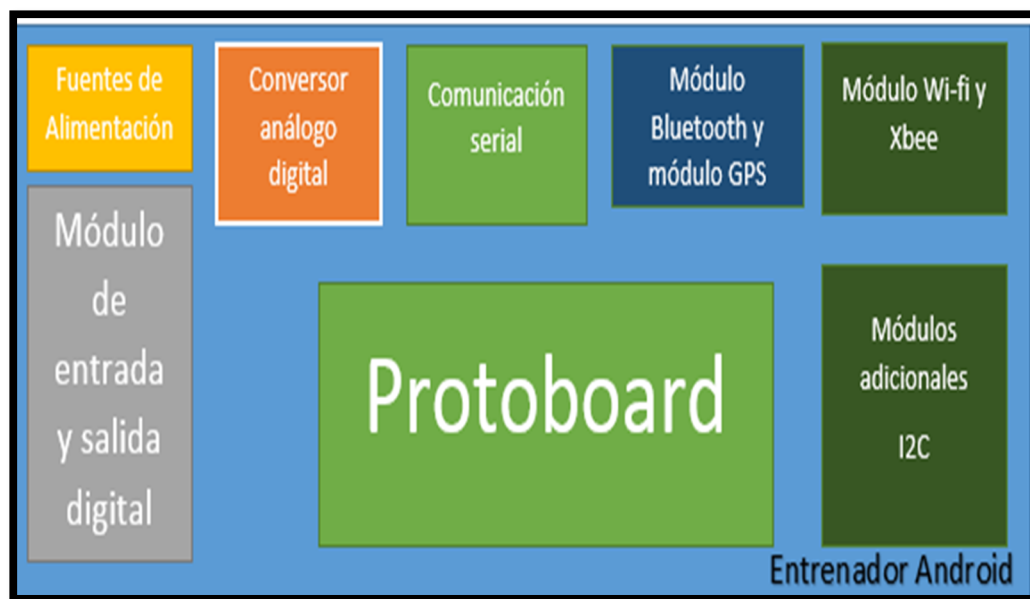
## CAPÍTULO III

### 3 DISEÑO Y CONSTRUCCIÓN DEL ENTRENADOR

#### 3.1 DESCRIPCIÓN GENERAL DEL ENTRENADOR

El entrenador se divide en 8 secciones, cada sección es un módulo del entrenador con la cual se realizará las prácticas, de igual manera cada sección consta de varios elementos electrónicos de acuerdo al módulo al que pertenezcan, además se incluye un protoboard con el fin de facilitar la conexión de dispositivos que no estén contemplados en el entrenador.

Los 8 bloques constitutivos del entrenador son: módulos: entrada y salida digital, conversión análogo-digital, comunicación serial, comunicación ZigBee y Wi-fi, comunicación Bluetooth y Gps, módulos adicionales, fuente de alimentación, y protoboard.



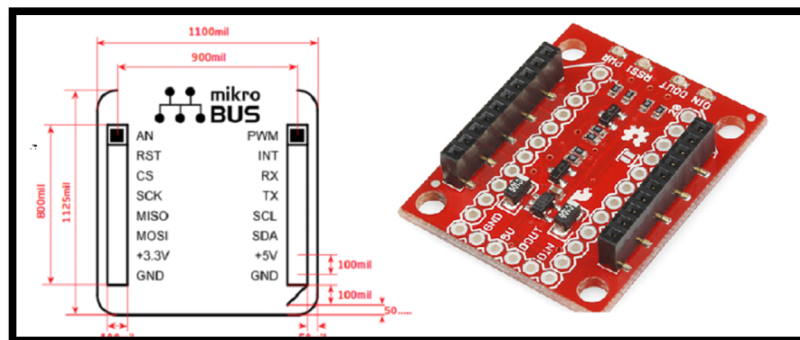
**Figura 15.** Descripción general del entrenador

**Referencia:** Elaborado por Byron Valenzuela.

Los módulos Bluetooth, y Gps, cuentan con el estándar mikroBus de miKroelectrónica, este estándar hace que los dispositivos anteriormente mencionados compartan el mismo socalo de conexión, eliminando así la conexión de accesorios adicionales para el funcionamiento de los equipos y la necesidad de realizar una interfaz por cada módulo que se requiera conectar.

El conector mikroBus consiste en 16 pines de conexión, entre los cuales tiene comunicación: SPI, UART, I2C, también tiene pines de PWM, interrupciones, entrada analógica, reset y chip select. Este sócalo es común en los módulos mencionados, y la configuración y uso es independiente de cada uno

De igual manera los módulos ZigBee y Wi-fi cuentan con un sócalo en común por el mismo hecho de tener las mismas dimensiones, el dispositivo que comparten es un regulador para equipos Xbee denominado Xbee Explorer, este dispositivo cuenta con 10 pines y es capaz de regular la alimentación y los niveles de voltaje de transmisión y recepción de 5V a 3.3V tanto del módulo ZigBee y del Wi-fi.



**Figura 16.** Conector mickorBus(i), Conector Xbee Explorer (d).

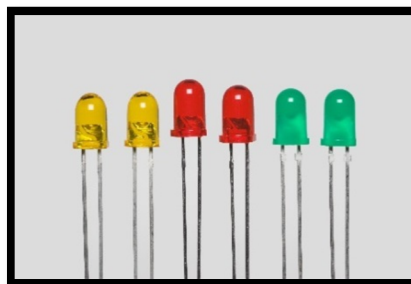
**Referencia:** ( mikroElektronika, 2012), (Sparkfun 2012).

## 3.2 MÓDULO DE ENTRADA Y SALIDA DIGITAL

Entrada y salida digital es el módulo que constará de varios elementos electrónicos como son.

### 3.2.1 SALIDA DIGITAL

Led's<sup>25</sup>.



**Figura 17.** Led's

**Referencia:** Led's. Recuperado de <http://hytare-p.blogspot.com/2010/08/clase-1-leds.html>

<sup>25</sup> LED: Light-Emitting Diode (Diodo Emisor de Luz)

El diodo emisor de luz (led) es un dispositivo semiconductor que emite luz cuando se polariza directamente la unión P-N, y circula por él una corriente comprendida entre los 10 y 50 mA y el voltaje desde 1.8V hasta los 5V (datasheet); los valores de corriente y voltaje dependen del color del led.

El entrenador cuenta con 8 diodos led 5mm de color rojo, que ayudan a identificar los niveles de voltaje de salida uno y cero lógico e interactuar con la placa IOIO.

La hoja de datos (datasheet) de los led's rojos especifican que los valores a trabajar para tener una buena iluminación son: corriente entre 10 y 20mA y un voltaje de diodo de 2.0 V; con estos datos se hace el respectivo cálculo de la resistencia que irá en serie con el led y la potencia del mismo.

$$I = \frac{V}{R} ; \text{donde } V = (V_{pin} - V_d) ; I = I_d$$

*despejando R se obtiene*

$$R = \frac{(V_{pin} - V_d)}{I_d} ; \text{ y la potencia es } Pot = V * I$$

$$R = \frac{(3.3 - 2.0)V}{10mA} ; Pot = 3.3V * 10mA$$

$$R = \frac{(1.3)V}{10mA} ; R = 130\Omega ; Pot = 33mW$$

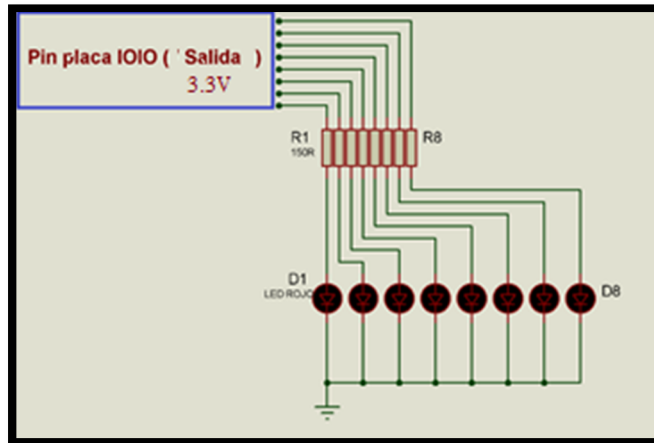
V<sub>pin</sub> (3.3V) = Voltaje que se obtendrá de los pines de la placa IOIO.

V<sub>d</sub> (2.0V) = Voltaje del diodo (especificado en las hoja de datos).

I<sub>d</sub> (10mA) = Corriente del diodo ((especificado en las hoja de datos).

El valor de la resistencia en serie es de 130Ω de 33mW; debido a que en el mercado no se encuentra una resistencia de ese valor, el entrenador posee una resistencia de 150 Ωde ¼ de wattio es decir trabaja a 250 mW; que además es compatible para trabajar con los pines tolerantes a 5V.





**Figura 18.** Esquema de conexión led's  
**Fuente:** Elaborado por Byron Valenzuela

## Buzzer



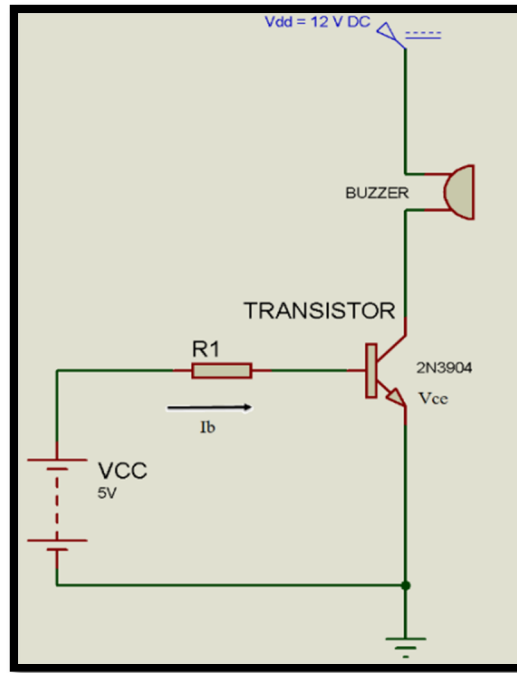
**Figura 19.** Buzzer electrónico

**Referencia:** Buzzer. Recuperado de: <http://www.electronicamagnabit.com/tienda/buzzer/765-buzzer-1-2v-12mm.html>

Buzzer o zumbador, es un transductor electro acústico que emite sonido continuo al ser polarizado, este dispositivo se usa como alarma frente algún evento ya sea temporizadores, aviso de ingreso de usuario, error de clave entre otras aplicaciones.

El entrenador cuenta con un buzzer HSD que funciona a 12V DC, para polarizar el buzzer se usa un transistor en corte y saturación, con el propósito de proteger al pin IOIO, ya que el buzzer utiliza una corriente de 40mA (datasheet) a 12V.

El transistor que usa el buzzer es un transistor NPN 2N3904, para que este transistor trabaje en corte y saturación se requiere una corriente de base igual a 1.0 mA información obtenida de la hoja de datos del dispositivo, con este dato se realiza los cálculos respectivos.



**Figura 20.** Circuito de polarización  
**Fuente:** Elaborado por Byron Valenzuela

$I_b (sat) = 1.0 \text{ mA}$ ;  $V_{be} = 0.65 \text{ mA}$  (Datasheet)

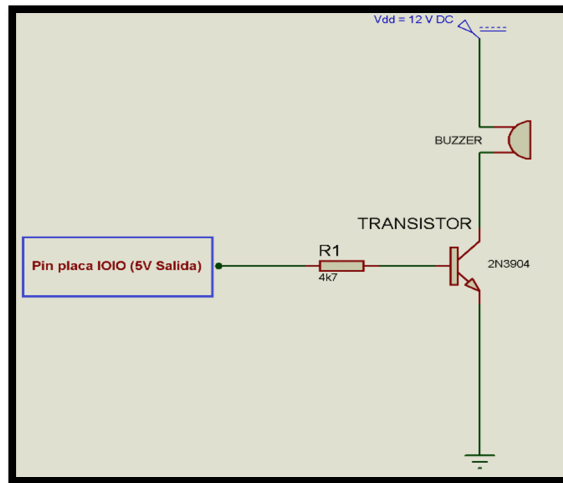
$R1 = R_b$  ;  $Pot = V * I$

$$R1 = \frac{V_{cc} - V_{be}}{I_b (sat)} ; Pot = 5V * 1.0 \text{ mA}$$

$$R1 = \frac{(5 - 0.65) \text{ V DC}}{1.0 \text{ mA}} ; Pot = 5mW$$

$$R1 = 5k\Omega = R_b$$

El valor de R1 es de 5kΩ a 5mW, pero comercialmente no existe ese valor, por tal razón se escoge una resistencia de 4.7kΩ a ¼ de wattio. Y la conexión con la IOIO es la siguiente.

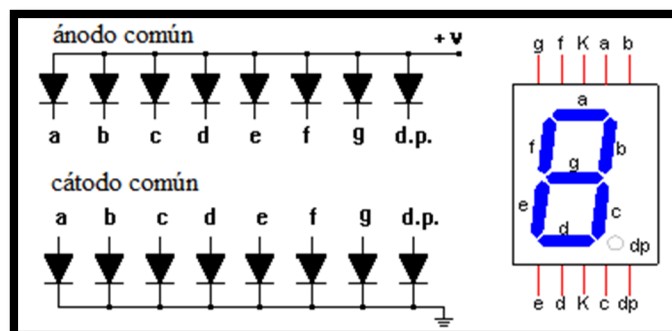


**Figura 21.** Esquema de conexión buzzer  
**Fuente:** Elaborado por Byron Valenzuela

### Display 7 segmentos

Dispositivo electrónico que consta de 7 diodos led que forman una especie de número 8 y su principal uso es para visualizar los números del 0 al 9; aunque se puede visualizar los caracteres que se desee encendiendo y apagando los leds que conforman el display.

Existen dos tipos de display 7 segmentos, los displays ánodo común y cátodo común; aunque en la forma física son semejantes lo que les hace diferentes es la forma en que los leds comparten la conexión común. En los ánodo común todos los led's comparten la conexión ánodo y esta a su vez a la fuente de alimentación; mientras que los cátodo común comparten la conexión de sus cátodos y va a Gnd.



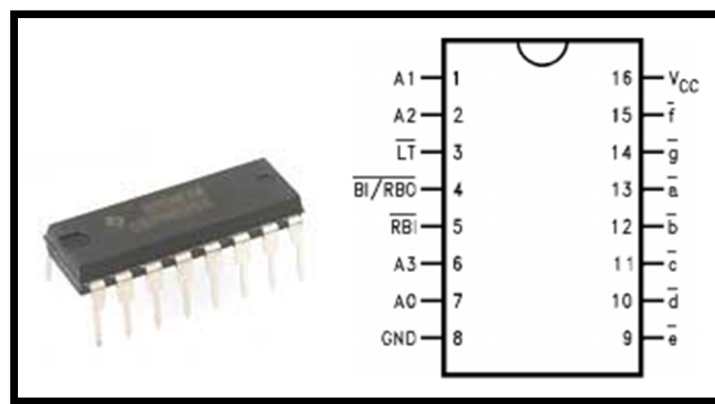
**Figura 22.** Display 7 segmentos

**Fuente:** Elaborado por Byron Valenzuela. Recuperado de Display 7 segmentos

[http://www.unicrom.com/Tut\\_display-7-segmentos.asp](http://www.unicrom.com/Tut_display-7-segmentos.asp)

Para hacer uso de este display se emplea de un decodificador para display 7 segmentos anodo común 74LS47, el entrador incluye dos display ánodo común con sus respectivos decodificadores con el propósito de ahorrar pines ya que el decodificador usa 4 bits para controlar el display mientras que sin decodificador se necesitaría 9 líneas para controlarlo.

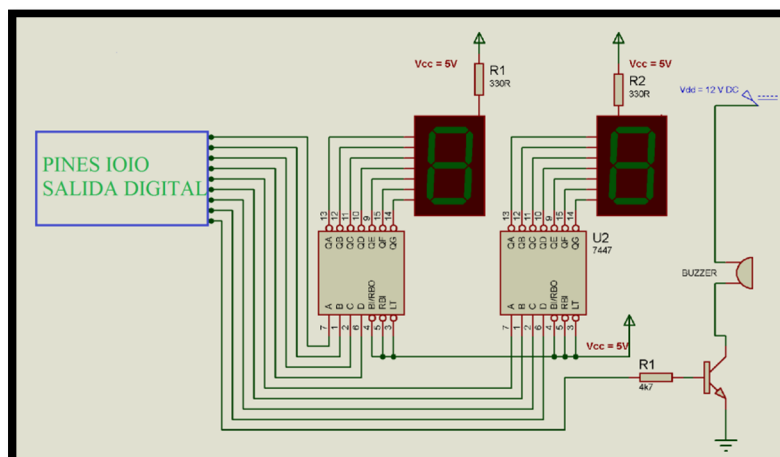
El integrado 74LS47 es un decodificador BCD a 7 segmentos, es capaz de controlar los leds del display directamente gracias a sus salidas a colector abierto. Sus principales características son: salidas colector abierto con el objetivo de controlar directamente al display, capacidad de supresión de 0 en una configuración en cascada, y un pin de activación o desactivación del controlador (FAIRCHILD, 2000).



**Figura 23.** Decodificador 74LS47

**Fuente:** Elaborado por Byron Valenzuela. Recuperado de (FAIRCHILD, 2000)

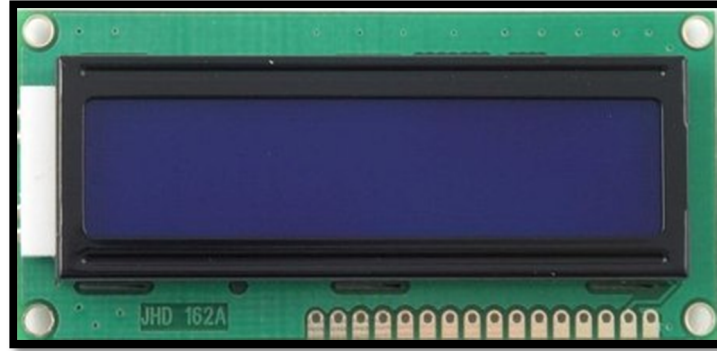
La conexión con la placa IOIO se muestra en la siguiente figura.



**Figura 24.** Esquema de conexión IOIO-displays

**Fuente:** Elaborado por Byron Valenzuela

## Pantalla LCD<sup>26</sup>



**Figura 25.** Pantalla LCD 16x2

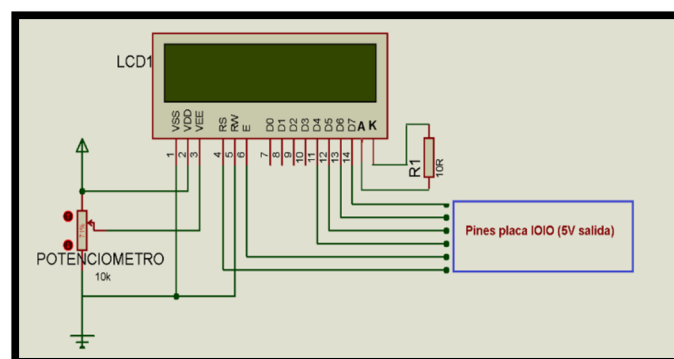
**Fuente:** Pantalla LCD 16x2 color azul. Recuperado de:

[http://articulo.mercadolibre.com.ec/MEC-402443448-pantalla-lcd-16x2-azul-microcontrolador-pic-avr-arduino-\\_JM](http://articulo.mercadolibre.com.ec/MEC-402443448-pantalla-lcd-16x2-azul-microcontrolador-pic-avr-arduino-_JM)

La pantalla LCD es un dispositivo electrónico que sirve para la representación de caracteres, símbolos e incluso dibujos, el entrenador contiene una LCD de 16x2 es decir dieciséis columnas, dos filas y cuenta con las siguientes características.

- Permite visualizar mensajes alfanuméricos y ASCII.
- Alimentación de 5V DC.
- Desplazamiento de caracteres a la izquierda o derecha.
- Capaz de proporcionar la dirección absoluta del carácter.
- Conexión a un procesador usando 4 u 8 bits.

La conexión con la IOIO es de la siguiente forma.



**Figura 26.** Diagrama de conexión LCD

**Fuente:** Elaborado por Byron Valenzuela

<sup>26</sup> LCD: Liquid Crystal Display, Display de cristal liquid

El potenciómetro que se observa en la anterior figura es un potenciómetro que controla el contraste de la lcd, el valor es de 10 kΩ (datasheet), y la resistencia que va a ánodo y cátodo es de 10Ω (datasheet) para proteger el diodo interno.

## Relé

El relé es un dispositivo electromecánico formado por una pequeña barra de hierro llamado núcleo y alrededor una bobina de hilo de cobre, funciona como una especie de interruptor accionado por un electroimán al existir corriente eléctrica por la bobina el núcleo se magnetiza y por ende este se acciona. El relé es un mecanismo que permite controlar un circuito eléctrico de más potencia con la que se esta trabajando.

El entrenador cuenta con un relé de 12V de propósito general para polarizar este dispositivo es necesario un transistor en corte y saturación. Para esta aplicación se va a proceder a calcular la resistencia de base con una corriente de base de 10mA (datasheet).

$$I_b (sat) = 10 \text{ mA}$$

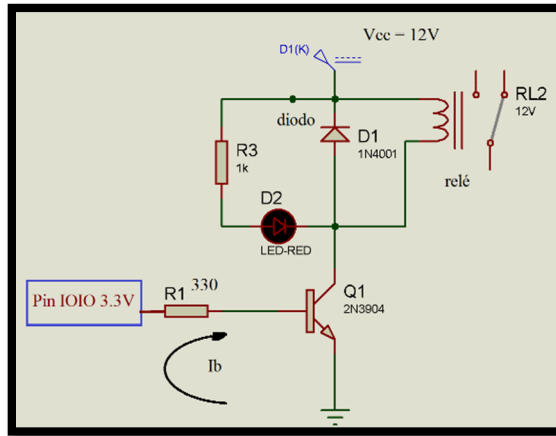
$$R1 = Rb$$

$$R1 = \frac{V_{cc}}{I_b (sat)} ; Pot = V * I$$

$$R1 = \frac{3.3V \text{ DC}}{10 \text{ mA}} ; Pot = 3.3V * 10mA$$

$$R1 = 330\Omega = Rb ; Pot = 33mW$$

La resistencia de base es de 330 Ω a ¼ de wattio que es un valor comercial; para la configuración del relé es necesario un diodo rectificador de 12V en paralelo que impide que al crearse el campo magnético la corriente inductiva que se genera en el núcleo no circule en sentido contrario de la corriente de colector y esta a su vez dañe el transistor. El diodo 1N4001 soporta hasta 50V (datasheet), A continuación se presenta una figura con la configuración del relé. El diodo led D2 junto con la resistencia que se muestra en la figura es un aviso visual que indica que el relé se activo

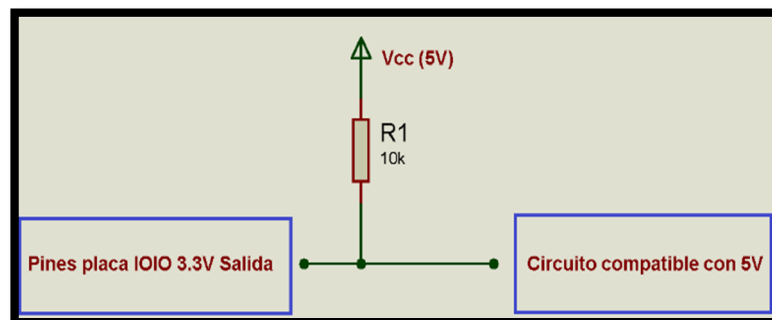


**Figura 27.** Conexión relé

**Fuente:** Elaborado por Byron Valenzuela

**Salidas a 5V**

En el capítulo I se mencionó que no todos los pines de la IOIO manejan el nivel de salida de 5V, esta sección adicional que tiene el entrenador, es el módulo salida 5V está específicamente diseñada para los pines de la IOIO que no manejan los niveles lógicos TTL. El diseño se basa en una resistencia pull-up con la fuente de 5V, el entrenador contiene 10 salidas en este módulo; el pin deberá estar trabajando en modo colector abierto esto hará que en 0V el pin se conecte directamente con el circuito externo y en 5V el pin se abra y entre a funcionar la fuente de 5V y teniendo 5V de salida. La resistencia pull-up es de  $10k\Omega$  (GitHub, 2008).



**Figura 28.** Esquema de conexión resistencia pull-up

**Fuente:** Elaborado por Byron Valenzuela

### 3.2.2 EENTRADA DIGITAL

#### Dip-Switch.

Este dispositivo electrónico es la unión de interruptores separados individualmente, el entrenador contiene un dip-switch de cuatro posiciones, usado para identificar el estado de un pin de entrada de la IOIO ya sea este un cero o un uno lógico.



**Figura 29.** Dip-switch 8 posiciones

**Fuente:** Dip-Switch. Recuperado de:

[http://www.olimex.cl/product\\_info.php?currency=CLP&products\\_id=3377](http://www.olimex.cl/product_info.php?currency=CLP&products_id=3377)

El dip-switch viene acompañado de una resistencia en serie con el fin de proteger el pin del dispositivo al que va a ser conectado en este caso la IOIO.

Cálculo de la resistencia para el dip-switch.

$$I = \frac{V_{cc}}{R}$$

*despejando R se obtiene*

$$R = \frac{(V_{cc})}{I} ; P = V * I$$

$$R = \frac{(5)V}{15mA} ; P = 5V * (15mA)$$

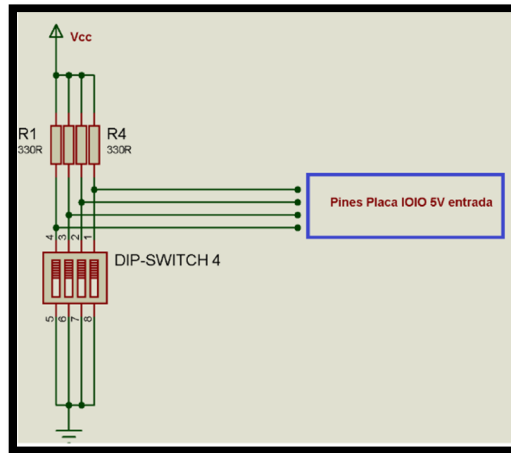
$$R = \frac{(5)V}{15mA} ; R = 333\Omega ; P = 75 Mw$$

Vcc (5V) = Voltaje de la fuente.

Id (15mA) = Corriente de entrada, lo máximo es 20mA así que se escoge 15mA.



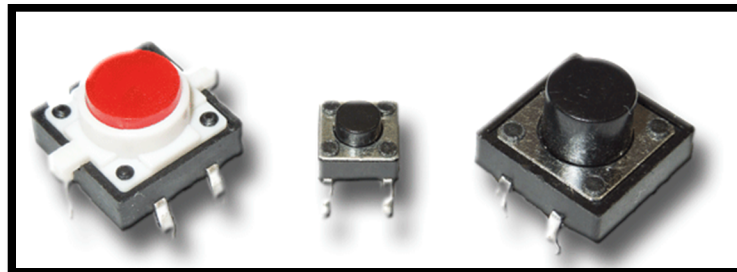
El valor de la resistencia que acompaña al dip-switch es de  $330\Omega$  a  $75\text{ mW}$  que quiere decir que se debe colocar una resistencia de  $330\ \Omega$  a  $\frac{1}{4}$  de wattio que es un valor comercial.



**Figura 30.** Esquema conexión dip-switch

**Fuente:** Elaborado por Byron Valenzuela.

## Pulsadores



**Figura 31.** Pulsadores

**Referencia:** Pulsadores. Recuperado de: <http://www.arduteka.com/2012/10/comparativa-pulsadores-para-proyectos/>

Un pulsador o un botón es un dispositivo electrónico que permite cambiar de estado lógico el pin al que está conectado, es decir es un interruptor instantáneo que cambia de estado siempre y cuando se mantenga presionado el mismo. Existen dos tipos de pulsadores normalmente abiertos y normalmente cerrados.

Los pulsadores normalmente cerrados al activarse pasarán a abrirse y los pulsadores normalmente abiertos pasarán a cerrarse, permitiendo así el paso o corte de corriente y por consecuencia el cambio de estado lógico de acuerdo como este configurado el mismo.

El entrenador cuenta con cuatro pulsadores normalmente abiertos conectados en serie con una resistencia a 5V, el estado de reposo es 0V, cuando se activa el botón permite el paso de corriente y cambia de estado a 5V.

Cálculo de la resistencia para el pulsador.

$$I = \frac{V_{cc}}{R}$$

*despejando R se obtiene*

$$R = \frac{(V_{cc})}{I} ; P = V * I$$

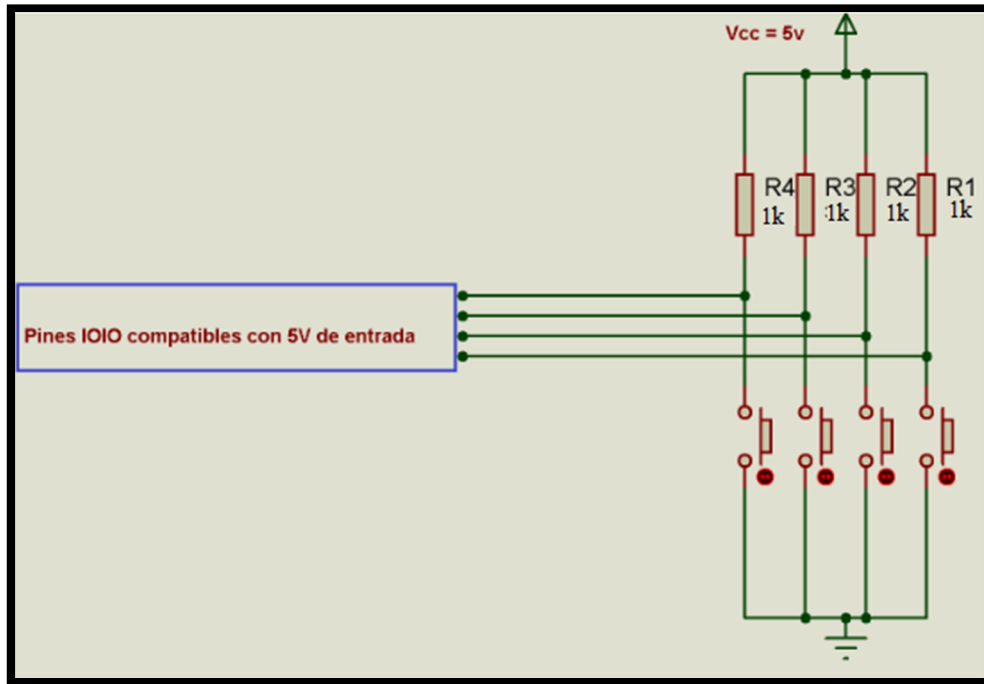
$$R = \frac{(5)V}{5mA} ; P = 5V * 5mA$$

$$R = \frac{(5)V}{5mA} ; R = 1000\Omega ; P = 25mW$$

Vcc (5V) = Voltaje de la fuente.

Id (5mA) = Corriente de entrada, lo máximo es 20mA para este caso se escoge 5mA.

La resistencia en serie con el pulsador es de 1kΩ A 25mW; por tal razón se coloca una resistencia de ¼ de wattio. Y la conexión se observa en la siguiente figura.



**Figura 32.** Conexión pulsadores con la placa IOIO

**Fuente:** Elaborado por Byron Valenzuela

En la figura anterior se observa los cuatro pulsadores normalmente abiertos conectados en serie con la resistencia de  $1k\Omega$  y la fuente de alimentación que al momento de cerrarse van a permitir el paso de corriente hacia los pines de la IOIO para realizar las prácticas de laboratorio.

### 3.3 MÓDULO DE CONVERSIÓN ANÁLOGO DIGITAL

El módulo de conversión análoga digital del entrenador cuenta con el dispositivo electrónico LM-35 un circuito integrado en forma de transistor, que es un sensor de temperatura con las siguientes características.

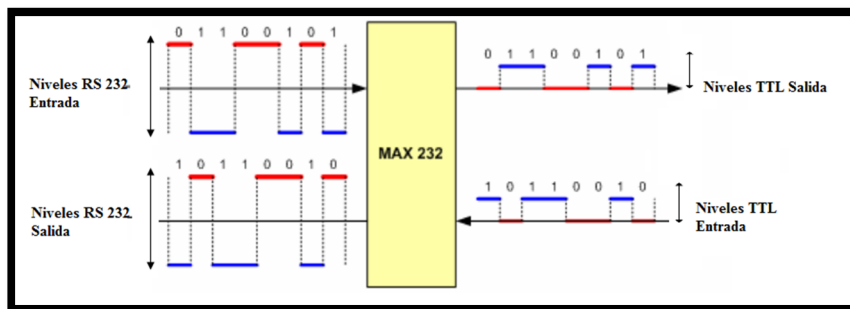
- Trabaja directamente con grados centígrados.
- Voltaje de salida proporcional a la temperatura.
- La escala de conversión es de  $10.0mV/^\circ C$ .
- Rango de lectura desde  $-55^\circ C$  a  $+150^\circ C$ .
- Admite una alimentación de 4V a 30V.



La figura anterior muestra la conexión del potenciómetro de 10kΩ conectado en serie con la fuente de alimentación, al momento de variar la resistencia el voltaje varía y así se puede realizar las prácticas con el módulo análogo digital.

### 3.4 MÓDULO DE COMUNICACIÓN SERIAL

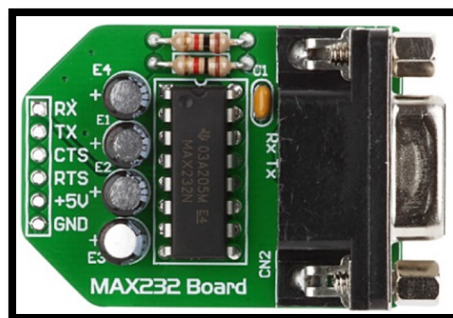
Este módulo consta de un circuito integrado Max232 que permite la comunicación serial con la IOIO, este integrado es capaz de convertir voltajes de nivel RS232 (3V a 15V en uno lógico y -3V a -15V en cero lógico), a voltajes TTL (5V en uno lógico y 0V en cero lógico) y viceversa.



**Figura 36.** Niveles de conversión RS 232 a TTL y viceversa

**Referencia:** (GARCÍA CELI & SANTILLÁN LARA, 2005)

El entrenador cuenta con un módulo Max232 del fabricante Mikroelektronika el módulo es Max232 Board. A continuación se presenta dicho módulo que servirá de interfaz de comunicación entre la PC y la IOIO.



**Figura 37.** Max232 Board

**Referencia:** Max232 Board. <http://www.mikroe.com/add-on-boards/communication/max232/>

### 3.5 MÓDULO ZIGBEE

Este es uno de los módulos adicionales al entrenador, es decir este módulo no forma parte de la placa central del entrenador, pero tiene su sócalo para conectarlo fácilmente y hacer uso del mismo.

Existen dos versiones idénticas de módulos Xbee de MaxStream que son Xbee® y Xbee-PRO® las dos versiones cuentan con características similares como son: bajo consumo de potencia, confiabilidad, y bajo costo, además trabajan en la banda ISM de 2.4GHz.

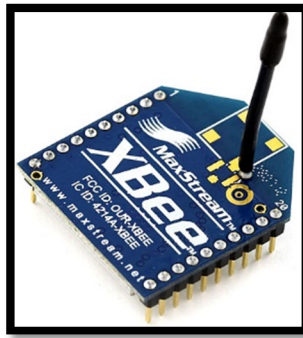
Las especificaciones de Xbee® y Xbee-PRO® se detallan en la siguiente tabla.

**Tabla 7.** Especificaciones Xbee® y Xbee-PRO®

Especificación	XbeeS1®	Xbee-PRO®
Alcance interno	Mayor a 30m	Mayor a 90m
Alcance externo con línea de vista	Mayor a 90m	Mayor a 1600m
Potencia de transmisión	1mW	63mW
Sensibilidad del receptor	-92dBm	-100 dBm
Voltaje de alimentación	2.8V a 3.4V DC	2.8V a 3.4V DC
Corriente de transmisión	45 mA a 3.3V	250mA a 3.3V
Corriente de recepción	50 mA a 3.3V	55mA a 3.3V
Banda de frecuencia	ISM 2.4 GHz	ISM 2.4 GHz
Comunicación	UART	UART
Dimensiones	2.438cm x 2.761cm	2.438cm x 3.294cm
Temperatura de operación	-40 to 85° C	-40 to 85° C

**Referencia:** (Digi International, Inc., 2009)

El entrenador es un equipo de laboratorio que será usado para motivos académicos por tal razón no se necesita de muchas distancias de operación, ni considerable potencia de transmisión, es así que el entrenador cuenta con el módulo XbeeS1®.

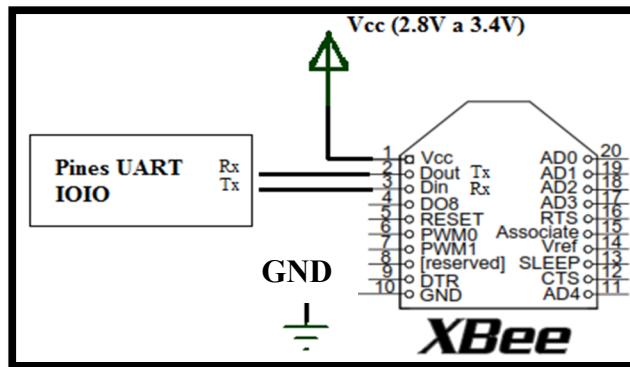


**Figura 38.** Módulo Xbee®.

**Referencia:** Módulo Xbee®. Recuperado de:

<http://embedsoftdev.com/embedded/more-on-xbee-part-one/>

XbeeS1® posee comunicación serial a través de la cual se comunica con la placa IOIO la conexión del módulo es la siguiente.



**Figura 39.** Conexión Xbee® con la IOIO

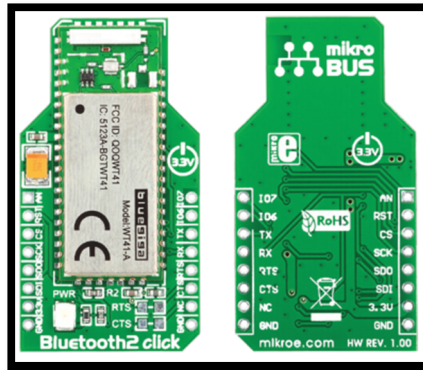
**Referencia:** Elaborado por Byron Valenzuela

La hoja de datos de Xbee® especifica que se debe realizar un divisor de voltaje con el objetivo de proteger el pin de recepción de datos (Din) del módulo ya que este módulo trabaja con un voltaje de 3.3V pero para esta aplicación no se debe realizar ningún divisor de voltaje debido a que el UART de la IOIO de igual forma trabaja con 3.3V.

### 3.6 MÓDULO BLUETTOH

De la misma manera Bluetooth es un módulo adicional al entrenador, para este dispositivo se dispone del conector mikroBus en la placa principal donde se lo puede insertar fácilmente y utilizar para las aplicaciones deseadas.

El dispositivo escogido para esta sección es el módulo Bluetooth2 Click fabricado por la empresa Mikroelektronika, este dispositivo puede alcanzar una cobertura de hasta 1000m, ofrece un bajo consumo de energía, e interfaz Uart que permite la comunicación con los dispositivos exteriores.



**Figura 40.** Módulo Bluetooth2 Click

**Referencia:** Bluetooth2 Click. Recuperado <http://www.mikroe.com/click/bluetooth2/>

Bluetooth2 Click cuenta con las siguientes especificaciones.

**Tabla 8.** Especificaciones BlueTooth2 Click

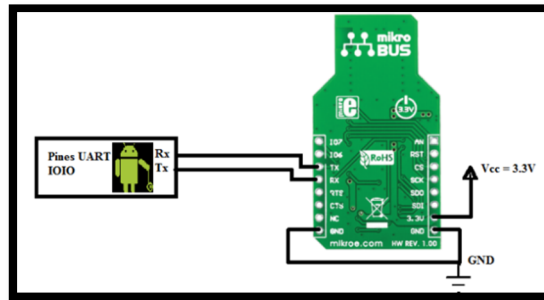
Características	Bluetooth2 Click
Comunicación	UART
Versión	Bluetooth 2.1 + EDR <sup>27</sup>
Banda de frecuencia	2.4 GHz
Velocidad de Transmisión	Hasta 3Mbps
Potencia de transmisión	20 dBm
Sensibilidad del receptor	-90 dBm
Temperatura de trabajo	-40C a +85 C
Cobertura	Exteriores hasta 1000m con línea de vista
Voltaje de alimentación	3.3V y 5V DC
Tamaño	14 mm x 35 mm x 5.65 mm
Peso	0,055 oz

**Referencia:** BlueTooth2 Click. Recuperado de <http://www.mikroe.com/click/bluetooth2/>

<sup>27</sup> EDR: Enhanced Data Rate, especificación que permite mejorar la velocidad de transmisión.



BlueTooth2 Stick se comunica con la IOIO mediante la comunicación Uart, y la forma de conexión con la IOIO es la siguiente.



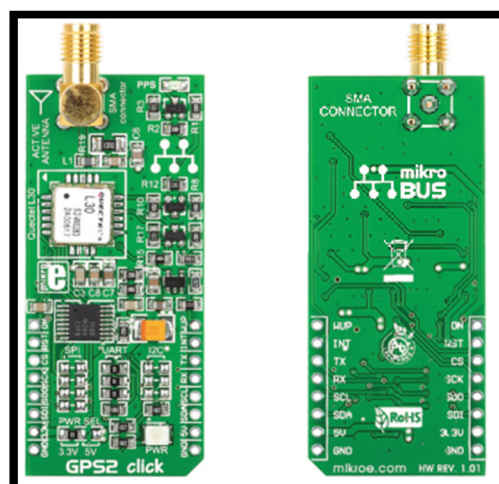
**Figura 41.** Conexión BlueTooth2 Click con la IOIO

**Referencia:** Elaborado por Byron Valenzuela

Los niveles de voltaje que maneja el Uart del BlueTooth2 Click son compatibles con los niveles Uart de la IOIO es por eso que la conexión es directa como se muestra en la figura anterior.

### 3.7 MÓDULO GPS

El módulo GPS adoptado para esta sección es el dispositivo GPS2 Click de mikroelektronika, es un equipo compacto que se adapta al sócalo mikroBus, este módulo soporta navegación, localización, y aplicaciones industriales como GPS C/A, S-BAS, y A-GPS, es compatible con comunicación UART, SPI, I2C configurable mediante jumpers o puentes en la placa, ofrece bajo consumo de potencia y alto rendimiento (Quectel, 2013).



**Figura 42.** GPS2 Click

**Fuente:** GPS2 Click. Recuperado de: <http://www.mikroe.com/click/gps2/>

Este dispositivo cuenta con las siguientes especificaciones técnicas.

**Tabla 9.** Especificaciones GPS2 Click

Características	GPS2 Click
Comunicación	UART, SPI, I2C
Protocolos	NMEA OSP
Banda de frecuencia	L1 (1575.42)MHz
Canales	48 canales
Precisión posición horizontal	< 2.5 m CEP
Precisión de la velocidad	< 0.01 m/s
Tiempo de readquisición	< 1s
Velocidad de Transmisión	4800 bps
Potencia de adquisición	130 dBm
Sensibilidad de navegación	-159 dBm
Altitud máxima	18288 m
Velocidad máxima	514 m/s
Voltaje de alimentación	3.3V y 5V DC
Tamaño	14 mm x 35 mm x 5.65 mm
Peso	0,6 gr

**Fuente:** (Quectel, 2013)

GPS2 click está diseñado para usar antenas activas, estas son antenas de alto rendimiento con protección de circuitos de radio frecuencia, integran un amplificador de bajo ruido (LNA), y cable coaxial capaz de alcanzar los 20m de distancia.



**Figura 43.** Antena activa DLGPS-E2

**Fuente:** Antena activa. Recuperado de: <http://www.mikroe.com/search/>

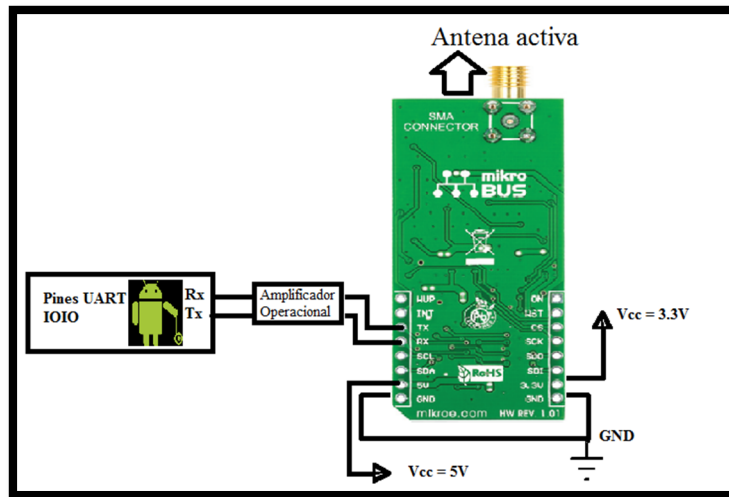
El módulo GPS incluye una antena DLGPS-E2; es una antena externa magnética y resistente al agua que cuenta con las siguientes especificaciones.

**Tabla 10.** Especificaciones antena DLGPS-E2

<b>Rango de frecuencia</b>	<b>1575.42Mhz</b>
<b>Ancho de banda</b>	+/-5Mhz
<b>Impedancia de entrada</b>	50 ohm
<b>Ganancia pico</b>	> 3dBic
<b>Polarización</b>	RHCP
<b>Ganancia LNA con cable</b>	27 dB
<b>Ruido</b>	1.5 dB
<b>VSWR</b>	≤ 2.0:1
<b>DC Voltaje</b>	3 - 5V
<b>DC Corriente</b>	Max 10mA
<b>Tamaño de la antena</b>	48.5 mm* 39mm * 15mm
<b>Longitud del cable</b>	5m
<b>Tipo de conector</b>	SMA
<b>Color de la antena</b>	Negro
<b>Montaje de la antena</b>	Magnética
<b>Temperatura de funcionamiento</b>	-30°C a +75°C
<b>Resistente al agua</b>	100%Resistente al agua

**Referencia:** DLGPS-E2. Recuperado de: <http://www.dolinele.com/DetailP.aspx?id=257>

El dispositivo GPS2 Click interactúa con la IOIO a través de la interfaz de comunicación serial, debido a que los niveles uart no son TTL es necesario utilizar un amplificador operacional para el uso del GPS y la forma de conexión es la siguiente.

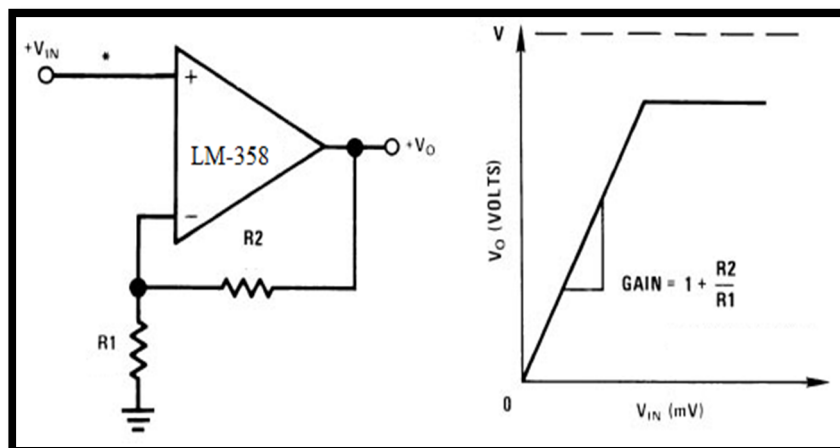


**Figura 44.** Conexión GPS2 Click con la IOIO

**Fuente:** Elaborado por Byron Valenzuela

El motivo de usar el amplificador operacional es por los niveles de voltaje que maneja el transmisor y receptor serial del módulo GPS2 Click que no son compatibles con los niveles de la placa IOIO (3.3V). Puesto que GPS2 Click maneja para transmisor 1.8V y para receptor hasta 3.6V

A continuación se presenta los cálculos realizados para la configuración del amplificador operacional. El modo de operación es amplificador DC no inversor.



**Figura 45.** Configuración del amplificador no inversor

**Fuente:** Elaborado por Byron Valenzuela. Basado en

<http://electronica.webcindario.com/componentes/lm358.htm>

La ecuación para calcular el voltaje de salida es la siguiente

$$V_0 = Gain * V_{in}$$

*Datos:*

$$Gain = \frac{5V}{1.8V} = 2.8$$

$$R2 = 100k\Omega$$

$$V_0 = Gain * V_{in} ; \text{Donde (1)}$$

$$Gain = 1 + \frac{R2}{R1} ; \text{Entonces (2)}$$

$$V_0 = \left(1 + \frac{R2}{R1}\right) * V_{in} \text{ (3)}$$

$$\left(1 + \frac{R2}{R1}\right) = Gain \text{ (5)}$$

$$R2 = (2.8 - 1) * R1 \text{ (6)}$$

$$R1 = 100k\Omega / (2.8 - 1) \text{ (7)}$$

$$R1 = 55.5 k \Omega \text{ (8)}$$

Como resultado se escoge una resistencia de 68kΩ que es la que se puede encontrar en el mercado.

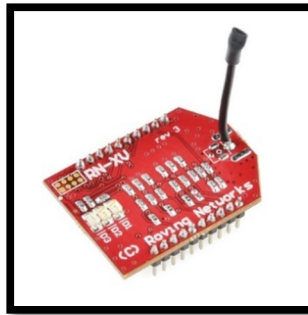
### 3.8 MÓDULO WI-FI

Wi-fi es un módulo adicional del entrenador, cuenta con el sócalo Xbee Explorer en la placa principal donde se lo puede insertar y hacer el uso correspondiente.

El dispositivo Wi-fi usado para esta sección es el equipo WiFly RN-XV fabricado por Sparkfun; WiFly RN-XV es una solución compacta para añadir comunicación Wi-fi bajo el estándar IEEE<sup>28</sup> 802.11 en los diseños electrónicos. Este dispositivo tiene las siguientes características: posee una antena integrada capaz de alcanzar 100m de distancia, 8 pines de propósito general, 3 conversores análogo digitales, y una interfaz de comunicación UART.

---

<sup>28</sup> IEE: Institute of Electrical and Electronics Engineers



**Figura 46.** WiFly RN-XV

**Referencia:** WiFly RN-XV. Recuperado de: <https://www.sparkfun.com/products/10822>

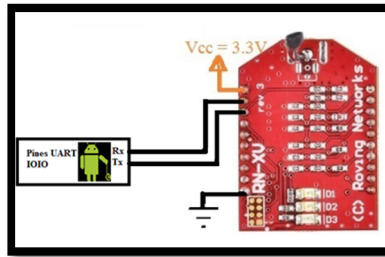
WiFly RN-XV tiene las siguientes especificaciones técnicas.

**Tabla 11.** Especificaciones WiFly RN-XV

Estándar	IEE 802.11 b/g
Banda de frecuencia	Banda ISM 2.400–2.484 GHz
Tasa de transmisión	1 – 11Mbps for 802.11b / 6 – 54Mbps for 802.11
Comunicación	UART TTL
Antena	PCB Integrada
Alcance	Mayor a 100m
Voltaje de alimentación	3.3V DC
Temperatura de trabajo	-40°C a + 85°C
Corriente de recepción	40 mA
Corriente de transmisión	180 mA
Potencia de transmisión	10dBm
Sensibilidad del receptor	-83 dBm
Frecuencia	2402 ~ 2480MHz
Seguridad	WEP, WPA-PSK, and WPA-2-PSK
Tamaño	14 mm x 35 mm x 5.65 mm
Intervalo de canales	1-14
<b>Soporta infraestructura Ad-hoc</b>	

**Fuente:** (ROVING NETWORKS, 2011)

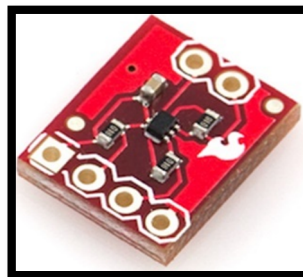
La conexión se muestra en la siguiente figura.



**Figura 47.** Conexión Wi-Fly con la IOIO  
**Fuente:** Elaborado por Byron Valenzuela

### 3.9 MÓDULO I2C

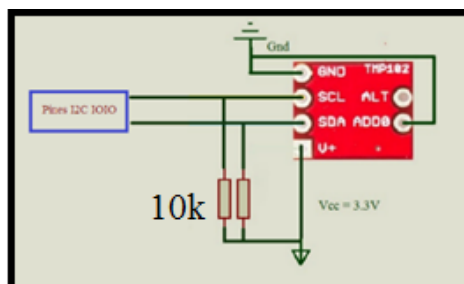
Este dispositivo es otro de los módulos adicionales al entrenador, el equipo I2C escogido para esta sección es el sensor de temperatura TMP-102, es un sensor creado por la compañía Sparkfun Electronics, usa un convertor análogo digital de 12 bits con una resolución de de 0.0625°C, puede medir temperaturas desde -25°C a +85°C, tiene un bajo consumo de potencia 10uA en estado activo, 1uA en estado pasivo, y una alimentación desde 1.6V a 3.3V.



**Figura 48.** Módulo I2C TMP 102

**Fuente:** Digital Temperature Sensor Breakout - TMP102. Recuperado de <https://www.sparkfun.com/products/retired/9418>

La conexión del sensor de temperatura con la IOIO se muestra en la siguiente figura.



**Figura 49.** Conexión del módulo I2C con la IOIO  
**Fuente:** Elaborado por Byron Valenzuela

Las dos resistencias de 10kΩ que acompaña a los pines SDA y SCL respectivamente son valores predeterminados que el fabricante dispone en la hoja de datos (datasheet).

### 3.10 FUENTE DE ALIMENTACIÓN

Este bloque del sistema es el encargado de suministrar la energía necesaria para que todos los módulos del entrenador puedan trabajar, esta sección a su vez se divide en dos fuentes de alimentación totalmente independientes la una se encarga de alimentar a el entrenador en sí y cuenta con varios pines de salida donde se puede alimentar los circuitos externos, la otra es la encargada de energizar a los módulos adicionales que cuenta el entrenador como son GSP, Bluetooth, Wi-fi, I2C, Xbee.

El diseño de la fuente del entrenador está realizado en base a los requerimientos de todos los módulos del mismo, es así que a continuación se tiene una tabla donde se detalla el voltaje necesario para cada módulo.

**Tabla 12.** Requerimientos de voltaje de lo módulos del entrenador

Módulo	Voltaje de alimentación
Salida digital	5V y 12V DC
Entrada digital	5V DC
Conversión análoga digital	5V DC
Comunicación Serial	5V DC
ZigBee	3.3V DC
Bluetooth	3.3V y 5V DC
GPS	3.3V y 5V DC
I2C	3.3V
Wi-fi	3.3V DC

**Fuente:** Elaborado por Byron Valenzuela



La fuente de alimentación del entrenador está formada principalmente por un transformador de 120VCA en 24V de corriente alterna CA a 1 amperio, luego pasa por un rectificador de onda completa que convertirá la corriente alterna en corriente continua CD formado por diodos rectificadores 1N4001 capaz de soportar 50VCA a 1 amperio (A) que es lo que la fuente necesita, seguido de esto se compone de un capacitor de 2200µF (cálculos en la parte posterior) y un capacitor de 47µf (datasheet) en paralelo con el objetivo de eliminar los picos de voltaje aún existentes. A la salida de este proceso ya se cuenta con 12VDC.

A continuación viene el proceso de regulación, los voltajes requeridos son de 12VDC, 5VDC, y 3.3VDC; para esto se hace uso de los reguladores LM7812, LM7805, LM1117V33 respectivamente acompañado de un capacitor de 33µf en paralelo que determina la hoja de datos del integrado.

Cálculo del condensador para la fase de rectificación.

*Datos*

$$V_{rms} = V_p * \sqrt{2}$$

$$V_p = 24V$$

$$V_{rms} = 24V * \sqrt{2} = 33.94V$$

$$Freq = 60Hz$$

$$I = 1A$$

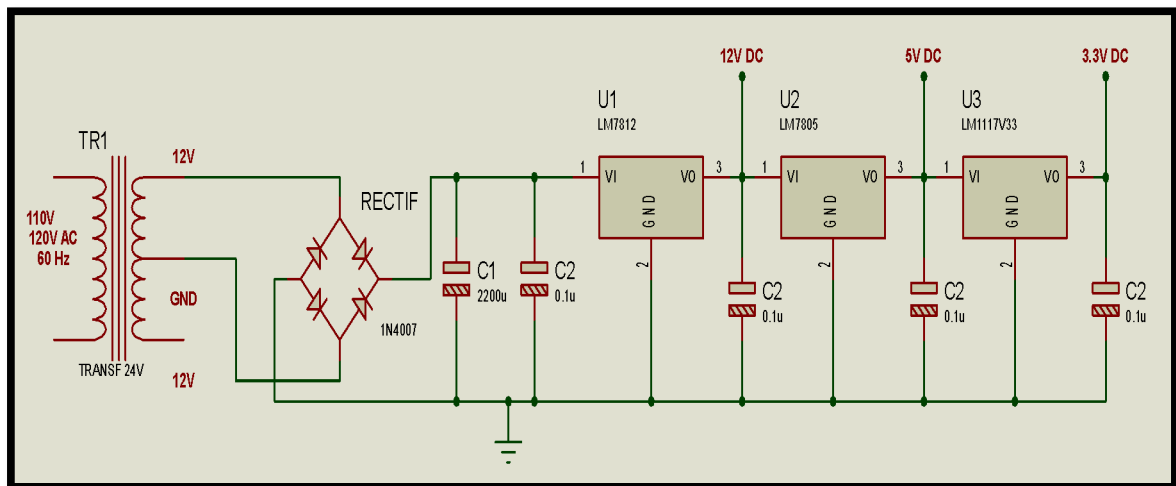
$$C = \frac{5 * I}{F * V_S *}$$

$$C = \frac{5 * 1 A}{60 Hz * 33.94V}$$

$$C = 0.02255$$

$$C = 2200 mF$$

El siguiente diagrama muestra la fuente de alimentación del entrenador.



**Figura 50.** Diagrama de la fuente de alimentación del entrenador

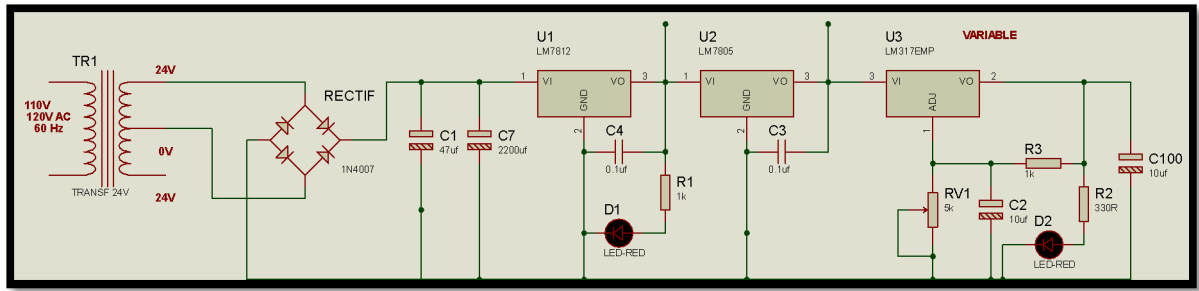
**Referencia:** Elaborado por Byron Valenzuela

La fuente de alimentación para los módulos adicionales es muy similar a la principal la única diferencia es que cuenta con una salida de voltaje variable de 0VDC a 12VDC; de la misma manera esta cuenta con un transformador de 120V en 24V de corriente alterna CA a 1 amperio, un rectificador de onda completa formado por los mismos diodos rectificadores de la fuente principal, el proceso de filtrado cuenta con los mismos capacitores de 2200µF y de 47µF.

Para el proceso de regulación se tiene los mismos voltajes anteriores y adicionalmente el voltaje variable 0V-12V y para esto se usa el regulador LM317 que es un regulador capaz de emitir un voltaje variable de 1.2VDC a 37VDC con una corriente de 1.5A (datasheet), el fabricante impone su propia configuración para el trabajo de este integrado; la configuración tiene un capacitor de 1µF en paralelo con la salida, una resistencia de 1kΩ entre los pines VO (voltaje de salida) y VADJ (voltaje variable) y un potenciómetro o resistencia variable de 5k Ω entre los pines VADJ y GND.

Adicional a la fuente se tiene dos diodos led con su respectiva resistencia de protección que actúan como indicadores visuales el diodo 1 uno se activa cuando la fuente se enciende, el diodo led 2 se enciende directamente proporcional al voltaje variable que esta saliendo de la fuente es decir que mientras mas voltaje exista el led emitirá más luz.

A continuación se tiene un diagrama de la fuente de alimentación para los módulos adicionales.



**Figura 51.** Fuente de alimentación variable.

Referencia: Elaborado por Byron Valenzuela

### 3.11 RESUPUESTO

El presupuesto referencial para la construcción y adquisición de los materiales y módulos con los que el entrenador cuenta se detalla a continuación en la siguiente tabla.

**Tabla 13.** Costos de los módulos y materiales para la construcción del entrenador

DESCRIPCIÓN	FABRICANTE	CANTIDAD	VALOR	TOTAL
<b>Módulos del entrenador</b>				
Placa IOIO	Sparkfun	1	150,00	150,00
Módulo Bluetooth (Bluetooth2 click)	Mikroelectronika	1	80,00	80,00
Módulo ZigBee (Xbee S1)	Digi Internacional	2	60,00	120,00
Módulo Wi-fi (RN-XV WiFly)	Sparkfun	1	80	80,00
Módulo GPS (GPS2 Click)	Mikroelectronika	1	95,00	95,00
Max232 Board	Mikroelectronika	1	25,00	25,00
Zócalo Xbee Xplorer	Sparkfun	1	25,00	25,00
Sensor LM-317	N/A	1	2,50	2,50
Matriz de led's	N/A	1	8,00	8,00
LCD 16x2	N/A	1	15,00	15,00
Sensor de temperatura (TMP-102)	Sparkfun	1	25,00	25,00

DESCRIPCIÓN	FABRICANTE	CANTIDAD	VALOR	TOTAL
Transporte módulos del exterior	N/A	1	60,00	60,00
			<b>Total módulos</b>	<b>685,50</b>
<b>Materiales y elementos electrónicos</b>				
Baquelita de fibra vegetal 1 lado	N/A	1	8,4	8,4
Transformador 12V 1ª	N/A	2	6,13	12,26
Regulador 5V 7805	N/A	2	0,42	0,84
Regulador 5V 7812	N/A	2	0,42	0,84
Regulador LM 1117	N/A	1	1,25	1,25
Capacitor electrolítico 1000uf/25V	N/A	1	1,00	1,00
Capacitor electrolítico 2200uf/25V	N/A	1	1,00	1,00
Capacitor electrolítico 47uf/25V	N/A	1	1,00	1,00
Capacitor electrolítico 10uf/25V	N/A	2	1,00	2,00
Capacitor cerámico 104	N/A	3	0,08	0,24
Didodos rectificadores 1N4001	N/A	8	0,10	0,80
Diodo Led	N/A	8	0,10	0,80
Resistencias varios valores 1/4W	N/A	35	0,05	1,75
Dipswitch 4 posiciones	N/A	1	0,67	0,67
Pulsador	N/A	4	0,20	0,80
Relé 12V	N/A	1	0,80	0,80
Transistor 2n3904	N/A	2	0,50	1,00
Buzzer electrónico	N/A	1	1,00	1,00
Decodificador 74LS47	N/A	2	1,20	2,40

DESCRIPCIÓN	FABRICANTE	CANTIDAD	VALOR	TOTAL
Display 7 segmentos Ánodo común	N/A	2	0,80	1,60
Potenciómetro	N/A	4	0,25	1,00
Amplificador OP. LM358	N/A	1	1,00	1,00
Bornera 2 posiciones	N/A	2	0,25	0,50
Bornera 3 posiciones	N/A	1	0,50	0,50
Espadin macho simple	N/A	8	0,60	4,80
Espadin hembra	N/A	8	0,78	6,24
Espadin maquinado hembra	N/A	8	0,80	6,40
Zócalo 16 pines	N/A	2	0,10	0,20
Zócalo 8 pines	N/A	1	0,10	0,10
Protoboard 1 regleta	N/A	1	6,00	6,00
Base para el entrenador	N/A	1	15,00	15,00
			<b>Total materiales</b>	<b>82,19</b>
			<b><u>Total costo</u></b>	<b><u>767,69</u></b>

**Fuente:** Elaborado por Byron Valenzuela.

Como se observa en la tabla el valor total para la construcción del entrenador es de 617.69\$ este precio incluye todos los módulos y elementos electrónicos que el entrenador posee, además se incluye el precio de la base que soporta la placa principal del entrenador.

Todos los elementos y módulos fueron adquiridos en una tienda electrónica en la ciudad de Quito, los precios son relativamente más económicos si se los puede adquirir directamente al fabricante.

## **CAPÍTULO IV**

### **4 PROPUESTA DE GUÍAS DE PRÁCTICAS**

#### **4.1 PRÁCTICAS CON EL MÓDULO ENTRADA Y SALIDA DIGITAL**

##### **4.1.1 JUEGO DE LUCES**

- **Tema:** Juego de luces
- **Objetivos:**
  - Objetivo general

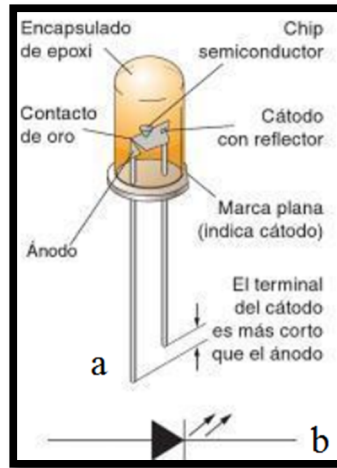
Implementar un juego de luces secuenciales, mediante el uso de los pines de salida digitales de la placa IOIO, para comprobar el funcionamiento del módulo de salida digital del entrenador.

- **Objetivos específicos**
  - Investigar el funcionamiento de la IOIO referente al módulo de salida digital.
  - Conocer los parámetros y funcionamiento electrónico de los diodos led.
  - Realizar el flujo grama para este caso de estudio.
  - Crear el programa que controle el juego de luces y ejecutarlo desde el dispositivo Android.
  - Implementar el juego de luces mediante el entrenador y la placa IOIO.

- **Marco teórico**

##### **Led's**

El diodo emisor de luz (Led), es un dispositivo electrónico, compuesto por un semiconductor llamado arseniuro de galio (AsGa), que emite luz cuando este se polariza de forma directa. Existen led's de diferentes colores como: rojo, azul, amarillo, verde, etc.



**Figura 52.** a) Esquema diodo led b) símbolo diodo led

**Referencia:** (A. Carretero, 2009)

La caída de tensión que experimentan varía de acuerdo a su color y va desde 1.5V a 3V.

El uso del diodo led es muy amplio, puede ser usado como un indicador visual en algún tipo de alarma, se los encuentra en los display's siete segmentos, tableros de instrumentos en los automóviles y otras diversas aplicaciones.

Para esta aplicación se usa diodos led color rojo con las siguientes especificaciones.

**Tabla 14.** Especificaciones diodo led rojo

Items	Symbol	Absolute máximo rating	Unit
Forward current DC	$I_f$	50	mA
Peak forward current	$I_{fp}$	100	mA
Reverse voltaje	$V_r$	5	V
Power dissipation	$P_d$	150	mW
Operation temperatura	$T_{opr}$	-20 ~ +95	°C

**Referencia:** (zhongzhouOpto)

A continuación se detalla los requerimientos para el desarrollo de esta aplicación.

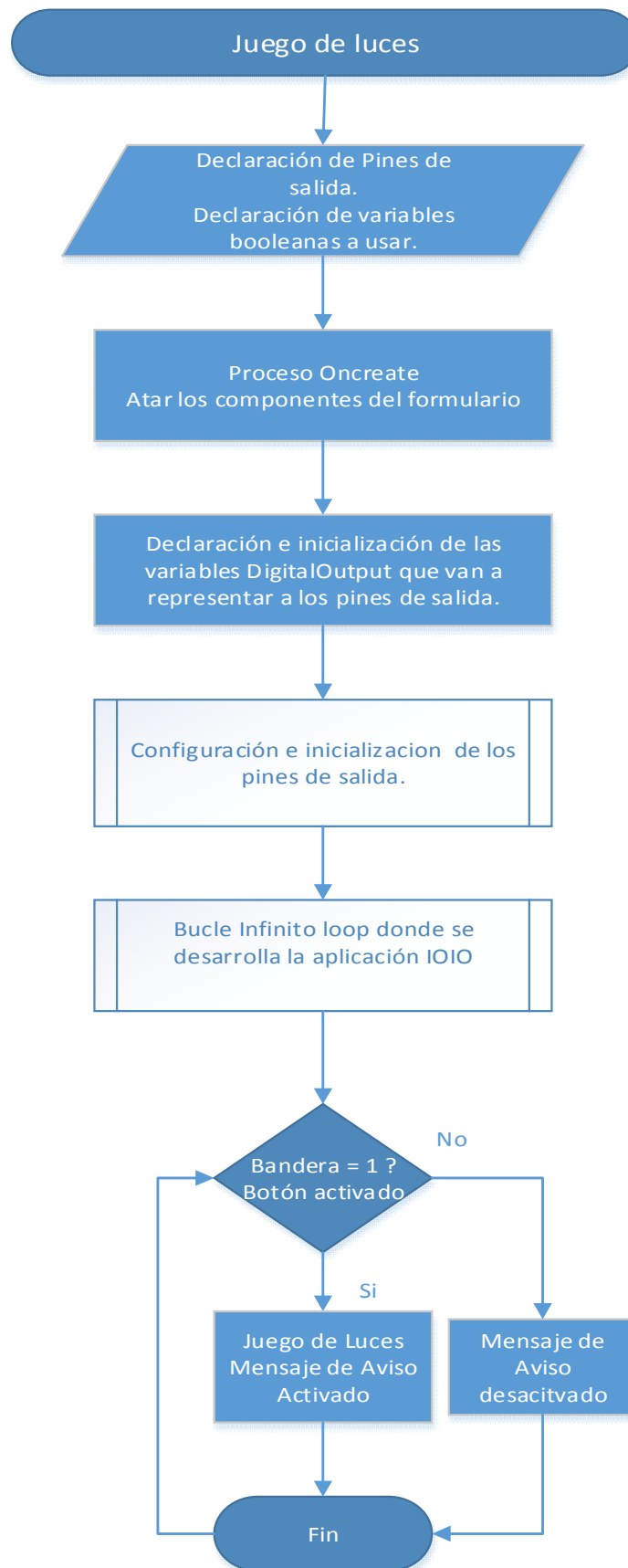
**Tabla 15.** Requerimientos de la aplicación juego de luces

<b>Dispositivo</b>	<b>Requerimientos</b>
<b>Led's</b>	Ocho led's rojos con los que cuenta el entrenador
<b>Tarjeta IOIO</b>	Pines digitales 3.3V.
<b>Equipo móvil Android</b>	Equipo que interactúa con la IOIO
<b>Dispositivo Bluetooth o cable de comunicación usb del equipo Android</b>	Interfaz de comunicación IOIO – Dispositivo móvil Android.

**Referencia:** Elaborado por Byron Valenzuela



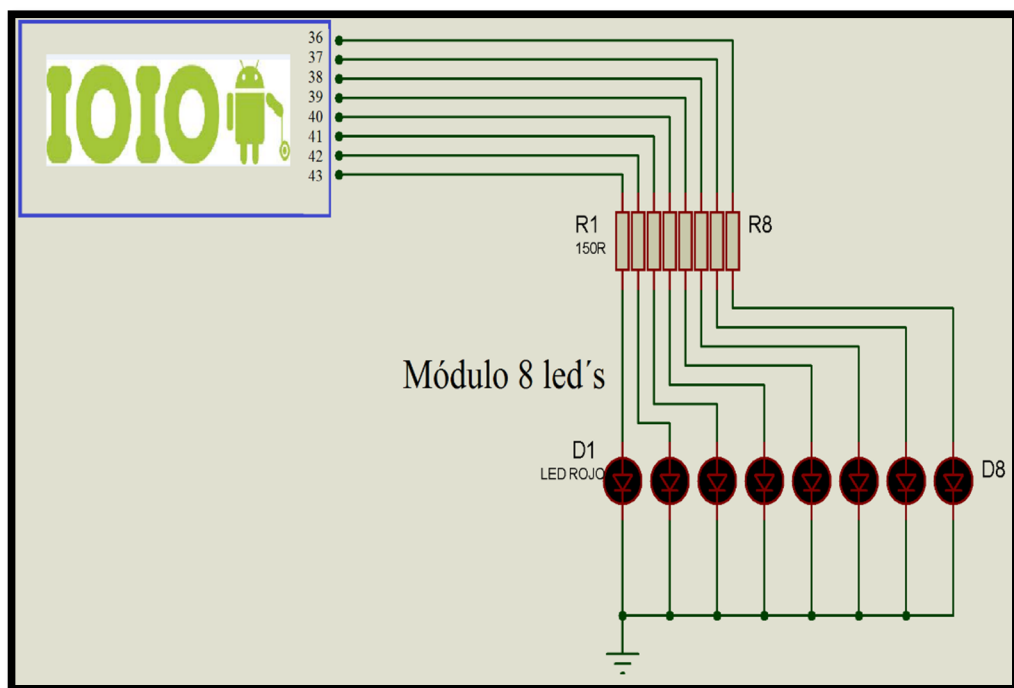
- Flujo grama



- **Desarrollo**

**Enunciado:** Desarrollar un juego de luces secuenciales de 8 led's de izquierda a derecha, conectados a los pines de salida digital no tolerantes a 5V de la IOIO, con un retardo de 500ms en cada transición.

- Los pines digitales escogidos son 36, 37, 38, 39, 40, 41, 42, 43.
- El entrenador cuenta con 8 led's disponibles para ser usados en esta aplicación, a continuación se presenta una figura del esquema de conexión de la IOIO con el entrenador.



**Figura 53.** Diagrama de conexión de la aplicación juego de luces

**Referencia:** Elaborada por Byron Valenzuela

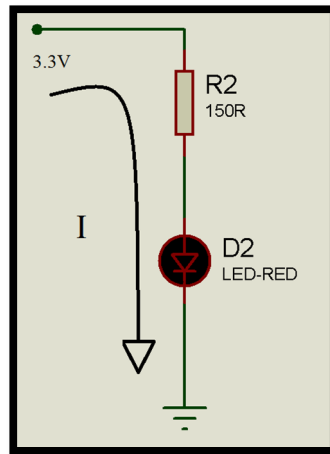
- **Análisis de resultados**

Los resultados obtenidos en esta práctica son los siguientes:

- Para el estado 0 lógico

Se tiene un voltaje de 0V y por ende la corriente es de 0mA.

- Para el estado 1 lógico



**Figura 54.** Circuito equivalente para el estado 1 lógico

**Referencia:** Elaborado por Byron Valenzuela

Datos.

$$I = \frac{V}{R} ; \text{donde } V = (V_{\text{pin}} - V_d) ; I = I_d$$

$$I = \frac{(V_{\text{pin}} - V_d)}{R} ;$$

$$I = \frac{(3.3 - 2.0)V}{150\Omega}$$

$$I = \frac{(1.3)V}{150\Omega} ; I = 8.6\text{Ma}$$

El valor de corriente que se obtiene del circuito es aproximadamente 7.8mA , y no sobrepasa la corriente máxima recomendada para la IOIO que es de 20mA. En cuanto a voltaje el valor es de 3.34V aproximadamente

- **Conclusiones**

- Los valores de corriente obtenidos en la teoría, están reflejados en la práctica al obtener mediciones que varían por muy pocas cifras.
- El método loop del código del programa es un bucle infinito en el cual se desarrolla la aplicación.

- La librería IOIOLibBt de la IOIO permitir la comunicación entre el dispositivo Bluetooth conectado a la IOIO y el Bluetooth del dispositivo Android de manera automática, sin escribir código adicional en la aplicación.
- La caída de voltaje varía de acuerdo al color del led que se esté usando.
- La IOIO dispone de pines tolerantes y no tolerantes a niveles de voltaje TTL.

- **Recomendaciones**

- Realizar de manera correcta los cálculos de manera que no sobrepase la corriente máxima soportada por la placa IOIO, y que pueda ocasionar averías en el mismo.
- Se recomienda verificar el funcionamiento de la aplicación mediante el cable usb y vía inalámbrica mediante Bluetooth.
- No olvidar colocar los permisos de internet y Bluetooth dentro de la aplicación Android ya que sin estos permisos el programa no se ejecuta.

- **Bibliografía.**

mikroElektronika. (2012). *mikroBUS pinout Standar Especification*.

A. Carretero, F. F. (2009). *Electrónica*. Editex.

Agüero, R. (s.f.). *Redes Inalámbricas de Área Local y Personal WLAN El estándar IEEE 802.11*. Cantabria.

Argüello, D. J. (2012). *Desarrollo de una aplicación que permita la captura almacenamiento, reproducción, administración y envío de archivos de vide, audio e imagenes utilizando la tecnología bluetooth, para dispositivos móviles basados en el sistema operativo Android*. Quito.

Carballar, J. (2010). *WI-FI lo que necesita conocer*. Madrid: RC Libros.

Carrillo Pérez, M. d. (2006). *Estudio y análisis de rendimiento Bluetooth*. Madrid.

Castillo, D. (2012). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ALARMA COMUNITARIA A BASE DE MÓDULOS INALÁMBRICOS UTILIZANDO TECNOLOGÍA ZIGBEE*. Ibarra.

Clanar Internacional. (s.f.). *Internet y redes inalámbricas*. Arequipa: Clanar.

Collaguazo, G. (2009). *Sistemas Basados en Microprocesadores*. Ibarra.

Correia, P. (2002). *Guía práctica del GPS*. Barcelona: Marcombo.

Creative Commons . (s.f.). *Arduino*. Obtenido de <http://arduino.cc/en/Main/CopyrightNotice>

Daniel Benchimol. (2011). *Microcontroladores*. Buenos Aires: DALAGA S.A.

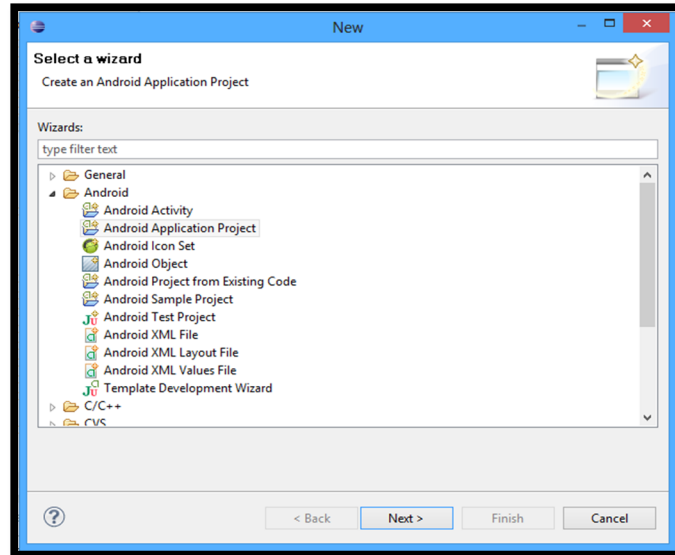
- Digi International, Inc. (2009). *XBee®/XBee-PRO® RF Modules*. New York.
- Dignani, J. (2011). *Análisis del protocolo ZigBee*.
- Dignani, J. P. (2011). *Análisis del protocolo ZigBee*.
- Fairchild Semiconductor. (1999). *MM74C922 • MM74C923*.
- FAIRCHILD. (2000). *DM74LS47 BCD to 7-Segment Decoder/Driver with Open-Collector Outputs*. FAIRCHILD SEMICONDUCTOR.
- Fernández, A. M. (2004). *EL BUS I2C*. Córdoba.
- GARCÍA CELI, H. M., & SANTILLÁN LARA, L. A. (2005). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE TARIFACIÓN PARA LOCUTORIOS*. Quito.
- GitHub. (2008). *Analog Input*. Obtenido de <https://github.com/ytai/ioio/wiki/Analog-Input>
- GitHub. (2008). *Digital IO*. Obtenido de <https://github.com/ytai/ioio/wiki/Digital-IO>
- GitHub. (2008). *Getting To Know The Board*. Obtenido de <https://github.com/ytai/ioio/wiki/Getting-To-Know-The-Board>
- GitHub. (2008). *IOIO Over Bluetooth*. Obtenido de <https://github.com/ytai/ioio/wiki/IOIO-Over-Bluetooth>
- GitHub. (2008). *IOIOLibBasics*. Obtenido de <https://github.com/ytai/ioio/wiki/IOIOLib-Basics>
- GitHub. (2008). *Power Supply*. Obtenido de <https://github.com/ytai/ioio/wiki/Power-Supply>
- GitHub. (2008). *Power Supply*. Obtenido de <https://github.com/ytai/ioio/wiki/Power-Supply>
- GitHub. (2008). *Pwm Output*. Obtenido de <https://github.com/ytai/ioio/wiki/PWM-Output>
- GitHub. (2008). *SPI*. Obtenido de <https://github.com/ytai/ioio/wiki/SPI>
- GitHub. (2008). *TWI*. Obtenido de <https://github.com/ytai/ioio/wiki/TWI>
- GitHub. (2008). *UART*. Obtenido de <https://github.com/ytai/ioio/wiki/UART>
- Granadino, C., & Suárez, J. (s.f.). *El bus I2C*. Chile: Universidad Técnica Federico Santa María .
- Huerta, E., Manguiaterra, A., & Gustavo, N. (2005). *Posicionamiento satelital*. Argentina: A.U.G.M.
- IOIO for Android*. (s.f.). Obtenido de <https://www.sparkfun.com/products/10748>
- Jara, P., & Nazar, P. (s.f.). *Estándar IEEE 802.11 X de las WLAN*. Buenos Aires: Edutecne.
- Microchip. (2010). *PIC24FJ256DA210 FAMILY*. Microchip.
- Microchip. (2013). *MRF24WB0MA/MRF24WB0MB*.
- Molina, F. (s.f.). *Global positioning system*. España.

- Monk, S. (2012). *Making Android Accessories With IOIO* (Vol. I). O'Reilly Media.
- Pérez, E. L. (s.f.). Curso de Redes de Microcontroladores PIC (PROTOCOLO SPI). México: Ingeniería en Microcontroladores.
- Pérochon, S. (2012). *Android Guia de desarrollo de aplicaciones para smartphones y tabletas*. Barcelona: Ediciones ENI.
- Quectel. (2011). *L30 Quectel GPS Engine*. Shanghai: Quectel.
- Quectel. (2013). *Quectel L30 Compact GPS Module*.
- Raúl Esteve Bosch, J. F. (2005). *Fundamentos de electrónica digital*. Valencia.
- RobotFreak. (2008). *IOIO-Rover*. Obtenido de <http://letsmakerobots.com/node/33968>
- ROVING NETWORKS. (2011). *RN-XV Data Sheet*. Arizona: ROVING NETWORKS.
- sgoliver.net foro. (s.f.). *Estructura de un proyecto Android*. Obtenido de <http://www.sgoliver.net/blog/?p=1278>
- Texas Instrument. (2000). *LM35*.
- Texas Instrument. (2004). *LM-358 DUAL OPERATIONAL AMPLIFIERS*. Dallas: Texas Instrument.
- Ucontrol. (2008). Matrices de LEDs. *Ucontrol Electrónica General Pic's en particular*, 68.
- Universidad Politécnica de Valencia. (Abril de 2013). *Elementos de un proyecto Android*. Obtenido de <http://www.androidcurso.com/index.php/recursos-didacticos/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/148-elementos-de-un-proyecto-android>
- USERS. (s.f.). Microcontroladores funcionamiento programación y usos prácticos. *USERS*, 70-76.
- Valverde Rebaza, J. C. (2007). *El Estándar Inalámbrico ZigBee*. Trujillo: Perú.
- Valverde, J. (2007). *El Estándar Inalámbrico ZigBee*. Trujillo.
- Xatakandroid. (2005). *¿Qué es Android?* Obtenido de <http://www.xatakandroid.com/sistema-operativo/que-es-android>
- Zambrano, B. J. (2011). *DISEÑO E IMPLEMENTACIÓN DE UN KIT DE APLICACIONES PARAPERSONAS CON DISCAPACIDAD VISUAL UTILIZANDO LA*. Sangolqui.
- zhongzhouOpto. (s.f.). *SPECIFICATION FOR ZHONGZHOU LED LAMP* .

## Anexos

### Creación del proyecto en el software Eclipse.

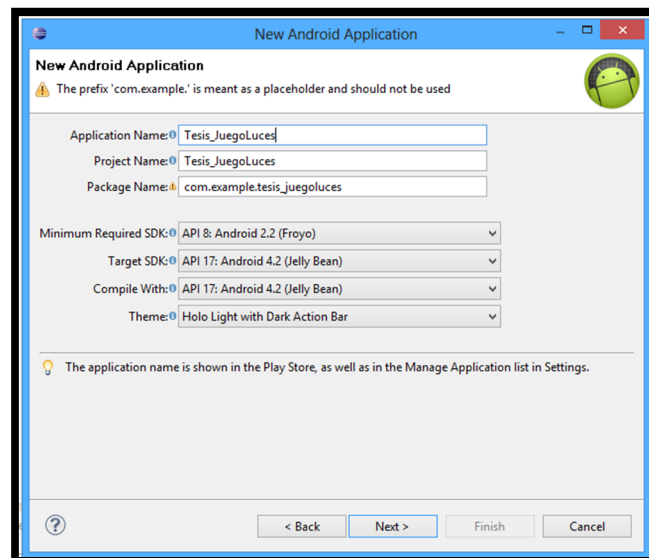
1. Crear un nuevo proyecto. File, New, Other, Android Application Project.



**Figura 55.** Creación de una nueva aplicación Android

**Referencia:** Elaborado por Byron Valenzuela

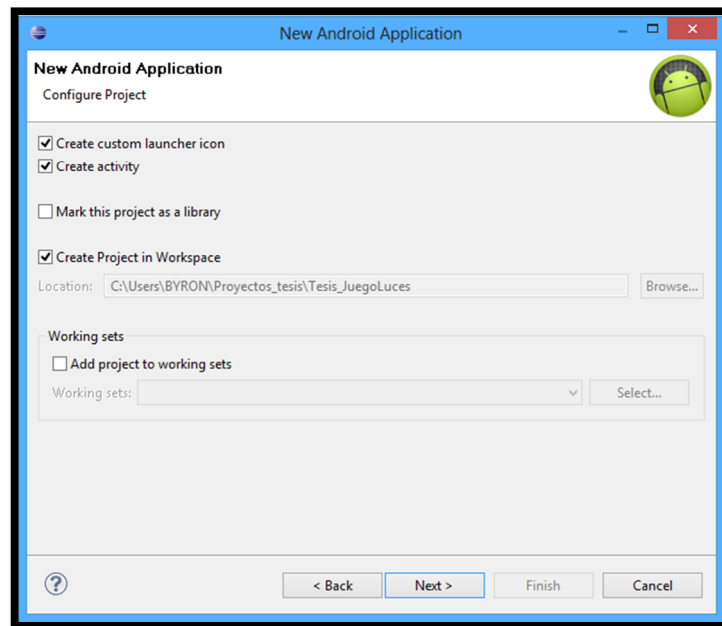
2. Llenar los campos correspondientes a la nueva aplicación.



**Figura 56.** Información correspondiente a la aplicación Android a crear

**Referencia:** Elaborado por Byron Valenzuela

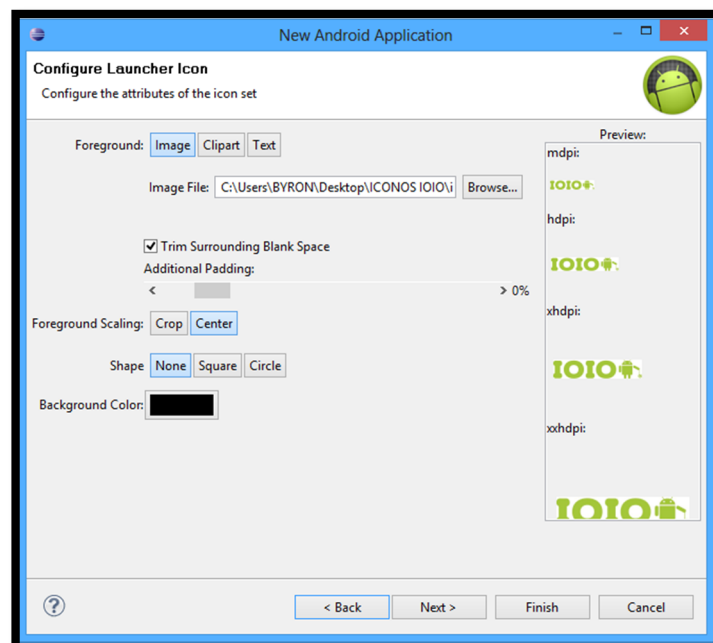
### 3. Configurar el nuevo proyecto.



**Figura 57.** Configuración del nuevo proyecto Android

**Referencia:** Elaborado por Byron Valenzuela

### 4. Establecer el icono que tendrá la aplicación.

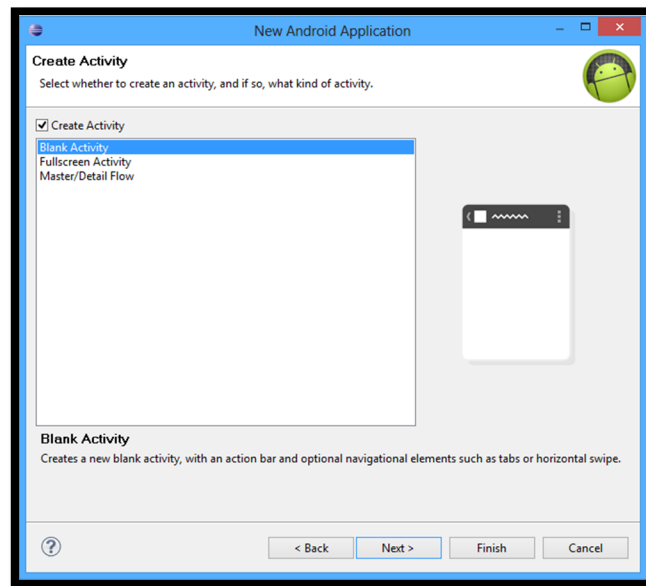


**Figura 58.** Configuración del icono del proyecto

**Referencia:** Elaborado por Byron Valenzuela



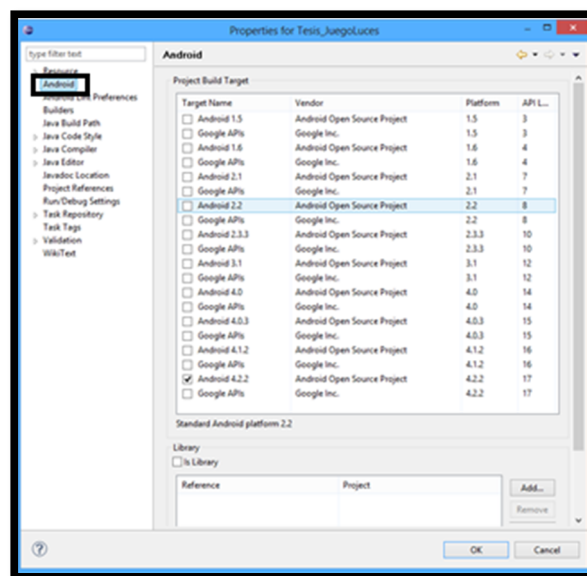
## 5. Elegir el tipo de formulario deseado.



**Figura 59.** Selección del tipo de formulario  
**Referencia:** Elaborado por Byron Valenzue

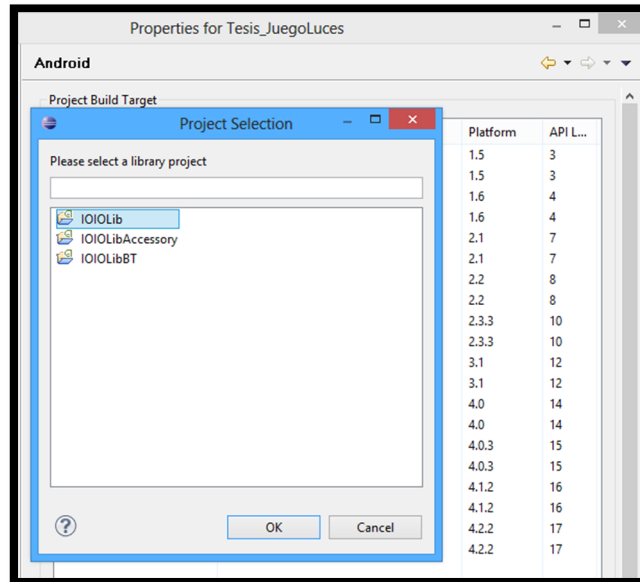
## Incluir librerías en el proyecto Android.

1. Click derecho sobre el proyecto a incluir librerías y acceder a la opción properties. En la parte izquierda colocar en la opción Android.



**Figura 60.** Propiedades Android del Proyecto  
**Referencia:** Elaborado por Byron Valenzuela

2. Dar click en la opción Add localizada en la parte inferior derecha. Y seleccionar las librerías IOIO disponibles (Nota: las librerías aparecerán siempre y cuando ya se haya descomprimido y ejecutado el paquete de librerías de la IOIO).



**Figura 61.** Librerías IOIO a incluir en el proyecto

**Referencia:** Elaborado por Byron Valenzuela

### Código de la aplicación en Eclipse.

// Juego de Luces Secuenciales, conectados al entrenador IOIO con 8 diodos led

**package** android.tesis.juegoluces;

**import** ioio.lib.api.DigitalOutput;

**import** ioio.lib.api.DigitalOutput.Spec;

**import** ioio.lib.api.exception.ConnectionLostException;

**import** ioio.lib.util.AbstractIOIOActivity;

**import** android.os.Bundle;

**import** android.tesis.juegoluces.R.id;

**import** android.view.View;

**import** android.view.View.OnClickListener;

**import** android.widget.Button;

```

public class JuegoLucesActivity extends AbstractIOActivity implements
OnClickListener {

    // Declaracion de los pines que van a ser utilizados en este programa

    private final int Led_0pin = 36;
    private final int Led_1pin = 37;
    private final int Led_2pin = 38;
    private final int Led_3pin = 39;
    private final int Led_4pin = 40;
    private final int Led_5pin = 41;
    private final int Led_6pin = 42;
    private final int Led_7pin = 43;

    /// Declaracion de la variable Button para enlazar con el boton del formulario

    private Button btnActivar;

    /// variables booleanas que definiran el estado de los leds

    private boolean led0state = false;
    private boolean led1state = false;
    private boolean led2state = false;
    private boolean led3state = false;
    private boolean led4state = false;
    private boolean led5state = false;
    private boolean led6state = false;
    private boolean led7state = false;
    private boolean bandera1 = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}

```

```

        // Atar los componentes del formulario del proyecto
        setContentView(R.layout.juego_luces_activity);

        btnActivar = (Button)findViewById(id.btnEncender);

        btnActivar.setOnClickListener(this);
    }

    class IOIOThread extends AbstractIOActivity.IOIOThread
    {
        // Declaracion de las variables DigitalOutput a ser utilizadas en el proyecto

        private DigitalOutput led0;

        private DigitalOutput led1;

        private DigitalOutput led2;

        private DigitalOutput led3;

        private DigitalOutput led4;

        private DigitalOutput led5;

        private DigitalOutput led6;

        private DigitalOutput led7;

        @Override

        protected void setup () throws ConnectionLostException
        {
            // Abrir y configurar el modo y estado inicial de los pines

            led0 = ioio_.openDigitalOutput(Led_0pin, Spec.Mode.OPEN_DRAIN, false); // Abrir
            el pin 4 y el estado inicial en 0 logico

            led1 = ioio_.openDigitalOutput(Led_1pin, Spec.Mode.OPEN_DRAIN, false);
            led2 = ioio_.openDigitalOutput(Led_2pin, Spec.Mode.OPEN_DRAIN, false);
            led3 = ioio_.openDigitalOutput(Led_3pin, Spec.Mode.OPEN_DRAIN, false);
            led4 = ioio_.openDigitalOutput(Led_4pin, Spec.Mode.OPEN_DRAIN, false);
            led5 = ioio_.openDigitalOutput(Led_5pin, Spec.Mode.OPEN_DRAIN, false);

            led6 = ioio_.openDigitalOutput(Led_6pin, false);

```

```

led7 = ioio_.openDigitalOutput(Led_7pin, false);
}
@Override
protected void loop() throws ConnectionLostException
{
// verificacion si se desea activar o desactivar el juego de luces
    if (bandera1 == true)
    {
// Encender el led0 correspondiente al pin 4 los demas apagados
        led0.write(true);
        led1.write(false);
        led2.write(false);
        led3.write(false);
        led4.write(true);
        led5.write(false);
        // retardo de 500 ms
        try {
            sleep(500);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
// Encender el led1 correspondiente al pin 5 los demas apagados
        led0.write(false);
        led1.write(true);
        led2.write(false);
        led3.write(false);
        try {

```

```

        sleep(500);
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

// Encender el led2 correspondiente al pin los demas apagados

led0.write(false);
led1.write(false);
led2.write(true);
led3.write(false);

try {
    sleep(500);
} catch (InterruptedException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

led0.write(false);
led1.write(false);
led2.write(false);
led3.write(true);

try {
    sleep(500);
} catch (InterruptedException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

} else

```

```

// caso contrario si se desea desactivar el juego de luces
{
// Apagar todos los leds, es decir poner en estado 0L a los pines
    // respectivos
    led0.write(false);
    led1.write(false);
    led2.write(false);
    led3.write(false);
}
try {
    sleep(100);

} catch (InterruptedException e) {
    // TODO: handle exception
}
}
}
@Override
protected AbstractIOActivity.IOIOThread createIOIOThread()
{
    return new IOIOThread();
}

@Override
public void onClick(View v) {
    // presiono el boton activar juego de luces ???
    bandera1 = !bandera1;
    if (bandera1==true)

```

```
{  
    btnActivar.setText(" Juego de luces activado");  
    // mensaje Juego de Luces Activado  
}else  
{  
    btnActivar.setText("Juego de luces desactivado");  
}  
}  
  
// TODO Auto-generated method stub  
}
```



#### 4.1.2 CONTADOR DE PULSOS

- **Tema: Contador de Pulsos**

- **Objetivos**

- **Objetivo General**

Desarrollar un contador de pulsos, a través de los pines de entrada digital 5V de la IOIO, para comprobar el módulo de entrada digital del entrenador.

- **Objetivos específicos**

- Conocer el funcionamiento de los pulsadores digitales.
- Investigar la forma de trabajo del módulo de entrada digital de la IOIO.
- Realizar el flujo grama respectivo para este caso de estudio.
- Plasmar el programa necesario para el contador de pulsos.
- Ejecutar el programa realizado, e implementarlo con el dispositivo Android y el entrenador.

- **Marco teórico**

##### **Pulsador digital electrónico.**

Pulsadores o botones electrónicos son dispositivos que permiten ingresar una señal digital en este caso a la tarjeta IOIO, la cual será capaz de percibir el estado en que estos se encuentran y tomar acciones según sea el caso, el entrenador cuenta con 4 pulsadores normalmente abiertos.

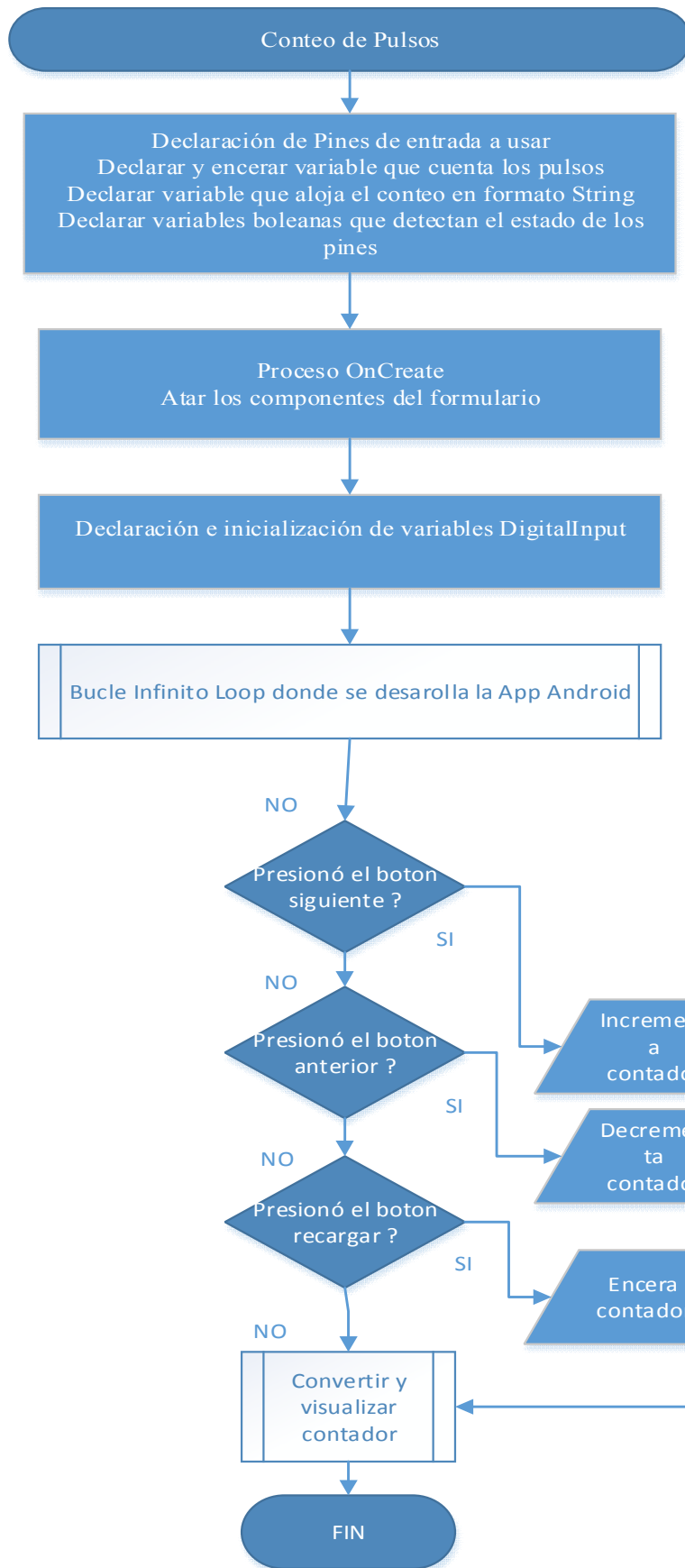
Los requerimientos para realizar esta aplicación son los siguientes:

**Tabla 16.** Requerimientos para la aplicación conteo de pulsos

<b>Dispositivo</b>	<b>Requerimientos</b>
<b>Pulsadores</b>	Tres de cuatro pulsadores con los que cuenta el entrenador
<b>Tarjeta IOIO</b>	Módulo de entrada digital 5V.
<b>Equipo móvil Android</b>	Equipo que interactúa con la IOIO
<b>Dispositivo Bluetooth o cable de comunicación usb del equipo Android</b>	Interfaz de comunicación IOIO – Dispositivo móvil Android.

**Referencia:** Elaborado por Byron Valenzuela

- **Flujo Grama**



- **Desarrollo**

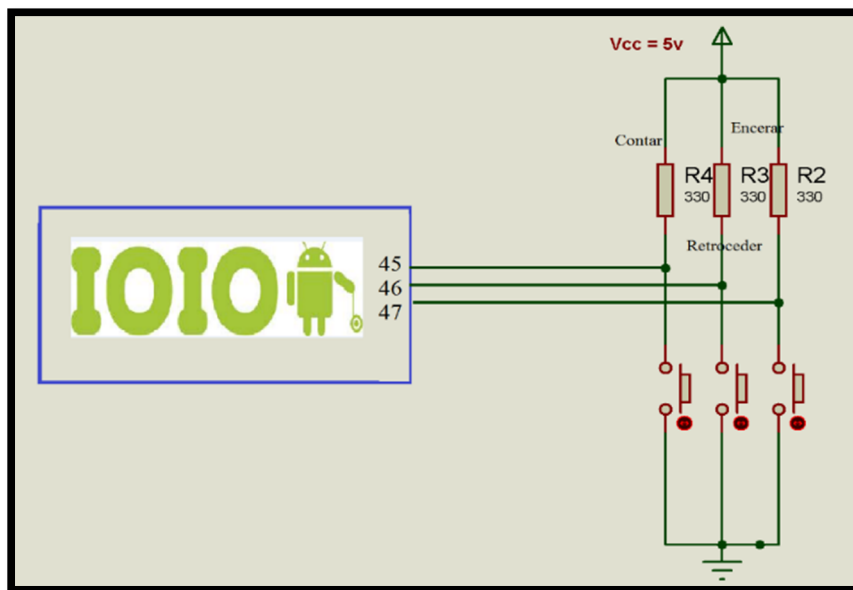
Enunciado: Crear una aplicación que permita verificar los pulsos provenientes de tres pulsadores, adelantar, retroceder y encerrar respectivamente. Para esta aplicación hacer uso de la entrada digital a 5V.

**Tabla 17.** Descripción de los pines y módulos usados en la aplicación

Placa IOIO	Pines a usar: 45(Adelantar), 46 (Retroceder), 47 (Encerrar).
ENTRENADOR	Se usa tres pulsadores aleatorios.

**Referencia:** Elaborado por Byron Valenzuela

A continuación se presenta una descripción gráfica de la conexión que tiene la aplicación.



**Figura 62.** Diagrama de conexión de la aplicación contador de pulsos

**Referencia:** Elaborado por Byron Valenzuela

- **Análisis de resultados**

Al utilizar un multímetro para analizar el circuito formado por el pulsador y la placa IOIO se obtuvo los siguientes resultados.

**Tabla 18.** Tabla de resultados Contador de Pulsos

	<b>Voltaje (V)</b>	<b>Corriente (I)</b>	<b>Resistencia (R)</b>
<b>Circuito Abierto</b>	0V	15mA	330 $\Omega$
<b>Circuito Cerrado</b>	5V	15mA	330 $\Omega$

**Referencia:** Elaborado por Byron Valenzuela

- **Conclusiones**

- El módulo de entrada digital de la IOIO es capaz de percibir dos valores como uno lógico 5V y 3.3V respectivamente según los o el pin que se esté usando.
- A través de un objeto de la clase DigitalInput, se accede a declarar e inicializar el pin de entrada digital.
- El método waitForValue se mantiene en espera hasta leer el dato que se desea recibir.
- Los pulsadores electrónicos envían señales digitales al periférico al que se encuentran conectados.

- **Recomendaciones**

- Declarar el estado al que van a estar conectados los pines inicialmente, caso contrario la IOIO no reaccionara y la aplicación fallará.
- Verificar el nivel de voltaje que pueden soportar los pines a escoger para realizar una aplicación de entrada digital.

- **Anexos**

**Como anexo se adjunta el código del proyecto.**

```
// Programa que permite contar los pulsos que se aplican a la IOIO
// con la capacidad de retroceder, adelantar y encerrar el contador

package tesis.conteopulsos;

import ioio.lib.api.DigitalInput;

import ioio.lib.api.exception.ConnectionLostException;

import ioio.lib.util.AbstractIOActivity;

import tesis.conteopulsos.R.id;

import android.os.Bundle;

import android.widget.TextView;

public class ConteoActivity extends AbstractIOActivity {

    // Pines que detectan el conteo de pulsos

    private final int contar = 45;

    private final int retroceder = 46;

    private final int reiniciar = 47;

    // Variable que cuenta los pulsos

    private int conta = 0;

    // Variable que contendra el numero de pulsos en formato Sting

    private String bandera = null;

    // Variable Text View a ser atada en el formulario

    private TextView tvContador1;

    // Variables que definiran el estado de los pulsadores

    private boolean estado,estado1,estado2;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    // atar los componentes del formulario
```

```
    setContentView(R.layout.conteo_activity);
```

```
    tvContador1 = (TextView)findViewById(id.tvContador);
```

```
}
```

```
class IOIOThread extends AbstractIOActivity.IOIOThread
```

```
{
```

```
    // Variables De entrada digital siguiente, a tras y encerrar
```

```
    private DigitalInput next;
```

```
    private DigitalInput previous;
```

```
    private DigitalInput reload;
```

```
    @Override
```

```
    protected void setup () throws ConnectionLostException
```

```
{
```

```
    try {
```

```
        // Inicializar los pines de entrada digital
```

```
        next = ioio_.openDigitalInput(contar,DigitalInput.Spec.Mode.PULL_UP);
```

```
        previous = ioio_.openDigitalInput(retroceder,DigitalInput.Spec.Mode.PULL_UP);
```

```
        reload = ioio_.openDigitalInput(reiniciar,DigitalInput.Spec.Mode.PULL_UP);
```

```
    } catch (ConnectionLostException e) {
```

```
        throw e;
```

```

    }
}

@Override

protected void loop () throws ConnectionLostException
{
    try {
        // leer el estado de los pines.

        estado = next.read();

        estado1 = previous.read();

        estado2 = reload.read();

        // presiono el pulsador siguiente ??

        if (estado == false)
        {
            sleep(200);

            // Espera hasta que el pulsador vuelva a la posicion normal

            next.waitForValue(true);

            // Incrementa contador

            conta = conta + 1;

        }

        // presiono el pulsador regresar ??

        if (estado1 == false)
        {
            sleep(200);

```

```

        previous.waitForValue(true);

        conta = conta - 1;
    }

    // presiono el pulsador encerar??

    if (estado2 == false)
    {

        sleep(200);

        reload.waitForValue(true);

        conta = 0;
    }

    // Convertir el numero conta a una variable string

    bandera = Integer.toString(conta);

    // SubProceso que asigna al textView el numero convertido

    setText(bandera);

    sleep(10);

} catch (InterruptedException e) {

    ioio_.disconnect();

} catch (ConnectionLostException e) {

    throw e;

}

}

}

```



```

@Override

protected AbstractIOActivity.IOIOThread createIOIOThread()
{
    return new IOIOThread();
}

public void setText (final String str1) {

runOnUiThread(new Runnable() {

@Override

public void run() {

    // Asigna al TextView una variable string

    tvContador1.setText(str1);

}

});

}

}

```

### 4.1.3 CONTADOR

- **Tema: Contador decimal dos dígitos con alarma.**
- **Objetivos**

- **Objetivo general**

Contador decimal de dos dígitos de 0-99 con alarma sucesiva en intervalo de 10 números, mediante el uso de los pines de salida digital a 3.3V y 5V, con el objetivo de mostrar el funcionamiento de salida digital del entrenador.

- **Objetivos específicos**

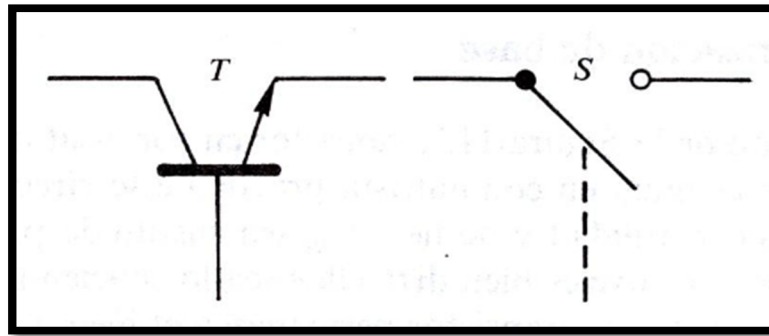
- Conocer el funcionamiento de los display 7 segmentos ánodo común, así como de los decodificadores para dichos display's.
- Investigar los parámetros de funcionamiento del buzzer electrónico.
- Comprender la forma de trabajo del transistor, en la zona de corte y saturación.
- Desarrollar el flujo grama respectivo.
- Escribir el código necesario para esta aplicación.
- Ejecutar el programa realizado, e implementarlo con el dispositivo Android y el entrenador.
- Comprobar el funcionamiento tanto del programa cargado en el dispositivo Android, como del entrenador electrónico.

- **Marco teórico**

#### **Transistor en Corte – Saturación**

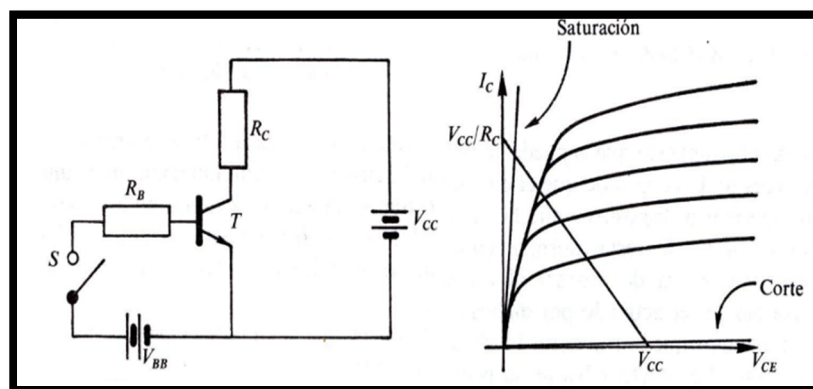
El transistor posee tres regiones de trabajo las cuales son: lineal o amplificación, corte y saturación. Para esta aplicación se trabaja en la región de corte y saturación, básicamente el funcionamiento de un transistor en corte y saturación es similar a un interruptor electrónico.

Cuando el transistor se encuentra en la región de corte, se asimila a un interruptor abierto, y cuando se halla en la región de saturación es similar a un interruptor cerrado. Las regiones de corte y saturación las determina el circuito asociado con la base. Se puede tener la siguiente relación.



**Figura 63.** Relación entre un transistor y un interruptor  
**Referencia:** Transistor como interruptor, (A. Carretero, 2009)

A continuación se presenta un circuito de polarización que trabaja en corte y saturación.



**Figura 64.** Circuito de polarización, zona de corte y saturación del transistor  
**Referencia:** Transistor como interruptor, (A. Carretero, 2009)

Si el interruptor S está abierto  $I_b = 0$ , esto quiere decir que el transistor permanece en corte, y que  $I_c = 0$  y  $V_{ce} = V_{cc}$ ; cuando el transistor se encuentra cerrado, existe suficiente corriente de base, y el transistor trabaja en la zona de saturación y por ende  $I_c = V_{cc}/R_c$  y  $V_{ce} = 0$  (A. Carretero, 2009).

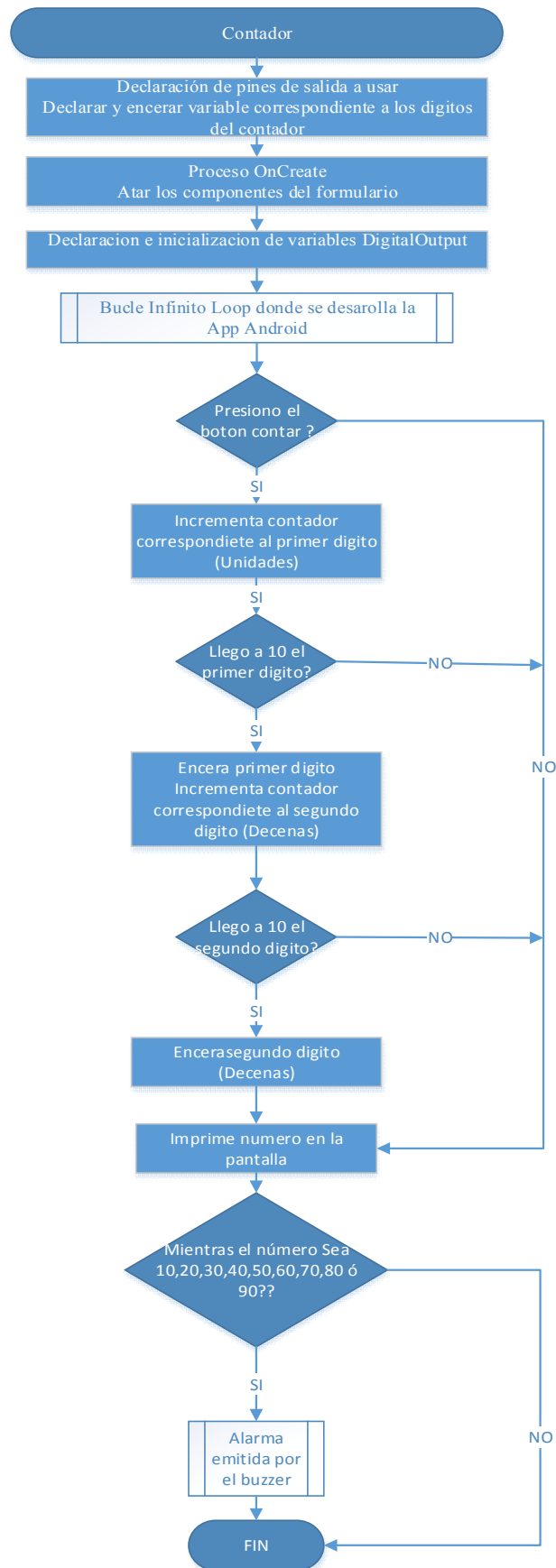
Para esta aplicación se requiere de los siguientes componentes:

**Tabla 19.** Requerimientos para la aplicación contador

<b>Dispositivo</b>	<b>Requerimientos</b>
<b>Display</b>	Dos display's ánodo común.
<b>Decodificadores</b>	Dos decodificadores 74ls47 para ánodo común.
<b>Buzzer</b>	Un buzzer o chicharra electrónica
<b>Tarjeta IOIO</b>	Módulo de salida digital 3.3V y 5V.
<b>Equipo móvil Android</b>	Equipo que interactúa con la IOIO
<b>Dispositivo Bluetooth o cable de comunicación usb del equipo Android</b>	Interfaz de comunicación IOIO – Dispositivo móvil Android.

**Fuente:** Elaborado por Byron Valenzuela

- Flujo grama



- Desarrollo

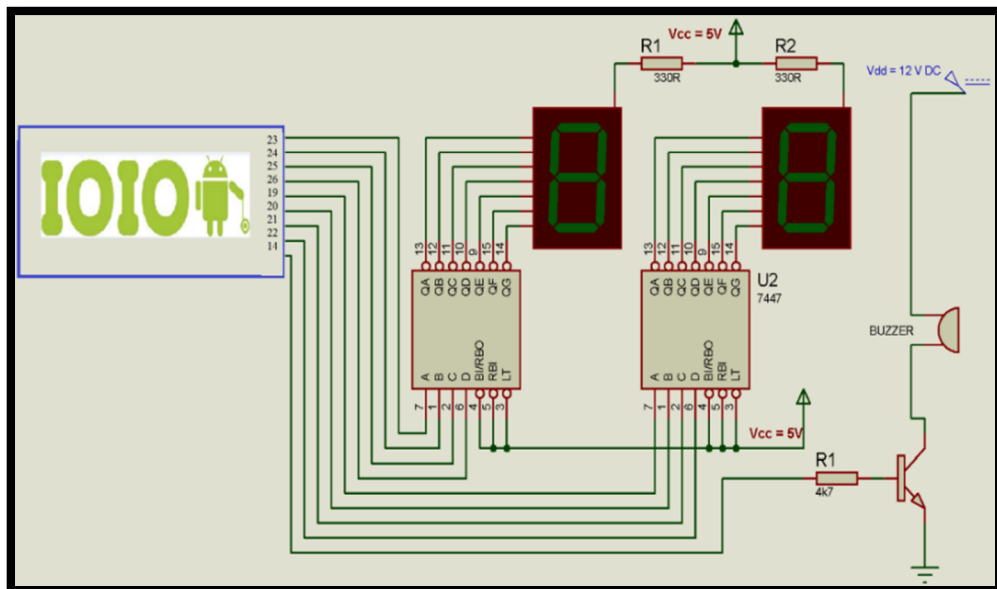
**Enunciado:** Elaborar un contador decimal 0-99, con alarma sonora en un intervalo de diez números.

**Tabla 20.** Descripción de los pines usados en la aplicación contador

<b>Placa IOIO</b>	Pines a usar: digito1 ( A0,A1,A2,A3, 19,20,21,22 respectivamente), digito2 (A0,A1,A2,A3, 23,24,25,26, respectivamente). Buzzer pin 14 compatible con 5V.
<b>ENTRENADOR</b>	Dos display 7 segmentos ánodo común con sus respectivos decodificadores, transistor 2N3904 y el buzzer.

**Referencia:** Elaborado por Byron Valenzuela

En la siguiente figura se observa el diagrama de conexión de los display's y buzzer con la placa IOIO.



**Figura 65.** Diagrama de conexión de la aplicación contador

**Fuente:** Elaborado por Byron Valenzuela

- **Análisis de resultados**

**Tabla 21.** Tabla de resultados de la aplicación contador

Ítem	Corte	Saturación
Ib	0mA	~1mA
Vce	10V	0V

**Fuente:** Elaborado por Byron Valenzuela

- **Conclusiones**

- A través del transistor en corte y saturación se crea una interfaz con la que se logra la conexión de dispositivos electrónicos con voltaje de alimentación superiores a los 5V.
- Los decodificadores 74LS194 permiten controlar los display 7 segmentos con los que cuenta el entrenador.
- Los decodificadores son capaces de reconocer la lógica de 3.3V con los que trabaja la IOIO perfectamente.

- **Recomendaciones**

- No conectar el buzzer directamente al pin de la IOIO, ya que se puede causar el daño parcial o total de un pin o de la placa entera.
- Con el objetivo de reducir resistencias, se recomienda conectar una sola resistencia al ánodo común de los displays.

- **Bibliografía**

mikroElektronika. (2012). *mikroBUS pinout Standar Especification*.

A. Carretero, F. F. (2009). *Electrónica*. Editex.

Agüero, R. (s.f.). *Redes Inalámbricas de Área Local y Personal WLAN El estándar IEEE 802.11*. Cantabria.

Argüello, D. J. (2012). *Desarrollo de una aplicación que permita la captura almacenamiento, reproducción, administración y envío de archivos de vídeo, audio e imágenes utilizando la tecnología bluetooth, para dispositivos móviles basados en el sistema operativo Android*. Quito.

- Carballar, J. (2010). *WI-FI lo que necesita conocer*. Madrid: RC Libros.
- Carrillo Pérez, M. d. (2006). *Estudio y análisis de rendimiento Bluetooth*. Madrid.
- Castillo, D. (2012). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ALARMA COMUNITARIA A BASE DE MÓDULOS INALÁMBRICOS UTILIZANDO TECNOLOGÍA ZIGBEE*. Ibarra.
- Clanar Internacional. (s.f.). *Internet y redes inalámbricas*. Arequipa: Clanar.
- Collaguazo, G. (2009). *Sistemas Basados en Microprocesadores*. Ibarra.
- Correia, P. (2002). *Guía práctica del GPS*. Barcelona: Marcombo.
- Creative Commons . (s.f.). *Arduino*. Obtenido de <http://arduino.cc/en/Main/CopyrightNotice>
- Daniel Benchimol. (2011). *Microcontroladores*. Buenos Aires: DALAGA S.A.
- Digi International, Inc. (2009). *XBee®/XBee-PRO® RF Modules*. New York.
- Dignani, J. (2011). *Análisis del protocolo ZigBee*.
- Dignani, J. P. (2011). *Análisis del protocolo ZigBee*.
- Fairchild Semiconductor. (1999). *MM74C922 • MM74C923*.
- FAIRCHILD. (2000). *DM74LS47 BCD to 7-Segment Decoder/Driver with Open-Collector Outputs*. FAIRCHILD SEMICONDUCTOR.
- Fernández, A. M. (2004). *EL BUS I2C*. Córdoba.
- GARCÍA CELI, H. M., & SANTILLÁN LARA, L. A. (2005). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE TARIFACIÓN PARA LOCUTORIOS*. Quito.
- GitHub. (2008). *Analog Input*. Obtenido de <https://github.com/ytai/ioio/wiki/Analog-Input>
- GitHub. (2008). *Digital IO*. Obtenido de <https://github.com/ytai/ioio/wiki/Digital-IO>
- GitHub. (2008). *Getting To Know The Board*. Obtenido de <https://github.com/ytai/ioio/wiki/Getting-To-Know-The-Board>
- GitHub. (2008). *IOIO Over Bluetooth*. Obtenido de <https://github.com/ytai/ioio/wiki/IOIO-Over-Bluetooth>
- GitHub. (2008). *IOIOLibBasics*. Obtenido de <https://github.com/ytai/ioio/wiki/IOIOLib-Basics>
- GitHub. (2008). *Power Supply*. Obtenido de <https://github.com/ytai/ioio/wiki/Power-Supply>
- GitHub. (2008). *Power Supply*. Obtenido de <https://github.com/ytai/ioio/wiki/Power-Supply>
- GitHub. (2008). *Pwm Output*. Obtenido de <https://github.com/ytai/ioio/wiki/PWM-Output>
- GitHub. (2008). *SPI*. Obtenido de <https://github.com/ytai/ioio/wiki/SPI>



GitHub. (2008). *TWI*. Obtenido de <https://github.com/ytai/ioio/wiki/TWI>

GitHub. (2008). *UART*. Obtenido de <https://github.com/ytai/ioio/wiki/UART>

Granadino, C., & Suárez, J. (s.f.). *El bus I2C*. Chile: Universidad Técnica Federico Santa María .

Huerta, E., Manguiaterra, A., & Gustavo, N. (2005). *Posicionamiento satelital*. Argentina: A.U.G.M.

*IOIO for Android*. (s.f.). Obtenido de <https://www.sparkfun.com/products/10748>

Jara, P., & Nazar, P. (s.f.). *Estándar IEEE 802.11 X de las WLAN*. Buenos Aires: Edu Tecne.

Microchip. (2010). *PIC24FJ256DA210 FAMILY*. Microchip.

Microchip. (2013). *MRF24WB0MA/MRF24WB0MB*.

Molina, F. (s.f.). *Global positioning system*. España.

Monk, S. (2012). *Making Android Accessories With IOIO* (Vol. 1). O'Reilly Media.

Pérez, E. L. (s.f.). *Curso de Redes de Microcontroladores PIC (PROTOCOLO SPI)*. México: Ingeniería en Microcontroladores.

Pérochon, S. (2012). *Android Guía de desarrollo de aplicaciones para smartphones y tabletas*. Barcelona: Ediciones ENI.

Quectel. (2011). *L30 Quectel GPS Engine*. Shanghai: Quectel.

Quectel. (2013). *Quectel L30 Compact GPS Module*.

Raúl Esteve Bosch, J. F. (2005). *Fundamentos de electrónica digital*. Valencia.

RobotFreak. (2008). *IOIO-Rover*. Obtenido de <http://letsmakerobots.com/node/33968>

ROVING NETWORKS. (2011). *RN-XV Data Sheet*. Arizona: ROVING NETWORKS.

sgoliver.net foro. (s.f.). *Estructura de un proyecto Android*. Obtenido de <http://www.sgoliver.net/blog/?p=1278>

Texas Instrument. (2000). *LM35*.

Texas Instrument. (2004). *LM-358 DUAL OPERATIONAL AMPLIFIERS*. Dallas: Texas Instrument.

Ucontrol. (2008). *Matrices de LEDs. Ucontrol Electrónica General Pic's en particular, 68*.

Universidad Politécnica de Valencia. (Abril de 2013). *Elementos de un proyecto Android*. Obtenido de <http://www.androidcurso.com/index.php/recursos-didacticos/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/148-elementos-de-un-proyecto-android>

USERS. (s.f.). *Microcontroladores funcionamiento programación y usos prácticos. USERS, 70-76*.

Valverde Rebaza, J. C. (2007). *El Estándar Inalámbrico ZigBee*. Trujillo: Perú.

Valverde, J. (2007). *El Estándar Inalámbrico ZigBee*. Trujillo.

Xatakandroid. (2005). *¿Qué es Android?* Obtenido de <http://www.xatakandroid.com/sistema-operativo/que-es-android>

Zambrano, B. J. (2011). *DISEÑO E IMPLEMENTACIÓN DE UN KIT DE APLICACIONES PARAPERSONAS CON DISCAPACIDAD VISUAL UTILIZANDO LA*. Sangolqui.

zhongzhouOpto. (s.f.). *SPECIFICATION FOR ZHONGZHOU LED LAMP* .

- **Anexos**

**A continuación se adjunta el código de la aplicación.**

```
// Contador decimal de dos digitos
```

```
// con alarma sonora en un intervalo de 10 numeros
```

```
package com.example.tesis_contador;
```

```
import io.io.lib.api.DigitalOutput;
```

```
import io.io.lib.api.DigitalOutput.Spec;
```

```
import io.io.lib.api.exception.ConnectionLostException;
```

```
import io.io.lib.util.AbstractIOActivity;
```

```
import android.os.Bundle;
```

```
import android.view.View;
```

```
import android.view.View.OnClickListener;
```

```
import android.widget.Button;
```

```
import android.widget.TextView;
```

```
public class ContadorActivity extends AbstractIOActivity implements OnClickListener  
{
```

```
    // Variable que identifica al pin usado para el buzzer
```

```
    private final int pin_buzzer = 14;
```

```
    // Variables que identifican los bits que componen el primer digito
```

```
    private final int bit0 = 19;
```

```
    private final int bit1 = 20;
```

```
    private final int bit2 = 21;
```

```

private final int bit3 = 22;
// Variables que identifican los bits que componen el segundo digito
private final int bit4 = 23;
private final int bit5 = 24;
private final int bit6 = 25;
private final int bit7 = 26;
// Variable que cuenta el progreso de cada digito
private int conta, conta1 = 0;
// Variables a ser atadas en el formulario
private Button BtnActivar;
private TextView TvNumero;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.contador_activity);
    BtnActivar = (Button)findViewById(R.id.btnContar); // Ata componentes del
TvNumero = (TextView)findViewById(R.id.tvNumero); // formulario
    BtnActivar.setOnClickListener(this);
}

class IOIOThread extends AbstractIOActivity.IOIOThread
{
    // Salida digital que va ser el buzzer
    private DigitalOutput buzzer;
    // Variables que definen las salidas digitales de cada digito
    private DigitalOutput A0;
    private DigitalOutput A1;
    private DigitalOutput A2;
    private DigitalOutput A3;
    //////////////////////////////////////
    private DigitalOutput B0;
    private DigitalOutput B1;
    private DigitalOutput B2;
}

```

```

private DigitalOutput B3;

@Override
protected void setup() throws ConnectionLostException
{
// Inicializa las salidas digitales tanto el buuzer como los
// bits de los dos digitos
buzzer=ioio_.openDigitalOutput(pin_buzzer,Spec.Mode.OPEN_DRAIN, false);
A0=ioio_.openDigitalOutput(bit0, false);
A1=ioio_.openDigitalOutput(bit1, false);
A2=ioio_.openDigitalOutput(bit2, false);
A3=ioio_.openDigitalOutput(bit3, false);

B0=ioio_.openDigitalOutput(bit4, false);
B1=ioio_.openDigitalOutput(bit5, false);
B2=ioio_.openDigitalOutput(bit6, false);
B3=ioio_.openDigitalOutput(bit7, false);

}

@Override
protected void loop() throws ConnectionLostException
{
// Si el conta correspondiente al digito 1 es decir la unidad
// esta en 0
// despliegue el numero 0
// en el display

if (conta == 0)
{
A0.write(false);
A1.write(false);
A2.write(false);
A3.write(false);
}
}

```

```
// De la misma manera el numero 1 y asi sucesivamente hasta el
// numero 9 que es el primer digito
```

```
    if (conta == 1)
    {
        A0.write(true);
        A1.write(false);
        A2.write(false);
        A3.write(false);
    }
    if (conta == 2)
    {
        A0.write(false);
        A1.write(true);
        A2.write(false);
        A3.write(false);
    }
    if (conta == 3)
    {
        A0.write(true);
        A1.write(true);
        A2.write(false);
        A3.write(false);
    }
    if (conta == 4)
    {
        A0.write(false);
        A1.write(false);
        A2.write(true);
        A3.write(false);
    }
    if (conta == 5)
    {
        A0.write(true);
        A1.write(false);
        A2.write(true);
```

```

        A3.write(false);
    }
    if (conta == 6)
    {
        A0.write(false);
        A1.write(true);
        A2.write(true);
        A3.write(false);
    }
    if (conta == 7)
    {
        A0.write(true);
        A1.write(true);
        A2.write(true);
        A3.write(false);
    }
    if (conta == 8)
    {

        A0.write(false);
        A1.write(false);
        A2.write(false);
        A3.write(true);
    }
    if (conta == 9)
    {

        A0.write(true);
        A1.write(false);
        A2.write(false);
        A3.write(true);
    }
}

```

// Si el conta 1 correspondiente al digito 2 es decir la decena

// esta en 0

// despliegue el numero 0

```

// en el display
    if (conta1 == 0)
    {

        B0.write(false);
        B1.write(false);
        B2.write(false);
        B3.write(false);

    }

// De la misma manera el numero 1 y asi sucesivamente hasta el
// numero 9 que es el primer digito
    if (conta1 == 1)
    {

        B0.write(true);
        B1.write(false);
        B2.write(false);
        B3.write(false);

    }

    if (conta1 == 2)
    {

        B0.write(false);
        B1.write(true);
        B2.write(false);
        B3.write(false);

    }

    if (conta1 == 3)
    {

        B0.write(true);
        B1.write(true);
        B2.write(false);
        B3.write(false);

    }

```

```
if (conta1 == 4)
{

    B0.write(false);
    B1.write(false);
    B2.write(true);
    B3.write(false);
}
if (conta1 == 5)
{

    B0.write(true);
    B1.write(false);
    B2.write(true);
    B3.write(false);
}
if (conta1 == 6)
{

    B0.write(false);
    B1.write(true);
    B2.write(true);
    B3.write(false);
}
if (conta1 == 7)
{

    B0.write(true);
    B1.write(true);
    B2.write(true);
    B3.write(false);
}
if (conta1 == 8)
{

    B0.write(false);
```



```

        B1.write(false);
        B2.write(false);
        B3.write(true);
    }
    if (conta1 == 9)
    {

        B0.write(true);
        B1.write(false);
        B2.write(false);
        B3.write(true);
    }
    // verifica si llego el display a 10
    // suena el buzzer
    // hats cambiar de numero

    if (conta1 == 1 && conta == 0)

    {

        do
        {

            buzzer.write(false); //enciende el buzzer
            try {
                sleep(500);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            buzzer.write(true); // apaga el buzzer

            try {
                sleep(500);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }

```

```

    }while ((conta1 == 1 && conta == 0));

}else
{
    buzzer.write(true);
}
// verifica si llego el display a 20
// suena el buzzer
// hats cambiar de numero

if (conta1==2 && conta == 0)
{
    do
    {
        buzzer.write(false);
        try {
            sleep(500);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        buzzer.write(true);

        try {
            sleep(500);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }while (conta1==2 && conta == 0);

}else
{
    buzzer.write(true);
}

```

```

// verifica si llego el display a 30
// suena el buzzer
// hats cambiar de numero

    if (conta1==3 && conta == 0)
    {
        do
        {

            buzzer.write(false);
            try {
                sleep(500);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            buzzer.write(true);

            try {
                sleep(500);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

        }while (conta1==3 && conta == 0);

    }else
    {
        buzzer.write(true);
    }
// Y asi sucesivamente hasta llegar al 90
    if (conta1==4 && conta == 0)
    {
        do

```

```

    {

        buzzer.write(false);
        try {
            sleep(500);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        buzzer.write(true);

        try {
            sleep(500);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        }while (conta1==4 && conta == 0);

    }else
    {
        buzzer.write(true);
    }

    if (conta1==5 && conta == 0)
    {
        do
        {

            buzzer.write(false);
            try {
                sleep(500);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }

```

```

    buzzer.write(true);

    try {
        sleep(500);
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    while (conta1==5 && conta == 0);

} else
{
    buzzer.write(true);
}

if (conta1==6 && conta == 0)
{
    do
    {

        buzzer.write(false);
        try {
            sleep(500);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        buzzer.write(true);

        try {
            sleep(500);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}

```

```

        }while (conta1==6 && conta == 0);

    }else
    {
        buzzer.write(true);
    }

    if (conta1==7 && conta == 0)
    {
        do
        {
            buzzer.write(false);
            try {
                sleep(500);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            buzzer.write(true);

            try {
                sleep(500);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

        }while (conta1==7 && conta == 0);

    }else
    {
        buzzer.write(true);
    }

```

```

if (conta1==8 && conta == 0)
{
    do
    {
        buzzer.write(false);
        try {
            sleep(500);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        buzzer.write(true);

        try {
            sleep(500);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }while (conta1==8 && conta == 0);

}else
{
    buzzer.write(true);
}

if (conta1==9 && conta == 0)
{
    do
    {
        buzzer.write(false);
        try {
            sleep(500);
        } catch (InterruptedException e) {

```

```

        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    buzzer.write(true);

    try {
        sleep(500);
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    }while (conta1==9 && conta == 0);

    }else
    {
        buzzer.write(true);
    }
try {
    sleep(100);

    } catch (InterruptedException e) {
        // TODO: handle exception
    }

}

}

@Override
protected AbstractIOActivity.IOIOThread createIOIOThread()
{
    return new IOIOThread();
}

// metodo a ser atado al boton contar
@Override

```



```

public void onClick(View v) {
    // incrementa contador
    conta = conta + 1;
    // primer digito llega a 9??
    if (conta == 10)
    {
        conta =0;// encera primer digito
        conta1 = conta1+1;// incrementa segundo digito
    }
    // segundo digito llego a 9??
    if (conta1== 10)
    {
        // encera segundo digito
        conta1=0;

    }
    // imprime "Numero actual + los contadores de los dos digitos "
    // en el TextView
    TvNumero.setText("Numero actual " + conta1 + conta );
    // TODO Auto-generated method stub

}

}

```

#### 4.1.4 LETRERO DINÁMICO

- **Tema: Letrero digital “Hola lolo” dinámico en la matriz de led’s**

- **Objetivos**

- Objetivos general

Implementar un letrero digital dinámico con el mensaje “Hola lolo” a través de una matriz de led’s, con el propósito de demostrar el funcionamiento del módulo de salida digital de la IOIO.

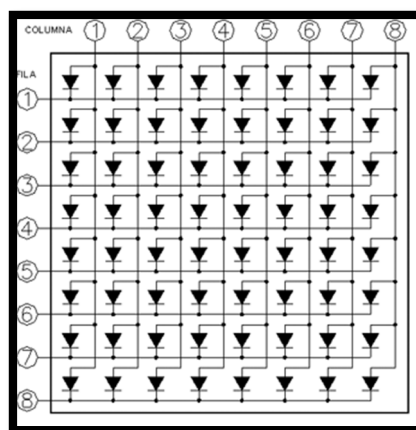
- Objetivos específicos

- Conocer la estructura y funcionamiento de la matriz de leds.
- Desarrollar el flujo grama respectivo.
- Escribir el código necesario para esta aplicación.
- Ejecutar el programa realizado, e implementarlo con el dispositivo Android y el entrenador.
- Comprobar el funcionamiento tanto del programa cargado en el dispositivo Android, como del entrenador electrónico.

- **Marco teórico**

##### **Matriz de led’s**

La matriz o arreglo de led’s es un finito número de diodos leds ordenamos en filas y columnas, de modo que todos los ánodos y cátodos comparten una conexión en común. Cada intersección de estas representa un pixel.



**Figura 66.** Diagrama de conexión de una matriz de leds de 8x8

Fuente: Matriz de leds 8x8. Recuperado de: <http://www.arduteka.com/2012/02/tutorial-arduino-0008-matriz-led-8x8-bicolor-74ch595/>

Al referirse a una matriz de NxM ejemplo 8x7 se está refiriendo a una matriz de Nfilas x Mcolumnas donde el producto de estos dos valores da como resultado el número de led's que se está manejando.

Un aspecto muy importante al momento de manejar la matriz de led's es saber diferenciar el formato de trabajo. En una matriz donde el número de filas sea diferente al número de columnas este aspecto no inmiscuiría ya que se diferenciaría si las filas son ánodos o cátodos, pero en una matriz donde se tenga el mismo número tanto de filas y columnas, se debería distinguir y escoger cuales van a ser los ánodos o los cátodos las filas o columnas.

El funcionamiento de la matriz se basa en el encendido y apagado de los led's, aplicando un nivel de voltaje 1 y 0 respectivamente en ánodo y cátodo se consigue que el led correspondiente a la fila y columna donde se aplicó la tensión se encienda

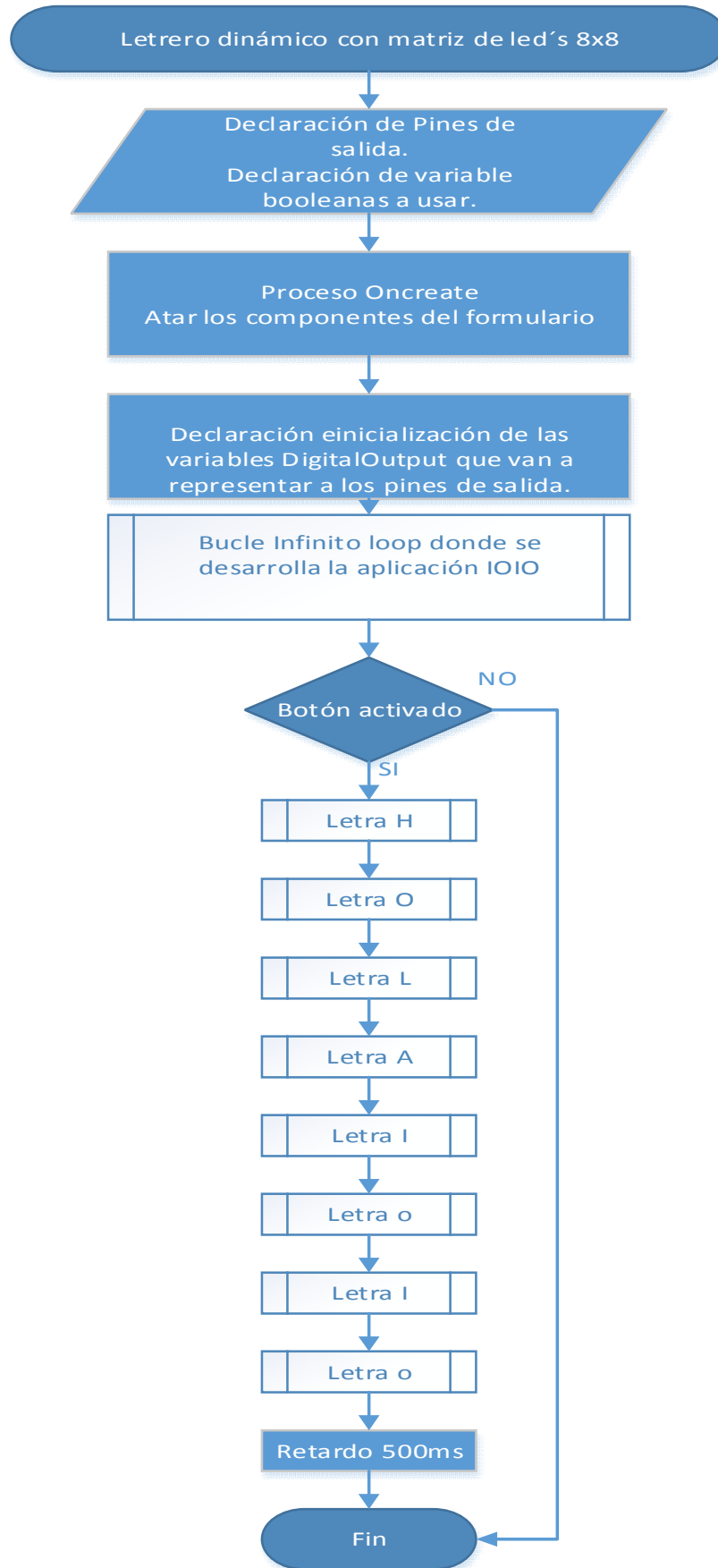
Para esta aplicación son necesarios los siguientes requerimientos.

**Tabla 22.** Requerimientos para la aplicación letrero dinámico

<b>Dispositivo</b>	<b>Requerimientos</b>
<b>Matriz de led's</b>	Matriz de led's 8x8
<b>Tarjeta IOIO</b>	Módulo de salida digital 3.3V
<b>Equipo móvil Android</b>	Equipo que interactúa con la IOIO
<b>Dispositivo Bluetooth o cable de comunicación usb del equipo Android</b>	Interfaz de comunicación IOIO – Dispositivo móvil Android.

**Fuente:** Elaborado por Byron Valenzuela.

- Flujo grama



- **Desarrollo**

Enunciado: Crear un letrero dinámico de izquierda a derecha con la frase “HOLA lolo” mediante la matriz de led’s.

Esta aplicación está realizada mediante el método de barrido de leds en las filas y las columnas respectivamente. El barrido consiste en prender y apagar los leds sin que la vista humana perciva el cambio, dando la sensación que la letra o figura está encendida.

La matriz del entrenador esta configurado de manera que las filas son los ánodos de los led’s de la matriz y las columnas los cátodos.

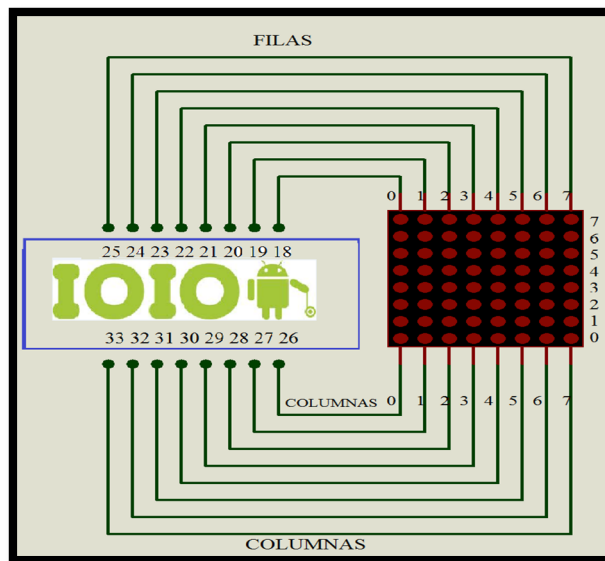
Los pines y materiales usados para esta aplicación son los siguientes.

**Tabla 23.** Pines y materiales usados en la aplicación letrero dinámico

<b>Placa IOIO</b>	Filas (0,1,2....7) pines: 18,19,20,21,22,23,24,25.  Columnas (0,1,2....7) pines: 26,27,28,29,30,31,32,33,.
<b>ENTRENADOR</b>	Matriz de led’s 8x8 color rojo

**Fuente:** Elaborado por Byron Valenzuela

El diagrama de la conexión de la matriz de led’s con la IOIO es el siguiente.



**Figura 67.** Diagrama de conexión de la IOIO con la matriz de led’s

**Fuente:** Elaborado por Byron Valenzuela

- **Conclusiones**

- El desarrollo de esta aplicación es de mucha utilidad al tratar de dar a conocer un mensaje, desde un lugar remoto y controlado desde el dispositivo móvil.
- Mientras mayor sea la velocidad de la técnica de encendido y apagado de los led's de la matriz, la vista humana será incapaz de percibir este fenómeno.
- Mediante un vector se puede agrupar los pines de la IOIO para trabajar de forma similar a los registros PORT de cualquier otro lenguaje que maneja microcontroladores.
- El barrido de led's es una técnica que permite el manejo de matriz de led's sin necesidad de ningún integrado adicional.

- **Recomendaciones**

- En caso de tener una matriz cuadrada de nxm asignar de manera correcta la polaridad de las filas y las columnas.

- **Bibliografía**

mikroElektronika. (2012). *mikroBUS pinout Standar Especification*.

A. Carretero, F. F. (2009). *Electrónica*. Editex.

Agüero, R. (s.f.). *Redes Inalámbricas de Área Local y Personal WLAN El estándar IEEE 802.11*. Cantabria.

Argüello, D. J. (2012). *Desarrollo de una aplicación que permita la captura almacenamiento, reproducción, administración y envío de archivos de vide, audio e imagenes utilizando la tecnología bluetooth, para dispositivos móviles basados en el sistema operativo Android*. Quito.

Carballar, J. (2010). *WI-FI lo que necesita conocer*. Madrid: RC Libros.

Carrillo Pérez, M. d. (2006). *Estudio y análisis de rendimiento Bluetooth*. Madrid.

Castillo, D. (2012). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ALARMA COMUNITARIA A BASE DE MÓDULOS INALÁMBRICOS UTILIZANDO TECNOLOGÍA ZIGBEE*. Ibarra.

Clanar Internacional. (s.f.). *Internet y redes inalámbricas*. Arequipa: Clanar.

Collaguazo, G. (2009). *Sistemas Basados en Microprocesadores*. Ibarra.

Correia, P. (2002). *Guía práctica del GPS*. Barcelona: Marcombo.

Creative Commons . (s.f.). *Arduino*. Obtenido de <http://arduino.cc/en/Main/CopyrightNotice>

Daniel Benchimol. (2011). *Microcontroladores*. Buenos Aires: DALAGA S.A.

Digi International, Inc. (2009). *XBee®/XBee-PRO® RF Modules*. New York.

Dignani, J. (2011). *Análisis del protocolo ZigBee*.

Dignani, J. P. (2011). *Análisis del protocolo ZigBee*.

Fairchild Semiconductor. (1999). *MM74C922 • MM74C923*.

FAIRCHILD. (2000). *DM74LS47 BCD to 7-Segment Decoder/Driver with Open-Collector Outputs*. FAIRCHILD SEMICONDUCTOR.

Fernández, A. M. (2004). *EL BUS I2C*. Córdoba.

GARCÍA CELI, H. M., & SANTILLÁN LARA, L. A. (2005). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE TARIFACIÓN PARA LOCUTORIOS*. Quito.

GitHub. (2008). *Analog Input*. Obtenido de <https://github.com/ytai/ioio/wiki/Analog-Input>

GitHub. (2008). *Digital IO*. Obtenido de <https://github.com/ytai/ioio/wiki/Digital-IO>

GitHub. (2008). *Getting To Know The Board*. Obtenido de <https://github.com/ytai/ioio/wiki/Getting-To-Know-The-Board>

GitHub. (2008). *IOIO Over Bluetooth*. Obtenido de <https://github.com/ytai/ioio/wiki/IOIO-Over-Bluetooth>

GitHub. (2008). *IOIOLibBasics*. Obtenido de <https://github.com/ytai/ioio/wiki/IOIOLib-Basics>

GitHub. (2008). *Power Supply*. Obtenido de <https://github.com/ytai/ioio/wiki/Power-Supply>

GitHub. (2008). *Power Supply*. Obtenido de <https://github.com/ytai/ioio/wiki/Power-Supply>

GitHub. (2008). *Pwm Output*. Obtenido de <https://github.com/ytai/ioio/wiki/PWM-Output>

GitHub. (2008). *SPI*. Obtenido de <https://github.com/ytai/ioio/wiki/SPI>

GitHub. (2008). *TWI*. Obtenido de <https://github.com/ytai/ioio/wiki/TWI>

GitHub. (2008). *UART*. Obtenido de <https://github.com/ytai/ioio/wiki/UART>

Granadino, C., & Suárez, J. (s.f.). *El bus I2C*. Chile: Universidad Técnica Federico Santa María .

Huerta, E., Manguiaterra, A., & Gustavo, N. (2005). *Posicionamiento satelital*. Argentina: A.U.G.M.

*IOIO for Android*. (s.f.). Obtenido de <https://www.sparkfun.com/products/10748>

Jara, P., & Nazar, P. (s.f.). *Estándar IEEE 802.11 X de las WLAN*. Buenos Aires: Edutecne.

Microchip. (2010). *PIC24FJ256DA210 FAMILY*. Microchip.

- Microchip. (2013). *MRF24WB0MA/MRF24WB0MB*.
- Molina, F. (s.f.). *Global positioning system*. España.
- Monk, S. (2012). *Making Android Accessories With IOIO* (Vol. I). O'Reilly Media.
- Pérez, E. L. (s.f.). Curso de Redes de Microcontroladores PIC (PROTOCOLO SPI). México: Ingeniería en Microcontroladores.
- Pérochon, S. (2012). *Android Guía de desarrollo de aplicaciones para smartphones y tabletas*. Barcelona: Ediciones ENI.
- Quectel. (2011). *L30 Quectel GPS Engine*. Shanghai: Quectel.
- Quectel. (2013). *Quectel L30 Compact GPS Module*.
- Raúl Esteve Bosch, J. F. (2005). *Fundamentos de electrónica digital*. Valencia.
- RobotFreak. (2008). *IOIO-Rover*. Obtenido de <http://letsmakerobots.com/node/33968>
- ROVING NETWORKS. (2011). *RN-XV Data Sheet*. Arizona: ROVING NETWORKS.
- sgoliver.net foro. (s.f.). *Estructura de un proyecto Android*. Obtenido de <http://www.sgoliver.net/blog/?p=1278>
- Texas Instrument. (2000). *LM35*.
- Texas Instrument. (2004). *LM-358 DUAL OPERATIONAL AMPLIFIERS*. Dallas: Texas Instrument.
- Ucontrol. (2008). Matrices de LEDs. *Ucontrol Electrónica General Pic's en particular*, 68.
- Universidad Politécnica de Valencia. (Abril de 2013). *Elementos de un proyecto Android*. Obtenido de <http://www.androidcurso.com/index.php/recursos-didacticos/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/148-elementos-de-un-proyecto-android>
- USERS. (s.f.). Microcontroladores funcionamiento programación y usos prácticos. *USERS*, 70-76.
- Valverde Rebaza, J. C. (2007). *El Estándar Inalámbrico ZigBee*. Trujillo: Perú.
- Valverde, J. (2007). *El Estándar Inalámbrico ZigBee*. Trujillo.
- Xatakandroid. (2005). *¿Qué es Android?* Obtenido de <http://www.xatakandroid.com/sistema-operativo/que-es-android>
- Zambrano, B. J. (2011). *DISEÑO E IMPLEMENTACIÓN DE UN KIT DE APLICACIONES PARAPERSONAS CON DISCAPACIDAD VISUAL UTILIZANDO LA*. Sangolqui.
- zhongzhouOpto. (s.f.). *SPECIFICATION FOR ZHONGZHOU LED LAMP* .



- Anexos

Como anexo se adjunta el código de la aplicación escrito en el software Eclipse.

```
//Letrero dinamico "HOLA lolo" con matriz de leds
// 8x8
package tesis.matrizleds;

import ioio.lib.api.DigitalOutput;
import ioio.lib.api.exception.ConnectionLostException;
import ioio.lib.util.AbstractIOActivity;
import tesis.matrizleds.R.id;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ToggleButton;

public class MatrizActivity extends AbstractIOActivity implements OnClickListener {
    // Variables que alojan los componentes del formulario
    private Button btnMensaje;
    private Boolean bandera = false;
    // declaracion de los pines de las filas y
    // columnas correspondientes a la IOIO
    int pinfilas [] = {18,19,20,21,22,23,24,25};
    int pincolumnas [] = {26,27,28,29,30,31,32,33};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // atar los componentes del formulario

        setContentView(R.layout.matriz_activity);
        btnMensaje = (Button)findViewById(id.btnMensaje);
    }
}
```

```

class IOIOThread extends AbstractIOActivity.IOIOThread
{
// declaracion de variables DigitalOutput correspondientes a las
// filas y columnas

private DigitalOutput fila0, fila1, fila2, fila3, fila4, fila5, fila6, fila7;
private DigitalOutput
columna0, columna1, columna2, columna3, columna4, columna5, columna6, columna7;

// Agrupar las filas y columnas en un vector para poder trabajar como un puerto
private DigitalOutput filas [] = {fila0, fila1, fila2, fila3, fila4, fila5, fila6, fila7};
private DigitalOutput columnas [] =
{columna0, columna1, columna2, columna3, columna4, columna5, columna6, columna7};

@Override
protected void setup () throws ConnectionLostException
{
// Inicializacion de los pines como salida digital con su respectivo
// valor inicial

for (int i=0; i<=7; i++)
{
filas [i] = ioio_.openDigitalOutput(pinfilas[i], false);
columnas [i] = ioio_.openDigitalOutput(pincolumnas[i], true);
}
}

@Override
public void loop() throws ConnectionLostException, InterruptedException
{

if (bandera == true)
{
// metodo para cada letra
// que aparece en la
// matriz de led's

```

```

        letraH();
        letraO();
        letraL();
        letraA();
        // retardo para la siguiente palabra
        sleep(100);
        letral();
        letrao();
        letral();
        letrao();
        // retardo para volver a repetir la frase
        sleep(500);
    }

}

//metodo para la letra H
public void letraH() throws ConnectionLostException, InterruptedException
{
    // Metodo que muestra en la matriz de led's las letras
    // todas tienen el mismo principio se realiza
    // un barrido de las filas y las columnas
    // el efecto es de prender y apagar los leds
    // de la letra sin que la vista humana pueda percibir
    // el cambio de encendido y apagado
    // segun la letra, luego de desplegada la letra se realiza
    // el movimiento de izquierda a derecha
    // ciclo for que hace que la letra se mueva
    // de izquierda a derecha
    for (int j =0;j<=4;j++)
    {
        // for que permite barrir las columnas de la letra H
        for (int i =1;i<=6;i++)
        {
            // prende las columnas de la letra H

```

```

        // es decir |  |
        columnas [4-j].write(false);
    columnas [7-j].write(false);
    filas [i].write(true);

        Thread.sleep(10);
        // apaga los leds que se prendio ateriormente
    filas [i].write(false);
    columnas [4-j].write(true);
    columnas [7-j].write(true);

    }

// for que permite barrer la linea horizontal de la letra
//H

    for (int i =5;i<=6;i++)
    {
        // prende la fila de la letra H
        // es decir -
        columnas [i-j].write(false);
        filas [4].write(true);

        Thread.sleep(10);
        // Apaga los leds prendidos
        columnas [i-j].write(true);
    }

        Thread.sleep(50);

    }
}

// metodo para la letra O
public void letraO() throws ConnectionLostException, InterruptedException
{

```

```

for(int j =0;j<=4;j++)
{
    for (int i =1;i<=6;i++)
    {
        columnas [4-j].write(false);
        columnas [7-j].write(false);
        filas [i].write(true);
        sleep(10);
        filas [i].write(false);
        columnas [4-j].write(true);
        columnas [7-j].write(true);

    }
    for (int i =5;i<=6;i++)
    {
        columnas [i-j].write(false);
        filas [1].write(true);
        filas [6].write(true);
        sleep (10);
        columnas [i-j].write(true);
    }
    sleep (50);
}

```

// Metodo para la letra L

```

public void letraL() throws ConnectionLostException, InterruptedException
{
    for(int j =0;j<=4;j++)
    {
        for (int i =1;i<=6;i++)
        {
            columnas [4-j].write(false);
            filas [i].write(true);
            sleep (10);
            filas [i].write(false);

```

```

        columnas [4-j].write(true);
    }
    for (int i =5;i<=7;i++)
    {
        columnas [i-j].write(false);
        filas [1].write(true);

        sleep (10);
        columnas [i-j].write(true);
    }
    sleep (50);
}
}

```

// Metodo para la Letra A

```
public void letraA() throws ConnectionLostException, InterruptedException
```

```

{

    for(int j =0;j<=4;j++)
    {
        for (int i =1;i<=6;i++)
        {
            columnas [4-j].write(false);
            columnas [7-j].write(false);
            filas [i].write(true);
            sleep (10);
            filas [i].write(false);
            columnas [4-j].write(true);
            columnas [7-j].write(true);

        }
        for (int i =5;i<=6;i++)
        {
            columnas [i-j].write(false);
            filas [6].write(true);
            filas [4].write(true);

```

```

        sleep (10);
        columnas [i-j].write(true);
    }
    sleep (50);
}
}

```

// Metodo para la letra l

**public void** letral() **throws** ConnectionLostException, InterruptedException

```

{

    for(int j =0;j<=4;j++)
    {
        for (int i =2;i<=6;i++)
        {
            //columnas [4-j].write(false);
            columnas [6-j].write(false);
            filas [i].write(true);
            sleep (10);
            filas [i].write(false);
            //columnas [4-j].write(true);
            columnas [6-j].write(true);

        }
        for (int i =5;i<=7;i++)
        {
            columnas [i-j].write(false);
            filas [6].write(true);
            filas [2].write(true);
            sleep (10);
            columnas [i-j].write(true);
        }
        sleep (50);
    }
}

```

// Metodo para la letra o

```

public void letrao() throws ConnectionLostException, InterruptedException
{
    for(int j =0;j<=4;j++)
    {
        for (int i =2;i<=6;i++)
        {
            columnas [4-j].write(false);
            columnas [7-j].write(false);
            filas [i].write(true);
            sleep(10);
            filas [i].write(false);
            columnas [4-j].write(true);
            columnas [7-j].write(true);

        }
        for (int i =5;i<=6;i++)
        {
            columnas [i-j].write(false);
            filas [2].write(true);
            filas [6].write(true);
            sleep (10);
            columnas [i-j].write(true);
        }
        sleep (50);
    }
}

```

@Override

```

protected AbstractIOActivity.IOIOThread createIOIOThread() {
    return new IOIOThread();
}

```

@Override

```

public void onClick(View v) {

```

```

    // presiono el boton Actvar mensaje ???

```



```
bandera = !bandera;

if (bandera==true)
{

    btnMensaje.setText(" Mensaje aActivado");
    // mensaje Juego de Luces Activado
}else
{
    btnMensaje.setText("Mensaje desactivado");
}
}
}
```

## 4.2 PRÁCTICA CON EL MÓDULO CONVERTOR ANALÓGICO DIGITAL

### 4.2.1 MEDIDOR DE VALORES ANALÓGICOS

- **Tema: Medidor de temperatura y lectura de voltaje de un potenciómetro**
- **Objetivos**
  - Objetivo general

Medir el voltaje y temperatura mediante un potenciómetro, sensor de temperatura y el módulo de conversión analógica de la IOIO, con el fin de demostrar el funcionamiento de la sección de conversión análogo digital del entrenador.

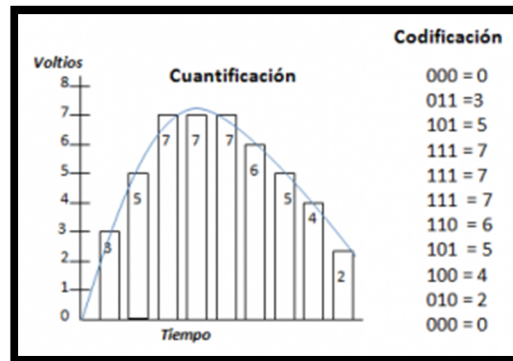
- **Objetivos específicos**
  - Comprender el funcionamiento del módulo de conversión analógica digital de la IOIO.
  - Conocer el funcionamiento del sensor de temperatura LM35 y el potenciómetro.
  - Escribir el flujo grama de la aplicación respectivo.
  - Desarrollar el código del programa que controla la aplicación de conversión análogo digital.
  - Compilar el código realizado e implementar la aplicación con la ayuda del entrenador y del dispositivo móvil Android.
  - Comprobar y corregir errores en el funcionamiento del programa y del entrenador.

- **Marco teórico**

#### **Convertidor Analógico- Digital (ADC)**

(USERS) El conversor análogo digital tiene como objetivo representar un valor analógico de tensión, en un conjunto de códigos binarios; es decir en un valor digital. El proceso que se realiza implícitamente al momento de realizar una conversión analógica digital es el de cuantificación y codificación.

La cuantificación es el proceso de asignar a cada valor analógico de entrada una serie de niveles discretos de salida, la cuantificación convierte la muestra analógica en un número binario dependiente de la resolución del conversor; una vez realizada la cuantificación el siguiente paso es la codificación que es el proceso de representar los valores muestreados por medio de valores numéricos previamente establecidos la forma de codificación más usada es la binaria



**Figura 68.** Cuantificación y codificación de una señal analógica

**Referencia:** El conflicto del audio, recuperado de <http://oirverycontar.wordpress.com/2011/09>

La resolución del convertor determina el número de bits con los cuales el convertor será capaz de representar la señal muestreada, mientras más bits de resolución se tengan, el resultado de la conversión será más exacto y se asemejará cada vez más al valor real. Existen convertidores de 8, 10, 12, 16, 20 y 24 bits; un ADC de  $n$  bits es capaz de representar  $2^n$  valores digitales, es decir que un ADC de 10 bits es capaz de representar 1024 valores digitales.

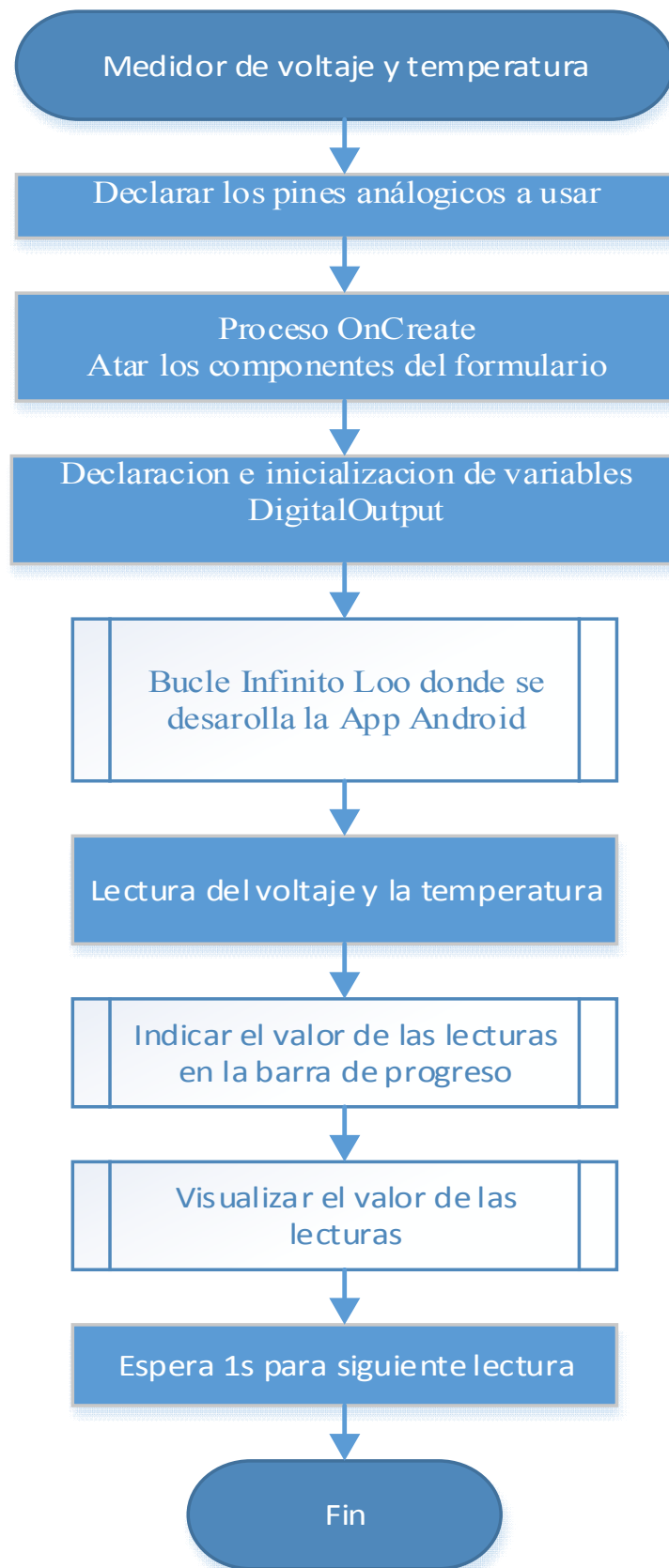
A continuación se presenta una tabla con los requerimientos necesarios para la aplicación de conversión analógica digital.

**Tabla 24.** Requerimientos de la aplicación con el módulo convertor analógico-digital

Dispositivo	Requerimientos
Sensor de temperatura y potenciómetro	Sensor LM35 y un potenciómetro de 10k que posee en entrenador
Tarjeta IOIO	Dos pines analógico digitales
Equipo móvil Android	Equipo en el cual se carga la aplicación Android e interactúa con la IOIO
Dispositivo Bluetooth o cable de comunicación usb del equipo Android	Interfaz de comunicación entre el móvil y la IOIO

**Fuente:** Elaborado por Byron Valenzuela

- Flujo grama



- **Desarrollo**

Enunciado: Elaborar un medidor de voltaje y temperatura, mediante el potenciómetro y el sensor de temperatura que forman parte del módulo análogo digital del entrenador.

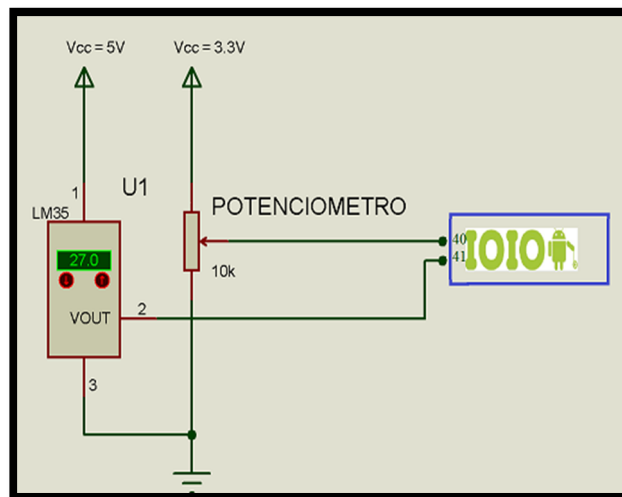
A continuación se describen los pines y materiales usados en la aplicación medidor de voltaje y temperatura.

**Tabla 25.** Descripción de los pines y materiales usados en la aplicación de conversión análoga digital

<b>Placa IOIO</b>	Dos pines análogo digitales (40 y 41) que permitan la lectura del voltaje y temperatura respectivamente
<b>Entrenador</b>	Sensor de temperatura LM35 y resistencia variable de 10k $\Omega$

**Referencia:** Elaborado por Byron Valenzuela

Luego de detallar los pines y dispositivos utilizados en la aplicación se presenta un gráfico con el esquema de conexión de los equipos usados y la placa IOIO.



**Figura 69.** Esquema de conexión potenciómetro y sensor de temperatura con la IOIO

**Referencia:** Elaborado por Byron Valenzuela

- **Análisis de resultados**

Para comprobar el funcionamiento se utiliza un multímetro digital, que sea capaz de medir tanto temperatura y voltaje, a continuación se presenta una tabla con las distintas medidas realizadas.

**Tabla 26.** Comparación de resultados

Multímetro Digital		Placa IOIO	
Voltaje	Temperatura °C	Voltaje	Temperatura °C
0V	10	0V	~9.8
5V	20	5V	~19.5
12V	30	12V	~29.6

**Referencia:** Elaborado por Byron Valenzuela

- **Conclusiones**

- La resolución de un conversor análogo digital juega un papel importante; mientras mayor sea la resolución del conversor, más exacto es el valor de la conversión
- Muchos de los fenómenos naturales como la presión, temperatura, la luminosidad, voz, etc. son de carácter analógico y sin un conversor análogo digital, no se lograría cuantificar el valor de estos fenómenos.
- El método read() de la clase AnalogInput retorna un valor entre 0 y 1, siendo 0 el valor mínimo de la lectura analógica y uno el valor máximo de la misma.

- **Recomendaciones**

- No exceder el valor de voltaje máximo de lectura analógica que es de 3.3V.
- Se recomienda realizar métodos exteriores en la aplicación Android, que imprima el valor de la lectura y coloque el valor en la barra de progreso, permitiendo la actualización cada vez que se realice la lectura.

- **Bibliografía**

mikroElektronika. (2012). *mikroBUS pinout Standar Especification*.

A. Carretero, F. F. (2009). *Electrónica*. Editex.

- Agüero, R. (s.f.). *Redes Inalámbricas de Área Local y Personal WLAN El estándar IEEE 802.11*. Cantabria.
- Argüello, D. J. (2012). *Desarrollo de una aplicación que permita la captura almacenamiento, reproducción, administración y envío de archivos de vídeo, audio e imágenes utilizando la tecnología bluetooth, para dispositivos móviles basados en el sistema operativo Android*. Quito.
- Carballar, J. (2010). *WI-FI lo que necesita conocer*. Madrid: RC Libros.
- Carrillo Pérez, M. d. (2006). *Estudio y análisis de rendimiento Bluetooth*. Madrid.
- Castillo, D. (2012). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ALARMA COMUNITARIA A BASE DE MÓDULOS INALÁMBRICOS UTILIZANDO TECNOLOGÍA ZIGBEE*. Ibarra.
- Clanar Internacional. (s.f.). *Internet y redes inalámbricas*. Arequipa: Clanar.
- Collaguazo, G. (2009). *Sistemas Basados en Microprocesadores*. Ibarra.
- Correia, P. (2002). *Guía práctica del GPS*. Barcelona: Marcombo.
- Creative Commons . (s.f.). *Arduino*. Obtenido de <http://arduino.cc/en/Main/CopyrightNotice>
- Daniel Benchimol. (2011). *Microcontroladores*. Buenos Aires: DALAGA S.A.
- Digi International, Inc. (2009). *XBee®/XBee-PRO® RF Modules*. New York.
- Dignani, J. (2011). *Análisis del protocolo ZigBee*.
- Dignani, J. P. (2011). *Análisis del protocolo ZigBee*.
- Fairchild Semiconductor. (1999). *MM74C922 • MM74C923*.
- FAIRCHILD. (2000). *DM74LS47 BCD to 7-Segment Decoder/Driver with Open-Collector Outputs*. FAIRCHILD SEMICONDUCTOR.
- Fernández, A. M. (2004). *EL BUS I2C*. Córdoba.
- GARCÍA CELI, H. M., & SANTILLÁN LARA, L. A. (2005). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE TARIFACIÓN PARA LOCUTORIOS*. Quito.
- GitHub. (2008). *Analog Input*. Obtenido de <https://github.com/ytai/ioio/wiki/Analog-Input>
- GitHub. (2008). *Digital IO*. Obtenido de <https://github.com/ytai/ioio/wiki/Digital-IO>
- GitHub. (2008). *Getting To Know The Board*. Obtenido de <https://github.com/ytai/ioio/wiki/Getting-To-Know-The-Board>
- GitHub. (2008). *IOIO Over Bluetooth*. Obtenido de <https://github.com/ytai/ioio/wiki/IOIO-Over-Bluetooth>
- GitHub. (2008). *IOIOLibBasics*. Obtenido de <https://github.com/ytai/ioio/wiki/IOIOLib-Basics>

GitHub. (2008). *Power Supply*. Obtenido de <https://github.com/ytai/ioio/wiki/Power-Supply>

GitHub. (2008). *Power Supply*. Obtenido de <https://github.com/ytai/ioio/wiki/Power-Supply>

GitHub. (2008). *Pwm Output*. Obtenido de <https://github.com/ytai/ioio/wiki/PWM-Output>

GitHub. (2008). *SPI*. Obtenido de <https://github.com/ytai/ioio/wiki/SPI>

GitHub. (2008). *TWI*. Obtenido de <https://github.com/ytai/ioio/wiki/TWI>

GitHub. (2008). *UART*. Obtenido de <https://github.com/ytai/ioio/wiki/UART>

Granadino, C., & Suárez, J. (s.f.). *El bus I2C*. Chile: Universidad Técnica Federico Santa María .

Huerta, E., Manguiaterra, A., & Gustavo, N. (2005). *Posicionamiento satelital*. Argentina: A.U.G.M.

*IOIO for Android*. (s.f.). Obtenido de <https://www.sparkfun.com/products/10748>

Jara, P., & Nazar, P. (s.f.). *Estándar IEEE 802.11 X de las WLAN*. Buenos Aires: Edutecne.

Microchip. (2010). *PIC24FJ256DA210 FAMILY*. Microchip.

Microchip. (2013). *MRF24WB0MA/MRF24WB0MB*.

Molina, F. (s.f.). *Global positioning system*. España.

Monk, S. (2012). *Making Android Accessories With IOIO* (Vol. I). O'Reilly Media.

Pérez, E. L. (s.f.). *Curso de Redes de Microcontroladores PIC (PROTOCOLO SPI)*. México: Ingeniería en Microcontroladores.

Pérochon, S. (2012). *Android Guia de desarrollo de aplicaciones para smartphones y tabletas*. Barcelona: Ediciones ENI.

Quectel. (2011). *L30 Quectel GPS Engine*. Shanghai: Quectel.

Quectel. (2013). *Quectel L30 Compact GPS Module*.

Raúl Esteve Bosch, J. F. (2005). *Fundamentos de electrónica digital*. Valencia.

RobotFreak. (2008). *IOIO-Rover*. Obtenido de <http://letsmakerobots.com/node/33968>

ROVING NETWORKS. (2011). *RN-XV Data Sheet*. Arizona: ROVING NETWORKS.

sgoliver.net foro. (s.f.). *Estructura de un proyecto Android*. Obtenido de <http://www.sgoliver.net/blog/?p=1278>

Texas Instrument. (2000). *LM35*.

Texas Instrument. (2004). *LM-358 DUAL OPERATIONAL AMPLIFIERS*. Dallas: Texas Instrument.

Ucontrol. (2008). *Matrices de LEDs. Ucontrol Electrónica General Pic's en particular*, 68.



Universidad Politécnica de Valencia. (Abril de 2013). *Elementos de un proyecto Android*. Obtenido de <http://www.androidcurso.com/index.php/recursos-didacticos/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/148-elementos-de-un-proyecto-android>

USERS. (s.f.). Microcontroladores funcionamiento programación y usos prácticos. *USERS*, 70-76.

Valverde Rebaza, J. C. (2007). *El Estándar Inalámbrico ZigBee*. Trujillo: Perú.

Valverde, J. (2007). *El Estándar Inalámbrico ZigBee*. Trujillo.

Xatakandroid. (2005). *¿Qué es Android?* Obtenido de <http://www.xatakandroid.com/sistema-operativo/que-es-android>

Zambrano, B. J. (2011). *DISEÑO E IMPLEMENTACIÓN DE UN KIT DE APLICACIONES PARAPERSONAS CON DISCAPACIDAD VISUAL UTILIZANDO LA*. Sangolqui.

zhongzhouOpto. (s.f.). *SPECIFICATION FOR ZHONGZHOU LED LAMP* .

- **Anexos**

**Se adjunta el código de la aplicación.**

```
// Meidor de Voltaje y temperatura
```

```
package tesis.analogodigital;
```

```
import ioio.lib.api.AnalogInput;
```

```
import ioio.lib.api.exception.ConnectionLostException;
```

```
import ioio.lib.util.AbstractIOActivity;
```

```
import android.R.string;
```

```
import android.os.Bundle;
```

```
import android.widget.SeekBar;
```

```
import android.widget.TextView;
```

```
public class AnalogoDigitalActivity extends AbstractIOActivity {
```

```
private final int PinPotenciometro = 40; // Pin conectado al Potenciometro
```

```
private final int PinTemperatura = 41; // Pin conectado al sensor de temp
```

```
private TextView tvPotenciometro; // TextView a visualizar
```

```
private SeekBar sbPotenciometro; // el voltaje y la temperatura
```

```
private TextView tvTemperatura; // barra de estado
private SeekBar sbTemperatura; // que indica el valor de la temp y volt.
```

```
@Override
```

```
public void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    setContentView(R.layout.analog_digital_activity);
```

```
    // atar los componentes del formulario
```

```
tvPotenciometro = (TextView) findViewById(R.id.tvPotenciometro);
```

```
sbPotenciometro = (SeekBar) findViewById(R.id.sbPotenciometro);
```

```
tvTemperatura = (TextView) findViewById(R.id.tvTemperatura);
```

```
sbTemperatura = (SeekBar) findViewById(R.id.sbTemperatura);
```

```
sbPotenciometro.setEnabled(false);
```

```
sbTemperatura.setEnabled(false);
```

```
}
```

```
class IOIOThread extends AbstractIOIOActivity.IOIOThread
```

```
{
```

```
    private AnalogInput pot; // Variable que leera el vaor
```

```
    private AnalogInput temp; // del voltaje y temperatura
```

```
@Override
```

```
public void setup() throws ConnectionLostException
```

```
{
```

```
    // Inicializacion de las variables AnalogInput
```

```
    pot = ioio_.openAnalogInput(PinPotenciometro);
```

```
    temp = ioio_.openAnalogInput(PinTemperatura);
```

```
}
```

```

@Override
public void loop() throws ConnectionLostException {
    try {

        // Lectura del valor de temperatura
        // y voltaje resoectivamente
        final float readPot = pot.read();
        final float readTemp = temp.read();
        // sub proceso que visualiza la barra de estado con
        // el valor de la temperatura y el voltaje
setSeekBar((int) (readPot * 100),(int)(readTemp *100));
// dub proceso que imprime el valor de la temperatura
        // y voltaje
setText((Float.toString((readPot * 33)/10)),(Float.toString((readTemp * 5)*100 )));
        // retardo de 1 s
        sleep(100);
    } catch (InterruptedException e) {
        ioio_.disconnect();
    } catch (ConnectionLostException e) {

        throw e;
    }
}
}
}

```

```

@Override
protected AbstractIOActivity.IOIOThread createIOIOThread() {
    return new IOIOThread();
}

```

```

// metodo que coloca el valor de la lectura en la barra de
// progreso
private void setSeekBar(final int pot,final int temp) {
    runOnUiThread(new Runnable() {
        @Override

```

```

    public void run() {
        sbPotenciometro.setProgress(pot);
        sbTemperatura.setProgress(temp);
    }
});
}

// metodo que emprime el valor de la lectura en la pantalla
// del movil
private void setText(final String strPot, final String strTemp) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            // imprime solo 4 digitos de la lectura
            tvPotenciometro.setText(strPot.substring(0,3));
            tvTemperatura.setText(strTemp.substring(0, 4));
        }
    });
}
}
}

```

### 4.3 PRÁCTICA PWM

- **Tema: Control de luminosidad mediante PWM**

- **Objetivos**

- **Objetivo general**

Controlar la luminosidad de un led, mediante el uso de la modulación PWM, y así demostrar el funcionamiento PWM de la placa.

- **Objetivos específicos**

- Entender el marco teórico referente a PWM.
- Investigar el modo de funcionamiento PWM de la IOIO.
- Realizar el flujo grama para el control de luminosidad.
- Escribir el código que controla el led PWM con la IOIO.
- Compilar y ejecutar el programa en el dispositivo Android.
- Verificar el funcionamiento y corregir errores de la aplicación.

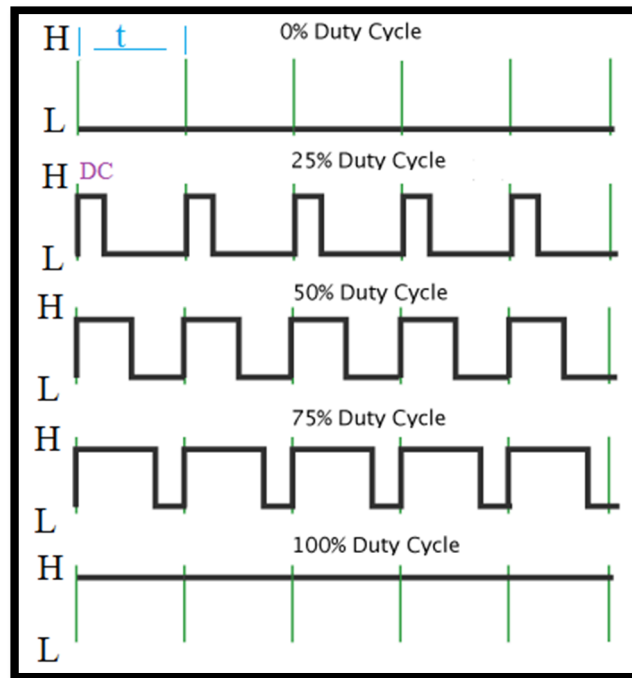
- **Marco teórico**

#### **PWM**

PWM o modulación por ancho de pulso es una técnica de modulación digital que permite simular una salida analógica con una salida digital, esta técnica consiste en generar ondas cuadradas con una frecuencia y ciclo de trabajo determinado.

En una señal cuadrada periódica el ciclo de trabajo o duty cycle (DC), es el tiempo que la señal se encuentra en la posición de alto o encendido frente al tiempo en que permanece en bajo o apagado, con esta técnica de variar el ciclo de trabajo logra controlar la cantidad de energía que se envía hacia una carga y fácilmente se controla la luminosidad de un led o la velocidad de giro un motor, etc.

En la siguiente figura se observa una representación de PWM donde  $t$  es el periodo de la señal, la frecuencia es el inverso del periodo, DC es el ciclo de trabajo claramente se observa la variación del DC.



**Figura 70.** Representación de PWM  
**Fuente:** Elaborado por Byron Valenzuela

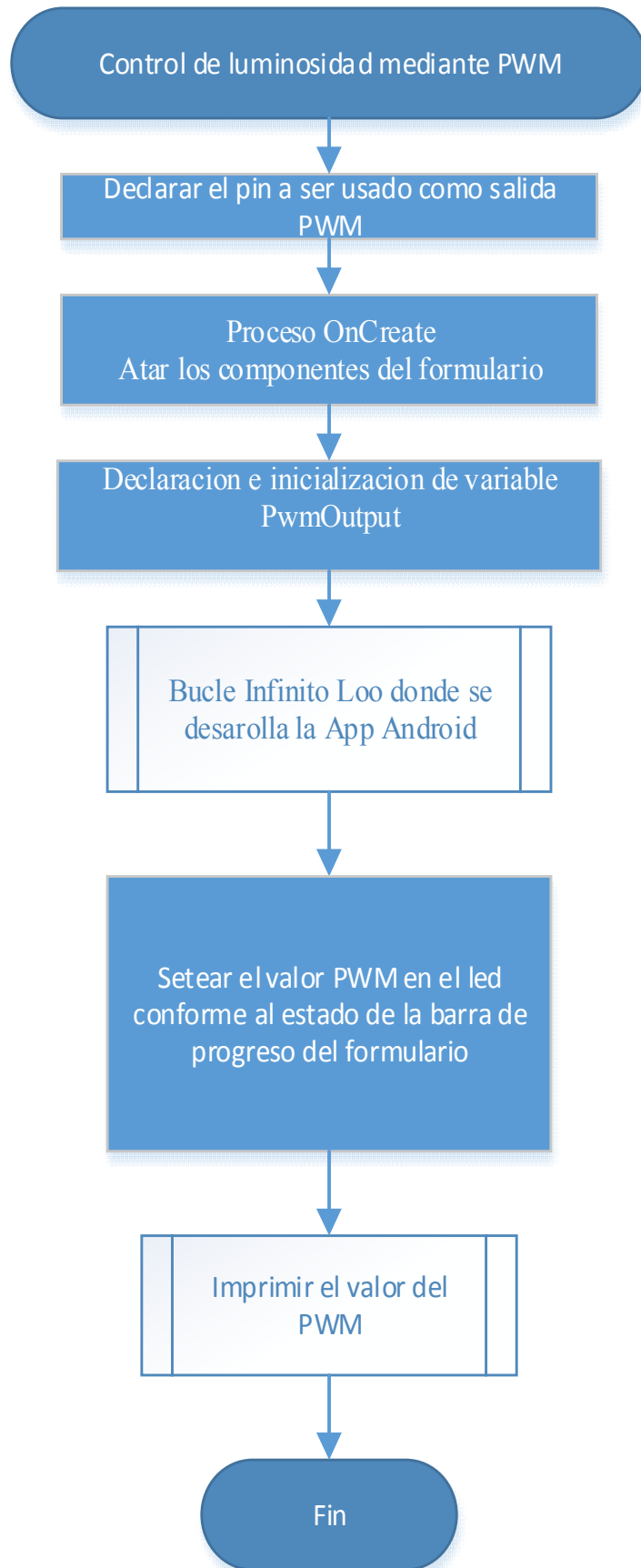
Los componentes requeridos para esta aplicación son los siguientes.

**Tabla 27.** Requerimientos de la aplicación PWM

Dispositivo	Requerimientos
Tarjeta IOIO	Módulo PWM.
Equipo móvil Android	Equipo que interactúa con la IOIO
Dispositivo Bluetooth o cable de comunicación usb del equipo Android	Interfaz de comunicación IOIO – Dispositivo móvil Android.

**Fuente:** Elaborado por Byron Valenzuela

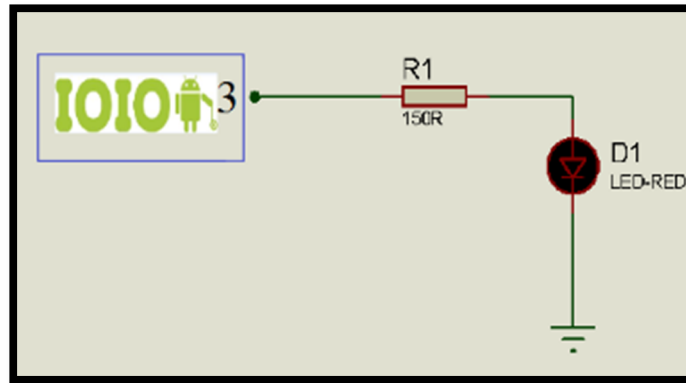
- Flujo grama



- **Desarrollo**

**Enunciado:** Elaborar una aplicación que controle la luminosidad de un led mediante PWM.

El diagrama para esta práctica es el siguiente.



**Figura 71.** Conexión de la aplicación PWM

**Referencia:** Elaborado por Byron Valenzuela

- **Conclusiones**

- A través de la técnica PWM es posible controlar la luminosidad de un led haciendo variar el ciclo de trabajo.
- El led stat integrado en la placa IOIO, no solo se lo puede usar en el módulo PWM, al contrario es de gran ayuda en los proyectos que se necesite de un led como salida digital.
- El método `setDutyCycle ()` de la IOIO, solo permite valores entre 0 y 1 para setear el valor del ciclo de trabajo.

- **Recomendaciones**

- Se recomienda ahondar en el tema PWM y realizar una aplicación que sea capaz de controlar un motor DC.

- **Bibliografía**

mikroElektronika. (2012). *mikroBUS pinout Standar Especification*.

A. Carretero, F. F. (2009). *Electrónica*. Editex.

Agüero, R. (s.f.). *Redes Inalámbricas de Área Local y Personal WLAN El estándar IEEE 802.11*. Cantabria.



- Argüello, D. J. (2012). *Desarrollo de una aplicación que permita la captura almacenamiento, reproducción, administración y envío de archivos de vídeo, audio e imágenes utilizando la tecnología bluetooth, para dispositivos móviles basados en el sistema operativo Android*. Quito.
- Carballar, J. (2010). *Wi-Fi lo que necesita conocer*. Madrid: RC Libros.
- Carrillo Pérez, M. d. (2006). *Estudio y análisis de rendimiento Bluetooth*. Madrid.
- Castillo, D. (2012). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ALARMA COMUNITARIA A BASE DE MÓDULOS INALÁMBRICOS UTILIZANDO TECNOLOGÍA ZIGBEE*. Ibarra.
- Clanar Internacional. (s.f.). *Internet y redes inalámbricas*. Arequipa: Clanar.
- Collaguazo, G. (2009). *Sistemas Basados en Microprocesadores*. Ibarra.
- Correia, P. (2002). *Guía práctica del GPS*. Barcelona: Marcombo.
- Creative Commons . (s.f.). *Arduino*. Obtenido de <http://arduino.cc/en/Main/CopyrightNotice>
- Daniel Benchimol. (2011). *Microcontroladores*. Buenos Aires: DALAGA S.A.
- Digi International, Inc. (2009). *XBee®/XBee-PRO® RF Modules*. New York.
- Dignani, J. (2011). *Análisis del protocolo ZigBee*.
- Dignani, J. P. (2011). *Análisis del protocolo ZigBee*.
- Fairchild Semiconductor. (1999). *MM74C922 • MM74C923*.
- FAIRCHILD. (2000). *DM74LS47 BCD to 7-Segment Decoder/Driver with Open-Collector Outputs*. FAIRCHILD SEMICONDUCTOR.
- Fernández, A. M. (2004). *EL BUS I2C*. Córdoba.
- GARCÍA CELI, H. M., & SANTILLÁN LARA, L. A. (2005). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE TARIFACIÓN PARA LOCUTORIOS*. Quito.
- GitHub. (2008). *Analog Input*. Obtenido de <https://github.com/ytai/ioio/wiki/Analog-Input>
- GitHub. (2008). *Digital IO*. Obtenido de <https://github.com/ytai/ioio/wiki/Digital-IO>
- GitHub. (2008). *Getting To Know The Board*. Obtenido de <https://github.com/ytai/ioio/wiki/Getting-To-Know-The-Board>
- GitHub. (2008). *IOIO Over Bluetooth*. Obtenido de <https://github.com/ytai/ioio/wiki/IOIO-Over-Bluetooth>
- GitHub. (2008). *IOIOLibBasics*. Obtenido de <https://github.com/ytai/ioio/wiki/IOIOLib-Basics>
- GitHub. (2008). *Power Supply*. Obtenido de <https://github.com/ytai/ioio/wiki/Power-Supply>

GitHub. (2008). *Power Supply*. Obtenido de <https://github.com/ytai/ioio/wiki/Power-Supply>

GitHub. (2008). *Pwm Output*. Obtenido de <https://github.com/ytai/ioio/wiki/PWM-Output>

GitHub. (2008). *SPI*. Obtenido de <https://github.com/ytai/ioio/wiki/SPI>

GitHub. (2008). *TWI*. Obtenido de <https://github.com/ytai/ioio/wiki/TWI>

GitHub. (2008). *UART*. Obtenido de <https://github.com/ytai/ioio/wiki/UART>

Granadino, C., & Suárez, J. (s.f.). *El bus I2C*. Chile: Universidad Técnica Federico Santa María .

Huerta, E., Manguiaterra, A., & Gustavo, N. (2005). *Posicionamiento satelital*. Argentina: A.U.G.M.

*IOIO for Android*. (s.f.). Obtenido de <https://www.sparkfun.com/products/10748>

Jara, P., & Nazar, P. (s.f.). *Estándar IEEE 802.11 X de las WLAN*. Buenos Aires: Edutecne.

Microchip. (2010). *PIC24FJ256DA210 FAMILY*. Microchip.

Microchip. (2013). *MRF24WB0MA/MRF24WBOMB*.

Molina, F. (s.f.). *Global positioning system*. España.

Monk, S. (2012). *Making Android Accessories With IOIO* (Vol. I). O'Reilly Media.

Pérez, E. L. (s.f.). *Curso de Redes de Microcontroladores PIC (PROTOCOLO SPI)*. México: Ingeniería en Microcontroladores.

Pérochon, S. (2012). *Android Guia de desarrollo de aplicaciones para smartphones y tabletas*. Barcelona: Ediciones ENI.

Quectel. (2011). *L30 Quectel GPS Engine*. Shanghai: Quectel.

Quectel. (2013). *Quectel L30 Compact GPS Module*.

Raúl Esteve Bosch, J. F. (2005). *Fundamentos de electrónica digital*. Valencia.

RobotFreak. (2008). *IOIO-Rover*. Obtenido de <http://letsmakerobots.com/node/33968>

ROVING NETWORKS. (2011). *RN-XV Data Sheet*. Arizona: ROVING NETWORKS.

sgoliver.net foro. (s.f.). *Estructura de un proyecto Android*. Obtenido de <http://www.sgoliver.net/blog/?p=1278>

Texas Instrument. (2000). *LM35*.

Texas Instrument. (2004). *LM-358 DUAL OPERATIONAL AMPLIFIERS*. Dallas: Texas Instrument.

Ucontrol. (2008). *Matrices de LEDs. Ucontrol Electrónica General Pic's en particular*, 68.

Universidad Politécnica de Valencia. (Abril de 2013). *Elementos de un proyecto Android*. Obtenido de <http://www.androidcurso.com/index.php/recursos-didacticos/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/148-elementos-de-un-proyecto-android>

USERS. (s.f.). Microcontroladores funcionamiento programación y usos prácticos. *USERS*, 70-76.

Valverde Rebaza, J. C. (2007). *El Estándar Inalámbrico ZigBee*. Trujillo: Perú.

Valverde, J. (2007). *El Estándar Inalámbrico ZigBee*. Trujillo.

Xatakandroid. (2005). *¿Qué es Android?* Obtenido de <http://www.xatakandroid.com/sistema-operativo/que-es-android>

Zambrano, B. J. (2011). *DISEÑO E IMPLEMENTACIÓN DE UN KIT DE APLICACIONES PARAPERSONAS CON DISCAPACIDAD VISUAL UTILIZANDO LA*. Sangolqui.

zhongzhouOpto. (s.f.). *SPECIFICATION FOR ZHONGZHOU LED LAMP* .

- **Anexos**

```
// Control de luminosidad del led integrado en la IOIO
```

```
package tesis.pwm;
```

```
import ioio.lib.api.PwmOutput;
```

```
import ioio.lib.api.exception.ConnectionLostException;
```

```
import ioio.lib.util.AbstractIOActivity;
```

```
import tesis.pwm.R.id;
```

```
import android.os.Bundle;
```

```
import android.widget.SeekBar;
```

```
import android.widget.TextView;
```

```
import android.widget.Toast;
```

```
public class PwmActivity extends AbstractIOActivity {
```

```
    private final int pwmPin = 3; // Pin 0 que es el led de la IOIO
```

```
    private final int freq = 1000; // Frecuencia de trabajo PWM
```

```
    private SeekBar SbLumi; // Variable tipo barra de estado
```

```
private TextView tvValor; // Text view que visualiza el ciclo de trabajo
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```

super.onCreate(savedInstanceState);
setContentView(R.layout.pwm_activity);
// atar los componentes al formulario
SbLumi = (SeekBar)findViewById(id.sbLed);
tvValor = (TextView)findViewById(id.tvValor);

}

```

```

class IOIOThread extends AbstractIOActivity.IOIOThread {
    private PwmOutput brillo; // Variable Tipo PwmOutput
    public void setup() throws ConnectionLostException {

        try {
            // Inicializa el pin PWM a ser usado en la aplicacion
                // con la frecuencia indicada por la variable freq
                brillo = ioio_.openPwmOutput(pwmPin, freq);
        } catch (ConnectionLostException e) {
            throw e;
        }
    }

    public void loop() throws ConnectionLostException {
        try {
            // impone el valor dl ciclo de trabajo segun el estado
                // de la barra de progreso
                // que se encuentra en el formulario
            brillo.setDutyCycle(((float) 1 - SbLumi.getProgress()) * (float) 0.01));
                // llamada al metodo
            // que imprime el valor del ciclo de trabajo en el text view
            setText(Double.toString((SbLumi.getProgress())*(float) 0.01));
                // retardo 10ms
            sleep(10);
        } catch (InterruptedException e) {
            ioio_.disconnect();
        } catch (ConnectionLostException e) {
            throw e;
        }
    }
}

```

```

    }
}
@Override
protected AbstractIOActivity.IOIOThread createIOIOThread() {
    return new IOIOThread();
}
// metodo que imprime el valor en el text view
private void setText(final String strPot ) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            tvValor.setText(strPot.substring(0,2));
        }
    });
}
}
}

```

## 4.4 PRÁCTICA CON EL MÓDULO I2C

- **Tema: Monitoreo de temperatura mediante el protocolo I2C**

- **Objetivos**

- **Objetivo general**

Monitorear la temperatura ambiente mediante la utilización de un módulo I2C conectado a la IOIO, con el propósito de comprobar el funcionamiento del módulo I2C del entrenador.

- **Objetivos específicos**

- Estudiar el protocolo de comunicación I2C.
- Investigar el funcionamiento del módulo TMP102 que contiene el entrenador electrónico.
- Realizar el flujo grama de la aplicación.
- Escribir el código necesario para controlar el monitoreo de la temperatura.
- Compilar el código y ejecutar la aplicación en el dispositivo móvil Android.
- Implementar la aplicación en el entrenador.
- Verificar el funcionamiento y corregir errores si en caso se presentan.

- **Marco teórico**

### **El bus I2C**

I2C fue creado por Philips en el año 1980 con el objetivo de comunicar varios dispositivos electrónicos de control originalmente a una velocidad de 100 kbits/s, gracias a su facilidad de manejo y versatilidad este protocolo se fue popularizando y hoy en día puede llegar a velocidades de 3.4 Megabits/s (Granadino & Suárez).

El bus I2C maneja dos líneas la de datos (SDA) y la de reloj (SCL), cada dispositivo trae consigo una dirección seleccionable por software o hardware según el fabricante, el bus transmite información serial de 8 bits e incluye un detector de colisiones con el propósito de hacer más efectiva la comunicación entre el master y los diferentes esclavos. La cantidad de dispositivos que se pueden conectar al bus es limitada, la capacidad máxima es de 400 pf (periféricos).

El protocolo I2C es el que permite la comunicación entre el master que generalmente un microcontrolador y los diferentes esclavos. Este protocolo define varios términos que son:

**Transmisor:** Es el dispositivo que envía los datos al bus.

**Receptor:** Dispositivo que recibe los datos el bus.

**Master (maestro):** Es el dispositivo que inicia la transferencia, genera la señal de reloj y termina un envío de datos.

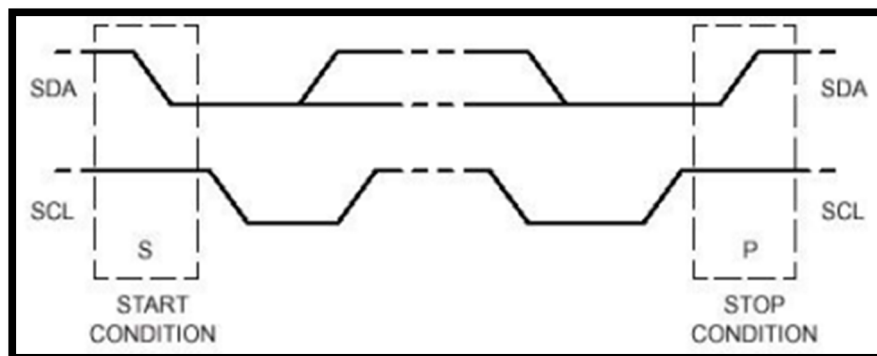
**Slave (esclavo):** Dispositivo que se comunica con el maestro.

**Multi-master:** El bus puede ser controlado por dos o más master.

**Arbitraje:** Proceso que asegura que solo un master pueda controlar el bus.

**Sincronización:** Proceso para generar la señal de reloj que ayuda a la comunicación entre master y slave.

Los dispositivos I2C generalmente son de colector abierto y se define tres modos de transferencia de información que son: modo estándar a 100 kbits/segundo, modo rápido a 400 kbits/segundo y modo de alta velocidad a 3.4 Mbits/segundo. Tanto la línea de datos SDA como la de reloj SCL son líneas bidireccionales, para que un dato sea válido debe existir un bit de inicio y un bit de parada; el bit de inicio es una transición de 1 a 0 en la línea SDA mientras la línea SCL está en 1, la señal de stop es una transición de 0 a 1 en la línea SDA y mientras SCL está en 1. (Granadino & Suárez)



**Figura 72.** Condición de inicio y parada en I2C

**Fuente:** (Granadino & Suárez)

Cada byte que se envíe debe estar seguido de un bit de reconocimiento, cuando un esclavo no da por finalizado su transmisión mantiene la línea SCL en 0 esto significa que el bus se encuentra en un estado de espera hasta que el mismo dispositivo sea capaz de desbloquear el bus y terminar la transferencia de información, cada terminal que recibe la información debe manejar ACK por cada byte de información que reciba con esto se asegura que los datos lleguen de manera correcta.

Los requisitos necesarios para esta aplicación son los siguientes.

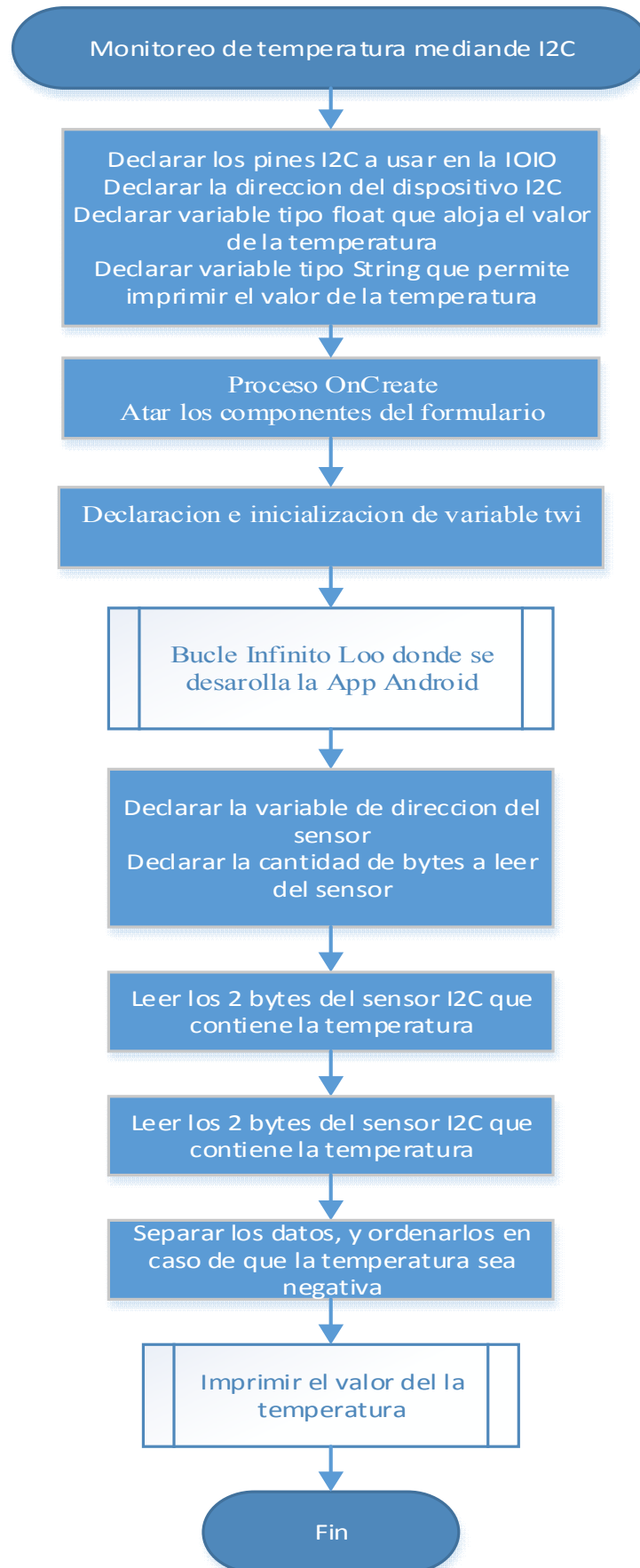
**Tabla 39.** Requerimientos para la aplicación I2C

<b>Dispositivo</b>	<b>Requerimientos</b>
<b>Tarjeta IOIO</b>	Módulo de comunicación I2C.
<b>Módulo I2C</b>	Sensor de temperatura I2C
<b>Equipo móvil Android</b>	Equipo que interactúa con la IOIO
<b>Dispositivo Bluetooth o cable de comunicación usb del equipo Android</b>	Interfaz de comunicación IOIO – Dispositivo móvil Android.

**Fuente:** Elaborado por Byron Valenzuela



- Flujo grama



- **Desarrollo**

**Enunciado:** Desarrollar un lector de temperatura mediante un dispositivo que trabaja bajo el protocolo de comunicación I2C.

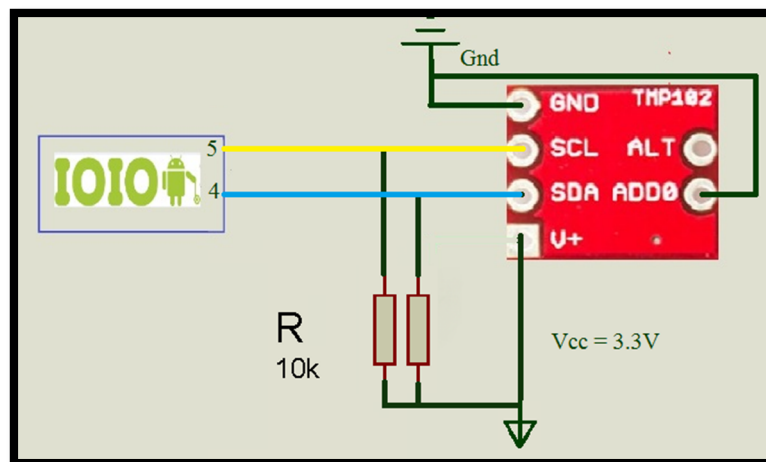
A continuación se describen los pines y materiales usados en esta aplicación.

**Tabla 28.** Materiales necesarios para la aplicación I2C.

<b>Placa IOIO</b>	Pines I2C SDA pin 4 y SCL pin 5 respectivamente
<b>Entrenador</b>	Módulo de temperatura I2C

**Fuente:** Elaborado por Byron Valenzuela.

La conexión del sensor de temperatura I2C con la IOIO se muestra en la siguiente figura.



**Figura 73.** Conexión sensor de temperatura I2C y la IOIO

**Fuente:** elaborado por Byron Valenzuela.

- **Análisis de resultados**

Con el propósito de comprobar el funcionamiento del lector de temperatura se realizó 3 capturas de temperatura teniendo los siguientes resultados.

**Tabla 29.** Capturas de temperatura obtenidas.

Medidor de temperatura	Placa IOIO
Temperatura oC	Temperatura oC
13	13
19	19
25	25

**Fuente:** Elaborado por Byron Valenzuela

Como se observa en la tabla los valores obtenidos por un medidor de temperatura y el lector realizado con la IOIO no varían con mucho esto quiere decir que la temperatura que se obtiene desde el sensor mediante I2C es exacto y puede ser utilizado con toda confianza.

- **Conclusiones**

- El protocolo de comunicación I2C usa los mismos hilos de comunicación para conectar distintos sensores con un nodo principal o dispositivo master, esto favorece el uso eficiente de los pines del equipo master.
- Cada dispositivo I2C trae consigo una dirección con la que se identifica dentro de los equipos esclavos, que puede ser cambiada por hardware o software según sea el caso del módulo.
- La utilización del módulo I2C de cualquier dispositivo hace obligatorio la necesidad de usar dos resistencias pull-up en los pines de datos y reloj.
- Con la librería twi de la IOIO se logra obtener los bytes requeridos para la lectura de información de temperatura.

- **Recomendaciones**

- Obligatoriamente usar resistencias pull-up entre los pines de datos y reloj del módulo I2C.
- En el caso del módulo I2C de temperatura TMP 102 usado en esta práctica debe estar conectado el pin ADD0 a uno lógico es decir a 3.3V, caso contrario el módulo no funcionará.
- Se recomienda realizar métodos exteriores en la aplicación Android, que imprima el valor de la lectura y coloque el valor en la barra de progreso, permitiendo la actualización cada vez que se realice la lectura.

- **Bibliografía**

mikroElektronika. (2012). *mikroBUS pinout Standar Especification*.

A. Carretero, F. F. (2009). *Electrónica*. Editex.

Agüero, R. (s.f.). *Redes Inalámbricas de Área Local y Personal WLAN El estándar IEEE 802.11*. Cantabria.

Argüello, D. J. (2012). *Desarrollo de una aplicación que permita la captura almacenamiento, reproducción, administración y envío de archivos de vide, audio e imagenes utilizando la tecnología bluetooth, para dispositivos móviles basados en el sistema operativo Android*. Quito.

Carballar, J. (2010). *WI-FI lo que necesita conocer*. Madrid: RC Libros.

Carrillo Pérez, M. d. (2006). *Estudio y análisis de rendimiento Bluetooth*. Madrid.

Castillo, D. (2012). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ALARMA COMUNITARIA A BASE DE MÓDULOS INALÁMBRICOS UTILIZANDO TECNOLOGÍA ZIGBEE*. Ibarra.

Clanar Internacional. (s.f.). *Internet y redes inalámbricas*. Arequipa: Clanar.

Collaguazo, G. (2009). *Sistemas Basados en Microprocesadores*. Ibarra.

Correia, P. (2002). *Guía práctica del GPS*. Barcelona: Marcombo.

Creative Commons . (s.f.). *Arduino*. Obtenido de <http://arduino.cc/en/Main/CopyrightNotice>

Daniel Benchimol. (2011). *Microcontroladores*. Buenos Aires: DALAGA S.A.

Digi International, Inc. (2009). *XBee®/XBee-PRO® RF Modules*. New York.

Dignani, J. (2011). *Análisis del protocolo ZigBee*.

Dignani, J. P. (2011). *Análisis del protocolo ZigBee*.

- Fairchild Semiconductor. (1999). *MM74C922 • MM74C923*.
- FAIRCHILD. (2000). *DM74LS47 BCD to 7-Segment Decoder/Driver with Open-Collector Outputs*. FAIRCHILD SEMICONDUCTOR.
- Fernández, A. M. (2004). *EL BUS I2C*. Córdoba.
- GARCÍA CELI, H. M., & SANTILLÁN LARA, L. A. (2005). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE TARIFACIÓN PARA LOCUTORIOS*. Quito.
- GitHub. (2008). *Analog Input*. Obtenido de <https://github.com/ytai/ioio/wiki/Analog-Input>
- GitHub. (2008). *Digital IO*. Obtenido de <https://github.com/ytai/ioio/wiki/Digital-IO>
- GitHub. (2008). *Getting To Know The Board*. Obtenido de <https://github.com/ytai/ioio/wiki/Getting-To-Know-The-Board>
- GitHub. (2008). *IOIO Over Bluetooth*. Obtenido de <https://github.com/ytai/ioio/wiki/IOIO-Over-Bluetooth>
- GitHub. (2008). *IOIOLibBasics*. Obtenido de <https://github.com/ytai/ioio/wiki/IOIOLib-Basics>
- GitHub. (2008). *Power Supply*. Obtenido de <https://github.com/ytai/ioio/wiki/Power-Supply>
- GitHub. (2008). *Power Supply*. Obtenido de <https://github.com/ytai/ioio/wiki/Power-Supply>
- GitHub. (2008). *Pwm Output*. Obtenido de <https://github.com/ytai/ioio/wiki/PWM-Output>
- GitHub. (2008). *SPI*. Obtenido de <https://github.com/ytai/ioio/wiki/SPI>
- GitHub. (2008). *TWI*. Obtenido de <https://github.com/ytai/ioio/wiki/TWI>
- GitHub. (2008). *UART*. Obtenido de <https://github.com/ytai/ioio/wiki/UART>
- Granadino, C., & Suárez, J. (s.f.). *El bus I2C*. Chile: Universidad Técnica Federico Santa María .
- Huerta, E., Manguiaterra, A., & Gustavo, N. (2005). *Posicionamiento satelital*. Argentina: A.U.G.M.
- IOIO for Android*. (s.f.). Obtenido de <https://www.sparkfun.com/products/10748>
- Jara, P., & Nazar, P. (s.f.). *Estándar IEEE 802.11 X de las WLAN*. Buenos Aires: Educecne.
- Microchip. (2010). *PIC24FJ256DA210 FAMILY*. Microchip.
- Microchip. (2013). *MRF24WB0MA/MRF24WB0MB*.
- Molina, F. (s.f.). *Global positioning system*. España.
- Monk, S. (2012). *Making Android Accessories With IOIO* (Vol. I). O'Reilly Media.
- Pérez, E. L. (s.f.). *Curso de Redes de Microcontroladores PIC (PROTOCOLO SPI)*. México: Ingeniería en Microcontroladores.

- Pérochon, S. (2012). *Android Guia de desarrollo de aplicaciones para smartphones y tabletas*. Barcelona: Ediciones ENI.
- Quectel. (2011). *L30 Quectel GPS Engine*. Shanghai: Quectel.
- Quectel. (2013). *Quectel L30 Compact GPS Module*.
- Raúl Esteve Bosch, J. F. (2005). *Fundamentos de electrónica digital*. Valencia.
- RobotFreak. (2008). *IOIO-Rover*. Obtenido de <http://letsmakerobots.com/node/33968>
- ROVING NETWORKS. (2011). *RN-XV Data Sheet*. Arizona: ROVING NETWORKS.
- sgoliver.net foro. (s.f.). *Estructura de un proyecto Android*. Obtenido de <http://www.sgoliver.net/blog/?p=1278>
- Texas Instrument. (2000). *LM35*.
- Texas Instrument. (2004). *LM-358 DUAL OPERATIONAL AMPLIFIERS*. Dallas: Texas Instrument.
- Ucontrol. (2008). Matrices de LEDs. *Ucontrol Electrónica General Pic's en particular*, 68.
- Universidad Politécnica de Valencia. (Abril de 2013). *Elementos de un proyecto Android*. Obtenido de <http://www.androidcurso.com/index.php/recursos-didacticos/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/148-elementos-de-un-proyecto-android>
- USERS. (s.f.). Microcontroladores funcionamiento programación y usos prácticos. *USERS*, 70-76.
- Valverde Rebaza, J. C. (2007). *El Estándar Inalámbrico ZigBee*. Trujillo: Perú.
- Valverde, J. (2007). *El Estándar Inalámbrico ZigBee*. Trujillo.
- Xatakandroid. (2005). *¿Qué es Android?* Obtenido de <http://www.xatakandroid.com/sistema-operativo/que-es-android>
- Zambrano, B. J. (2011). *DISEÑO E IMPLEMENTACIÓN DE UN KIT DE APLICACIONES PARAPERSONAS CON DISCAPACIDAD VISUAL UTILIZANDO LA*. Sangolqui.
- zhongzhouOpto. (s.f.). *SPECIFICATION FOR ZHONGZHOU LED LAMP* .

- **Anexos**

Código de la aplicación en Eclipse

```
package tesis.teis_i2c;
```

```
import io.lib.api.TwiMaster;
```

```
import io.lib.api.exception.ConnectionLostException;
```

```

import ioio.lib.util.AbstractIOIOActivity;
import tesis.teis_i2c.R.id;
import android.os.Bundle;
import android.widget.TextView;

public class TesisActivity extends AbstractIOIOActivity {

    // variable que define el numero de modulo I2C de la IOIO
    private final int twiNum = 0;
    // direccion del dispositivo I2C
    private final int direccion = 45;
    //variable que aloja el valor de la temperatura en grados centigrados
    float celsius;
    // Variables que contienen el valor de la lectura
    String Valor, contador1;
    // Variables para asignacion de componetes del formulario
    private TextView tvTemperatura;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.tesis_activity);
        // atar los componentes del formulario
        tvTemperatura = (TextView)findViewById(id.tvTemperatura);
    }

    class IOIOthread extends AbstractIOIOActivity.IOIOThread
    {
        //Variable que permite la utilizacion del modulo I2C del
        //modulo
        private TwiMaster twi;

        @Override
        protected void setup () throws ConnectionLostException
        {
            // Inicializacion del modulo I2C en el modulo 0 a una frecuencia de 100

```

```

//khz y en modo colector abierto
twi = iio_.openTwiMaster(0, TwiMaster.Rate.RATE_100KHz,false);

}

@Override
protected void loop () throws ConnectionLostException
{
    // proceso que recoge la informacion del sensor
    //direccion del sensor
    byte[] request = new byte[] { 0x48 };
    //vector que almacena los datos del sensor
    byte[] tempdata = new byte[2];
    //variable que encera el sensor
    // lee la temperatura del sensor
    try {
        twi.writeRead(0x48, false, request, request.length, tempdata,
tempdata.length);
    } catch (InterruptedException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }
    //separo los datos obetenidos por el sensor via I2C
    byte MSB = tempdata[0];
    byte LSB = tempdata[1];
    // Operacion que permite imprimir el valor de la temperatura
    // en caso de ser negativa
    int TemperatureSum = ((MSB << 8) | LSB) >> 4;
    // transformo la temperatura a grados centigrados
    float celsius = (float) (TemperatureSum*0.0625);
    // convierte la temperatura de float a string
    Valor = Float.toString(celsius);
    // SubProceso que asigna al textView el numero convertido
    setText(Valor);
    try {
        sleep(1000);
    } catch (InterruptedException e) {

```



```

        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

@Override
protected AbstractIOActivity.IOIOThread createIOIOThread()
{
    return new IOIOThread();
}

public void setText (final String str1) {

    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            // Asigna al TextView una variable string
            tvTemperatura.setText(str1);
        }
    });
}
}

```

## 4.5 PRÁCTICAS CON EL MÓDULO DE COMUNICACIÓN SERIAL

### 4.5.1 MÓDULO BLUETOOTH

- **Tema: Lector de temperatura mediante Bluetooth**
- **Objetivos**
  - Objetivo General

Establecer una red Bluetooth punto a punto, para leer la temperatura de forma inalámbrica mediante el uso de un módulo Bluetooth conectado a la IOIO, con el fin de demostrar el funcionamiento de la sección Bluetooth del entrenador.

- **Objetivos específicos**
  - Conocer y comprender la tecnología Bluetooth para la transmisión de información.
  - Investigar el funcionamiento del módulo de comunicación serial de la IOIO.
  - Configurar el módulo Bluetooth2 click para el envío y recepción de información serial.
  - Realizar el flujo grama necesario para la aplicación.
  - Escribir el código que controla el lector de temperatura inalámbrica.
  - Compilar el código y ejecutar la aplicación en el dispositivo Android.
  - Comunicar la PC (Bluetooth) con el módulo Bluetooth2 Click conectado a la IOIO.
  - Verificar el funcionamiento y corregir errores si en caso se presentan.

- **Marco teórico**

#### **Bluetooth**

Bluetooth que en español significa “Diente azul”, nace en 1994 como un proyecto con el objetivo de conectar dispositivos electrónicos como celulares, laptops y Pc’s, sin la necesidad de usar cables; años más tarde, en 1998 un grupo de industrias entre ellas Nokia, Toshiba, Intel, IBM y Ericsson formaron un grupo llamado SIG (Special Interest Group) y crearon una especificación global para la transmisión sin hilos de corto alcance (Carrillo Pérez, 2006).

Bluetooth es una tecnología de transmisión perteneciente a las redes inalámbricas de área personal WPAN<sup>29</sup>, tiene un alcance de hasta 10 metros, y sus principales características son la robustez, bajo consumo de potencia y bajo costo; está especificado en el estándar IEE 802.15.1, trabaja en el rango de frecuencia de 2.402Ghz a 2.480Ghz

---

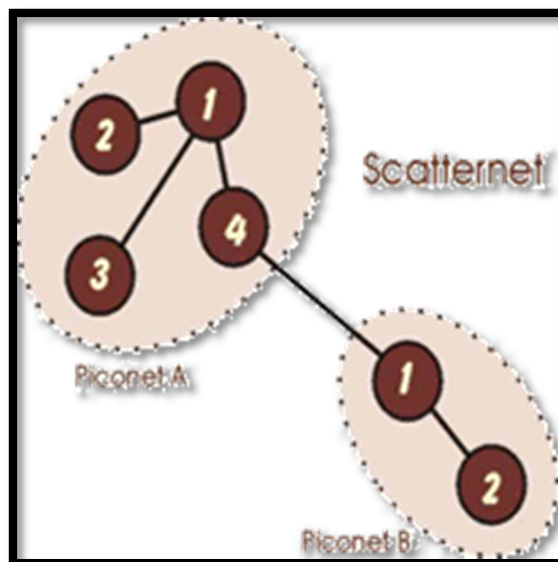
<sup>29</sup> WPAN: Wireless personal area network

con espectro ensanchado (spread spectrum) y saltos de frecuencia (frequency hopping). Es capaz de alcanzar velocidades de transmisión entre 720 Kbps y 1Mbps sin línea de vista.

En cuestión de seguridad Bluetooth presenta las siguientes características que aseguran privacidad y seguridad de la información (Carrillo Pérez, 2006):

- Rutina de pregunta respuesta para la autenticación
- Corriente cifrada de datos para la inscripción
- Generación de clave de sesión.

La topología de las redes Bluetooth pueden ser punto a punto o punto multipunto; estas redes se denominan piconets y pueden tener hasta 8 conexiones punto a punto. En una piconet, un dispositivo debe actuar como master y es el encargado de enviar la señal de reloj que es necesario para la sincronización y además la información referente a los saltos de frecuencia; el resto de dispositivos son esclavos. La unión de dos piconets se denomina scatternets.



**Figura 74.** Topologías Bluetooth

**Fuente:** Bluetooth. Recuperado de <http://artemisa.unicauca.edu.co/~dabravo/bluetooth/quees.htm>

Actualmente la gran mayoría de los dispositivos móviles traen consigo un módulo de comunicación Bluetooth, que permiten conectarse entre sí y compartir recursos como videos, imágenes, música, etc.

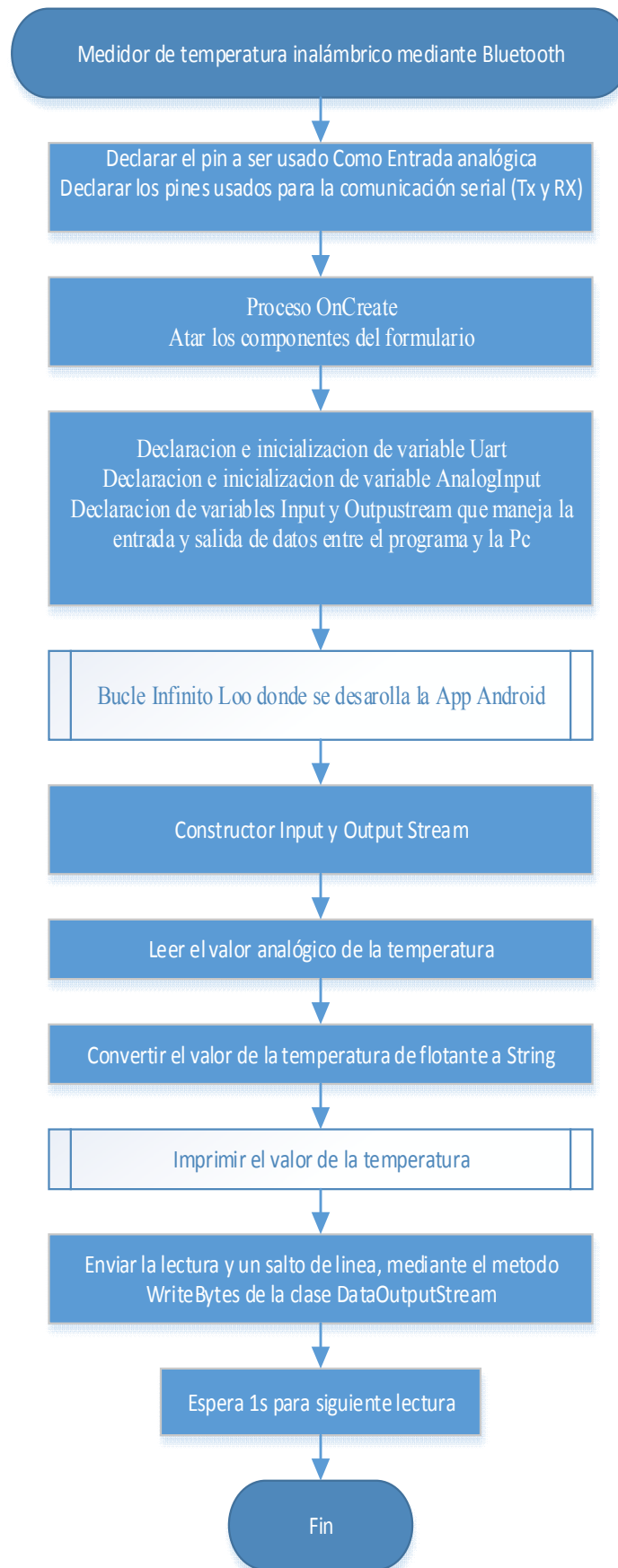
Para esta aplicación se demanda de los siguientes requerimientos.

**Tabla 30.** Requerimientos de la aplicación con el módulo Bluetooth

<b>Dispositivo</b>	<b>Requerimientos</b>
<b>Tarjeta IOIO</b>	Módulo análogo digital y módulo de comunicación serial.
<b>Sensor de temperatura</b>	Sensor de temperatura LM35
<b>Módulo Bluetooth</b>	Módulo Bluetooth que se conecta a la IOIO y permite enviar datos al Pc
<b>Computador</b>	Computador con módulo Bluetooth conectado al puerto USB.
<b>Equipo móvil Android</b>	Equipo que interactúa con la IOIO
<b>Dispositivo Bluetooth o cable de comunicación usb del equipo Android</b>	Interfaz de comunicación IOIO – Dispositivo móvil Android.

**Fuente:** Elaborado por Byron Valenzuela

- **Flujo grama**



- **Desarrollo**

**Enunciado: Desarrollar una aplicación que permita leer la temperatura de un sensor LM35 de forma inalámbrica mediante Bluetooth.**

A continuación se describen los pines y materiales usados en la aplicación lector de temperatura mediante Bluetooth.

**Tabla 31.** Materiales necesarios para la aplicación Bluetooth

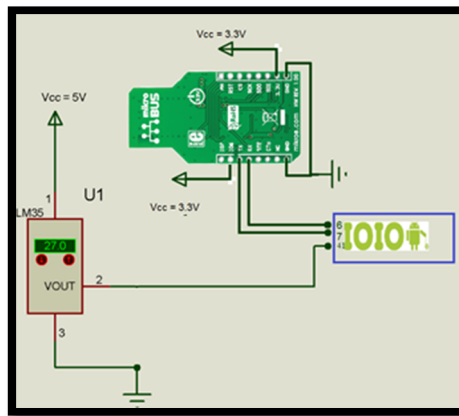
<b>Placa IOIO</b>	Pines de transmisión (Tx) y recepción (Rx) Uart pines 6 y 7 respectivamente, pin análogo digital pin 41
<b>Entrenador</b>	Sensor de temperatura LM35, y módulo Bluetooth2 Click
<b>Computador</b>	Computador con el programa hyperterminal o software semejante.
<b>Módulo Bluetooth</b>	Dispositivo Bluetooth conectado al puerto USB del computador.
<b>Cable Serial</b>	Convertidor serial a USB.

**Fuente:** Elaborado por Byron Valenzuela

- Antes de conectar el módulo Bluetooth a la IOIO es necesario configurar el dispositivo Bluetooth2 Click vía serial mediante comandos de configuración propios del fabricante, y definir parámetros necesarios para esta aplicación como: el nombre del dispositivo, la velocidad de transmisión, bits de parada, y paridad para la comunicación serial de la PC con el dispositivo.

- Posteriormente se realiza la vinculación del dispositivo Bluetooth conectado a la Pc, con el módulo Bluetooth2 Click para la recepción de la información en este caso de la temperatura.
- Conectar a la placa IOIO el módulo Bluetooth2 click para el envío de la información hacia la PC.
- Abrir el terminal (hyperterminal), colocar los parámetros configurados y verificar si la información está arribando de manera correcta.
- En caso de presentarse algún problema de comunicación seguir el proceso anterior con el fin de encontrar y corregir el error que imposibilita el funcionamiento de la comunicación.

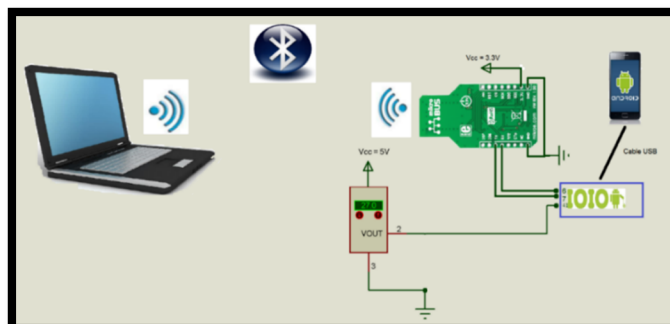
A continuación se presenta un esquema de conexión de la IOIO que actua como transmisor de temperatura en esta aplicación.



**Figura 75.** Esquema de conexión de la IOIO con el módulo Bluetooth2 Click

**Fuente:** Elaborado por Byron Valenzuela

Y el esquema general de la aplicación es la siguiente:



**Figura 76.** Diagrama general de la aplicación medidor de temperatura inalámbrico

**Fuente:** Elaborado por Byron Valenzuela

- **Análisis de resultados**

Efectivamente, al ejecutar la aplicación y verificar el funcionamiento, se puede observar que la temperatura que se obtiene en el equipo móvil es la misma que se recibe en el computador vía hyperterminal.

- **Conclusiones**

- Uno de los inconvenientes de la comunicación Bluetooth es la distancia de operación, ya que solo permite realizar aplicaciones con distancias de redes WPAN hasta 10m.
- Bluetooth es una tecnología usada para aplicaciones que requieran bajo consumo de potencia, es por tal razón que hoy en día esta tecnología forma parte de la gran mayoría de dispositivos móviles y equipos que permiten interactuar de manera rápida y eficiente con el usuario.
- Se ha desarrollado un lector de temperatura inalámbrico sencillo y útil capaz de brindar prestaciones en circunstancias donde se necesite de un monitoreo continuo de temperatura desde dos estaciones diferentes.

- **Recomendaciones**

- Se recomienda que se continúe con el desarrollo de la aplicación y no dejarlo como un monitoreo de temperatura, se podría incluir algún actuador y circuito exterior, para que funcione como un control de temperatura haciendo más eficiente el uso de la comunicación Bluetooth.
- Se recomienda no utilizar otros equipos que trabajen el rango de frecuencia de 2.4Ghz ya que se podría tener problemas con interferencia en la comunicación.

- **Bibliografía**

© 2013 Bluetooth SIG, Inc. Todos los derechos reservados, página oficial Bluetooth SIG recuperado de <http://www.bluetooth.com/>.

mikroElektronika. (2012). *mikroBUS pinout Standar Especification*.

A. Carretero, F. F. (2009). *Electrónica*. Editex.

Agüero, R. (s.f.). *Redes Inalámbricas de Área Local y Personal WLAN El estándar IEEE 802.11*. Cantabria.

Argüello, D. J. (2012). *Desarrollo de una aplicación que permita la captura almacenamiento, reproducción, administración y envío de archivos de vídeo, audio e imágenes utilizando la*



*tecnología bluetooth, para dispositivos móviles basados en el sistema operativo Android.*  
Quito.

Carballar, J. (2010). *WI-FI lo que necesita conocer*. Madrid: RC Libros.

Carrillo Pérez, M. d. (2006). *Estudio y análisis de rendimiento Bluetooth*. Madrid.

Castillo, D. (2012). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ALARMA COMUNITARIA A BASE DE MÓDULOS INALÁMBRICOS UTILIZANDO TECNOLOGÍA ZIGBEE*. Ibarra.

Clanar Internacional. (s.f.). *Internet y redes inalámbricas*. Arequipa: Clanar.

Collaguazo, G. (2009). *Sistemas Basados en Microprocesadores*. Ibarra.

Correia, P. (2002). *Guía práctica del GPS*. Barcelona: Marcombo.

Creative Commons . (s.f.). *Arduino*. Obtenido de <http://arduino.cc/en/Main/CopyrightNotice>

Daniel Benchimol. (2011). *Microcontroladores*. Buenos Aires: DALAGA S.A.

Digi International, Inc. (2009). *XBee®/XBee-PRO® RF Modules*. New York.

Dignani, J. (2011). *Análisis del protocolo ZigBee*.

Dignani, J. P. (2011). *Análisis del protocolo ZigBee*.

Fairchil Semiconductor. (1999). *MM74C922 • MM74C923*.

FAIRCHILD. (2000). *DM74LS47 BCD to 7-Segment Decoder/Driver with Open-Collector Outputs*.  
FAIRCHILD SEMICONDUCTOR.

Fernández, A. M. (2004). *EL BUS I2C*. Córdoba.

GARCÍA CELI, H. M., & SANTILLÁN LARA, L. A. (2005). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE TARIFACIÓN PARA LOCUTORIOS*. Quito.

GitHub. (2008). *Analog Input*. Obtenido de <https://github.com/ytai/ioio/wiki/Analog-Input>

GitHub. (2008). *Digital IO*. Obtenido de <https://github.com/ytai/ioio/wiki/Digital-IO>

GitHub. (2008). *Getting To Know The Board*. Obtenido de  
<https://github.com/ytai/ioio/wiki/Getting-To-Know-The-Board>

GitHub. (2008). *IOIO Over Bluetooth*. Obtenido de <https://github.com/ytai/ioio/wiki/IOIO-Over-Bluetooth>

GitHub. (2008). *IOIOLibBasics*. Obtenido de <https://github.com/ytai/ioio/wiki/IOIOLib-Basics>

GitHub. (2008). *Power Supply*. Obtenido de <https://github.com/ytai/ioio/wiki/Power-Supply>

GitHub. (2008). *Power Supply*. Obtenido de <https://github.com/ytai/ioio/wiki/Power-Supply>

GitHub. (2008). *Pwm Output*. Obtenido de <https://github.com/ytai/ioio/wiki/PWM-Output>

GitHub. (2008). *SPI*. Obtenido de <https://github.com/ytai/ioio/wiki/SPI>

GitHub. (2008). *TWI*. Obtenido de <https://github.com/ytai/ioio/wiki/TWI>

GitHub. (2008). *UART*. Obtenido de <https://github.com/ytai/ioio/wiki/UART>

Granadino, C., & Suárez, J. (s.f.). *El bus I2C*. Chile: Universidad Técnica Federico Santa María .

Huerta, E., Manguiaterra, A., & Gustavo, N. (2005). *Posicionamiento satelital*. Argentina: A.U.G.M.

*IOIO for Android*. (s.f.). Obtenido de <https://www.sparkfun.com/products/10748>

Jara, P., & Nazar, P. (s.f.). *Estándar IEEE 802.11 X de las WLAN*. Buenos Aires: Edutecne.

Microchip. (2010). *PIC24FJ256DA210 FAMILY*. Microchip.

Microchip. (2013). *MRF24WB0MA/MRF24WBOMB*.

Molina, F. (s.f.). *Global positioning system*. España.

Monk, S. (2012). *Making Android Accessories With IOIO* (Vol. I). O'Reilly Media.

Pérez, E. L. (s.f.). *Curso de Redes de Microcontroladores PIC (PROTOCOLO SPI)*. México: Ingeniería en Microcontroladores.

Pérochon, S. (2012). *Android Guía de desarrollo de aplicaciones para smartphones y tabletas*. Barcelona: Ediciones ENI.

Quectel. (2011). *L30 Quectel GPS Engine*. Shanghai: Quectel.

Quectel. (2013). *Quectel L30 Compact GPS Module*.

Raúl Esteve Bosch, J. F. (2005). *Fundamentos de electrónica digital*. Valencia.

RobotFreak. (2008). *IOIO-Rover*. Obtenido de <http://letsmakerobots.com/node/33968>

ROVING NETWORKS. (2011). *RN-XV Data Sheet*. Arizona: ROVING NETWORKS.

sgoliver.net foro. (s.f.). *Estructura de un proyecto Android*. Obtenido de <http://www.sgoliver.net/blog/?p=1278>

Texas Instrument. (2000). *LM35*.

Texas Instrument. (2004). *LM-358 DUAL OPERATIONAL AMPLIFIERS*. Dallas: Texas Instrument.

Ucontrol. (2008). *Matrices de LEDs. Ucontrol Electrónica General Pic's en particular*, 68.

Universidad Politécnica de Valencia. (Abril de 2013). *Elementos de un proyecto Android*. Obtenido de <http://www.androidcurso.com/index.php/recursos-didacticos/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/148-elementos-de-un-proyecto-android>

USERS. (s.f.). Microcontroladores funcionamiento programación y usos prácticos. *USERS*, 70-76.

Valverde Rebaza, J. C. (2007). *El Estándar Inalámbrico ZigBee*. Trujillo: Perú.

Valverde, J. (2007). *El Estándar Inalámbrico ZigBee*. Trujillo.

Xatakandroid. (2005). *¿Qué es Android?* Obtenido de <http://www.xatakandroid.com/sistema-operativo/que-es-android>

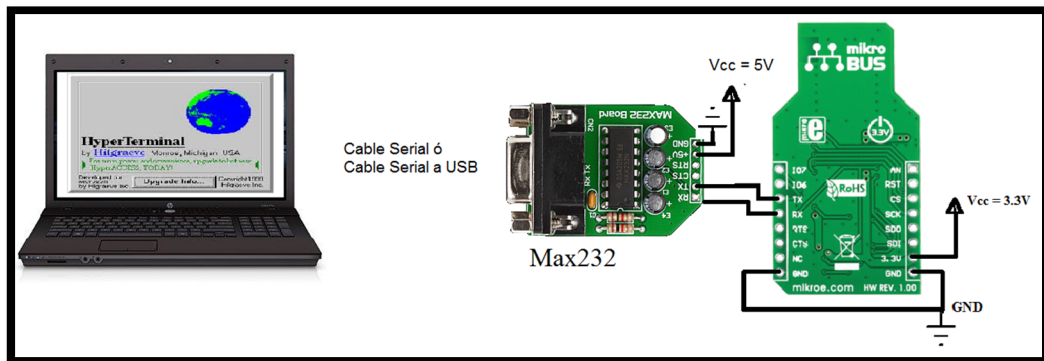
Zambrano, B. J. (2011). *DISEÑO E IMPLEMENTACIÓN DE UN KIT DE APLICACIONES PARAPERSONAS CON DISCAPACIDAD VISUAL UTILIZANDO LA*. Sangolqui.

zhongzhouOpto. (s.f.). *SPECIFICATION FOR ZHONGZHOU LED LAMP* .

- Anexos

## Configuración del módulo Bluetooth2 click vía comunicación serial

1. Conectar el módulo Bluetooth al PC de la siguiente manera.



**Figura 77.** Conexión del módulo Bluetooth con el computador

**Fuente:** Elaborado por Byron Valenzuela

2. Para configurarlo por primera vez, se debe acceder al módulo mediante las siguientes configuraciones.

Baud rate: 115200bps

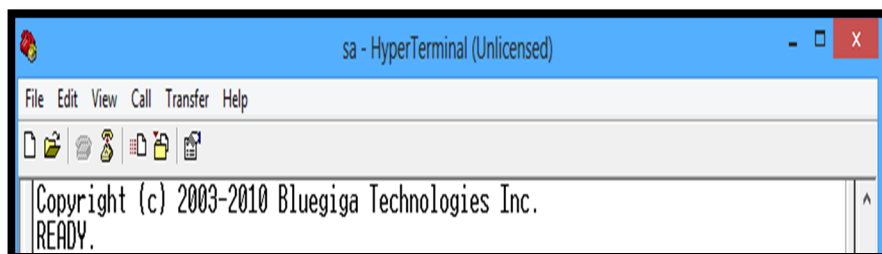
Data bits: 8

Stop bits: 1

Parity bit: No parity

HW Flow Control: Enabled

Al conseguir una conexión exitosa el módulo responde con un mensaje similar al de la siguiente figura.



**Figura 78.** Imagen de presentación del módulo

**Fuente:** Elaborado por Byron Valenzuela

3. Configurar los parámetros como: nombre del dispositivo, velocidad de transmisión, bits de parada y bits de paridad para la comunicación serial con el dispositivo mediante comandos propios del fabricante.

El comando SET BT NAME “Nombre del Dispositivo”.

Para la velocidad, bits de parada y bits de paridad el comando es el siguiente:

```
SET CONTROL BAUD {baud_rate},8{parity}{stop_bits}
```

Donde:

Baud\_rate es la velocidad en bps.

Parity: **n: No parity, e: Even parity, o: Odd parity.**

Stop bits: **1: One, 2: Two.**

Un ejemplo de configuración de este comando es el siguiente

**SET CONTROL BAUD 9600,8N1**

Este ejemplo define velocidad de transmisión 9600bps, 8 bits, sin paridad y un bit de parada.

### Vinculación con el dispositivo Bluetooth conectado al PC

1. En este proyecto se usa el módulo Bluetooth USB Trendnet TBW107-UB, el primer paso es instalar los controladores de este dispositivo.
2. Acceder al dispositivo instalado y proceder a buscar el módulo Bluetooth2 Click, con el respectivo nombre anteriormente configurado. Para este caso el nombre es BYRON.

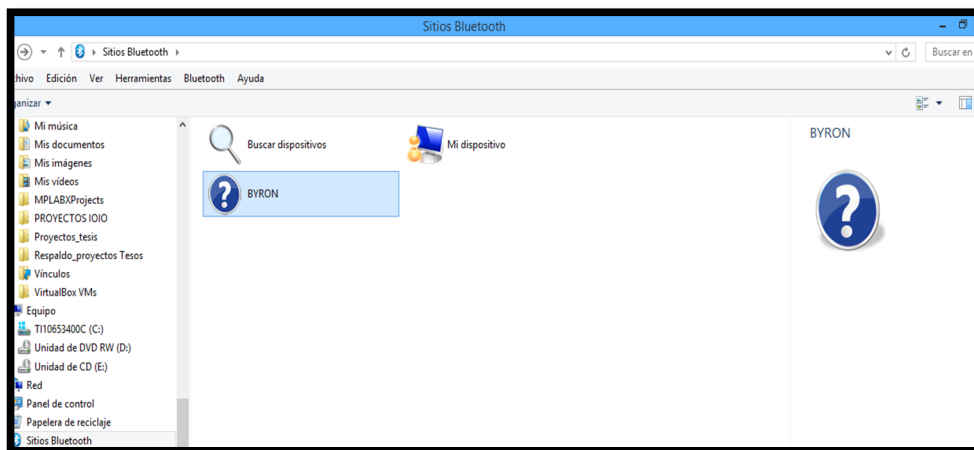
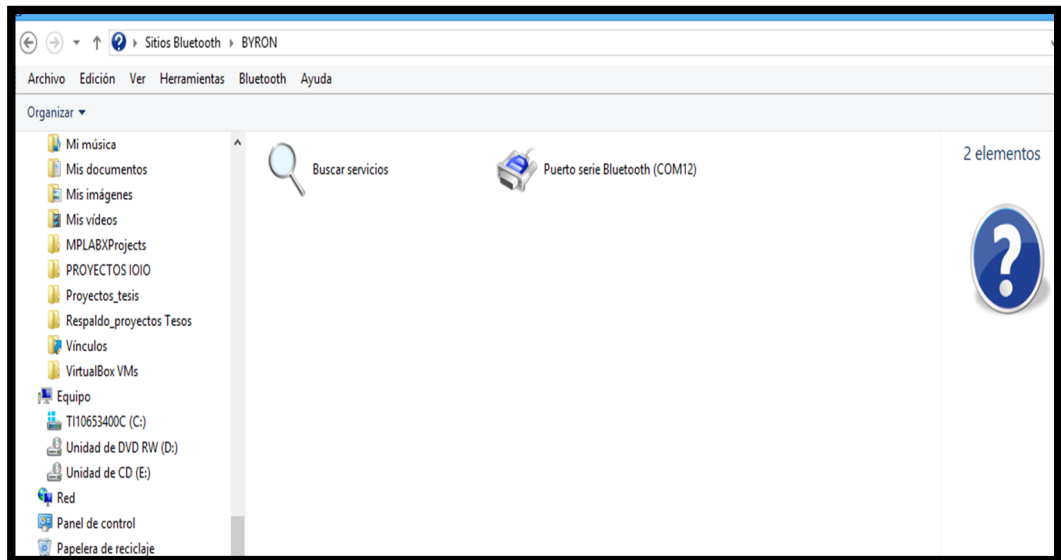


Figura 79. Dispositivos Bluetooth disponibles

Fuente: Elaborado por Byron Valenzuela

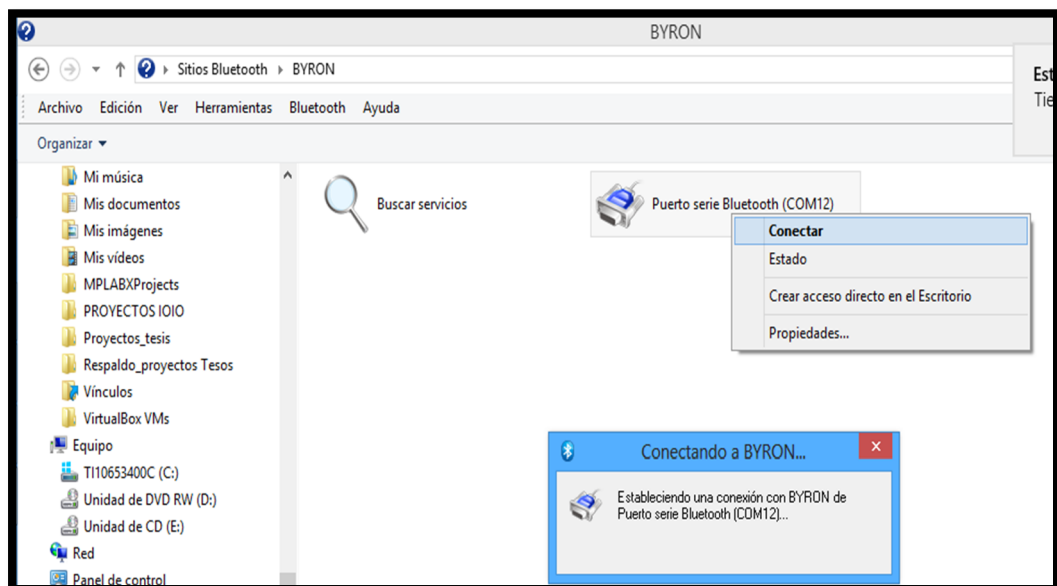
3. Ingresar a los servicios dando doble click al nombre del módulo y seleccionando el servicio de Puerto serie Bluetooth, el mismo indicará en que puerto COM va a trabajar.



**Figura 80.** Servicios del módulo Bluetooth

**Fuente:** Elaborado por Byron Valenzuela

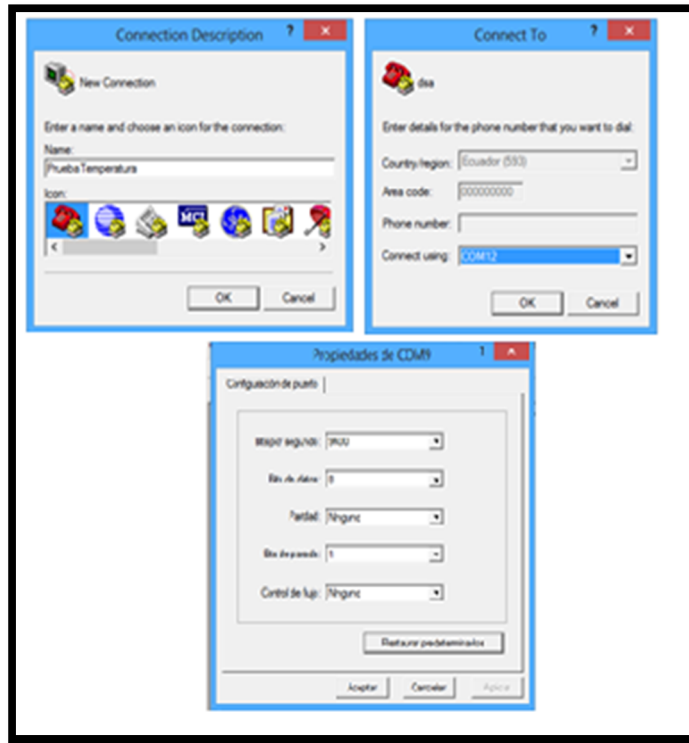
4. Establecer comunicación serial con el dispositivo Bluetooth



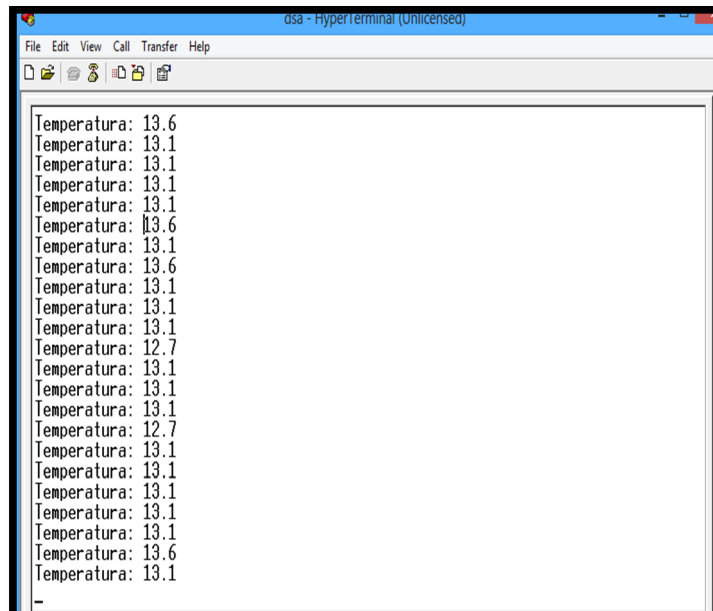
**Figura 81.** Estableciendo comunicación con el módulo Bluetooth

**Fuente:** Elaborado por Byron Valenzuela

5. Una vez creado el enlace entre el dispositivo Bluetooth y el Bluetooth2 Click, solo resta acceder al puerto COM desde el hyperterminal y recibir la información.



**Figura 82.** Configuración del puerto COM  
**Fuente:** Elaborado por Byron Valenzuela



**Figura 83.** Recepción de la temperatura en el puerto COM  
**Fuente:** Elaborado por Byron Valenzuela

## Código del Programa en Java

```
// // Aplicacion medidor de temperatura mediante BLuetooth
package com.example.teis_bluetoothtemperatura;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

import ioio.lib.api.AnalogInput;
import ioio.lib.api.Uart;
import ioio.lib.api.exception.ConnectionLostException;
import ioio.lib.util.AbstractIOActivity;
import android.os.Bundle;
import android.widget.Button;
import android.widget.TextView;

import com.example.teis_bluetoothtemperatura.R.id;

public class BluetoothActivity extends AbstractIOActivity {

    // Pin analogico que medira la temperatura
    private final int PinTemperatura = 41;

    // Pines usados para la comunicacion serial
    private final int Tx = 6;
    private final int Rx = 7;
    // TextView donde se imprime el valor de la temperatura
    private TextView tvTemperatura;
    // Variable tipo String que alojara la conversion analogica
    String lectura;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```



```

setContentView(R.layout.bluetooth_activity);
// atar los componentes del formulario
tvTemperatura = (TextView)findViewById(id.tvTemperatura);

}

```

```

class IOIOThread extends AbstractIOIOActivity.IOIOThread {
// Variable AnalogInput que realizara la lectura analogica de la
// temperatura
private AnalogInput temp;
// Variable tipo uart para la comunicacion serial con la IOIO
private Uart uart;
// Variable Input y OutputStrem para manejo de comunicacion I/O en
// java
private InputStream in;
private OutputStream out;

public void setup() throws ConnectionLostException
{
// inicializar la conversion analoga digital
temp = ioio_.openAnalogInput(PinTemperatura);
// inicializar la comunicacion serial de la IOIO
uart = ioio_.openUart(Rx,Tx, 9600,
                    Uart.Parity.NONE, Uart.StopBits.ONE);
// Acceder a los metodos de lectura y escritura serial
in = uart.getInputStream();
out = uart.getOutputStream();
}
public void loop() throws InterruptedException, ConnectionLostException
{
// Constuctor del DataInput y OutpuStream
DataOutputStream wr = new DataOutputStream(out);
DataInputStream rd = new DataInputStream (in);
// leer el conversor analogo digital (temperatura)
final float readTemp = temp.read();
// Convertir la lectura en String

```

```

lectura = ((Float.toString((readTemp * 5)*100 )).substring(0, 4);
// Imprimir el valor de la lectural en el textView
setText(lectura);
// retardo
sleep(10);
try {
    // envia la lectura analogica
    wr.writeBytes("Temperatura: " + lectura);
    // envia el caracter salto de linea
    wr.write(13);
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
//retardo 1s para proxima lectura
sleep(1000);
}
}
@Override
protected AbstractIOActivity.IOIOThread createIOIOThread() {
    return new IOIOThread();
}
// metodo que imprime el valor de la lectura el el textview
private void setText(final String strPot) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {

            tvTemperatura.setText(lectura);

        }
    });
}
}
}

```

## 4.5.2 MÓDULO ZIGBEE

- **Tema: Monitoreo de distancia mediante módulos Xbee.**
- **Objetivos:**
  - Objetivo general

Medir la distancia de un objeto, mediante módulos ZigBee configurados en una red punto a punto, con el propósito de comprobar el funcionamiento de la IOIO y el entrenador electrónico.

- **Objetivos específicos**
  - Entender el protocolo de comunicaciones ZigBee utilizado para la transmisión de información.
  - Investigar el funcionamiento del módulo de comunicación serial de la IOIO.
  - Configurar los módulos Xbee como medio de transmisión y recepción de información.
  - Realizar el flujo grama necesario para la aplicación.
  - Escribir el código que controla el medidor de distancia inalámbrico.
  - Compilar el código y ejecutar la aplicación en el dispositivo móvil Android.
  - Verificar el funcionamiento y corregir errores si en caso se presentan.

- **Marco teórico**

### **ZigBee**

IEEE 802.15.4 es el estándar que rige al protocolo de comunicación ZigBee, este protocolo fue creado por ZigBee Alliance una organización sin fines de lucro de la cual forman parte más de 200 grandes empresas, la mayoría de ellas fabricantes de semiconductores tales como: Mitsubishi, Honeywell, Philips, etc. Al igual que Bluetooth, el nombre ZigBee es muy peculiar, la idea vino de una colmena de abejas pululando alrededor de su panal y comunicándose entre ellas es por tal razón que ZigBee en español quiere decir “Zumbido de abejas (Valverde, 2007)

Las comunicaciones ZigBee se realizan en la banda libre de 2.4Ghz, este protocolo a diferencia de Bluetooth no utiliza FHSS (frecuecy hopping spread spectrum), utiliza una única frecuencia es decir un único canal que se escoge de entre los 16 posibles. ZigBee alcanza velocidades entre 20, 40 y 256 Kbps y un alcance en el rango de 10 a 75m, este alcance depende de la potencia de transmisión y del tipo de antena que se está utilizando.

IEEE 802.15.4 define dos tipos de dispositivos que son:

**Dispositivos con funciones completas (FFD)**

**Dispositivos con funciones reducidas (FRD)**

Mientras que ZigBee Alliance ha clasificado los dispositivos en tres tipos que son (Castillo, 2012):

**Coordinador:** Es el nodo que forma la red, establece el canal de comunicaciones y el PAN ID (identificador de red) para toda la red. Además Permite la asociación y desasociación de los dispositivos.

**Ruteador:** Dispositivo capaz de enrutar los mensajes entre los dispositivos de la red y soportar asociaciones.

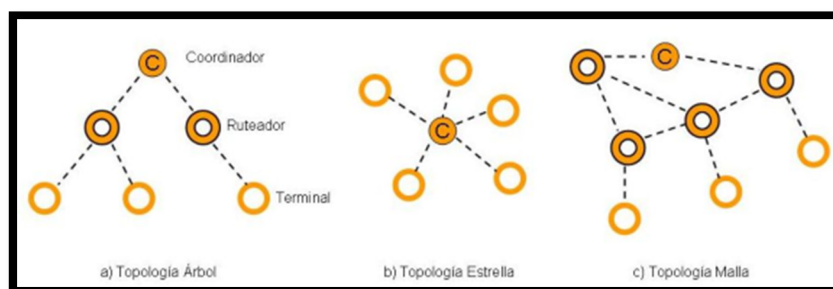
**Terminal:** Dispositivo donde se desarrollan las aplicaciones, este dispositivo debe comunicarse siempre con un ruteador o un coordinador.

Una red ZigBee puede tener hasta 65536 nodos por red y 255 por subred; soporta múltiples topologías como: estrella, árbol y malla.

**Topología estrella:** En una topología estrella se tiene un dispositivo tipo FFD que es quien asume el rol de coordinador de red y uno o varios nodos hijos; en esta topología el rango de la red está limitado al rango de transmisión del coordinador.

**Topología árbol:** Esta topología consiste en un coordinador con una o más ruteadores, donde un ruteador es capaz de tener nodos hijos y la comunicación directa existe solo a través de la relación padre e hijo.

**Topología malla:** Es muy similar a la topología tipo árbol, y es una extensión de comunicación entre pares (per to per). En la topología malla los dispositivos FFDs pueden comunicarse directamente entre sí, y los nodos terminales solo pueden intercambiar información con sus nodos padres.



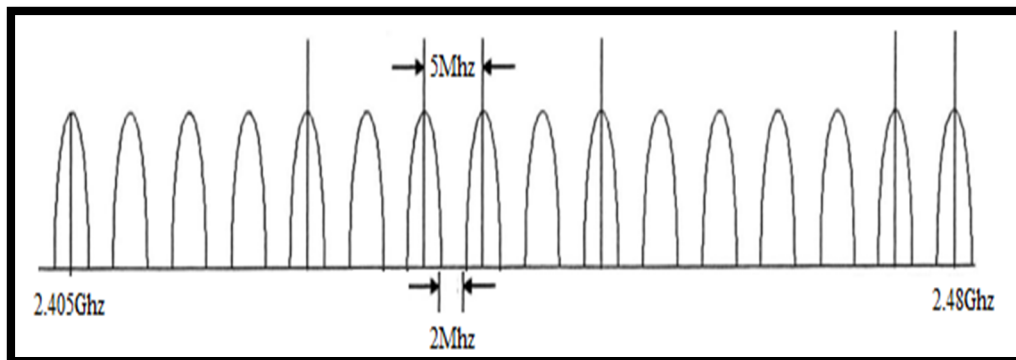
**Figura 84.** Topologías ZigBee

**Fuente:** (Dignani J. P., 2011)

## Arquitectura

ZigBee está definido por capas y se basa en el modelo de referencia OSI (Open System Interconnection) de 7 capas, de las cuales solo utiliza solo 4 con el propósito de simplificar la arquitectura de una red de transmisión simple, de baja potencia y bajo consumo (Dignani J. P., 2011).

**La capa física** define las frecuencias de radio (RF) que se utilizan para la transmisión de información, el estándar ofrece la posibilidad de trabajar en tres bandas de frecuencia que son: la banda 868Mhz que es la banda europea, 915Mhz que es la banda americana y la tercera que es la más utilizada que es la banda 2.4Ghz cuyo uso está permitido en todo el mundo. Esta banda proporciona 16 canales entre las frecuencias 2.405Ghz y 2.48Ghz con una separación de 5Mhz por canal.



**Figura 85.** Canales de transmisión de la banda 2.4Ghz para IEEE 802.15.4

**Fuente:** Elaborado por Byron Valenzuela, basado en (Digi International, Inc., 2009)

**La capa control de acceso al medio (MAC)** esta capa se encarga de la transmisión de las tramas, sincronización y provisión de un mecanismo de transmisión confiable, proporciona funciones de vinculación y desvinculación de dispositivos en la red.

**La capa de red** su objetivo primordial es proveer una interfaz simple para las aplicaciones de los usuarios. En esta capa se lleva a cabo el descubrimiento, y mantenimiento de la información de las rutas así como el descubrimiento de nuevos dispositivos en la red y la memorización de los mismos. En esta capa el coordinador es quien asigna la dirección a un nuevo dispositivo asociado.

**La capa aplicación** esta capa es una interfaz entre la aplicación del usuario y el nodo ZigBee, aquí se definen funciones tales como: mantener el rol que el nodo juega en la red, filtra paquetes a nivel de capa aplicación y simplifica el envío de información a diferentes nodos de la red.

Una nueva red ZigBee se establece por medio del coordinador, este al momento en que se inicializa busca nuevos coordinadores para el establecimiento de la nueva red mediante un identificador único PAN de 16 bits, en caso de existir inconvenientes con duplicidad de PAN ID, los coordinadores efectúan el proceso de resolución de dirección donde cambiará la dirección uno de los coordinadores. Cada coordinador tiene una tabla de vecindades en la cual posee las direcciones de los dispositivos (ruteadores o dispositivos finales) que están asociados a él.

En la actualidad el uso de la comunicación ZigBee se centra en la automatización de hogares y edificios, seguridad, alarmas, control de aire acondicionado, control de iluminación, monitorización de pacientes y equipos para la salud, control de procesos, sensores, etc.

Para esta aplicación son necesarios los siguientes requerimientos.

**Tabla 32.** Requerimientos de la aplicación con el módulo ZigBee

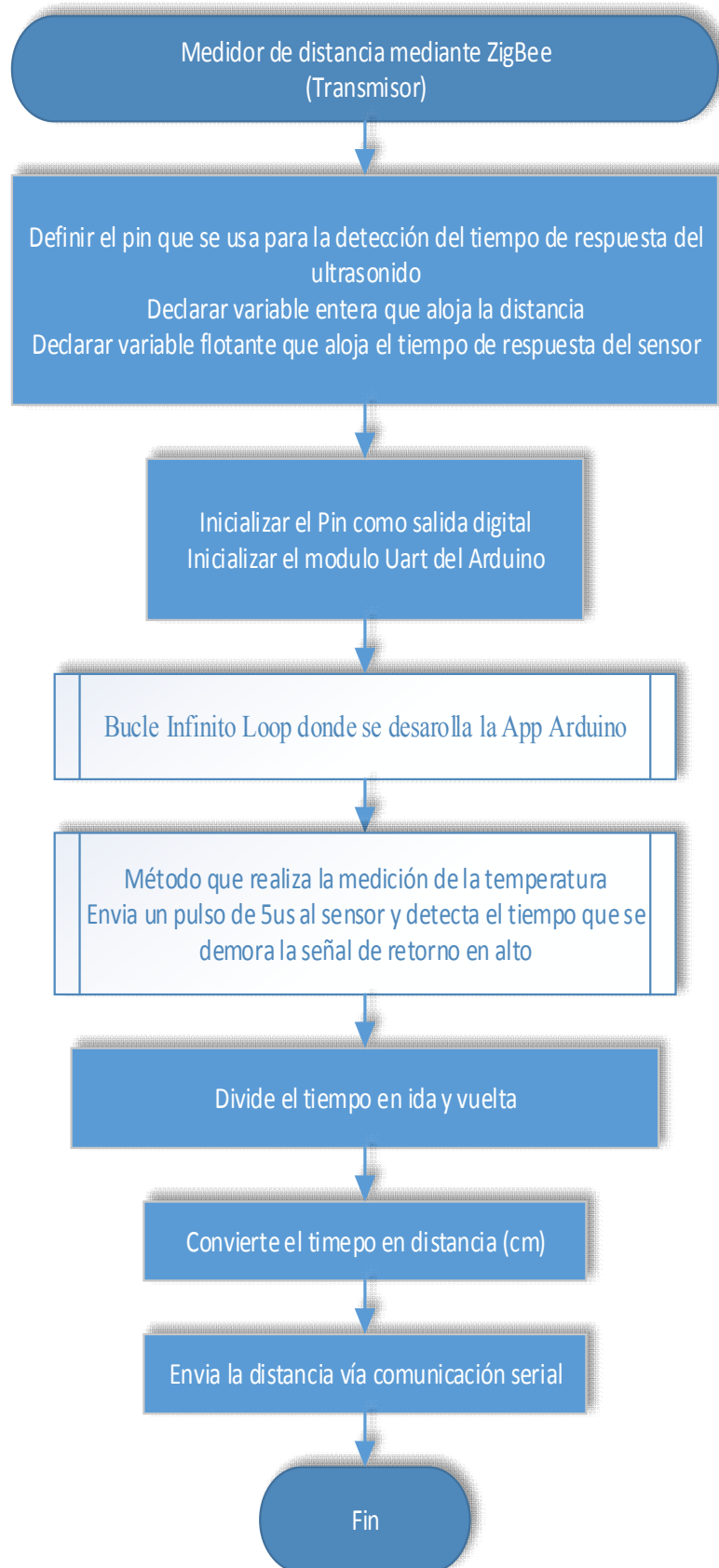
Dispositivo	Requerimientos
Tarjeta IOIO	Módulo de comunicación serial.
Tarjeta Arduino	Transmisor de la distancia
Módulos ZigBee	Transmisor y receptor permiten enviar y recibir la información de distancia
Sensor de distancia	Sensor de distancia ultrasonido
Equipo móvil Android	Equipo que interactúa con la IOIO
Dispositivo Bluetooth o cable de comunicación usb del equipo Android	Interfaz de comunicación IOIO – Dispositivo móvil Android.

**Fuente:** Elaborado por Byron Valenzuela.

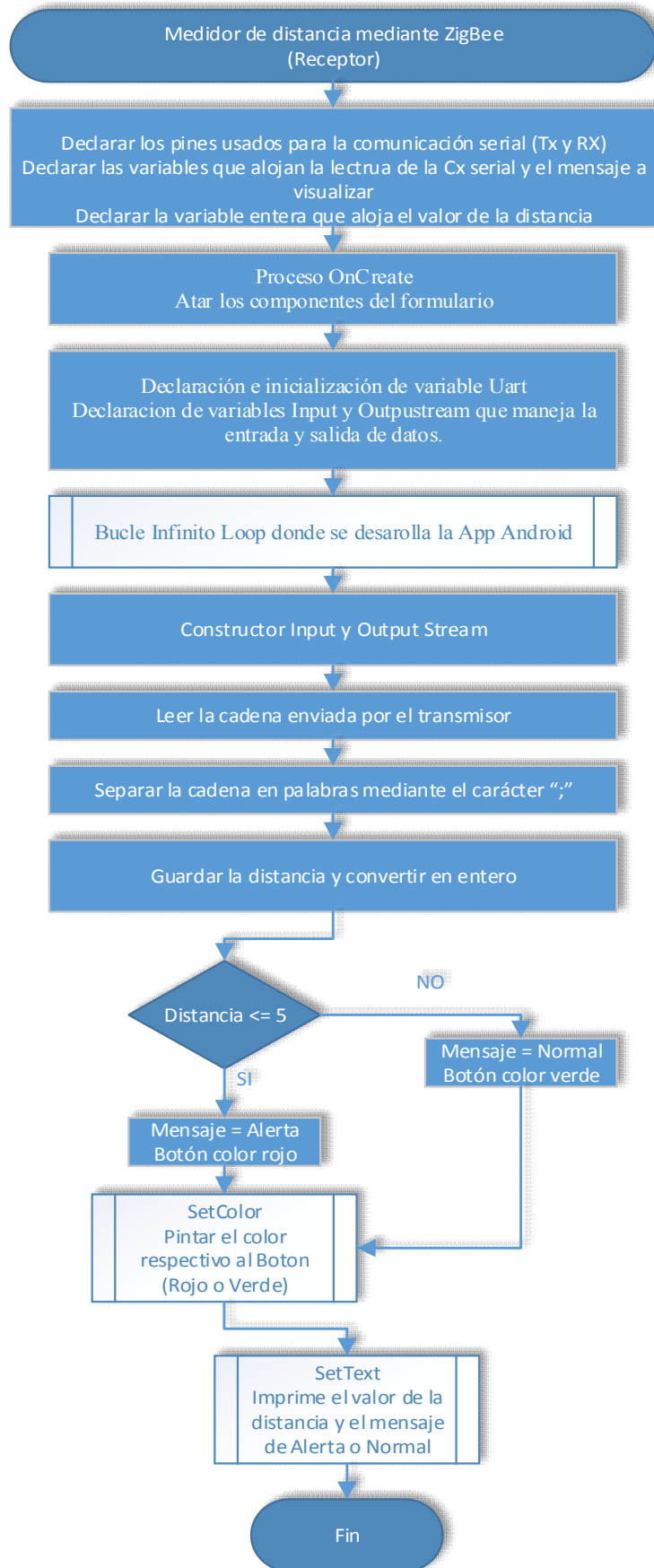
- **Flujo grama**

Para esta aplicación se tiene dos flujo gramas uno para el transmisor y otro para el receptor respectivamente, el flujo grama para el transmisor es para la placa Arduino y el flujo grama receptor es para la placa IOIO.

## Transmisor



## Receptor





- **Desarrollo**

**Enunciado:** Realizar un medidor de distancia mediante una red punto a punto con módulos ZigBee, el transmisor envía la distancia del sensor, el receptor recibe la información de la distancia y en caso de que la distancia sea menor a 5m se emite una alarma visual.

A continuación se describen los pines y materiales usados para esta aplicación.

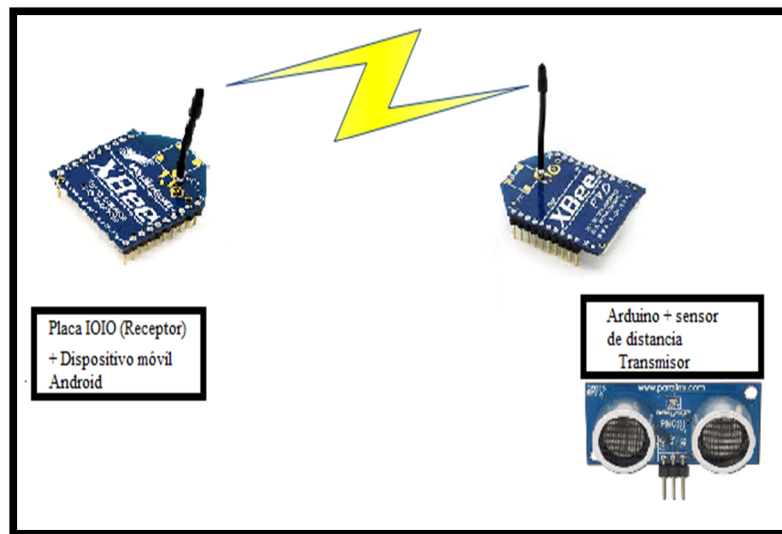
**Tabla 33.** Descripción de materiales usados en la aplicación medidor de distancia mediante módulos ZigBee

<b>Placa IOIO</b>	Pines de transmisión (Tx) y recepción (Rx) Uart pines 6 y 7 respectivamente.
<b>Placa Arduino</b>	Actúa como transmisor de la distancia mediante un módulo Xbee S1
<b>Sensor de distancia</b>	Sensor de distancia ultrasonido PING de Parallax
<b>Entrenador</b>	Módulo Xbee S1

**Fuente:** Realizado por Byron Valenzuela

Esta aplicación se divide en dos partes y circuitos independientes por un lado está el transmisor que viene a ser la placa Arduino con el sensor de distancia y por otro lado está el receptor que viene a ser la placa IOIO con el dispositivo Android.

A continuación se presenta un diagrama de la forma en que está constituida la aplicación.



**Figura 86.** Representación gráfica de la manera en que está organizada la aplicación

**Fuente:** Elaborado por Byron Valenzuela

Los módulos Xbee se comunican mediante los siguientes parámetros.

**Tabla 34.** Parámetros de configuración de los módulos XBee transmisor y receptor

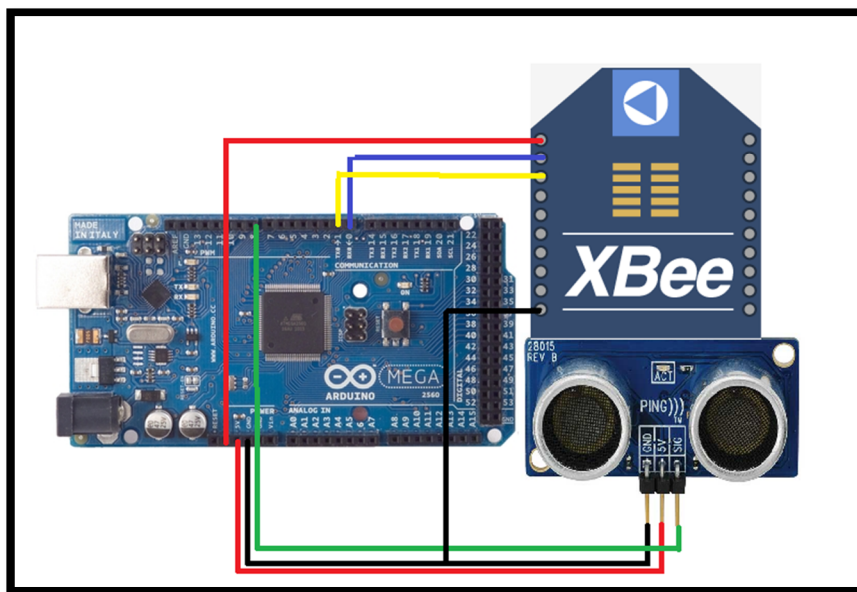
Parámetro	Transmisor	Receptor
Channel	0x10	0x10
Pan ID	9032	9032
MY	1	0
DL	0	1

**Fuente:** Elaborado por Byron Valenzuela

## Transmisor

En el lado del transmisor como se observa en la figura anterior se tiene una tarjeta Arduino, quien detecta la distancia y la envía por el módulo Xbee S1. A continuación se detalla el proceso seguido en el transmisor.

- Conectar y configurar la placa Arduino de modo que lea la distancia del sensor ultrasónico PING y la envíe vía serial al módulo Xbee.
- Configurar el módulo Xbee como transmisor y conectarlo a la Arduino de la siguiente forma



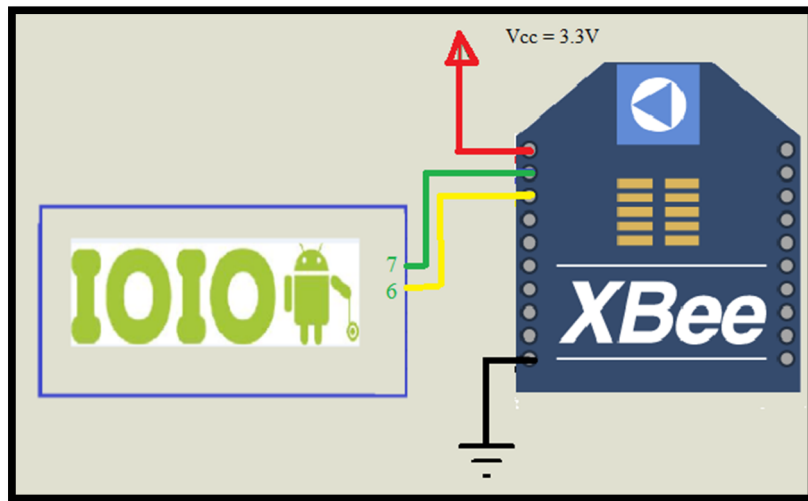
**Figura 87.** Sistema transmisor de la aplicación

**Fuente:** Elaborado por Byron Valenzuela.

## Receptor

En el lado del receptor se tiene la placa IOIO conectado al módulo Xbee, y al respectivo equipo móvil Android que es donde se visualiza la distancia enviada desde el transmisor.

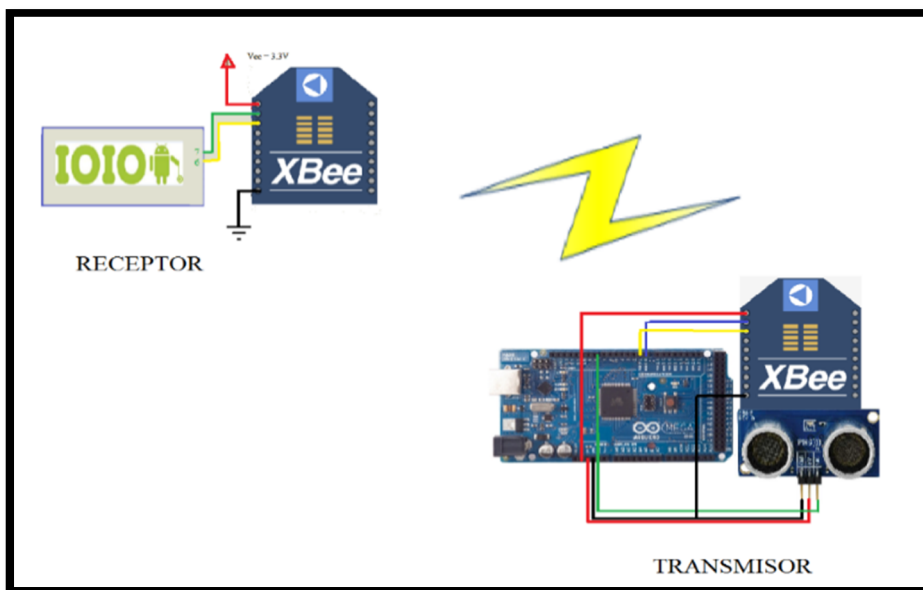
- Configurar el módulo Xbee S1 con los parámetros indicados anteriormente.
- Conectar la IOIO al Xbee mediante la comunicación serial, de modo que reciba la información enviada desde el transmisor. A continuación se presenta un diagrama de conexión de la IOIO con el Xbee.



**Figura 88.** Conexión de la IOIO con el módulo Xbee S1

**Fuente:** Elaborado por Byron Valenzuela

A continuación se tiene una representación gráfica de la conexión total de la aplicación.



**Figura 89.** Conexión total de la aplicación con módulos ZigBee

**Fuente:** Elaborado por Byron Valenzuela

- **Análisis de resultados**

Con el propósito de comprobar el funcionamiento de la aplicación a continuación se tiene una tabla de resultados medidos en el transmisor mediante una cinta métrica y en el receptor el resultado en el móvil Android.

**Tabla 35.** Resultados de las mediciones entre en transmisor y receptor

Distancia	Transmisor	Receptor
1cm	1cm	1cm
10cm	~9.8cm	~9.8cm
1m	~90.8cm	~90.8cm
2m	~1.96m	~1.96m

**Fuente:** Elaborado por Byron Valenzuela

- **Conclusiones**

- El desarrollo de esta aplicación sería de mucha utilidad en vehículos que no tengan un sistema de monitoreo de distancia al momento de dar marcha atrás.
- El protocolo de comunicación ZigBee tiene la capacidad de comunicar un mayor número de dispositivos, teniendo una capacidad de 65536 nodos de comunicación.
- El alcance de transmisión depende de la antena que utiliza el módulo, siendo capaz de alcanzar una distancia de 1km con línea de vista.

- **Recomendaciones**

- En esta aplicación se ha usado los dispositivos Xbee en una red punto a punto limitando su capacidad de trabajo, se recomienda utilizar una topología diferente con más sensores y nodos y así comprobar su funcionamiento.

- **Bibliografía**

ignani, J. P. (2011). Análisis del protocolo ZigBee.

GitHub. (2008). UART. Obtenido de <https://github.com/ytai/ioio/wiki/UART>

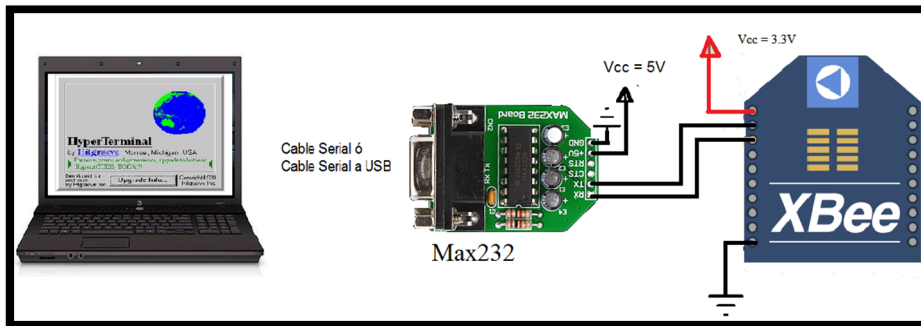
Valverde Rebaza, J. C. (2007). El Estándar Inalámbrico ZigBee. Trujillo: Perú.

- Anexos

## Configuración de los módulos Xbee S1

La configuración de los módulos Xbee S1 tanto para el transmisor como para el receptor son idénticos a diferencia de los parámetros para cada uno respectivamente. A continuación se detalla el proceso de configuración.

### 1. Conectar los módulos Xbee S1 al computador vía comunicación serial

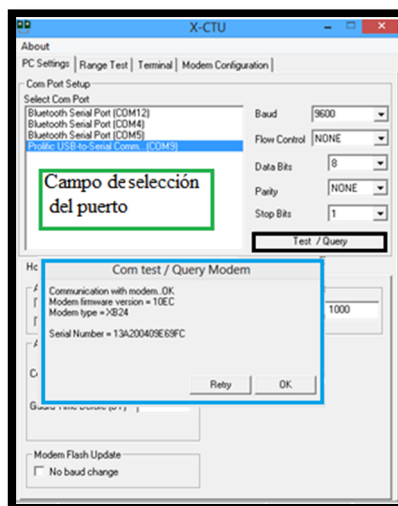


**Figura 90.** Conexión del Xbee al PC  
**Fuente:** Elaborado por Byron Valenzuela.

### 2. Descargar e instalar el software X'CTU

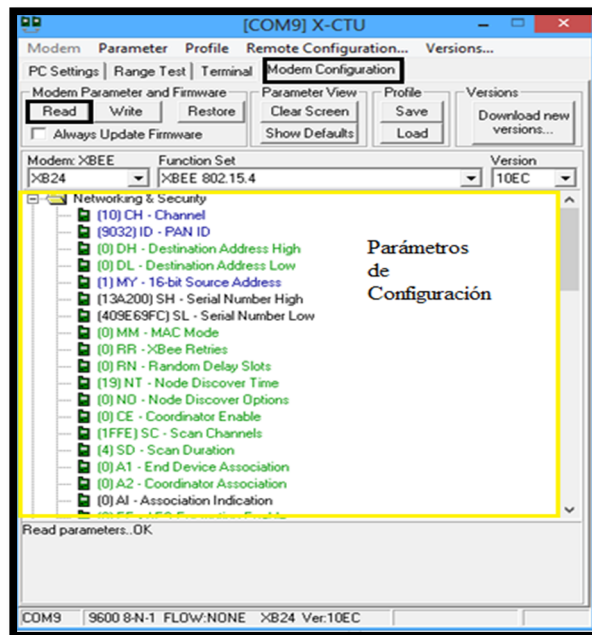
<http://www.digi.com/support/productdetail?pid=3352&osvid=57&type=utilities>

3. Ingresar al software X-CTU, seleccionar el puerto al que está conectado el cable serial y verificar si existe conectividad con el módulo dando click en Test / Query inmediatamente aparece un mensaje de confirmación del estado de la conexión.



**Figura 91.** Ventana de verificación de conectividad con el módulo  
**Fuente:** Elaborado por Byron Valenzuela

4. Ir a la pestaña Modem Configuration y presionar el botón Read, aquí aparece la información del Xbee que está conectado. Por ultimo configurar los parámetros dando click en los mismos y colocando el valor deseado.



**Figura 92.** Pestaña de configuración del módulo Xbee

**Fuente:** Elaborado por Byron Valenzuela

5. Por último cuando ya se haya cambiado los parámetros de configuración dar click en el botón Write inmediatamente se escribe y guarda los cambios en el módulo.

6. Repetir el proceso tanto para el transmisor como para el receptor de la aplicación.

### Código del Transmisor en el software Arduino

```
// Medidor de distancia Modulo PING
```

```
int signal=8;// Ping que detectara el tiempo calcula la distancia
```

```
int distance; // Variable que almacenara la distancia
```

```
unsigned long pulseduration=0; // variable que almacena el tiempo
```

```
    // de regreso de la señal
```

```
void setup()
```

```
{
```

```

pinMode(signal, OUTPUT); // Definicion del pin en modo salida
Serial.begin(9600); // inicializa la cx serial
}
// metodo que mide la distancia
void measureDistance()
{
// define el pin como salida
pinMode(signal, OUTPUT);
// envia un pulso cuadrado 5 Us
digitalWrite(signal, LOW);
delayMicroseconds(5);
digitalWrite(signal, HIGH);
delayMicroseconds(5);
digitalWrite(signal, LOW);
// define el pin como entrada digital
pinMode(signal, INPUT);
// y mide el tiempo que la serial de retorno esta en alto
pulseduration=pulseIn(signal, HIGH);
}
void loop()
{
// llama al metodo
measureDistance();
// divide el tiempo de duracion en ida y vuelta
pulseduration=pulseduration/2;
// convierte en cm el tiempo
distance = int(pulseduration/29);
// envia la distancia por el serial
Serial.print("Distance;");
Serial.print(distance);
Serial.println(";cm");
delay(500);
}

```

### **Código del Receptor en el software Eclipse**

```
// Medidor de distancia mediante ZigBee
```

```
package tesis.xbee;
```



```

import ioio.lib.api.Uart;
import ioio.lib.api.exception.ConnectionLostException;
import ioio.lib.util.AbstractIOIOActivity;
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import tesis.xbee.R.id;
import android.graphics.PorterDuff.Mode;
import android.os.Bundle;
import android.widget.Button;
import android.widget.TextView;
import android.widget.ToggleButton;

public class XbeeActivity extends AbstractIOIOActivity {

    //Pine que van a ser usados por el modulo serial de la IOIO
    private final int Tx = 6;
    private final int Rx = 7;
    // Variables TextView para visualizar la distancia
    private TextView tvMensaje,TvValor;
    // botn que emitira una alamra visual
    private Button btnAlerta;
    // Variables tipo String que alojan la lectura,
    //valor necesario para separar las cadenas y el mensaje distancia
    String lectura,valor,Mensaje;
    // variable que aloja el valor de la distancia
    int valorint =0;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.xbee_activity);
        // ata los componentes al formulario
        tvMensaje= (TextView)findViewById(id.tvDistancia);
        TvValor = (TextView)findViewById(id.tvValor);
    }
}

```

```

    btnAlerta = (Button)findViewById(id.btnAlerta);
}

class IOIOThread extends AbstractIOActivity.IOIOThread {
    // declarar variable tipo Uart para la comunicacion serial
    private Uart uart;
    // declaracion de variables tipo Input y OutputStream para
    // la comunicacion serial en java
    private InputStream in;
    private OutputStream out;
    public void setup() throws ConnectionLostException {
        try {
            //inicializar la comunicacion serial de la IOIO
            uart = ioio_.openUart(Rx,Tx, 9600,
                Uart.Parity.NONE, Uart.StopBits.ONE);
//Variable Input y OutputStrem para manejo de comunicacion I/O en
            // java
            in = uart.getInputStream();
            out = uart.getOutputStream();
        } catch (ConnectionLostException e) {
            throw e;
        }
    }
}

@SuppressWarnings("deprecation")
public void loop() throws ConnectionLostException {
    // Constuctor del DataInput y OutpuStream
    DataOutputStream wr = new DataOutputStream(out);
    DataInputStream rd = new DataInputStream (in);
    try {
        // lee la informacion tipo String que llega via serial
        lectura = rd.readLine();
        // separa la cadena en palabras separadas por ;
        // y las aloja en el vector palabras
        String[] palabras = lectura.split(";");
        // almacena el valor de la distancia que se encuentra en
        // el vector
        valor = palabras [1];
    }
}

```

```

        // convierte la distancia en entero
        valorint = Integer.parseInt(valor);
        // detecta si la distancia es menor a 5
        if(valorint <=5)
        {
            // imprime la palabra Alerta y
            // coloca de color rojo al boton
            Mensaje = "Alerta";
            setColor(Mensaje);
        }else
        {
            // imprime la palabra Normal y
            // coloca de color rojo al boton
            Mensaje = "Normal";
            setColor(Mensaje);
        }
        setText(valor,Mensaje);
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
}

```

```

@Override
protected AbstractIOActivity.IOIOThread createIOIOThread() {
    return new IOIOThread();
}
// Metodo que Imprime el valor de la temperatura y
// el estado Alerta o Normal
private void setText(final String valor,final String estado ) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            tvMensaje.setText(valor + "cm");
            TvValor.setText(estado);
        }
    });
}

```

```

    }
  });
}
// Metodo que coloca el mensaje y el color al boton
private void setColor(final String valor) {
  runOnUiThread(new Runnable() {
    @Override
    public void run() {
      // imprime el mensaje en el boton
      btnAlerta.setText(valor);
      // si esta en Alerta
      if (valor == "Alerta")
      {
        // Pone de color rojo al boton

        btnAlerta.setBackground().setColorFilter(0xFFFF0000,Mode.MULTIPLY);
      }
      else
      {
        // caso contrario pone de color verde

        btnAlerta.setBackground().setColorFilter(0xff0000ff,Mode.MULTIPLY);
      }
    }
  });
}
}

```

### 4.5.3 MÓDULO GPS

**Tema: Ubicación de un punto en el mapa mediante un módulo GPS conectado a la IOIO.**

- **Objetivos:**

- Objetivo general

Posicionar un punto dado por un módulo GPS conectado a la IOIO mediante el mapa de google, con el propósito de comprobar el módulo GPS del entrenador

- **Objetivos específicos**

- Estudiar la tecnología GPS, como medio de posicionamiento y ubicación global.
- Comprender el funcionamiento del módulo GPS2 Click que posee el entrenador electrónico.
- Investigar el uso de los mapas de google para el servicio de ubicación en dispositivos móviles.
- Realizar el flujo grama de la aplicación.
- Escribir el código necesario para ejecutar la aplicación.
- Compilar el código y ejecutar la aplicación en el dispositivo móvil Android.
- Verificar el funcionamiento y corregir errores si en caso se presentan.

- **Marco teórico**

#### **GPS**

GPS (Global Positioning System), sistema de posicionamiento global que nace alrededor de los años 1973 – 1978 cuando se hizo el lanzamiento del primer satélite, el responsable de este sistema fue el departamento de defensa de los Estados Unidos y fue concebido para uso militar (Huerta, Manguiaterra, & Gustavo, 2005).

El GPS es un sistema que tiene como principal objetivo la radionavegación basado en la recepción de señales desde una constelación de 24 satélites. Un sistema GPS consta de tres segmentos que son

Segmento espacial: Constituye el conjunto de satélites en órbita.

Segmento de control: Monitorea los satélites en órbita con el objeto de sincronizar los relojes a bordo, además verifica la integridad de las señales.

Segmento de usuario: Constituye los equipos receptores de la señal GPS.

El receptor GPS es un dispositivo electrónico utilizado por personas que navegan por aire mar o tierra capaz de captar las señales emitidas por los satélites, transformarlas en datos de posición y determinar un punto en el mapa.

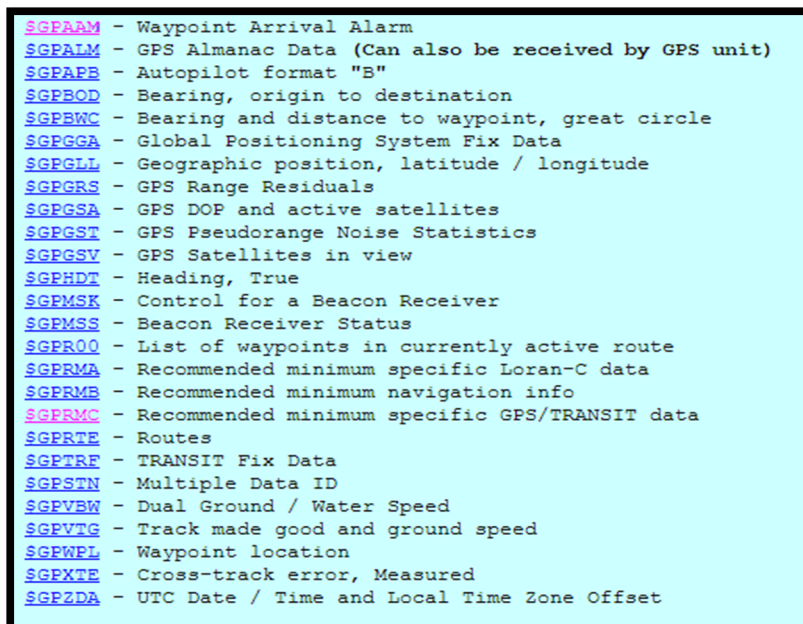
Estos instrumentos están constituidos básicamente por un receptor y una antena, el receptor consta de un mínimo de 4 canales que permiten simultáneamente procesar la información que se recibe de los satélites, la antena se conecta al receptor; las coordenadas que se calculan corresponden al centro radioeléctrico de la antena.

Los satélites emiten dos señales L1 y L2 en la banda de las microondas. La banda L1 trabaja en la frecuencia de 1575.42 Mhz es la encargada de transmitir la información de navegación y posicionamiento, mientras que L2 trabaja en la frecuencia 1227.60 Mhz, esta se emplea para el uso de posicionamiento de precisión y medición de retraso de propagación de señales en la atmósfera.

Los receptores GPS brindan una amplia gama de información al usuario entre ellas esta: los satélites localizados, satélites en seguimiento, intensidad de cada señal recibida, condición de cada satélite en seguimiento, posición, longitud, latitud, velocidad.

La mayoría de receptores GPS trabajan con el protocolo NMEA 0183 denominado así por ser creados por la Asociación Electrónica Marítima Nacional permite establecer una interfaz de comunicación con equipos marinos y receptores GPS.

Existen 26 diferentes tipos de sentencias o mensajes NMEA que el receptor GPS transmite al usuario todas ellas inician con el carácter "\$" cada sentencia presenta su información respectiva, a continuación se muestra una gráfica con las sentencias NMEA.



<a href="#">\$GPRMB</a>	- Waypoint Arrival Alarm
<a href="#">\$GPRMA</a>	- GPS Almanac Data (Can also be received by GPS unit)
<a href="#">\$GPRPB</a>	- Autopilot format "B"
<a href="#">\$GPRPD</a>	- Bearing, origin to destination
<a href="#">\$GPRPW</a>	- Bearing and distance to waypoint, great circle
<a href="#">\$GPRXA</a>	- Global Positioning System Fix Data
<a href="#">\$GPRXL</a>	- Geographic position, latitude / longitude
<a href="#">\$GPRRS</a>	- GPS Range Residuals
<a href="#">\$GPRSA</a>	- GPS DOP and active satellites
<a href="#">\$GPRST</a>	- GPS Pseudorange Noise Statistics
<a href="#">\$GPRSV</a>	- GPS Satellites in view
<a href="#">\$GPHDT</a>	- Heading, True
<a href="#">\$GPMSC</a>	- Control for a Beacon Receiver
<a href="#">\$GPMSS</a>	- Beacon Receiver Status
<a href="#">\$GPR00</a>	- List of waypoints in currently active route
<a href="#">\$GPRMA</a>	- Recommended minimum specific Loran-C data
<a href="#">\$GPRMB</a>	- Recommended minimum navigation info
<a href="#">\$GPRMC</a>	- Recommended minimum specific GPS/TRANSIT data
<a href="#">\$GPRTE</a>	- Routes
<a href="#">\$GPRTF</a>	- TRANSIT Fix Data
<a href="#">\$GPRTN</a>	- Multiple Data ID
<a href="#">\$GPRVB</a>	- Dual Ground / Water Speed
<a href="#">\$GPRVTG</a>	- Track made good and ground speed
<a href="#">\$GPRWL</a>	- Waypoint location
<a href="#">\$GPRXE</a>	- Cross-track error, Measured
<a href="#">\$GPRZDA</a>	- UTC Date / Time and Local Time Zone Offset

**Figura 93.** Sentencias NMEA

**Fuente:** NMEA sentence information. Recuperado de <http://home.mira.net/~gnb/gps/nmea.html#gprmc>

La trama que se usa en este proyecto es la "\$GPGGA" Global Positioning System Fix Data que tiene la siguiente descripción.

**Tabla 36:** Descripción de la trama "\$GPGGA"

Nombre	Ejemplo	Descripción
<b>Identificador</b>	\$GPGGA	Global Positioning System Fix Data
<b>Tiempo</b>	170834	17:08:34 UTC
<b>Latitud</b>	4124.8963, N	41d 24.8963' N or 41d 24' 54" N
<b>Longitud</b>	08151.6838, W	81d 51.6838' W or 81d 51' 41" W
- 0 = Invalid - 1 = GPS fix - 2 = DGPS fix	1	Data is from a GPS fix
<b>Número de satélites</b>	05	5 satélites activos
<b>Dilución de precisión horizontal(HDOP)</b>	1.5	Precisión relativa de la posición horizontal
<b>Altitud</b>	280.2, M	280.2 metros sobre el nivel del mar
<b>Altura del geoide por encima del elipsoide WGS84</b>	-34.0, M	-34.0 metros
<b>Tiempo desde la última actualización del DGPS</b>	Blank	Sin actualización
<b>Id de la estación DGPS de referencia</b>	Blank	Sin Id
<b>Checksum</b>	*75	Usado para verificar errores de transmisión

**Fuente:** NMEA sentence information. Recuperado de <http://home.mira.net/~gnb/gps/nmea.html#gpgga>

Los requisitos necesarios para esta aplicación son los siguientes.

**Tabla 37.** Requerimientos para la aplicación de Ubicación GPS

<b>Dispositivo</b>	<b>Requerimientos</b>
<b>Tarjeta IOIO</b>	Módulo de comunicación serial.
<b>Módulo GPS</b>	Transmisor de latitud y longitud
<b>Amplificador Operacional</b>	Amplificar los niveles de transmisión del módulo GPS
<b>Equipo móvil Android</b>	Equipo que interactúa con la IOIO
<b>Dispositivo Bluetooth o cable de comunicación usb del equipo Android</b>	Interfaz de comunicación IOIO – Dispositivo móvil Android.

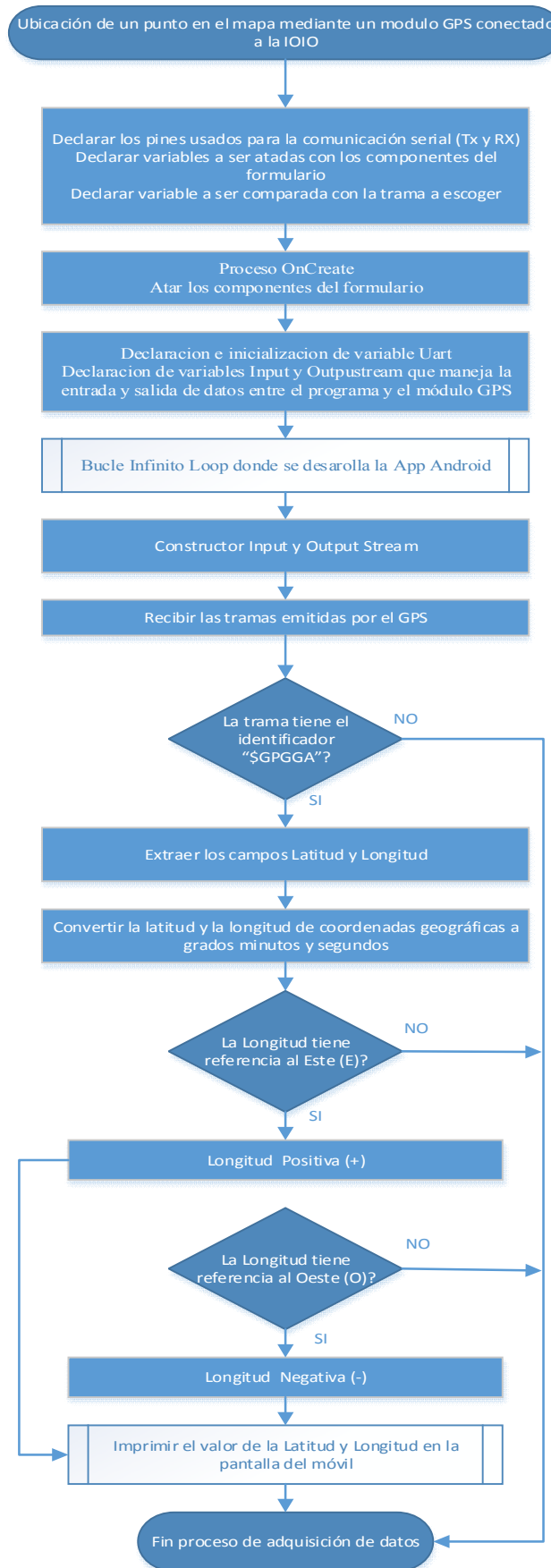
**Fuente:** Elaborado por Byron Valenzuela

- **Flujo grama**

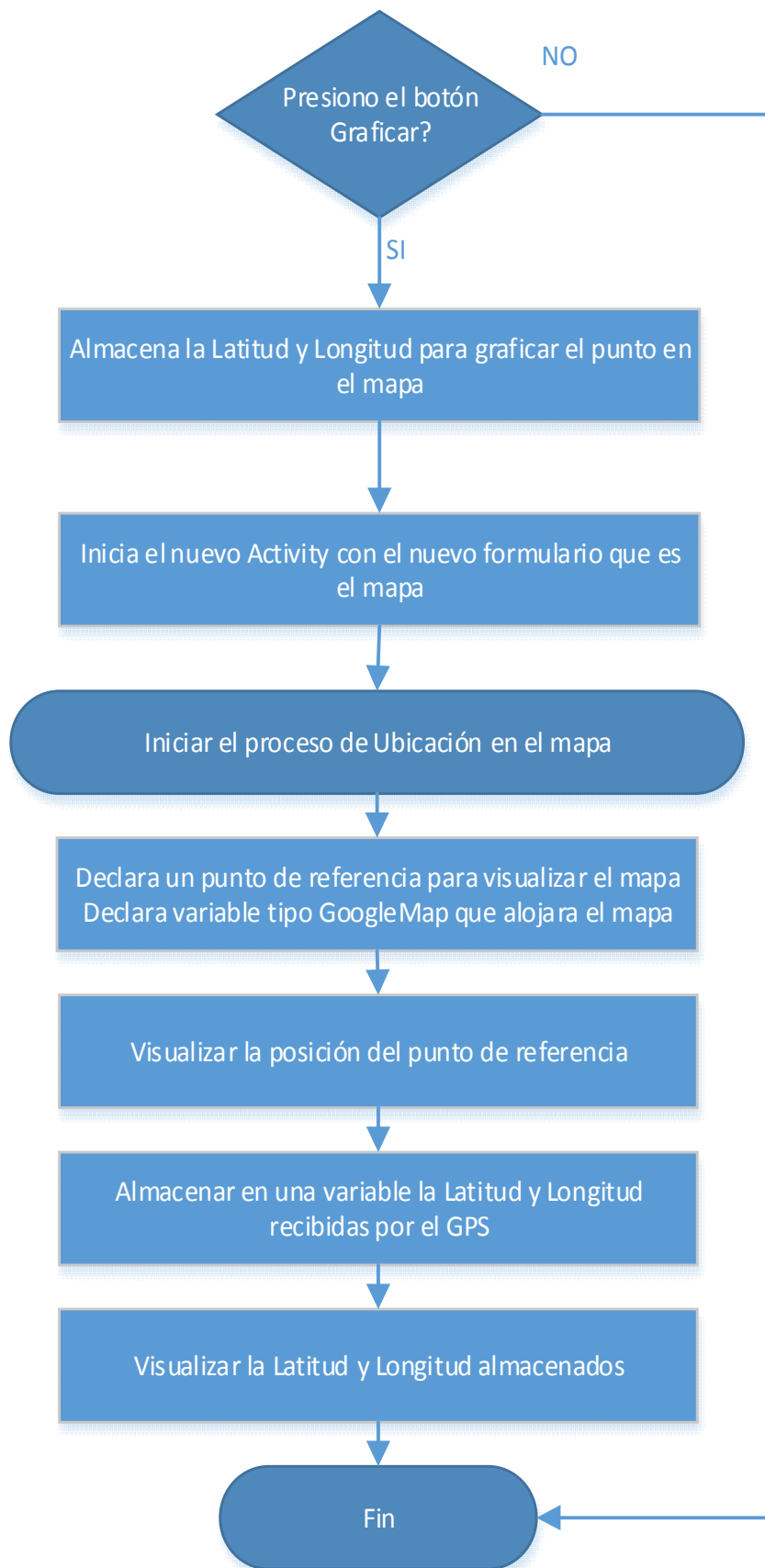
Para esta aplicación es necesario realizar dos flujo gramas que son: la recolección de datos del GPS, y el proceso de graficar el punto en el mapa.



## Recolección de datos del GPS.



## Ubicación del punto en el mapa



- **Desarrollo**

**Enunciado:** Realizar una aplicación que reciba la información de ubicación del GPS y grafique el punto recibido en un mapa.

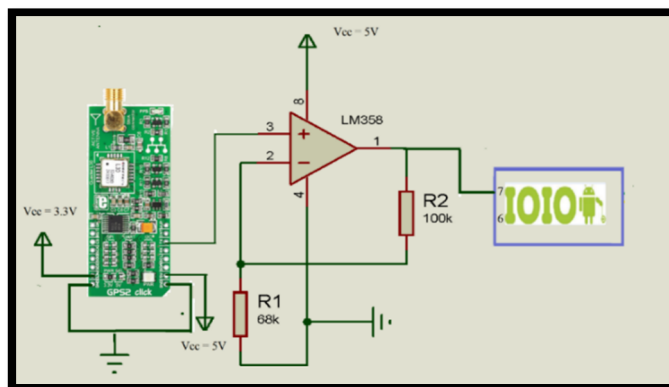
A continuación se presenta la descripción de los pines y materiales que son usados en esta aplicación.

**Tabla 38.** Descripción de los pines y requerimientos de la aplicación

<b>Placa IOIO</b>	Pines de transmisión (Tx) y recepción (Rx) Uart pines 6 y 7 respectivamente.
<b>Módulo GPS</b>	Módulo GPS2 Click que tiene el entrenador
<b>Entrenador</b>	Módulo amplificador operacional LM-358 para elevar los voltajes de transmisión y recepción del módulo GPS

**Fuente:** Elaborado por Byron Valenzuela

La conexión del módulo GPS2 Click con la IOIO se muestra en la siguiente figura.



**Figura 94.** Conexión del módulo GPS con la IOIO

**Fuente:** Elaborado por Byron Valenzuela

- **Análisis de resultados**

Con el propósito de comprobar el funcionamiento del GPS y del localizador se realizó 3 medidas en lugares distintos dentro de la ciudad de Ibarra obteniendo los siguientes resultados.

**Tabla 39.**Tabla de resultados de mediciones realizadas con el GPS

<b>Lugar</b>	<b>Medición realizada con GPS particular</b>	<b>Resultado obtenido por la aplicación</b>
Río Yasuní 1-62 y Río Tahuando	Posicion Indicada	Aproximadamente en la posición indicada a un alrededor de 20m
Av. 17 de Julio	Posicion indicada	Aproximadamente en la posición indicada a un alrededor de 20m
Laboratorio Ciercom Universidad Técnica del Norte	Posicion Indicada	Aproximadamente en la posición indicada a un alrededor de 20m

**Fuente:** Elaborado por Byron Valenzuela

- **Conclusiones**

- Este proyecto es de mucha utilidad para conocer la localización y ubicación remota de un equipo, al establecer conexión inalámbrica la IOIO con el equipo móvil.
- Para poder graficar el mapa en el dispositivo móvil es necesario hacer uso de los mapas de google.
- Las tramas NMEA tienen gran cantidad de información que reciben de los satélites y que pueden ser de mucha utilidad como: velocidad, latitud, longitud, altitud, y hora.

- **Recomendaciones**

- Profundizar más en el tema de aplicaciones GPS y realizar una aplicación que permita observar el recorrido realizado por el GPS.
- Necesariamente conectar el amplificador operacional de otra forma la IOIO es incapaz de establecer comunicación con el módulo GPS2 Click.
- Esperar varios minutos hasta que la información recibida por el GPS se estabilice y se pueda visualizar en el display del móvil. Para comenzar a trabajar con módulo GPS2 Click aplicar un pulso Low/High/Low en el pin On así lo recomienda el datasheet respectivo.

- **Bibliografía**

mikroElektronika. (2012). *mikroBUS pinout Standar Especification*.

A. Carretero, F. F. (2009). *Electrónica*. Editex.

Agüero, R. (s.f.). *Redes Inalámbricas de Área Local y Personal WLAN El estándar IEEE 802.11*. Cantabria.

Argüello, D. J. (2012). *Desarrollo de una aplicación que permita la captura almacenamiento, reproducción, administración y envío de archivos de vide, audio e imagenes utilizando la tecnología bluetooth, para dispositivos móviles basados en el sistema operativo Android*. Quito.

Carballar, J. (2010). *WI-FI lo que necesita conocer*. Madrid: RC Libros.

Carrillo Pérez, M. d. (2006). *Estudio y análisis de rendimiento Bluetooth*. Madrid.

Castillo, D. (2012). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ALARMA COMUNITARIA A BASE DE MÓDULOS INALÁMBRICOS UTILIZANDO TECNOLOGÍA ZIGBEE*. Ibarra.

Clanar Internacional. (s.f.). *Internet y redes inalámbricas*. Arequipa: Clanar.

Collaguazo, G. (2009). *Sistemas Basados en Microprocesadores*. Ibarra.

Correia, P. (2002). *Guía práctica del GPS*. Barcelona: Marcombo.

Creative Commons . (s.f.). *Arduino*. Obtenido de <http://arduino.cc/en/Main/CopyrightNotice>

Daniel Benchimol. (2011). *Microcontroladores*. Buenos Aires: DALAGA S.A.

Digi International, Inc. (2009). *XBee®/XBee-PRO® RF Modules*. New York.

Dignani, J. (2011). *Análisis del protocolo ZigBee*.

- Dignani, J. P. (2011). *Análisis del protocolo ZigBee*.
- Fairchild Semiconductor. (1999). *MM74C922 • MM74C923*.
- FAIRCHILD. (2000). *DM74LS47 BCD to 7-Segment Decoder/Driver with Open-Collector Outputs*. FAIRCHILD SEMICONDUCTOR.
- Fernández, A. M. (2004). *EL BUS I2C*. Córdoba.
- GARCÍA CELI, H. M., & SANTILLÁN LARA, L. A. (2005). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE TARIFACIÓN PARA LOCUTORIOS*. Quito.
- GitHub. (2008). *Analog Input*. Obtenido de <https://github.com/ytai/ioio/wiki/Analog-Input>
- GitHub. (2008). *Digital IO*. Obtenido de <https://github.com/ytai/ioio/wiki/Digital-IO>
- GitHub. (2008). *Getting To Know The Board*. Obtenido de <https://github.com/ytai/ioio/wiki/Getting-To-Know-The-Board>
- GitHub. (2008). *IOIO Over Bluetooth*. Obtenido de <https://github.com/ytai/ioio/wiki/IOIO-Over-Bluetooth>
- GitHub. (2008). *IOIOLibBasics*. Obtenido de <https://github.com/ytai/ioio/wiki/IOIOLib-Basics>
- GitHub. (2008). *Power Supply*. Obtenido de <https://github.com/ytai/ioio/wiki/Power-Supply>
- GitHub. (2008). *Power Supply*. Obtenido de <https://github.com/ytai/ioio/wiki/Power-Supply>
- GitHub. (2008). *Pwm Output*. Obtenido de <https://github.com/ytai/ioio/wiki/PWM-Output>
- GitHub. (2008). *SPI*. Obtenido de <https://github.com/ytai/ioio/wiki/SPI>
- GitHub. (2008). *TWI*. Obtenido de <https://github.com/ytai/ioio/wiki/TWI>
- GitHub. (2008). *UART*. Obtenido de <https://github.com/ytai/ioio/wiki/UART>
- Granadino, C., & Suárez, J. (s.f.). *El bus I2C*. Chile: Universidad Técnica Federico Santa María .
- Huerta, E., Manguiaterra, A., & Gustavo, N. (2005). *Posicionamiento satelital*. Argentina: A.U.G.M.
- IOIO for Android*. (s.f.). Obtenido de <https://www.sparkfun.com/products/10748>
- Jara, P., & Nazar, P. (s.f.). *Estándar IEEE 802.11 X de las WLAN*. Buenos Aires: Edutecne.
- Microchip. (2010). *PIC24FJ256DA210 FAMILY*. Microchip.
- Microchip. (2013). *MRF24WB0MA/MRF24WBOMB*.
- Molina, F. (s.f.). *Global positioning system*. España.
- Monk, S. (2012). *Making Android Accessories With IOIO (Vol. I)*. O'Reilly Media.

- Pérez, E. L. (s.f.). *Curso de Redes de Microcontroladores PIC (PROTOCOLO SPI)*. México: Ingeniería en Microcontroladores.
- Pérochon, S. (2012). *Android Guia de desarrollo de aplicaciones para smartphones y tabletas*. Barcelona: Ediciones ENI.
- Quectel. (2011). *L30 Quectel GPS Engine*. Shanghai: Quectel.
- Quectel. (2013). *Quectel L30 Compact GPS Module*.
- Raúl Esteve Bosch, J. F. (2005). *Fundamentos de electrónica digital*. Valencia.
- RobotFreak. (2008). *IOIO-Rover*. Obtenido de <http://letsmakerobots.com/node/33968>
- ROVING NETWORKS. (2011). *RN-XV Data Sheet*. Arizona: ROVING NETWORKS.
- sgoliver.net foro. (s.f.). *Estructura de un proyecto Android*. Obtenido de <http://www.sgoliver.net/blog/?p=1278>
- Texas Instrument. (2000). *LM35*.
- Texas Instrument. (2004). *LM-358 DUAL OPERATIONAL AMPLIFIERS*. Dallas: Texas Instrument.
- Ucontrol. (2008). *Matrices de LEDs. Ucontrol Electrónica General Pic's en particular*, 68.
- Universidad Politécnica de Valencia. (Abril de 2013). *Elementos de un proyecto Android*. Obtenido de <http://www.androidcurso.com/index.php/recursos-didacticos/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/148-elementos-de-un-proyecto-android>
- USERS. (s.f.). *Microcontroladores funcionamiento programación y usos prácticos. USERS*, 70-76.
- Valverde Rebaza, J. C. (2007). *El Estándar Inalámbrico ZigBee*. Trujillo: Perú.
- Valverde, J. (2007). *El Estándar Inalámbrico ZigBee*. Trujillo.
- Xatakandroid. (2005). *¿Qué es Android?* Obtenido de <http://www.xatakandroid.com/sistema-operativo/que-es-android>
- Zambrano, B. J. (2011). *DISEÑO E IMPLEMENTACIÓN DE UN KIT DE APLICACIONES PARAPERSONAS CON DISCAPACIDAD VISUAL UTILIZANDO LA*. Sangolqui.
- zhongzhouOpto. (s.f.). *SPECIFICATION FOR ZHONGZHOU LED LAMP* .

- Anexos

### Como obtener un mapa de google maps para un proyecto Android

1. Crear el proyecto Android como en anteriores ocasiones, tener en cuenta el nombre del proyecto y el paquete que contiene al proyecto. Importar las librerías de google play services, para esto abrir el SDK Manager y verificar la dirección donde se encuentra instalado, además se debe instalar el paquete Google play services.

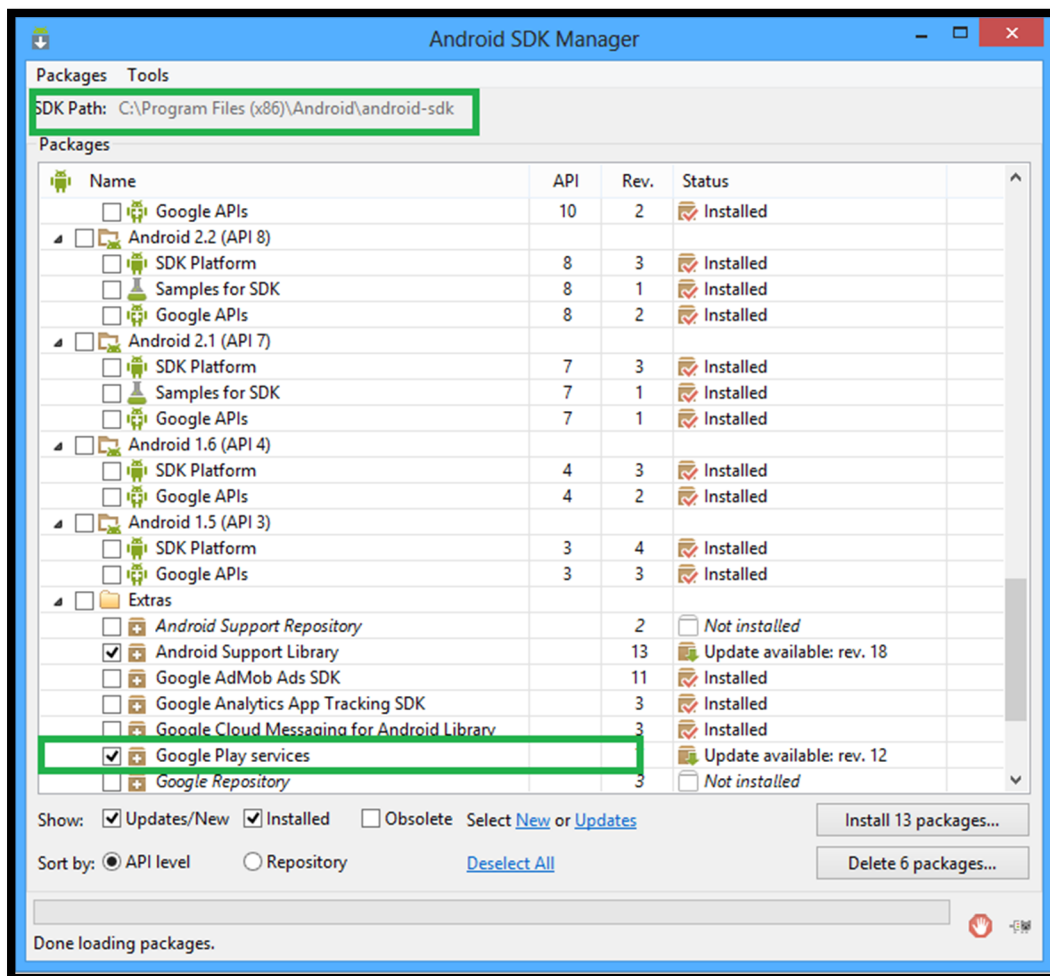
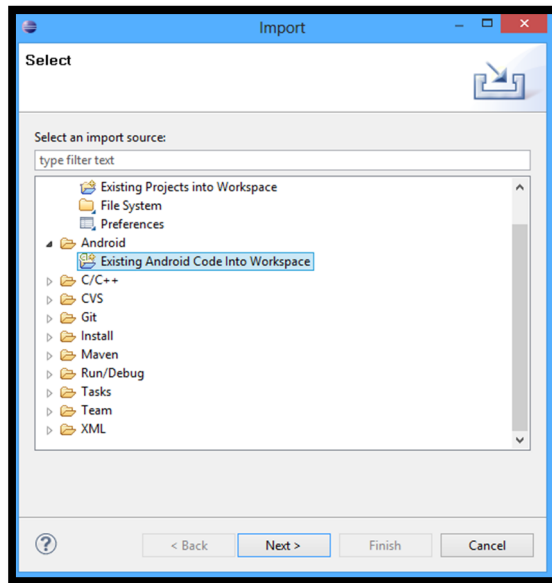


Figura 95. Ventana principal Android SDK Manager

Fuente: Elaborado por Byron Valenzuela



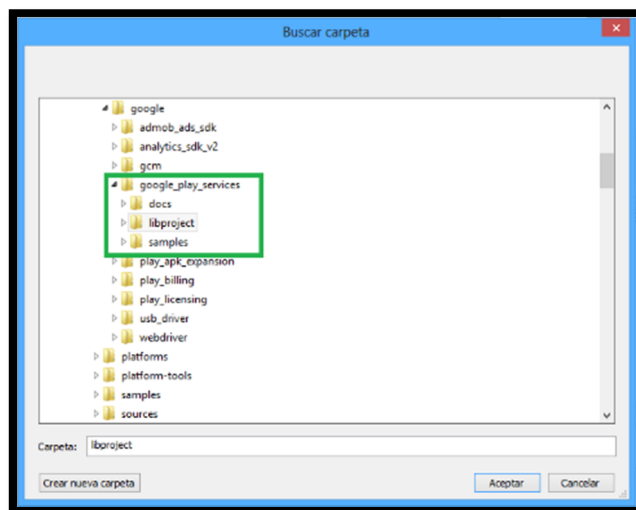
2. Con esta dirección importar las librerías de Google play service. Desde Eclipse ir a File, Import, Existing Android Code Into Workspace



**Figura 96.** Ventana de importación de proyectos en Eclipse

**Fuente:** Elaborado por Byron Valenzuela

3. Ir a la dirección del sdk, anteriormente verificada, carpeta extras, google, libproject y aceptar.



**Figura 97.** Interfaz para buscar la dirección de un proyecto a importar

**Fuente:** Elaborado por Byron Valenzuela

4. Por ultimo importar el proyecto.

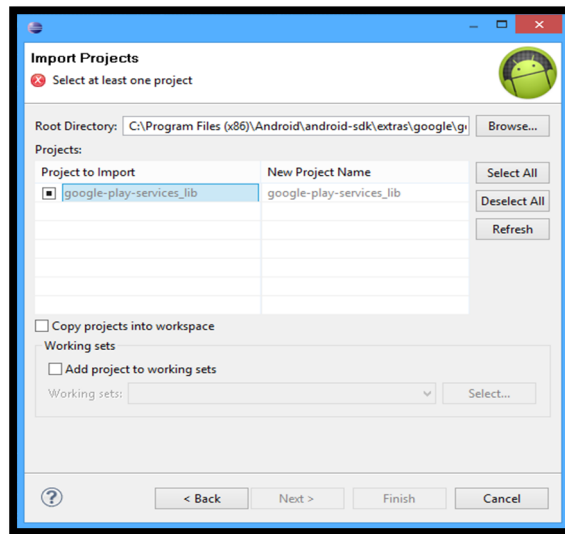


Figura 98. Ventana importar proyectos en Eclipse

Fuente: Elaborado por Byron Valenzuela

5. Añadir las librerías importadas al proyecto. Click derecho en el proyecto, Properties, submenú Android y en la parte inferior click en Add, y escoger google-play-services-lib.

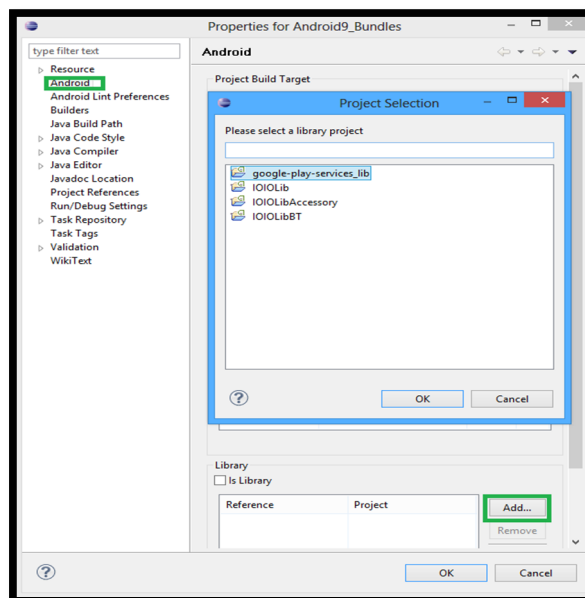
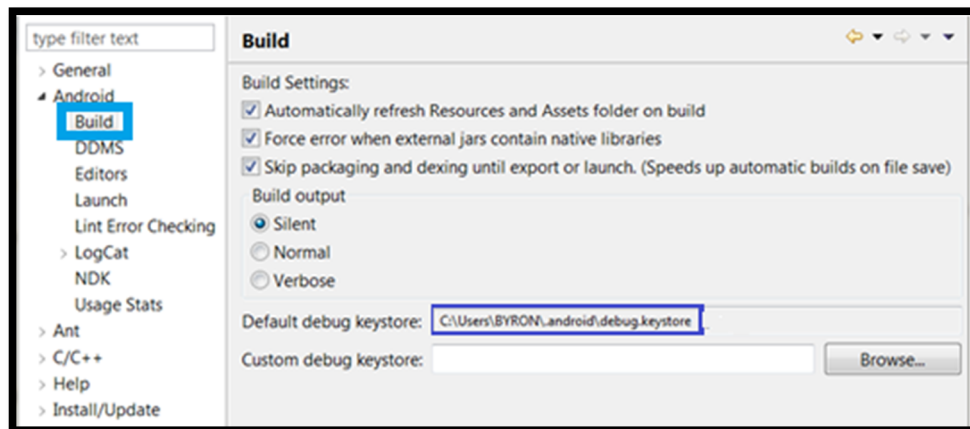


Figura 99. Ventana para añadir librerías al proyecto Android

Fuente: Elaborado por Byron Valenzuela

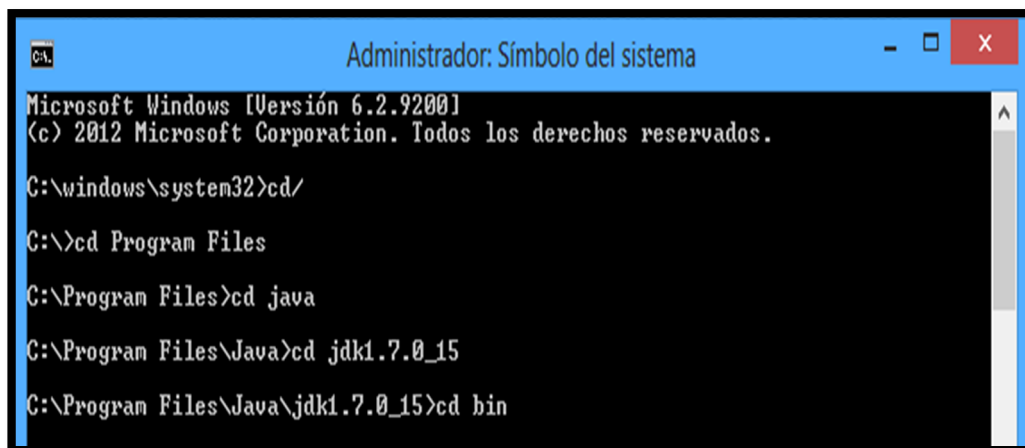
- Como se va utilizar el API de google es necesario la clave SHA1. Primeramente se debe descubrir donde se encuentra el certificado digital de depuración. Para esto dirigirse: Window, Preferences, Android, Build.



**Figura 100.** Ventana preferences de Eclipse

**Fuente:** Elaborado por Byron Valenzuela

- Buscar la dirección donde se encuentra instalado Java. Para este caso es C:\Program Files\Java\jdk1.7.0\_15\bin.
- Ingresar a la dirección de Java a través de la consola de comandos (CMD). Ejecutar como Administrador.

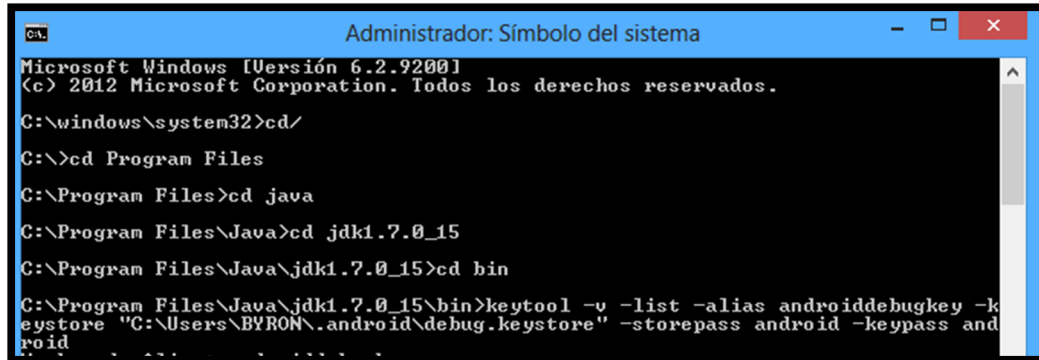


**Figura 101.** Ingreso a la dirección mencionada mediante el CMD

**Fuente:** Elaborado por Byron Valenzuela

- Una vez ubicados en esta dirección ejecutar el siguiente comando: `keytool -v -list -alias androiddebugkey -keystore "C:\Users\BYRON\.android\debug.keystore" -storepass android -keypass android`

Cambiar la parte subrayada por la dirección keystore obtenida en el paso 3.

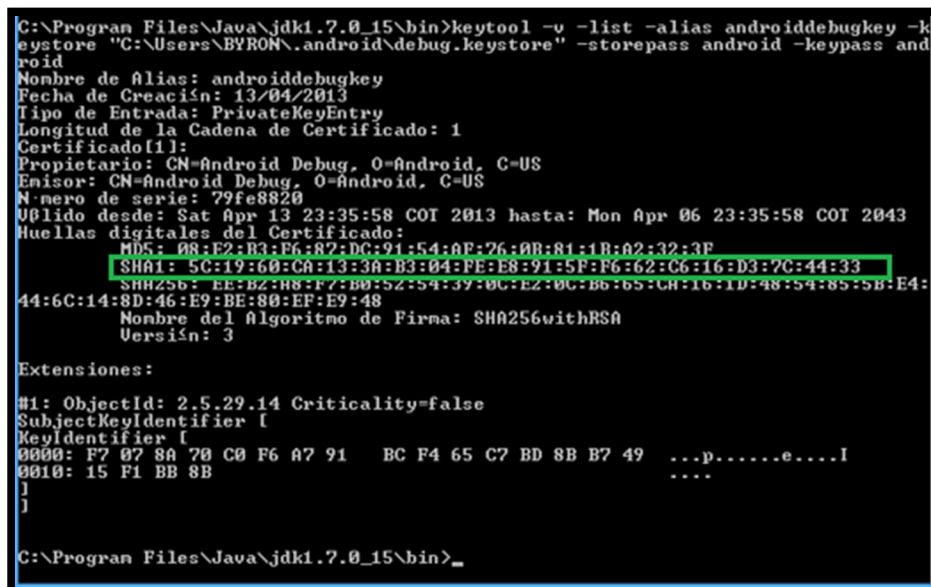


```
Administrador: Símbolo del sistema
Microsoft Windows [Versión 6.2.9200]
(c) 2012 Microsoft Corporation. Todos los derechos reservados.
C:\windows\system32>cd/
C:\>cd Program Files
C:\Program Files>cd java
C:\Program Files\Java>cd jdk1.7.0_15
C:\Program Files\Java\jdk1.7.0_15>cd bin
C:\Program Files\Java\jdk1.7.0_15\bin>keytool -v -list -alias androiddebugkey -keystore "C:\Users\BYRON\.android\debug.keystore" -storepass android -keypass android
```

Figura 102. Ejecución del comando para verificar la clave SHA1

Fuente: Elaborado por Byron Valenzuela

- Al momento de ejecutar el comando aparece en la pantalla varias claves, de las tantas la que interesa es la clave SHA1.



```
C:\Program Files\Java\jdk1.7.0_15\bin>keytool -v -list -alias androiddebugkey -keystore "C:\Users\BYRON\.android\debug.keystore" -storepass android -keypass android
Nombre de Alias: androiddebugkey
Fecha de Creación: 13/04/2013
Tipo de Entrada: PrivateKeyEntry
Longitud de la Cadena de Certificado: 1
Certificado[]:
Propietario: CN=Android Debug, O=Android, C=US
Emisor: CN=Android Debug, O=Android, C=US
Número de serie: 79fe8820
Válido desde: Sat Apr 13 23:35:58 COT 2013 hasta: Mon Apr 06 23:35:58 COT 2014
Huellas digitales del Certificado:
MD5: 08:E2:B3:F6:82:DC:91:54:0F:76:0B:81:1B:02:32:3F
SHA1: 5C:19:60:CA:13:3A:B3:04:FE:E8:91:5F:F6:62:C6:16:D3:7C:44:33
SHA256: EE:B2:88:F7:B0:52:54:39:0C:E2:0C:B6:65:CA:16:1D:48:54:85:5B:E4:
44:6C:14:8D:46:E9:BE:80:EF:E9:48
Nombre del Algoritmo de Firma: SHA256withRSA
Versión: 3

Extensiones:
#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: F7 07 8A 70 C0 F6 A7 91 BC F4 65 C7 BD 8B B7 49 ...p.....e....I
0010: 15 F1 BB 8B .....
]
]
C:\Program Files\Java\jdk1.7.0_15\bin>_
```

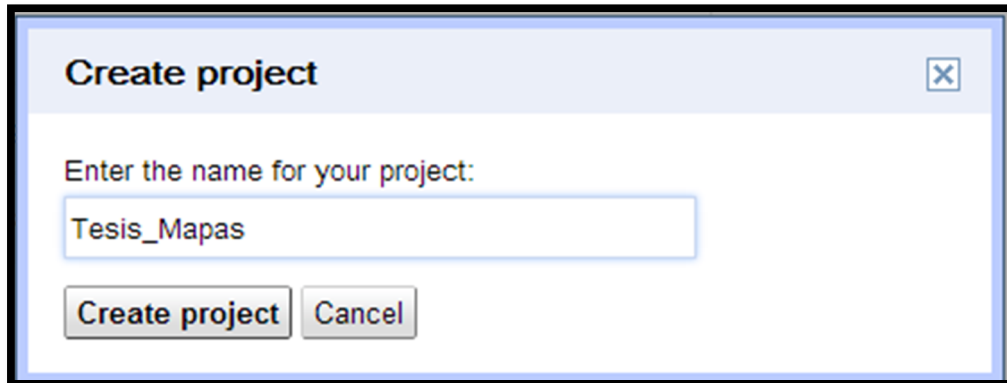
Figura 103. Clave SAH1

Fuente: Elaborado por Byron Valenzuela

Esta clave sirve para hacer uso de los mapas de google.

11. Luego de conseguir la clave SHA1. Ir a la página <https://code.google.com/apis/console/> para obtener el permiso de uso de los mapas. Para acceder es necesario tener una cuenta en gmail.

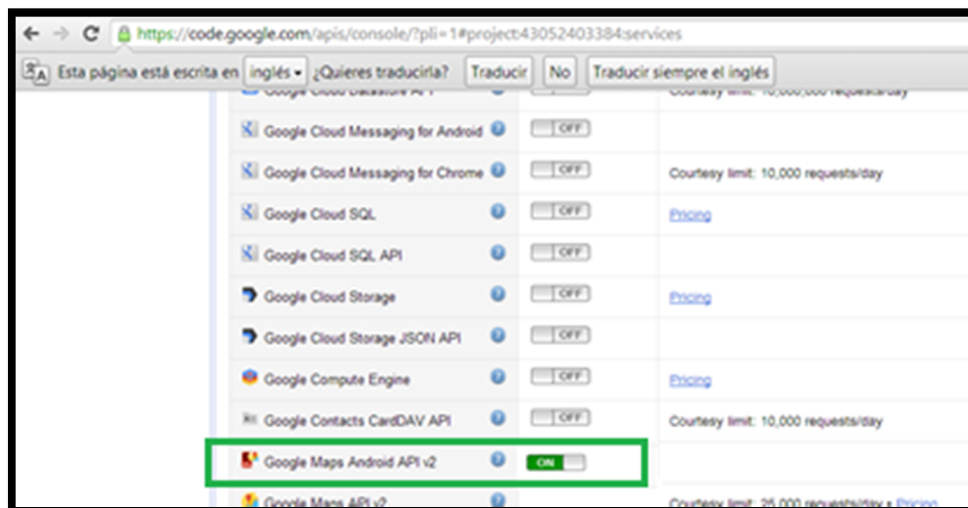
12. En esta página crear un nuevo proyecto. Debe tener el mismo nombre con el que se creó en Eclipse.



**Figura 104.** Pestaña de creación de un nuevo proyecto

**Fuente:** Elaborado por Byron Valenzuela

13. Acceder a la opción services y activar la opción Google Maps Android API V2.



**Figura 105.** Activación del servicio de mapas para Android

**Fuente:** Elaborado por Byron Valenzuela

14. Posteriormente escoger Api Access entre las opciones de ventana.

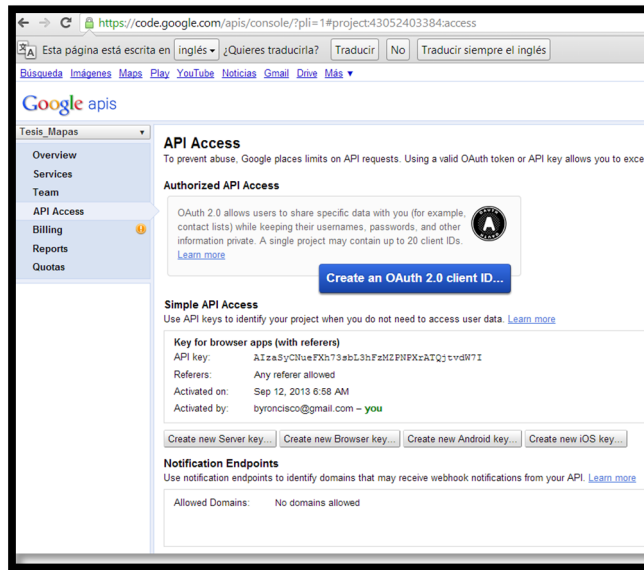


Figura 106. Ventana API Access

Fuente: Elaborado por Byron Valenzuela

15. En esta venta acceder a Create new Android Key y colocar la clave SHA1 anteriormente obetida seguida de ; y el nombre del paquete que contiene el proyecto Ej.

5C:19:60:CA:13:3A:B3:04:FE:E8:91:5F:F6:62:C6:16:D3:7C:44:33;tesis.mapas

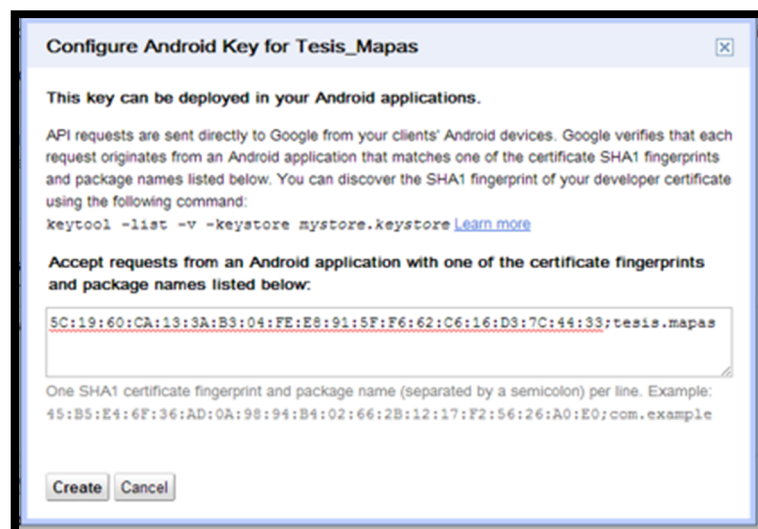


Figura 107. Ventana de configuración para obtener la clave Android para el uso de los mapas

Fuente: Elaborado por Byron Valenzuela

16. Por ultimo tomar nota de la clave generada para el proyecto Android.

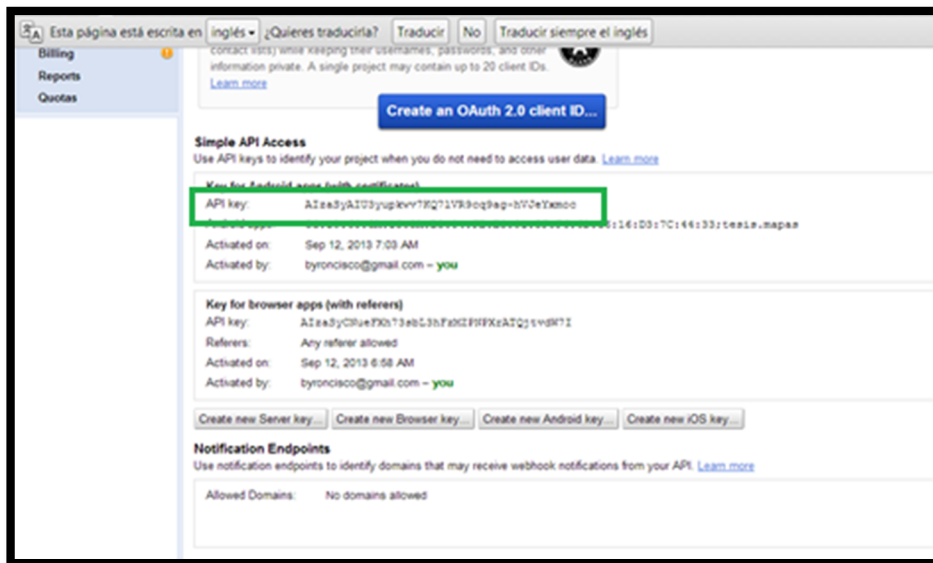


Figura 108. Ventana donde se observa la clave generada para hacer uso de los mapas de google

Fuente: Elaborado por Byron Valenzuela

17. Dentro del AndroidManifest del proyecto dentro del elemento <application> anadir las siguientes líneas.

<meta-data

android:name="com.google.android.maps.v2.API\_KEY"

android:value="AlzaSyCXV7RcLaKvNaOpJuViK1J9kbKHoucJ7Zg"/>

Reemplazar el valor marcado por la clave que se generó anteriormente.

18. Añadir los siguientes permisos.

<permission

android:name="tesis.gps.permission.MAPS\_RECEIVE"

android:protectionLevel="signature"/>

<uses-permission android:name="tesis.gps.permission.MAPS\_RECEIVE"/>

<uses-permission android:name="android.permission.INTERNET"/>

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"/>

<uses-permission

android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>

```
<uses-permission  
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
```

19. A continuación de los permisos añadir el siguiente código.

```
<uses-feature android:glEsVersion="0x00020000"  
    android:required="true"/>
```

20. Reemplazar el código del layout (formulario) por el siguiente.

```
<RelativeLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    tools:context=".MainActivity">  
    <fragment  
        android:id="@+id/map"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        class="com.google.android.gms.maps.SupportMapFragment"/>  
</RelativeLayout>
```

21. La clase que contenga el mapa deberá derivarse de FragmentActivity Ej.

```
public class MainActivity extends FragmentActivity
```

22. Al ejecutar la aplicación se pintara el mapa en la pantalla del móvil Android.



## Anexo 2

### Código de aplicación.

Para esta aplicación se tiene dos Activities GpsActivity que es donde se recoge los datos del GPS conectado a la IOIO y MapaActivity es donde se grafica el punto en el mapa

#### GpsActivity

```
// Ubicacion de un punto en el mapa mediante
// un modulo GPS conectado a la IOIO
package tesis.gps;

import ioio.lib.api.Uart;
import ioio.lib.api.exception.ConnectionLostException;
import ioio.lib.util.AbstractIOIOActivity;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

import tesis.gps.R.id;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class GpsActivity extends AbstractIOIOActivity {

    private final int Tx = 6; // Pines a ser usados para
    private final int Rx = 7; // la comunicacion serial
    // Variable que aloja los componentes del formulario
    private TextView tvValor, tvValor1;
    private Button btnUbicar;
```

```

// Variable que aloja la lectura del GPS, y permite la
// separacion de campos en la trama
String lecturaGps,valor,Mensaje;
// Variable para comprara el identificador de la trama a escoger
String comparacion = "$GPGGA";
// Variable que aloja la latitud, y longitud .
String latitud,latitud1,longitud,longitud1;
// Vector que permite alojar los campos de la trama
String palabras [];
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.gps_activity);
    // atar los componentes al formulario
    tvValor = (TextView)findViewById(id.tvValor);
    tvValor1 = (TextView)findViewById(id.tvValor1);
    btnUbicar = (Button)findViewById(id.btnGraficar);
}

```

```

class IOIOThread extends AbstractIOIOActivity.IOIOThread {

    //Declara la variable Uart para la comunicacion serial de la IOIO
    private Uart uart;
    // Variable Input y OutputStrem para manejo de comunicacion I/O en
    // java
    private InputStream in;
    private OutputStream out;
    public void setup() throws ConnectionLostException {

        try {
            // Inicializa la comunicacion serial de la IOIO
            uart = ioio_.openUart(Rx,Tx, 4800,
                Uart.Parity.NONE, Uart.StopBits.ONE);
            // Acceder a los metodos de lectura y escritura serial
            in = uart.getInputStream();
            out = uart.getOutputStream();
        }
    }
}

```

```

    } catch (ConnectionLostException e) {
        throw e;
    }
}
@SuppressWarnings("deprecation")
public void loop() throws ConnectionLostException {
    DataOutputStream wr = new DataOutputStream(out);
    DataInputStream rd = new DataInputStream (in);

    try {
        // Lee la cadena recibida por el serial de la IOIO
        String s = rd.readLine();
        // Separa la cadena cada vez que encuentre el caracter ,
        palabras = s.split(",");
        // Almacena el identificador de la trama
        valor = palabras [0];
        // El identificador es el mismo que se necesita "$GPGGA"?
        if (valor.equals(comparacion))
        {
            // Separar y almacenar latitud y longitud
            latitud = palabras [2] ;
            longitud = palabras [4];
            // Convierte de String a Flotante
            // los grados, minuto y segundos
            Float gradoslat = Float.parseFloat(latitud.substring(0,2));
            Float miutoslat = Float.parseFloat(latitud.substring(2,4));
            Float seclat = Float.parseFloat(latitud.substring(5,7));
            Float milisec = Float.parseFloat(latitud.substring(7,9));

            // Convierte de String a Flotante
            // los grados, minuto y segundos
            Float gradoslong = Float.parseFloat(longitud.substring(1,3));
            Float miutoslong = Float.parseFloat(longitud.substring(3,5));
            Float seclong = Float.parseFloat(longitud.substring(6,8));
            Float miliseclong = Float.parseFloat(longitud.substring(8,9));
        }
    }
}

```

```

        // Conversion de coordenadas geograficas a grados munitos y
        //segundos
Float resultlat = (gradoslat)+(miutoslat / 60)+ (seclat / 3600)+ (milisec / 216000);
        /// latitud covertida
        latitud1 = latitud1.valueOf(resultlat);
        // Verifica el simbolo de la longitud
        if (palabras [5].equals("E"))
        {
            // Si tiene direccin Este el simbolo es +
Float resultlong = (gradoslong)+(miutoslong / 60)+ (seclong / 3600)+ (miliseclong /
216000);
            longitud1 = longitud1.valueOf(resultlong);
        }
        if (palabras [5].equals("W"))
        {
            // Si tiene direccin Este el simbolo es -
Float resultlong = ((gradoslong)+(miutoslong / 60)+ (seclong / 3600)+ (miliseclong /
216000))*-1;
            longitud1 = longitud1.valueOf(resultlong);
        }

        // Metodo que imprime el valor de la longitud y latitud
        setText(latitud1,longitud1) ;
    }

    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
}

@Override
protected AbstractIOActivity.IOIOThread createIOIOThread() {
    return new IOIOThread();
}
}

```

```

// Imprime el valor de la longitud y latitud en el textView
private void setText(final String valor,final String estado ) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            tvValor.setText(valor);
            tvValor1.setText(estado);
        }
    });
}

// Metodo que Activa el mapa almacena las variables
// de latitud y longitud recibidas por el Gps
public void ubicacion(View v){
    // Obtiene el valor de la latitud y longitud
    String latitud = tvValor.getText().toString();
    String longitud = tvValor1.getText().toString();
    // Activa El mapa
    Intent intentSiguiente=new Intent(this, MapaActivity.class);
    // Almacena la longitud y latitud para ser usados en la siguiente clase
    intentSiguiente.putExtra("platitude", latitud);
    intentSiguiente.putExtra("plongitud", longitud);
    // Muestra el siguiente formulario (Mapa)
    startActivity (intentSiguiente);
}

}

```

## MapaActivity

```

package tesis.gps;

import android.os.Bundle;
import android.support.v4.app.FragmentActivity;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;

```

```
import com.google.android.gms.maps.GoogleMap.OnMapClickListener;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
```

```
public class MapaActivity extends FragmentActivity implements OnMapClickListener {
    // Declara la latitud y longitud de Ibarra
    private static final LatLng ibarra = new LatLng(0.34,-78.12);
    // Variable que aloja el mapa de google
    private GoogleMap mapa;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_mapas_gps);
        // atar los componentes del formulario
        mapa = ((SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.map)).getMap();
        // mueve la posicion del mapa a latitud ibarra
        mapa.moveCamera(CameraUpdateFactory.newLatLng(ibarra));
        mapa.animateCamera(CameraUpdateFactory.zoomTo(15), 200, null);
        mapa.setOnMapClickListener(this);
        // recupera las variables de latitud y longitud
        // de la amnterior clase
        Bundle bundleDatosPersonales = getIntent().getExtras();
        String latitud = bundleDatosPersonales.getString("latitud");
        String longitud = bundleDatosPersonales.getString("plongitud");
        // convierte las variables String a Flotantes
        Float lat = Float.parseFloat(latitud);
        Float longi = Float.parseFloat(longitud);
        // Declara e inicializa la variable tipo latitud, longitud con los
        // valores recibidos por el GPS
        final LatLng ibarra = new LatLng(lat,longi);
        // Imprime un marcador en la posicion recibida
        mapa.addMarker(new MarkerOptions().position(ibarra).title("Posicion nueva"));
    }
    // Toast con la latitud y longitud del punto
    Toast.makeText(this,
```

```
        "Latitud: " + lat + " Longitud: " + longi,  
        2000).show();  
  
    }  
  
    @Override  
    public void onMapClick(LatLng point) {  
        // TODO Auto-generated method stub  
  
    }  
}
```

#### 4.5.4 MÓDULO WI-FI

- **Tema: Sistema de Monitoreo y control de humedad de suelo, a través de un módulo Wi-Fi.**

- **Objetivos**

- Objetivo general

Monitorear y controlar la humedad de suelo, mediante un dispositivo WiFly XV conectado a la IOIO, con el propósito de comprobar el funcionamiento del módulo WI-FI del entrenador electrónico.

- Objetivos específicos

- Estudiar la tecnología de comunicación inalámbrica WI-FI.
- Investigar el funcionamiento del WiFly XV que posee el entrenador electrónico.
- Realizar el flujo grama de la aplicación.
- Escribir el código necesario para controlar esta aplicación.
- Compilar el código y ejecutar la aplicación en el dispositivo móvil Android.
- Implementar la aplicación en el entrenador.
- Verificar el funcionamiento y corregir errores si en caso se presentan.

- **Marco teórico**

##### **Tecnología WI-FI**

Wi-Fi es una tecnología de comunicación sin cables, es decir de manera inalámbrica, esta tecnología permite la interconexión de varios dispositivos de usuario final tales como: impresoras, laptops, celulares, tablets y todos los equipos que contengan un adaptador de red inalámbrico.

En la actualidad es sumamente indispensable contar con un terminal que pueda conectarse a la red, navegar por internet y compartir los recursos de red disponibles; es por tal razón que la mayoría de dispositivos electrónicos de comunicación hoy en día traen consigo un adaptador de red inalámbrico Wi-fi.

Wi-fi pertenece al grupo de redes WLAN (Wireless local Area Network) redes de área local inalámbricas, el estándar que rige a Wi-fi es el estándar IEE 802.11 que especifica las capas MAC y Física para redes WLAN y hace uso del protocolo TCP/IP para la transmisión de información.

IEE 802.11 dispone de varios grupos de trabajo entre los más importantes están los siguientes (Agüero).



- **IEEE 802.11a** (1999) tiene una tasa de transmisión de hasta 54Mbps, opera en la banda de 5Ghz y utiliza OFDM multiplexación por división de frecuencias ortogonales.
- **IEEE 802.11b** (1999) tiene una velocidad de transmisión de 5.5Mbps y 11Mbps, opera en la banda de 2.4Ghz y utiliza DSSS espectro ensanchado por secuencia directa.
- **IEEE 802.11g** (2003) velocidad de transmisión de hasta 54Mbps, opera en la banda de 2.4Ghz, este es compatible con el estándar IEEE 802.11b
- **IEEE 802.11n** (2009) velocidades de transmisión de hasta 600Mbps opera en ambas bandas 2,4Ghz y 5Ghz, usa la tecnología MIMO Multiples entradas-Multiples salidas.

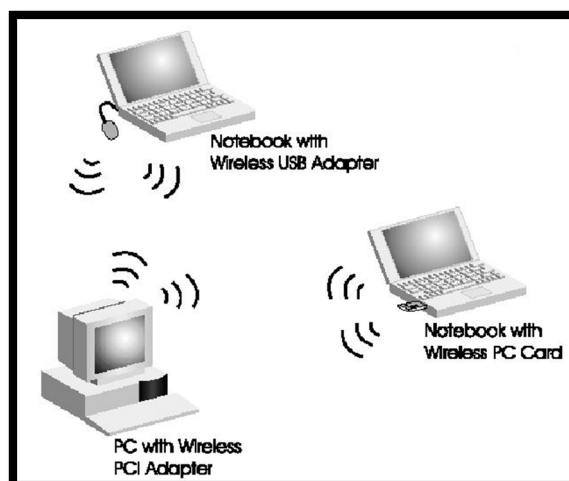
En la actualidad la mayoría de adaptadores de red inalámbricos traen compatibilidad con los estándares 802.11b, g, n.

### Topología de red en 802.11

El estándar 802.11 define el concepto de Conjunto básico de servicio (BSS Basic set service) que es nada mas que uno o varios nodos inalámbricos capaces de establecer comunicación entre ellos. Las BSS pueden intercambiar información de dos modos diferentes (Jara & Nazar).

- **Ad-Hoc (Peer to Peer)**

Se le conoce como IBSS (Independet Basic Service Set), las estaciones se comunican entre sí y no tienen un punto de acceso (AP) en común, generalmente esta topología es temporal ya que se establecen para compartir archivos.



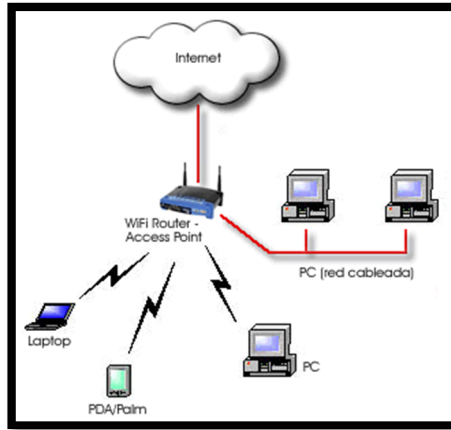
**Figura 109.** Red Ad-Hoc

**Fuente:** Tipos de redes Wi-Fi inalámbricas. Recuperado de

<http://resources.kodak.com/support/shtml/es/manuals/urg00336/urg00336c6s3.shtml>

- **Red de infraestructura**

A esta topología se la conoce como BSS (Basic Service Set), todas las estaciones se conectan a un punto de acceso en común denominado AP que actúa como repetidor y brindará la señal de cobertura para los usuarios que se conecten a la red.



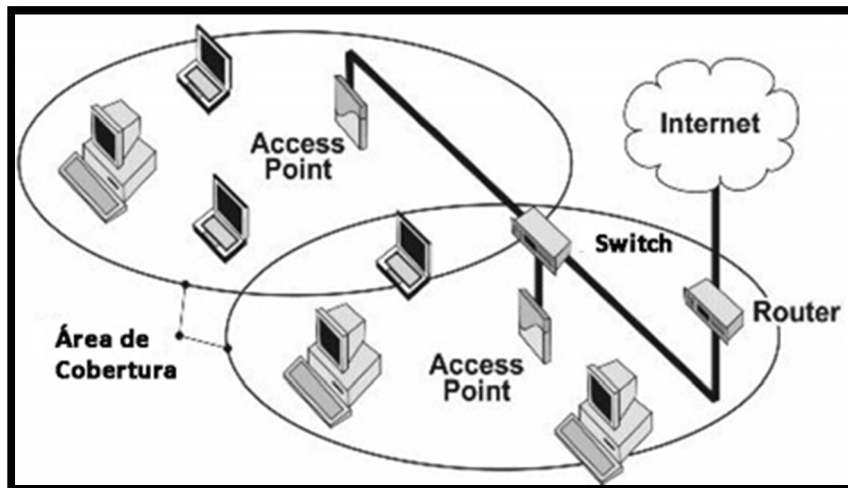
**Figura 110.** Red de infraestructura

**Fuente:** Las redes wi-fi en la palestra. Recuperado de

[http://newtechandeducation.blogspot.com/2010\\_01\\_01\\_archive.html](http://newtechandeducation.blogspot.com/2010_01_01_archive.html)

- **Red extendida**

También conocida como ESS (Extended Service Set), esta topología consiste en sobreponer varios BSS cada uno con su respectivo AP; esto permite que los usuarios tengan la posibilidad de moverse libremente de un AP a otro esta técnica se la conoce como Roaming.



**Figura 111.** Red extendida

**Fuente:** Introducción a las Redes Inalámbricas WiFi Profesionales. Recuperado

<http://foro.tecnicasprofesionales.com/index.php?topic=788.0>

## Seguridad en redes Wi-Fi

El mismo hecho de usar el aire como medio de transmisión hace que la seguridad en este tipo de redes sea vulnerable a posibles intrusos que podrían sacar provecho de la red inalámbrica y poner en riesgo la seguridad informática de la misma.

Existen muchos ataques a los que las redes Wi-Fi son expuestas como la inserción de un usuario no permitido, la negación del servicio a causa de una posible interferencia, duplicidad de IP o Mac de clientes autorizados con el objetivo de revelar los password con los que los AP's se encuentran configurados.

Para contrarrestar estos ataques existen varias técnicas y métodos que permiten asegurar la red inalámbrica al máximo.

- **CNAC (Closed Network Access Control)**

Este mecanismo se basa en el SSID que es un identificador de red de 32 caracteres como máximo el cual identifica a cada red inalámbrica; se debe impedir que el SSID sea visible de esta manera los únicamente los autorizados a conectarse conocerán este nombre y podrán acceder a la red mientras que los usuarios que no conozcan y no sean autorizados no podrán conectarse.

- **ACL (Access Control List)**

Lo que hace este método es permitir el acceso a la red a las Mac de los usuarios que el administrador ha considerado permitir el acceso a la red.

- **WEP (Wired Equivalent Privacy)**

Wep es un protocolo de seguridad incluido en el estandar IEEE 802.11 que permite que la información que cruza por la red sea cifrada mediante el algoritmo RC4 y utiliza una clave de 64 o 128 bits.

- **WPA (Wi-Fi Protected Access)**

Este Sistema de cifrado fue creado por la alianza Wi-Fi, WPA utiliza el algoritmo RC4 con una clave de 128 bits y fue creado con el objetivo de fortalecer las debilidades de WEP. WPA fue diseñado para ser utilizado con un servidor de autenticación (AS) que participa en la distribución de claves a cada usuario además se puede usar el modo de clave compartida PSK para usuarios de hogares y oficinas donde no se tenga un servidor de autenticación.

- **WPA2 (Wi-Fi Protected Access2)**

Sistema basado en el estandar IEEE 802.11i WPA2 define a la clave compartida PSK como clave personal, utiliza un algoritmo de cifrado AES (Advanced Encryption Standard) y se afirma que es la versión certificada de WPA.

### Ventajas y desventajas de las redes Wi-Fi

**Tabla 40.** Ventajas y desventajas de las redes Wi-Fi

Ventajas	Desventajas
Movilidad: Los usuarios pueden moverse libremente dentro del área de cobertura.	Seguridad: Al ser el aire el medio de transmisión, el área de cobertura puede extenderse y así tener varios usuarios que quieran infiltrarse en la red.
Portabilidad: Contar con un dispositivo móvil que pueda conectarse a la red y compartir los recursos es más cómodo para los usuarios.	Interferencias: Es bastante perceptible a interferencias ocasionadas por equipos que trabajen en el mismo rango de frecuencia.
Flexibilidad: No solo los dispositivos móviles pueden conectarse a una red Wi-Fi basta con instalar un adaptador de red inalámbrico a un computador de escritorio para ser parte de esta.	Inversión inicial: Los equipos inalámbricos son más costosos que los cableados.
Ahorro de costos: Los costos son menores frente a una red cableada.	Alcance: El alcance de la red Wi-Fi está definida por la potencia de los equipos y la ganancia de la antena.
Escalabilidad: Es mucho más fácil expandir una red inalámbrica que una red cableada.	Velocidad: Las redes Wi-Fi operan a menor velocidad que las redes cableadas.

**Fuente:** Elaborado por Byron Valenzuela. Basado en (Clanar Internacional).

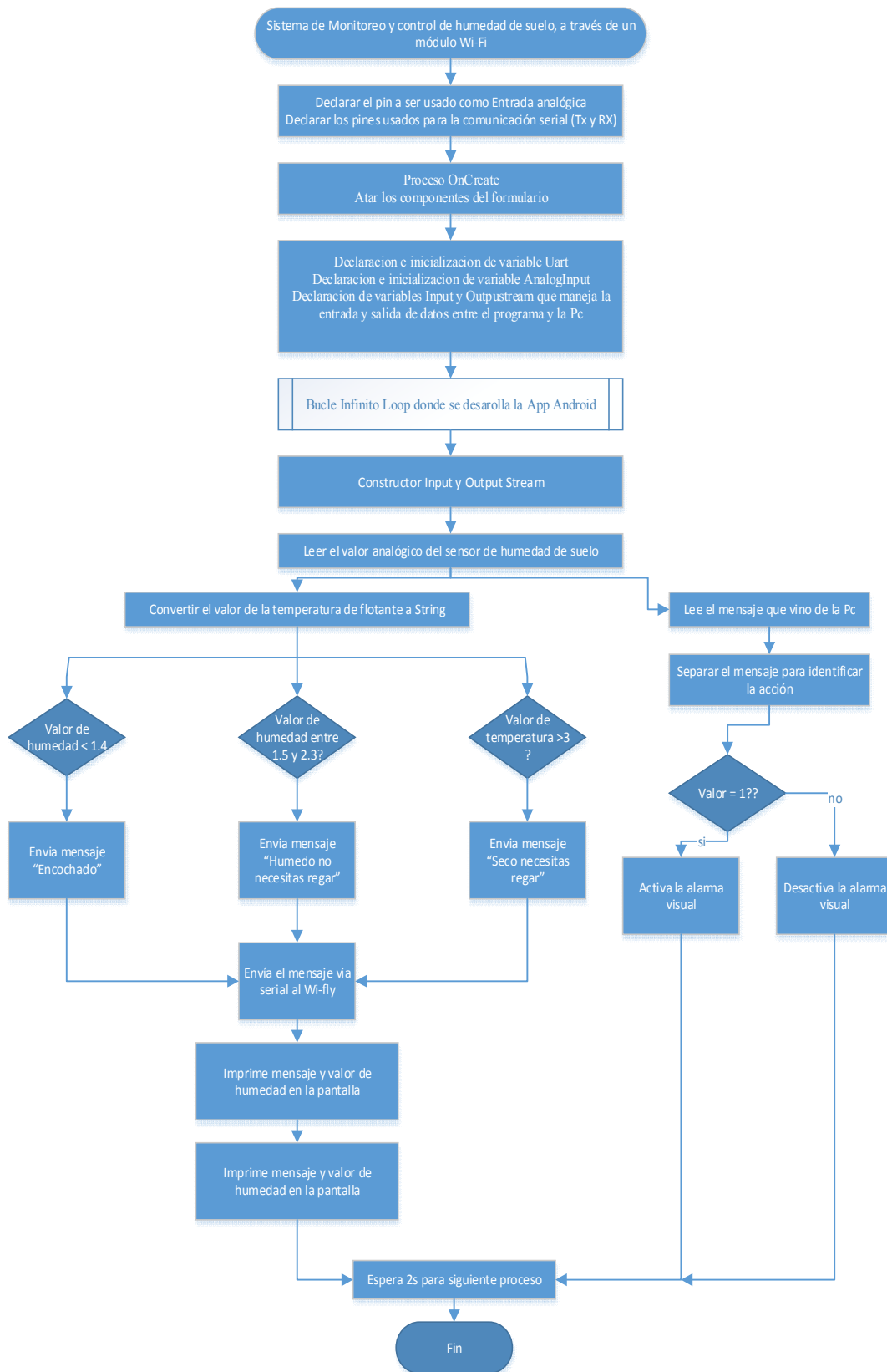
Los requisitos necesarios para esta aplicación son los siguientes.

**Tabla 38.** Requerimientos para la aplicación Wi-Fi

<b>Dispositivo</b>	<b>Requerimientos</b>
<b>Tarjeta IOIO</b>	Módulo de comunicación serial.
<b>Módulo Wi-Fi</b>	Dispositivo que permite la comunicación con la red Wi-fi
<b>Sensor de humedad de suelo</b>	Sensor de humedad de suelo YL-69
<b>Relé</b>	Interfaz para el indicador visual
<b>Equipo móvil Android</b>	Equipo que interactúa con la IOIO
<b>Dispositivo Bluetooth o cable de comunicación usb del equipo Android</b>	Interfaz de comunicación IOIO – Dispositivo móvil Android.

**Fuente:** Elaborado por Byron Valenzuela

• Flujo grama



- **Desarrollo**

**Enunciado: Desarrollar una aplicación que permita monitorear remotamente desde la Pc la humedad de suelo mediante un módulo Wi-Fi; en caso que se necesite regar enviar una alarma visual de una bombilla a 110VCA.**

A continuación se describen los pines y materiales usados en esta aplicación.

**Tabla 41.** Materiales necesarios para la aplicación Wi-Fi

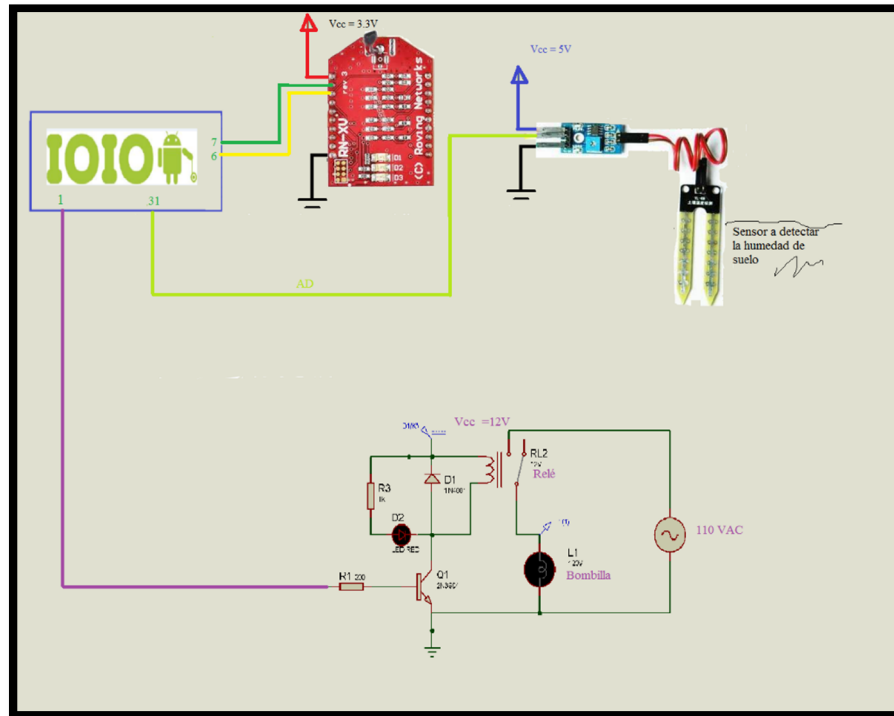
<b>Placa IOIO</b>	Pines de transmisión (Tx) y recepción (Rx) Uart pines 6 y 7 respectivamente, pin digital 1, pin análogo digital pin 31
<b>Entrenador</b>	Relé, Módulo Wi-Fly RN-XV, sensor de humedad de suelo.
<b>Computador</b>	Aplicación C# donde se observa el estado de la humedad de suelo

**Fuente:** Elaborado por Byron Valenzuela.

- Como primer paso para realizar esta aplicación es configurar el dispositivo de red inalámbrica que actuara como AP y a la cual el módulo Wi-Fi se conectará; este paso varia de acuerdo a la marca y modelo del equipo Access Point. Los ítems a configurar son: SSID, clave de la red, canal en la cual trabaja y que actue como servidor DHCP.
- Posteriormente se debe configurar al módulo Wi-Fly RN-XV mediante comunicación serial con comandos y software propios del fabricante, los parámetros a configurar son los anteriormente mencionados SSID, clave, y que el módulo se asocie a la red por medio del servidor DHCP; con estos parámetros configurados el módulo se conectará exitosamente a la red y actuará como un host dentro de la red. Este trabajará con una IP determinada por el DHCP y un puerto determinado.
- Para realizar la aplicación en cualquier lenguaje de programación e IDE y usar el módulo es necesario usar los socket que es la unión de una IP y un puerto. En este caso el lenguaje de programación es C# y el IDE es Visual estudio 2010.
- Conectar a la placa IOIO el módulo Wi-Fly, el sensor de humedad de suelo y el relé que servirá como interfaz para la bombilla a 110V.

- Antes de iniciar la aplicación verificar si el módulo se conecta con éxito a la red; si es así arrancar la aplicación y verificar la información del sensor de humedad de suelo.

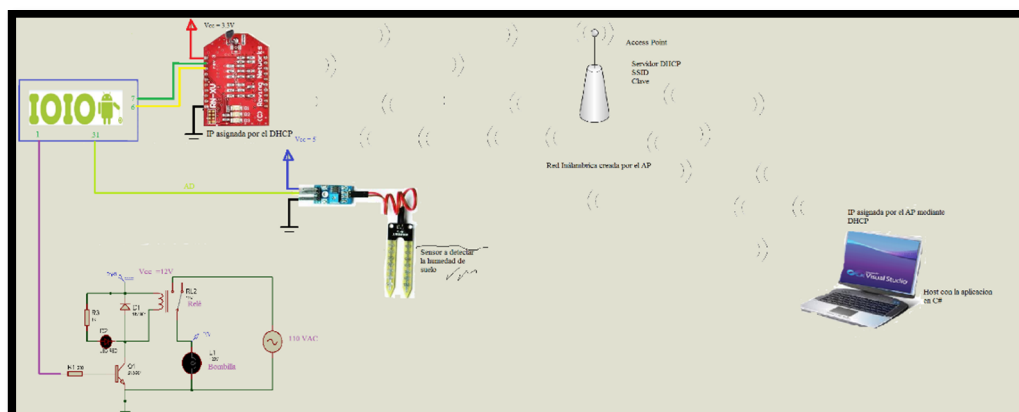
En la siguiente figura se presenta un esquema de conexión de la IOIO que actúa como transmisor y actuador según la humedad del suelo.



**Figura 112.** Esque de conexión IOIO, sensor de humedad de suelo y Wifly

**Fuente:** Elaborado por Byron Valenzuela

Y el esquema general de la aplicación es el siguiente.



**Figura 113.** Esquema general de la aplicación con el módulo Wi-Fly

**Fuente:** Elaborado por Byron Valenzuela



- **Análisis de resultados**

Al ejecutar esta aplicación en tres clases de suelos diferentes: seco, húmedo y encochado; efectivamente se observa que a la interfaz de la aplicación en la PC le recibe los estados reales del suelo estando en la capacidad activar o desactivar la alarma visual.

- **Conclusiones**

- Con esta aplicación se puede tener un monitoreo constante de un jardín en casa y en caso de ser necesario encender una alarma que comunique que el mismo necesita riego.
- Las redes Wi-fi trabajan bajo el protocolo TCP/IP y pertenecen a las redes wlan con un alcance máximo de 100m.
- El alcance de la red depende de la ganancia de la antena que trae consigo el AP.
- La red Wi-Fi depende del AP ya que es este quien crea la red para que los dispositivos de usuario final en este caso la Pc y el Wi-fly puedan conectarse.
- La cantidad de host que se pueden conectar a la red Wi-Fi depende del equipo que este creando la red (AP) y además de la mascara de sub red con la que se este trabajando.

- **Recomendaciones**

- La alimentación del módulo es a 3.3V no exeder en el voltaje porque causaría daños permanentes en el módulo.
- El fabricante recomienda el uso de Tera Term ojo no usar hyperterminal como terminal emulador.
- Antes de proceder a realizar la aplicación en el IDE verificar con un ping si el módulo esta conectado a la red.
- Conocer o configurar de manera correcta los parámetros en el AP que harán que el módulo se conecte con satisfacción a la misma.

- **Bibliografía**

mikroElektronika. (2012). *mikroBUS pinout Standar Especification*.

A. Carretero, F. F. (2009). *Electrónica*. Editex.

Agüero, R. (s.f.). *Redes Inalámbricas de Área Local y Personal WLAN El estándar IEEE 802.11*. Cantabria.

Argüello, D. J. (2012). *Desarrollo de una aplicación que permita la captura almacenamiento, reproducción, administración y envío de archivos de vide, audio e imagenes utilizando la*

*tecnología bluetooth, para dispositivos móviles basados en el sistema operativo Android.*  
Quito.

Carballar, J. (2010). *WI-FI lo que necesita conocer*. Madrid: RC Libros.

Carrillo Pérez, M. d. (2006). *Estudio y análisis de rendimiento Bluetooth*. Madrid.

Castillo, D. (2012). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ALARMA COMUNITARIA A BASE DE MÓDULOS INALÁMBRICOS UTILIZANDO TECNOLOGÍA ZIGBEE*. Ibarra.

Clanar Internacional. (s.f.). *Internet y redes inalámbricas*. Arequipa: Clanar.

Collaguazo, G. (2009). *Sistemas Basados en Microprocesadores*. Ibarra.

Correia, P. (2002). *Guía práctica del GPS*. Barcelona: Marcombo.

Creative Commons . (s.f.). *Arduino*. Obtenido de <http://arduino.cc/en/Main/CopyrightNotice>

Daniel Benchimol. (2011). *Microcontroladores*. Buenos Aires: DALAGA S.A.

Digi International, Inc. (2009). *XBee®/XBee-PRO® RF Modules*. New York.

Dignani, J. (2011). *Análisis del protocolo ZigBee*.

Dignani, J. P. (2011). *Análisis del protocolo ZigBee*.

Fairchil Semiconductor. (1999). *MM74C922 • MM74C923*.

FAIRCHILD. (2000). *DM74LS47 BCD to 7-Segment Decoder/Driver with Open-Collector Outputs*.  
FAIRCHILD SEMICONDUCTOR.

Fernández, A. M. (2004). *EL BUS I2C*. Córdoba.

GARCÍA CELI, H. M., & SANTILLÁN LARA, L. A. (2005). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE TARIFACIÓN PARA LOCUTORIOS*. Quito.

GitHub. (2008). *Analog Input*. Obtenido de <https://github.com/ytai/ioio/wiki/Analog-Input>

GitHub. (2008). *Digital IO*. Obtenido de <https://github.com/ytai/ioio/wiki/Digital-IO>

GitHub. (2008). *Getting To Know The Board*. Obtenido de  
<https://github.com/ytai/ioio/wiki/Getting-To-Know-The-Board>

GitHub. (2008). *IOIO Over Bluetooth*. Obtenido de <https://github.com/ytai/ioio/wiki/IOIO-Over-Bluetooth>

GitHub. (2008). *IOIOLibBasics*. Obtenido de <https://github.com/ytai/ioio/wiki/IOIOLib-Basics>

GitHub. (2008). *Power Supply*. Obtenido de <https://github.com/ytai/ioio/wiki/Power-Supply>

GitHub. (2008). *Power Supply*. Obtenido de <https://github.com/ytai/ioio/wiki/Power-Supply>

GitHub. (2008). *Pwm Output*. Obtenido de <https://github.com/ytai/ioio/wiki/PWM-Output>

GitHub. (2008). *SPI*. Obtenido de <https://github.com/ytai/ioio/wiki/SPI>

GitHub. (2008). *TWI*. Obtenido de <https://github.com/ytai/ioio/wiki/TWI>

GitHub. (2008). *UART*. Obtenido de <https://github.com/ytai/ioio/wiki/UART>

Granadino, C., & Suárez, J. (s.f.). *El bus I2C*. Chile: Universidad Técnica Federico Santa María .

Huerta, E., Manguiaterra, A., & Gustavo, N. (2005). *Posicionamiento satelital*. Argentina: A.U.G.M.

*IOIO for Android*. (s.f.). Obtenido de <https://www.sparkfun.com/products/10748>

Jara, P., & Nazar, P. (s.f.). *Estándar IEEE 802.11 X de las WLAN*. Buenos Aires: Edutecne.

Microchip. (2010). *PIC24FJ256DA210 FAMILY*. Microchip.

Microchip. (2013). *MRF24WB0MA/MRF24WBOMB*.

Molina, F. (s.f.). *Global positioning system*. España.

Monk, S. (2012). *Making Android Accessories With IOIO* (Vol. I). O'Reilly Media.

Pérez, E. L. (s.f.). *Curso de Redes de Microcontroladores PIC (PROTOCOLO SPI)*. México: Ingeniería en Microcontroladores.

Pérochon, S. (2012). *Android Guía de desarrollo de aplicaciones para smartphones y tabletas*. Barcelona: Ediciones ENI.

Quectel. (2011). *L30 Quectel GPS Engine*. Shanghai: Quectel.

Quectel. (2013). *Quectel L30 Compact GPS Module*.

Raúl Esteve Bosch, J. F. (2005). *Fundamentos de electrónica digital*. Valencia.

RobotFreak. (2008). *IOIO-Rover*. Obtenido de <http://letsmakerobots.com/node/33968>

ROVING NETWORKS. (2011). *RN-XV Data Sheet*. Arizona: ROVING NETWORKS.

sgoliver.net foro. (s.f.). *Estructura de un proyecto Android*. Obtenido de <http://www.sgoliver.net/blog/?p=1278>

Texas Instrument. (2000). *LM35*.

Texas Instrument. (2004). *LM-358 DUAL OPERATIONAL AMPLIFIERS*. Dallas: Texas Instrument.

Ucontrol. (2008). *Matrices de LEDs. Ucontrol Electrónica General Pic's en particular*, 68.

Universidad Politécnica de Valencia. (Abril de 2013). *Elementos de un proyecto Android*. Obtenido de <http://www.androidcurso.com/index.php/recursos-didacticos/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/148-elementos-de-un-proyecto-android>

USERS. (s.f.). Microcontroladores funcionamiento programación y usos prácticos. *USERS*, 70-76.

Valverde Rebaza, J. C. (2007). *El Estándar Inalámbrico ZigBee*. Trujillo: Perú.

Valverde, J. (2007). *El Estándar Inalámbrico ZigBee*. Trujillo.

Xatakandroid. (2005). *¿Qué es Android?* Obtenido de <http://www.xatakandroid.com/sistema-operativo/que-es-android>

Zambrano, B. J. (2011). *DISEÑO E IMPLEMENTACIÓN DE UN KIT DE APLICACIONES PARAPERSONAS CON DISCAPACIDAD VISUAL UTILIZANDO LA*. Sangolqui.

zhongzhouOpto. (s.f.). *SPECIFICATION FOR ZHONGZHOU LED LAMP* .

## Anexos

Código de la aplicación en Eclipse.

```
package tesis.wifi;

import ioio.lib.api.AnalogInput;
import ioio.lib.api.DigitalOutput;
import ioio.lib.api.Uart;
import ioio.lib.api.exception.ConnectionLostException;
import ioio.lib.util.AbstractIOActivity;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

import tesis.wifi.R.id;
import android.os.Bundle;
import android.widget.TextView;

public class WiFiActivity extends AbstractIOActivity {

    // Pin analogico que medira la humedad de suelo
    private final int PinHumeda = 31;
```

```

private final int pin_accion =1;
// Pines usados para la comunicacion serial
private final int Tx = 6;
private final int Rx = 7;
// TextView donde se imprime el valor de la temperatura
// Variable tipo String que alojara la conversion analogica
String lectura = "hoola";
String lectura1,accion,estado;
private TextView tvHumedad,TvValor;
int valorint =0;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.wifi_activity);
    tvHumedad = (TextView)findViewById(R.id.tvHumedad);
    TvValor=(TextView)findViewById(id.tvValor);
}

class IOIOThread extends AbstractIOActivity.IOIOThread
{
    // Variable AnalogInput que realizara la lectura analogica de la
    // humedad de suelo
    private AnalogInput temp;
    // Variable tipo uart para la comunicacion serial con la IOIO
    private DigitalOutput pin;
    // pin que controla la bombilla a 110V
    private Uart uart;
    // Variable Input y OutputStrem para manejo de comunicacion I/O en
    // java
    private InputStream in;
    private OutputStream out;

    public void setup() throws ConnectionLostException
    {
        // inicializar la conversion analoga digital
        temp = ioio_.openAnalogInput(PinHumeda);
    }
}

```

```

// inicializar la comunicacion serial de la IOIO
uart = ioio_.openUart(Rx,Tx, 9600,
                    Uart.Parity.NONE, Uart.StopBits.ONE);
// Acceder a los metodos de lectura y escritura serial
pin=ioio_.openDigitalOutput(pin_accion, false);
// inicializacion del pin que controla la bombilla
in = uart.getInputStream();
out = uart.getOutputStream();
}
public void loop() throws ConnectionLostException, InterruptedException {

// Constuctor del DataInput y OutpuStream
DataOutputStream wr = new DataOutputStream(out);
DataInputStream rd = new DataInputStream (in);

final float readTemp = 0;
sleep(50);
final float readTemp = temp.getVoltage();
sleep(20);
lectura = ((Float.toString((readTemp))));

if (readTemp <= 1.4)
{
    lectura1 = "Encochado";
}

if ((readTemp > 1.4) && (readTemp <= 2.3))
{
    lectura1 = "Humedo no Necesita Regar";
}

if (readTemp > 3.0)
{
    lectura1 = "Seco necesitas regar";
}
}

```

```

try {
    wr.writeBytes(lectura1);
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
setText(lectura1,lectura);
sleep (2000);
//lectura de datos

try {
    // lee la informacion tipo String que llega via serial
    accion = rd.readLine();
    // separa la cadena en palabras separadas por ;
    // y las aloja en el vector palabras
    String[] palabras = accion.split(";");
    // almacena el valor de la variable que se encuentra en
    // el vector
    estado = palabras [1];
    // convierte la distancia en entero
    valorint = Integer.parseInt(estado);
    // detecta si la distancia es menor a 5
    if(valorint == 0)
    {
        //enciende la bombilla
        pin.write(true); //enciende el buzzer

    }else
    {
        //Apaga la bombilla
        pin.write(false); //enciende el buzzer
    }
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

```

```

        ///
    }
}
@Override
protected AbstractIOActivity.IOIOThread createIOIOThread() {
    return new IOIOThread();
}

// metodo que imprime el valor de la lectura el el textview
private void setText(final String strTemp,final String strTemp1) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            //imprime los 4 primeros digitos de la lectrua
            tvHumedad.setText(strTemp);
            TvValor.setText(strTemp1);

        }
    });
}
}
}

```

Código de la aplicación en Visio 2010

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.Threading;

```



```

namespace WifiApp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
        }

        private void btnConectar_Click(object sender, EventArgs e)
        {
            sock.RemotePort = 2000;
            sock.RemoteHost = "192.168.1.3";
            sock.Connect();
            btnConectar.Enabled = false;
        }

        private void sock_DataArrival(object sender,
AxMSWinsockLib.DMSWinsockControlEvents_DataArrivalEvent e)
        {
            String data = "";
            Object dat = (object)data;
            String nuevostring = "";
            Byte caracter;
            sock.GetData(ref dat);
            data = (string)dat;
            tbMensaje.Text = data;
            richTextBox1.Text = data;
        }
        private void btnEnviar_Click(object sender, EventArgs e)

```

```
{
    String Datod = "ALERTA;0";
    sock.SendData(Datod);
}
private void tbMensaje_TextChanged(object sender, EventArgs e)
{
}
}
}
```

## **CAPÍTULO V**

### **5 CONCLUSIONES Y RECOMENDACIONES**

#### **5.1 CONCLUSIONES**

Al culminar este proyecto de titulación el estudiante de la carrera de Ingeniería en Electrónica y Redes de Comunicación podrá familiarizarse más con el sistema operativo Android y realizar varias prácticas electrónicas con los diferentes dispositivos con los que el entrenador cuenta.

Este trabajo ayudará a muchos estudiantes que les gusta el campo de la electrónica y el mundo Android a desarrollar aplicaciones primeramente básicas, conocer los parámetros de operación de la placa y el desarrollo de software Android, para luego manejar módulos adicionales y con esto realizar proyectos que permitan el avance tecnológico de la Universidad y del país.

Las prácticas electrónicas están desarrolladas según el sílabo de la materia de microprocesadores de la carrera de Ingeniería en Electrónica y Redes de Comunicación de la Universidad Técnica del Norte, que siguen el proceso de desarrollo de aplicaciones elementales como manejo de puertos de entrada y salida, manejo de conversores hasta lo más complejo que es el manejo de dispositivos electrónicos con comunicaciones serial e I2C.

Android es uno de los sistemas operativos más usados en los dispositivos móviles en la actualidad, al ser de código abierto es libre esto quiere decir que cualquier persona puede desarrollar aplicaciones para su equipo móvil.

Android cuenta con el soporte oficial y herramientas necesarias para las personas que se interesan en desarrollar las aplicaciones, esto quiere decir que brinda paso a paso capacitación, ejemplos y tutoriales para adentrarse en el apasionante mundo del desarrollo de aplicaciones.

La placa de desarrollo electrónico IOIO cuenta con un firmware incluido que facilita el uso y la comunicación ya sea por cable o vía Bluetooth, además cuenta con librerías para el uso de los diferentes módulos que la junta posee como son: módulo de entrada y salida digital, comunicación serial, PWM, I2C, y conversores análogo digitales.

Al finalizar este proyecto se logró comunicar la IOIO con varios módulos adicionales, con el propósito de que el estudiante tenga la capacidad de interactuar Android con los distintos dispositivos electrónicos como Xbee, Wi-Fi, Bluetooth, GPS y pueda controlarlos desde su equipo móvil.

El entrenador electrónico Android puede ser usado para realizar prácticas con la tarjeta IOIO o con cualquier otro microcontrolador gracias al protoboard que trae incorporado, solo basta tener precaución de los voltajes y corrientes con los que se va a trabajar y a conectar cada módulo.

El lenguaje de programación oficial para desarrollar aplicaciones Android es Java un lenguaje libre orientado a objetos y el IDE es Eclipse de igual forma un entorno de desarrollo gratuito que trabaja bajo la plataforma de java.

Al finalizar las prácticas con la IOIO vemos que es obligatorio incluir los permisos de uso Bluetooth e Internet dentro de la aplicación a desarrollarse, además se debe adjuntar las librerías IOIO para poder acceder a todos los métodos y librerías que controlan los módulos electrónicos de la misma.

La implementación del entrenador electrónico para dispositivos Android permitió afianzar los conocimientos adquiridos en las aulas de clase debido a que el sistema abarca conceptos tanto de redes como electrónica.

Todas las aplicaciones realizadas en esta obra son prácticas que el estudiante de Ingeniería en Electrónica y Redes de Comunicación está en la capacidad de realizarlas con un previo conocimiento adquirido en las aulas.

## **1.1. RECOMENDACIONES**

Antes de realizar las prácticas es muy importante conocer la estructura física e interna que la IOIO posee para no causar daños parciales o totales dentro de la placa, al circuito externo, módulos adicionales con los que se este trabajando e incluso al dispositivo móvil al que esta conectado.

De igual manera previo al uso de los distintos módulos adicionales el estudiante debe estar familiarizado con el dispositivo, conocer los parámetros de funcionamiento tanto eléctricos como de configuración con el objetivo de impedir que los módulos sufran averías.

Luego de haber desarrollado la aplicación Android IOIO, probar el funcionamiento de la aplicación conectando la placa al móvil vía cable Usb y mediante Bluetooth y así comprobar el funcionamiento a través de las dos conexiones disponibles.

Al momento de realizar las prácticas con el módulo Gps, es muy conveniente comprobar el funcionamiento de la aplicación en un ambiente libre de obstáculos para así evitar interferencias al momento de recibir los datos satelitales de longitud y latitud.

En el presente trabajo se usó la IOIO como dispositivo conectado a un terminal Android, la versión IOIO OTG también es capaz de trabajar como un host conectado a un computador, se recomienda continuar la investigación y realizar prácticas con la IOIO conectado al ordenador.

Adicionalmente a las prácticas presentadas, el estudiante una vez que haya dominado el tema puede incursionar en el desarrollo de aplicaciones electrónicas con Android; si la aplicación es necesaria e innovadora subirla al App Store y venderla obteniendo así una remuneración por el trabajo realizado.

## REFERENCIAS BIBLIOGRÁFICAS

- mikroElektronika. (2012). *mikroBUS pinout Standar Especification*.
- A. Carretero, F. F. (2009). *Electrónica*. Editex.
- Agüero, R. (s.f.). *Redes Inalámbricas de Área Local y Personal WLAN El estándar IEEE 802.11*. Cantabria.
- Argüello, D. J. (2012). *Desarrollo de una aplicación que permita la captura almacenamiento, reproducción, administración y envío de archivos de vide, audio e imagenes utilizando la tecnología bluetooth, para dispositivos móviles basados en el sistema operativo Android*. Quito.
- Carballar, J. (2010). *WI-FI lo que necesita conocer*. Madrid: RC Libros.
- Carrillo Pérez, M. d. (2006). *Estudio y análisis de rendimiento Bluetooth*. Madrid.
- Castillo, D. (2012). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ALARMA COMUNITARIA A BASE DE MÓDULOS INALÁMBRICOS UTILIZANDO TECNOLOGÍA ZIGBEE*. Ibarra.
- Clanar Internacional. (s.f.). *Internet y redes inalámbricas*. Arequipa: Clanar.
- Collaguazo, G. (2009). *Sistemas Basados en Microprocesadores*. Ibarra.
- Correia, P. (2002). *Guía práctica del GPS*. Barcelona: Marcombo.
- Creative Commons . (s.f.). *Arduino*. Obtenido de <http://arduino.cc/en/Main/CopyrightNotice>
- Daniel Benchimol. (2011). *Microcontroladores*. Buenos Aires: DALAGA S.A.
- Digi International, Inc. (2009). *XBee®/XBee-PRO® RF Modules*. New York.
- Dignani, J. (2011). *Análisis del protocolo ZigBee*.
- Dignani, J. P. (2011). *Análisis del protocolo ZigBee*.
- Fairchil Semiconductor. (1999). *MM74C922 • MM74C923*.
- FAIRCHILD. (2000). *DM74LS47 BCD to 7-Segment Decoder/Driver with Open-Collector Outputs*. FAIRCHILD SEMICONDUCTOR.
- Fernández, A. M. (2004). *EL BUS I2C*. Córdoba.
- GARCÍA CELI, H. M., & SANTILLÁN LARA, L. A. (2005). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE TARIFACIÓN PARA LOCUTORIOS*. Quito.
- GitHub. (2008). *Analog Input*. Obtenido de <https://github.com/ytai/ioio/wiki/Analog-Input>
- GitHub. (2008). *Digital IO*. Obtenido de <https://github.com/ytai/ioio/wiki/Digital-IO>

- GitHub. (2008). *Getting To Know The Board*. Obtenido de <https://github.com/ytai/ioio/wiki/Getting-To-Know-The-Board>
- GitHub. (2008). *IOIO Over Bluetooth*. Obtenido de <https://github.com/ytai/ioio/wiki/IOIO-Over-Bluetooth>
- GitHub. (2008). *IOIOLibBasics*. Obtenido de <https://github.com/ytai/ioio/wiki/IOIOLib-Basics>
- GitHub. (2008). *Power Supply*. Obtenido de <https://github.com/ytai/ioio/wiki/Power-Supply>
- GitHub. (2008). *Power Supply*. Obtenido de <https://github.com/ytai/ioio/wiki/Power-Supply>
- GitHub. (2008). *Pwm Output*. Obtenido de <https://github.com/ytai/ioio/wiki/PWM-Output>
- GitHub. (2008). *SPI*. Obtenido de <https://github.com/ytai/ioio/wiki/SPI>
- GitHub. (2008). *TWI*. Obtenido de <https://github.com/ytai/ioio/wiki/TWI>
- GitHub. (2008). *UART*. Obtenido de <https://github.com/ytai/ioio/wiki/UART>
- Granadino, C., & Suárez, J. (s.f.). *El bus I2C*. Chile: Universidad Técnica Federico Santa María .
- Huerta, E., Manguiaterra, A., & Gustavo, N. (2005). *Posicionamiento satelital*. Argentina: A.U.G.M.
- IOIO for Android*. (s.f.). Obtenido de <https://www.sparkfun.com/products/10748>
- Jara, P., & Nazar, P. (s.f.). *Estándar IEEE 802.11 X de las WLAN*. Buenos Aires: Edutecne.
- Microchip. (2010). *PIC24FJ256DA210 FAMILY*. Microchip.
- Microchip. (2013). *MRF24WB0MA/MRF24WBOMB*.
- Molina, F. (s.f.). *Global positioning system*. España.
- Monk, S. (2012). *Making Android Accessories With IOIO* (Vol. I). O'Reilly Media.
- Pérez, E. L. (s.f.). Curso de Redes de Microcontroladores PIC (PROTOCOLO SPI). México: Ingeniería en Microcontroladores.
- Pérochon, S. (2012). *Android Guia de desarrollo de aplicaciones para smartphones y tabletas*. Barcelona: Ediciones ENI.
- Quectel. (2011). *L30 Quectel GPS Engine*. Shanghai: Quectel.
- Quectel. (2013). *Quectel L30 Compact GPS Module*.
- Raúl Esteve Bosch, J. F. (2005). *Fundamentos de electrónica digital*. Valencia.
- RobotFreak. (2008). *IOIO-Rover*. Obtenido de <http://letsmakerobots.com/node/33968>
- ROVING NETWORKS. (2011). *RN-XV Data Sheet*. Arizona: ROVING NETWORKS.

- sgoliver.net foro. (s.f.). *Estructura de un proyecto Android*. Obtenido de <http://www.sgoliver.net/blog/?p=1278>
- Texas Instrument. (2000). *LM35*.
- Texas Instrument. (2004). *LM-358 DUAL OPERATIONAL AMPLIFIERS*. Dallas: Texas Instrument.
- Ucontrol. (2008). Matrices de LEDs. *Ucontrol Electrónica General Pic's en particular*, 68.
- Universidad Politécnica de Valencia. (Abril de 2013). *Elementos de un proyecto Android*. Obtenido de <http://www.androidcurso.com/index.php/recursos-didacticos/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/148-elementos-de-un-proyecto-android>
- USERS. (s.f.). Microcontroladores funcionamiento programación y usos prácticos. *USERS*, 70-76.
- Valverde Rebaza, J. C. (2007). *El Estándar Inalámbrico ZigBee*. Trujillo: Perú.
- Valverde, J. (2007). *El Estándar Inalámbrico ZigBee*. Trujillo.
- Xatakandroid. (2005). *¿Qué es Android?* Obtenido de <http://www.xatakandroid.com/sistema-operativo/que-es-android>
- Zambrano, B. J. (2011). *DISEÑO E IMPLEMENTACIÓN DE UN KIT DE APLICACIONES PARAPERSONAS CON DISCAPACIDAD VISUAL UTILIZANDO LA*. Sangolqui.
- zhongzhouOpto. (s.f.). *SPECIFICATION FOR ZHONGZHOU LED LAMP* .



## **ANEXOS**

### **ANEXO A PASOS PARA LA CONEXIÓN MEDIANTE BLUETOOTH**

Para conseguir la conexión se debe realizar los siguientes pasos:

Conectar el dispositivo Bluetooth al conector USB que tiene el adaptador de la placa.

1. Limitar la cantidad de corriente que será suministrado al Bluetooth.
2. En el móvil Android activar la conexión Bluetooth y explorar los dispositivos que actualmente están activos.
3. Dentro de la lista de los dispositivos activos aparece un equipo con el nombre (IOIO: xx:xx) donde x representa los cuatro últimos dígitos de la dirección de la IOIO.
4. Al momento de conectarse al dispositivo se solicita un código Pin de conexión el cual es 4545 para cualquier placa IOIO.

## ANEXO B INSTALACIÓN DEL ENTORNO DE DESARROLLO PARA APLICACIONES ANDROID

### INSTALACIÓN DE JAVA

1. Descargar el paquete que contiene el Java (JDK) del siguiente link:  
<http://www.oracle.com/technetwork/es/java/javase/downloads/index.html>

**Java SE 7u21**  
This release includes important security fixes. Oracle strongly recommends that all Java SE 7 users upgrade to this release.  
[Learn more](#) ▶

**Which Java package do I need?**

- **JDK:** (Java Development Kit). For Java Developers. Includes a complete JRE plus tools for developing, debugging, and monitoring Java applications.
- **Server JRE:** (Server Java Runtime Environment) For deploying Java applications on servers. Includes tools for JVM monitoring and tools commonly required for server applications, but does not include browser integration (the Java plug-in), auto-update, nor an installer. [Learn more](#) ▶
- **JRE:** (Java Runtime Environment). Covers most end-users needs. Contains everything required to run Java applications on your system.

JDK	Server JRE	JRE
<a href="#">DOWNLOAD</a> ↓	<a href="#">DOWNLOAD</a> ↓	<a href="#">DOWNLOAD</a> ↓
<b>JDK 7 Docs</b> <ul style="list-style-type: none"><li>• Installation Instructions</li><li>• ReadMe</li><li>• Release Notes</li><li>• Oracle License</li><li>• Java SE Products</li><li>• Third Party Licenses</li><li>• Certified System Configurations</li></ul>	<b>Server JRE 7 Docs</b> <ul style="list-style-type: none"><li>• Installation Instructions</li><li>• ReadMe</li><li>• Release Notes</li><li>• Oracle License</li><li>• Java SE Products</li><li>• Third Party Licenses</li><li>• Certified System Configurations</li></ul>	<b>JRE 7 Docs</b> <ul style="list-style-type: none"><li>• Installation Instructions</li><li>• ReadMe</li><li>• Release Notes</li><li>• Oracle License</li><li>• Java SE Products</li><li>• Third Party Licenses</li><li>• Certified System Configurations</li></ul>

Figura 1. Página web de descarga del JDK

2. Seleccionar el sistema operativo en el que se va instalar el JDK.

Producto / Descripción del archivo	Tamaño del archivo	Descargar
Linux ARM v6v7 Soft Float ABI	65.09 MB	<a href="#">jdk-7u21-linux-arm-sfp.tar.gz</a>
Linux x86	80.35 MB	<a href="#">jdk-7u21-linux-i586.rpm</a>
Linux x86	93.06 MB	<a href="#">jdk-7u21-linux-i586.tar.gz</a>
Linux x64	81.43 MB	<a href="#">jdk-7u21-linux-x64.rpm</a>
Linux x64	91.81 MB	<a href="#">jdk-7u21-linux-x64.tar.gz</a>
Mac OS X x64	144.18 MB	<a href="#">jdk-7u21-macosx-x64.dmg</a>
Solaris x86 (SVR4 paquete)	135.84 MB	<a href="#">jdk-7u21-solaris-i586.tar.Z</a>
Solaris x86	92.08 MB	<a href="#">jdk-7u21-solaris-i586.tar.gz</a>
Solaris x64 (SVR4 paquete)	22.67 MB	<a href="#">jdk-7u21-solaris-x64.tar.Z</a>
Solaris x64	15.02 MB	<a href="#">jdk-7u21-solaris-x64.tar.gz</a>
Solaris SPARC (SVR4 paquete)	136.09 MB	<a href="#">jdk-7u21-solaris-sparc.tar.Z</a>
Solaris SPARC	95.44 MB	<a href="#">jdk-7u21-solaris-sparc.tar.gz</a>
Solaris SPARC de 64 bits (SVR4 paquete)	22.97 MB	<a href="#">jdk-7u21-solaris-sparcv9.tar.Z</a>
Solaris SPARC de 64 bits	17.58 MB	<a href="#">jdk-7u21-solaris-sparcv9.tar.gz</a>
Windows x86	88.98 MB	<a href="#">jdk-7u21-windows-i586.exe</a>
Windows x64	90.57 MB	<a href="#">jdk-7u21-windows-x64.exe</a>

**Java SE Development Kit 7u21 Demos y muestras Descargas**

Figura 2. Selección del instalador según el S.O

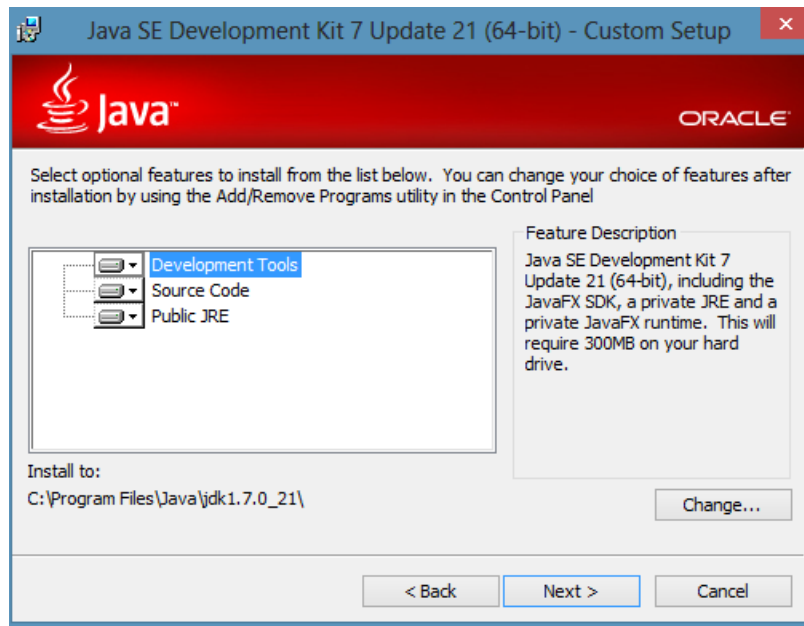
3. Se descargará el instalador JDK .exe.

4. Iniciar la instalación del JDK.



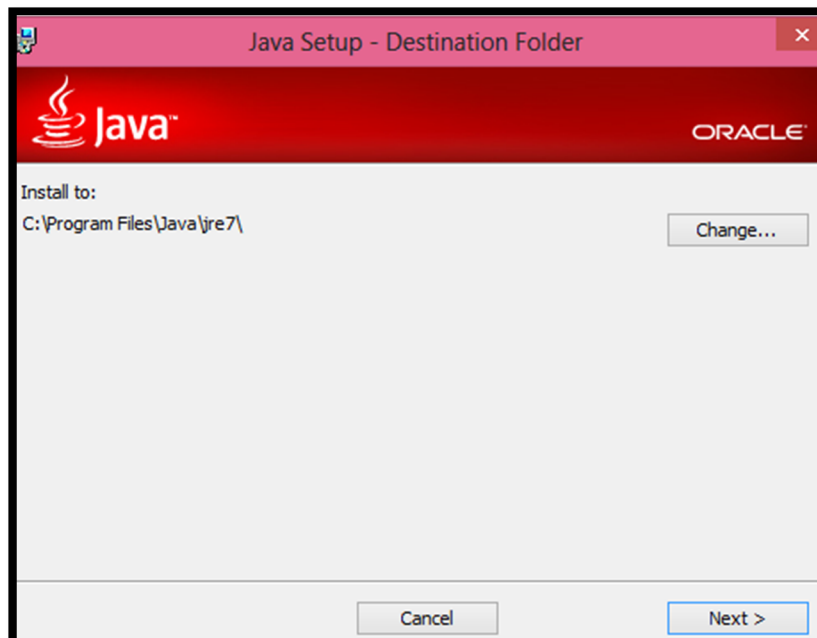
Figura 3. Inicio de la instalación

5. Escoger las características que tendrá el JDK.



**Figura 4.** Ventana de selección de características

6. Seleccionar el directorio a instalar JDK.



**Figura 5.** Ubicación de la carpeta de instalación

## 7. Finalizar la instalación.



Figura 6. Ventana de finalización de la instalación

## INSTALACIÓN DE ECLIPSE IDE

1. Descargar Eclipse IDE for Java Developers del siguiente link, escoger el sistema operativo donde se va instalar, en este caso Windows de 64 bits.

<http://www.eclipse.org/downloads/>.

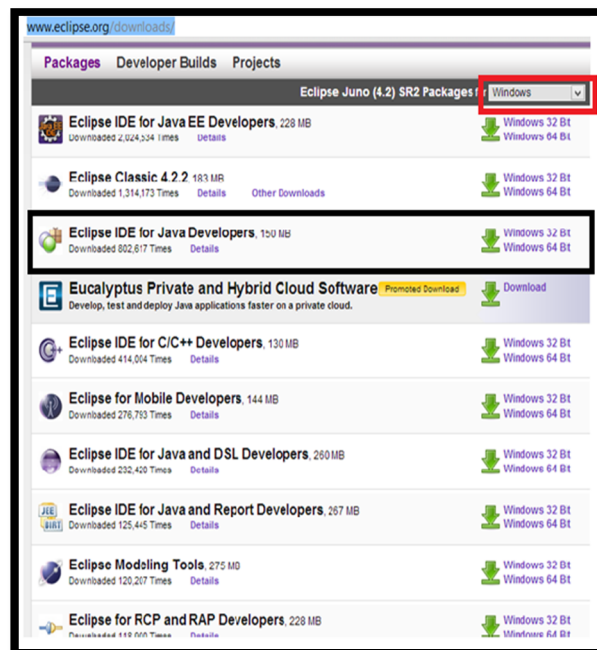
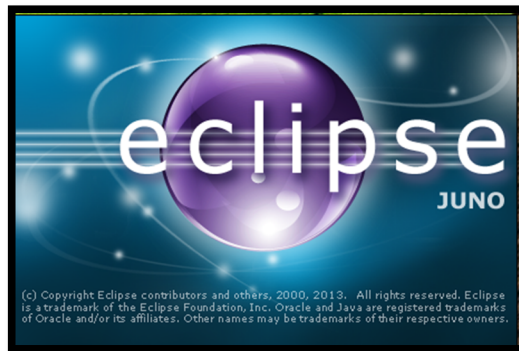


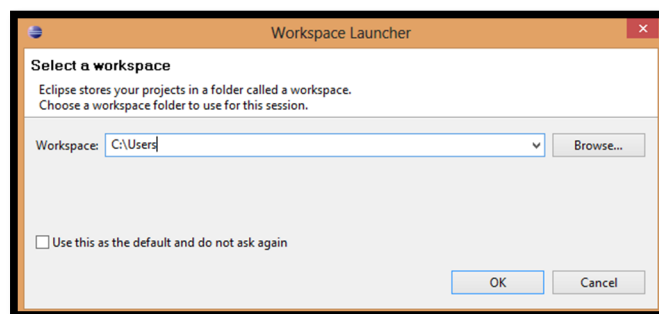
Figura 7. Página web de descarga Eclipse

2. Posteriormente se descargará un paquete .zip.
3. Descomprimir el paquete descargado en la dirección que se desea instalar el Eclipse.  
Por ejemplo: C:\Program Files (x86).
4. Abrir la carpeta descomprimida y ejecutar la aplicación Eclipse.exe



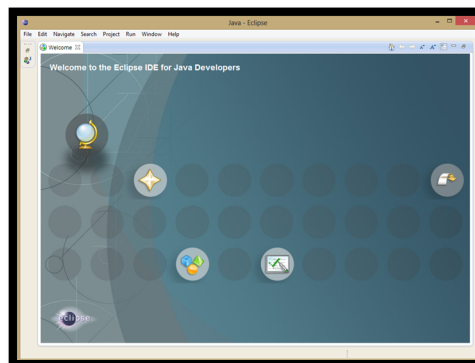
**Figura 8.** Interfaz inicial de Eclipse

5. Escoger el directorio de trabajo.



**Figura 9.** Ubicación del directorio de trabajo

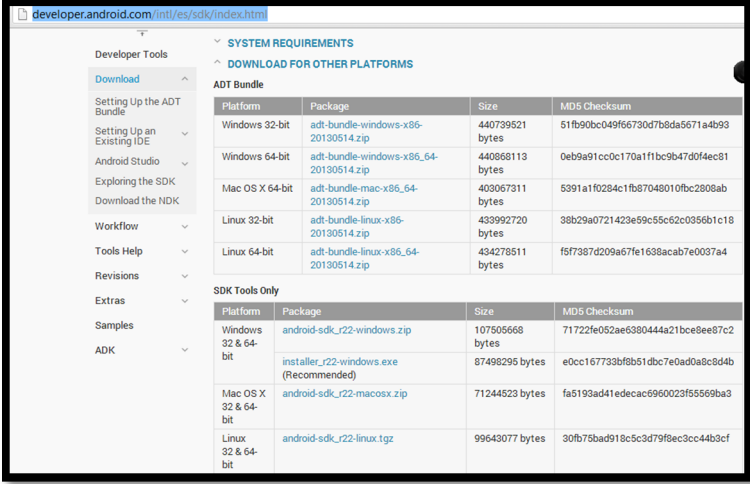
6. Finalmente, aparecerá la ventana con la interfaz del Eclipse IDE.



**Figura 10.** Interfaz de Eclipse

## INSTALAR EL SDK DE ANDROID

1. Descargar el SDK android del siguiente link, escoger el sistema operativo a instalar el SDK. En el caso de Windows se puede elegir el archivo .zip o el ejecutable (.exe). <http://developer.android.com/intl/es/sdk/index.html>

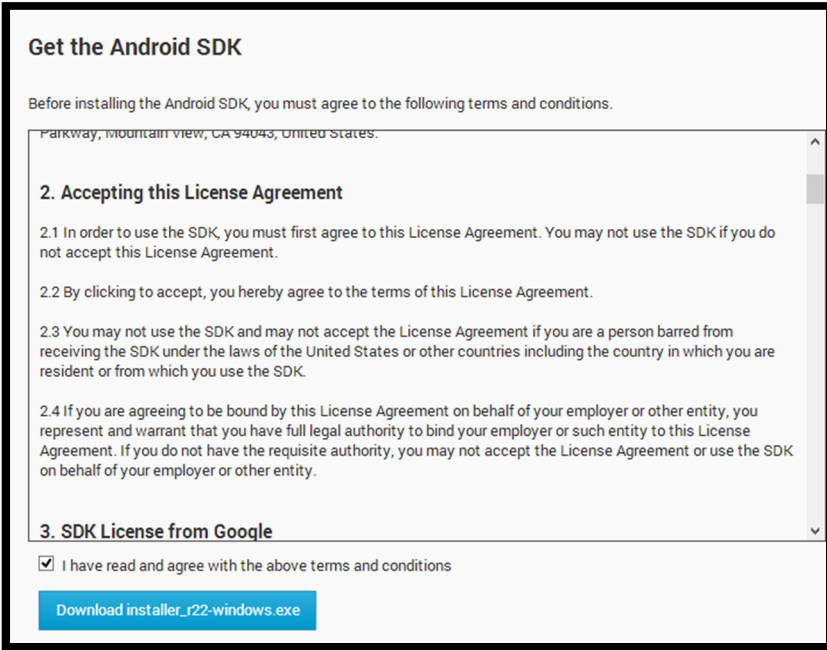


The screenshot shows the 'Download' section of the Android SDK page. It lists 'ADT Bundle' and 'SDK Tools Only' for various platforms. The 'ADT Bundle' section includes links for Windows 32-bit, Windows 64-bit, Mac OS X 64-bit, Linux 32-bit, and Linux 64-bit. The 'SDK Tools Only' section includes links for Windows 32 & 64-bit, Mac OS X 32 & 64-bit, and Linux 32 & 64-bit. Each link is accompanied by the package name, size, and MD5 checksum.

Platform	Package	Size	MD5 Checksum
Windows 32-bit	adt-bundle-windows-x86-20130514.zip	440739521 bytes	51fb90bc049f66730d7b8da5671a4b93
Windows 64-bit	adt-bundle-windows-x86_64-20130514.zip	440868113 bytes	0eb9a91cc0c170a1f1bc9b47d0f4ec81
Mac OS X 64-bit	adt-bundle-mac-x86_64-20130514.zip	403067311 bytes	5391a1f0284c1fb87048010fbc2808ab
Linux 32-bit	adt-bundle-linux-x86-20130514.zip	433992720 bytes	38b29a0721423e59c55c62c0356b1c18
Linux 64-bit	adt-bundle-linux-x86_64-20130514.zip	434278511 bytes	ff7387d209a57fe1638acab7e0037a4
SDK Tools Only			
Platform	Package	Size	MD5 Checksum
Windows 32 & 64-bit	android-sdk_r22-windows.zip	107505668 bytes	71722fe052ae6380444a21bce8ee87c2
	installer_r22-windows.exe (Recommended)	87498295 bytes	e0cc167733f8b51dbc7e0ad0a8c8d4b
Mac OS X 32 & 64-bit	android-sdk_r22-macosx.zip	71244523 bytes	fa5193ad41edecac6960023f55569ba3
Linux 32 & 64-bit	android-sdk_r22-linux.tgz	99643077 bytes	30fb75bad918c5c3d79f8ec3cc44b3cf

Figura 11. Pagina web de descarga SDK

2. Aceptar los términos y condiciones del uso del programa.



The screenshot shows the 'Get the Android SDK' page with the license agreement. It includes a checkbox for 'I have read and agree with the above terms and conditions' and a 'Download installer\_r22-windows.exe' button.

**Get the Android SDK**

Before installing the Android SDK, you must agree to the following terms and conditions.

Parkway, Mountain View, CA 94043, United States.

### 2. Accepting this License Agreement

2.1 In order to use the SDK, you must first agree to this License Agreement. You may not use the SDK if you do not accept this License Agreement.

2.2 By clicking to accept, you hereby agree to the terms of this License Agreement.

2.3 You may not use the SDK and may not accept the License Agreement if you are a person barred from receiving the SDK under the laws of the United States or other countries including the country in which you are resident or from which you use the SDK.

2.4 If you are agreeing to be bound by this License Agreement on behalf of your employer or other entity, you represent and warrant that you have full legal authority to bind your employer or such entity to this License Agreement. If you do not have the requisite authority, you may not accept the License Agreement or use the SDK on behalf of your employer or other entity.

### 3. SDK License from Google

I have read and agree with the above terms and conditions

[Download installer\\_r22-windows.exe](#)

Figura 12. Términos y condiciones del SDK Android

3. Ejecutar el instalador descargado.

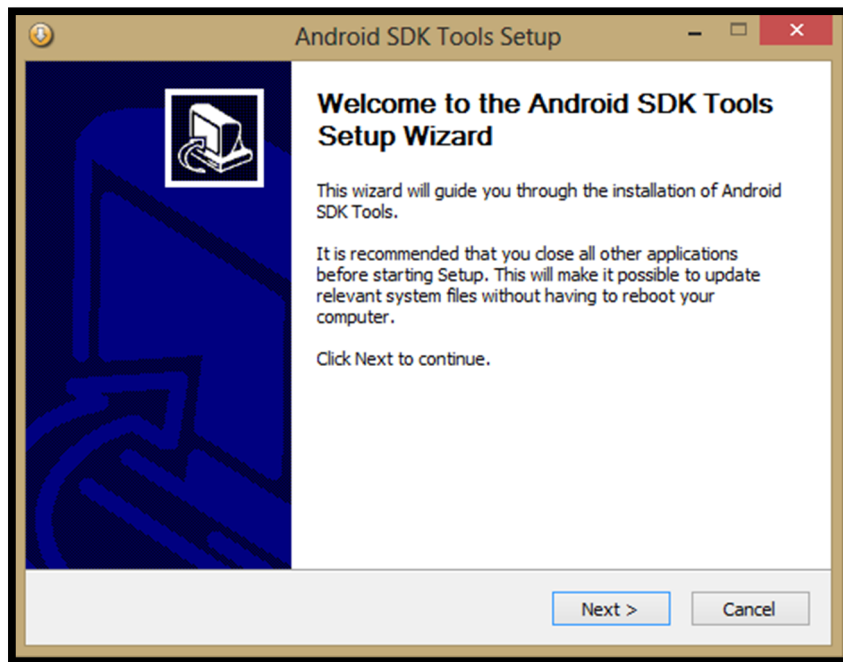


Figura 13. Instalacion del SKD de Android.

4. Seleccionar los usuarios que tendrán acceso al SDK.

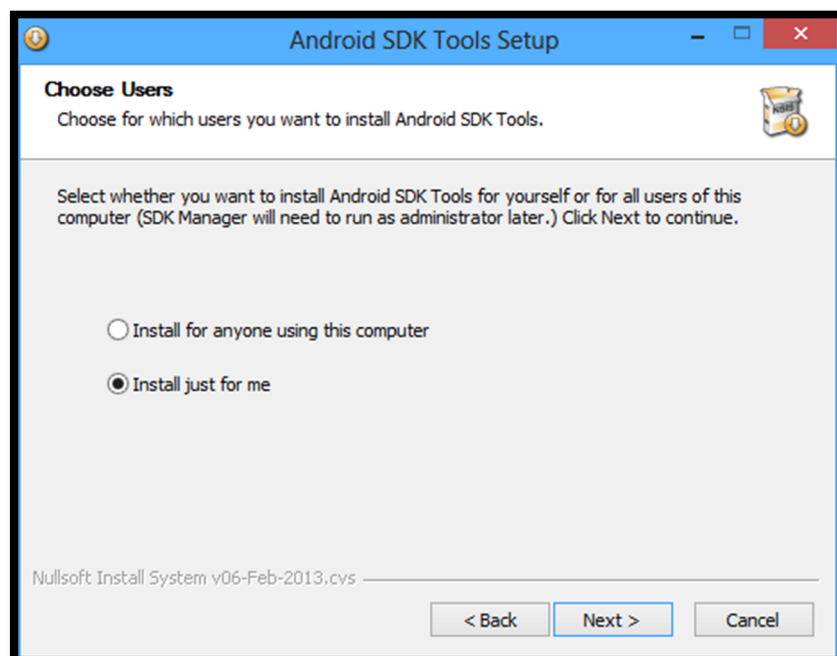


Figura 14. Ventana de selección de los usuarios del SDK



5. Seleccionar la ubicación donde se instalará el SDK.

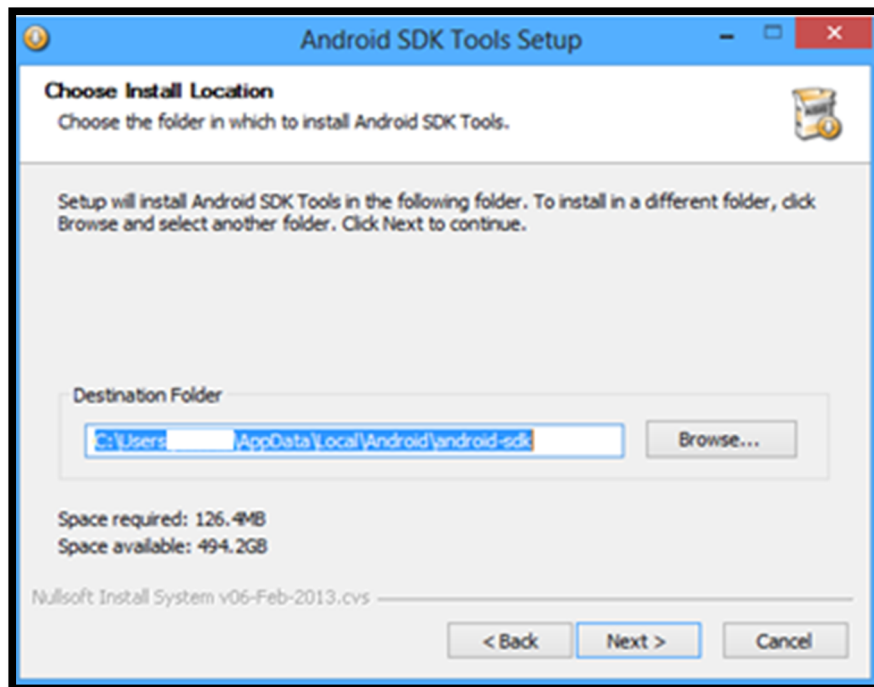


Figura 15. Ubicación de la carpeta de instalación

6. Elegir el destino del acceso directo.

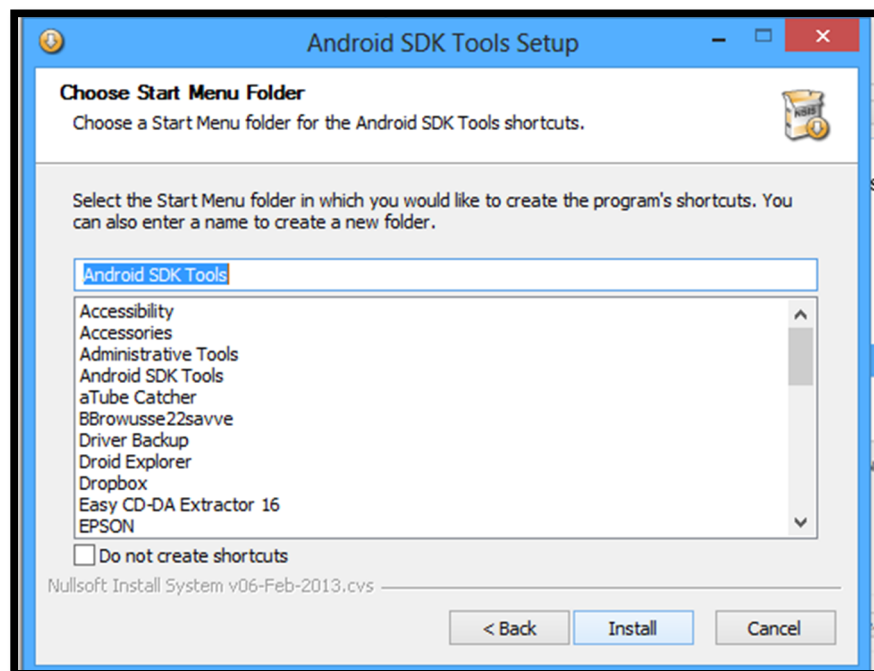


Figura 16. Destino del Acceso directo

## 7. Finalizar la instalación.

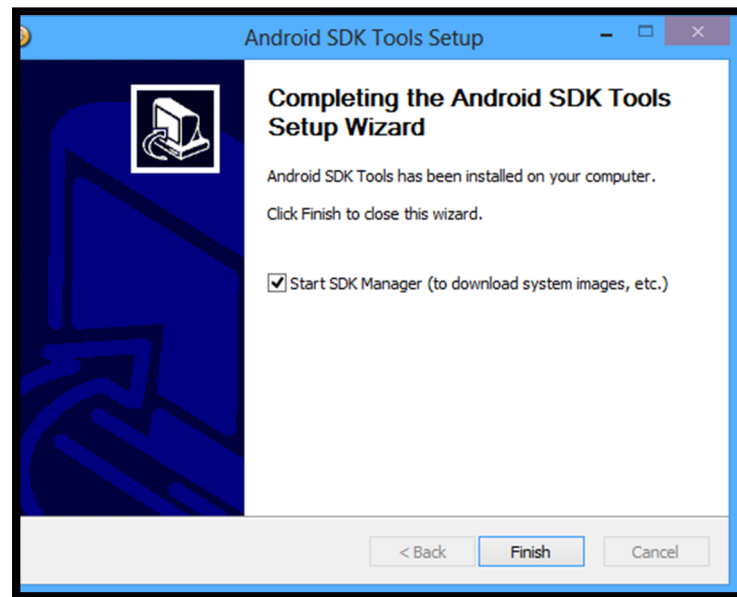


Figura 17. Ventana de finalización de la instalación del SDK

## 8. Ejecutar el SDK manager.

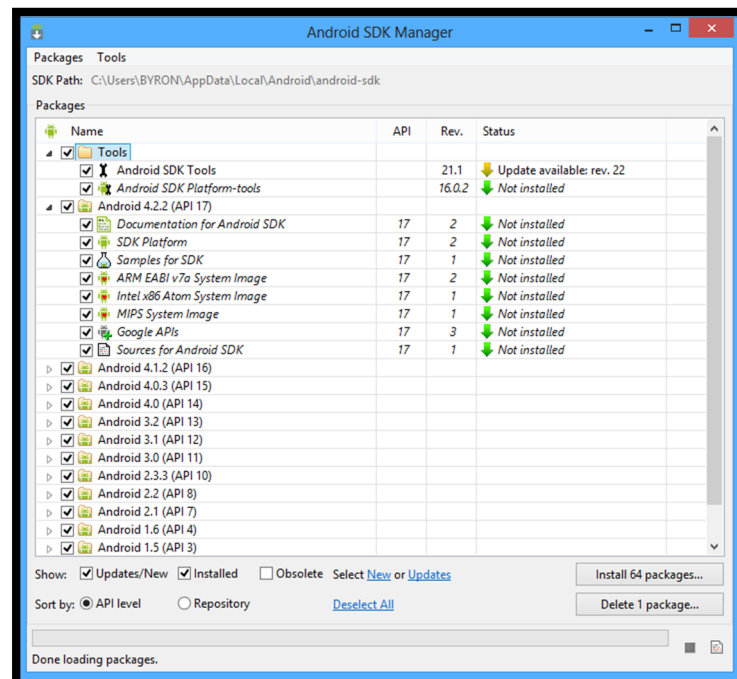


Figura 18. Interfaz del SDK Manager

# INSTALACIÓN Y CONFIGURACIÓN DEL PLUGIN ATD PARA ECLIPSE

1. Ejecutar Eclipse.

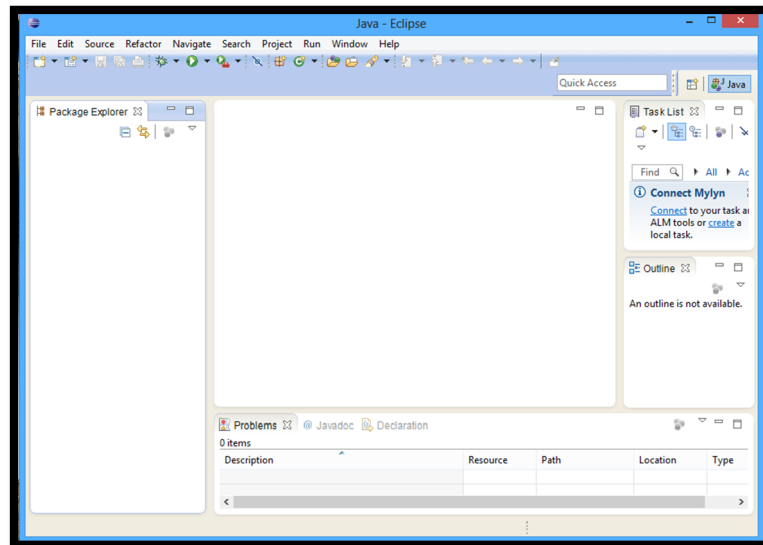


Figura 19. Ventana principal de Eclipse.

2. Click a Help – Install new software.

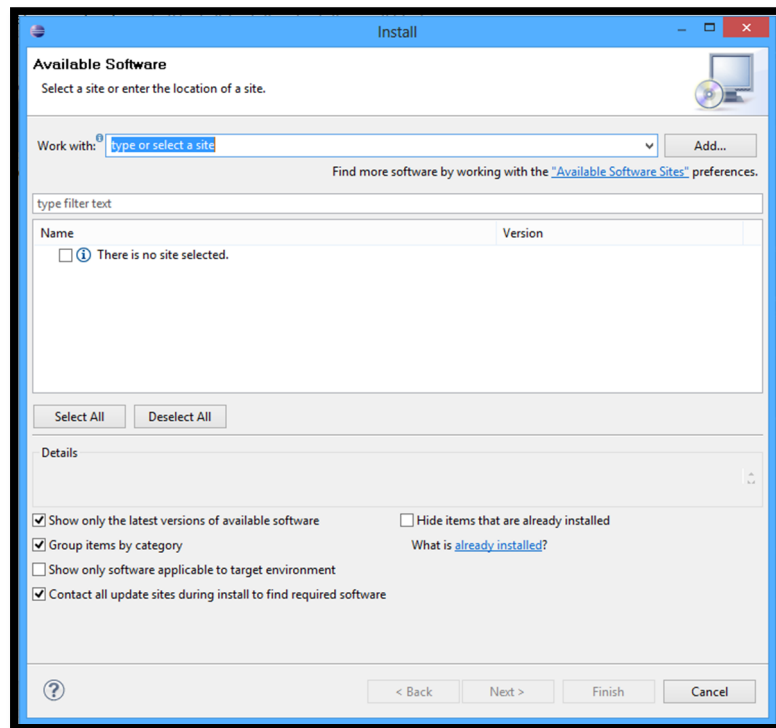
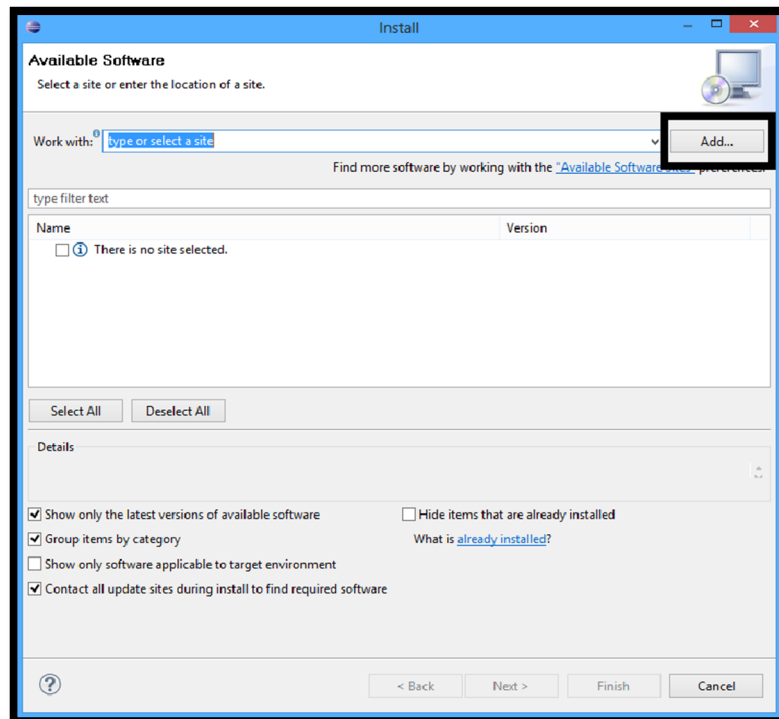


Figura 20. Ventana Install new Software

3. Ir a Add (Añadir).

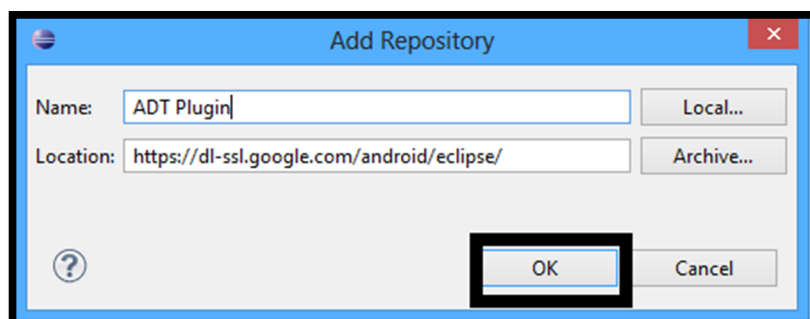


**Figura 21.** Ventana que indica añadir

4. Luego de dar click en añadir aparecerá la siguiente ventana donde se procederá a llenar los campos con la siguiente información.

Name: ADT Plugin

Location: <https://dl-ssl.google.com/android/eclipse/>



**Figura 22.** Ventana añadir un nuevo repositorio

5. Seleccionar todos los ítems que aparecerán en la ventana de instalación y dar a next (Siguiete).

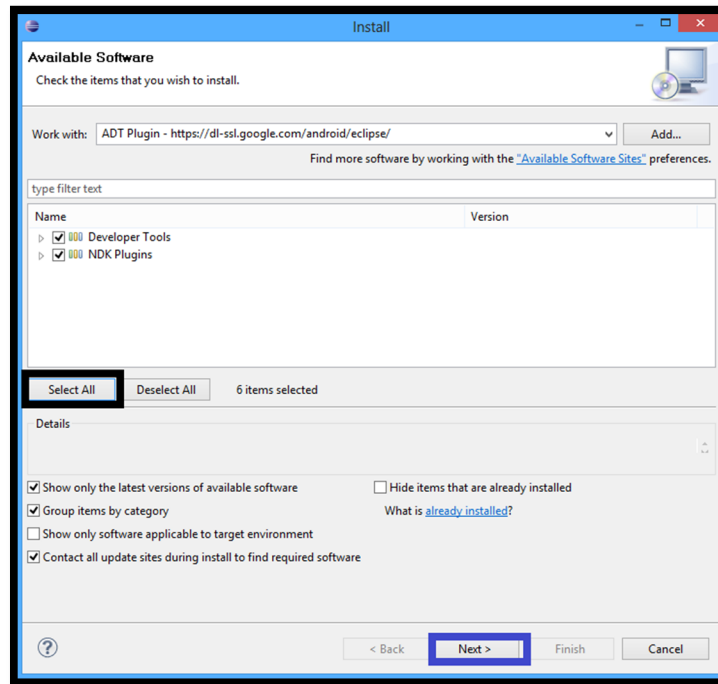


Figura 23. Items a instalar.

6. Continuar la instalación.

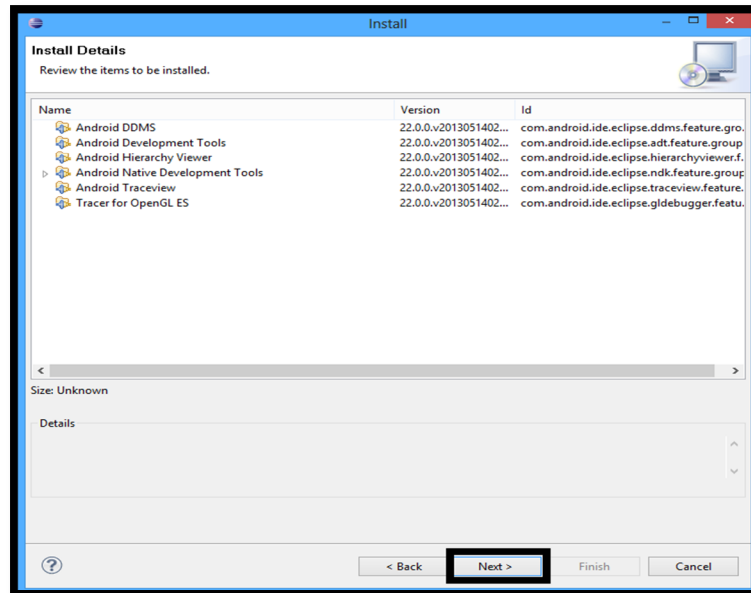


Figura 24. Paquetes a ser instalados

7. Aceptar los términos y condiciones del programa.

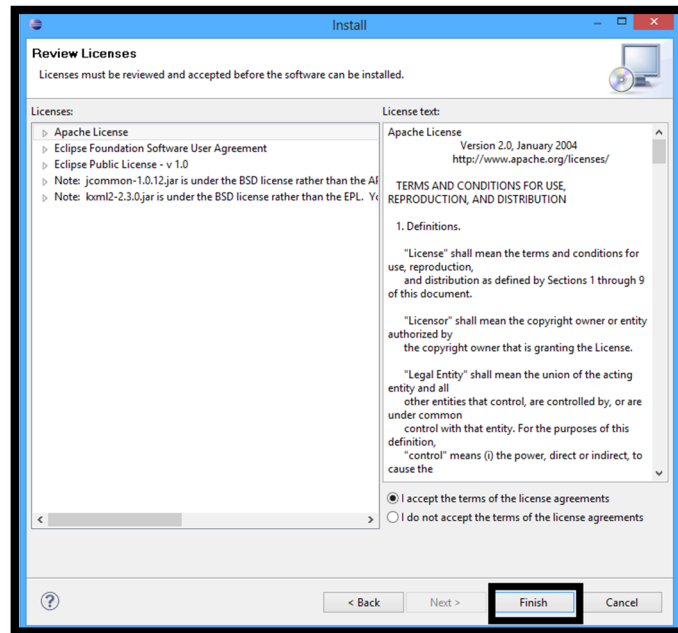


Figura 25. Términos y condiciones del programa

8. Al finalizar la instalación se reiniciará el Eclipse.

9. Indicar a Eclipse el directorio donde se instaló el SDK Android. En el menú Window escoger preferences (preferencias).

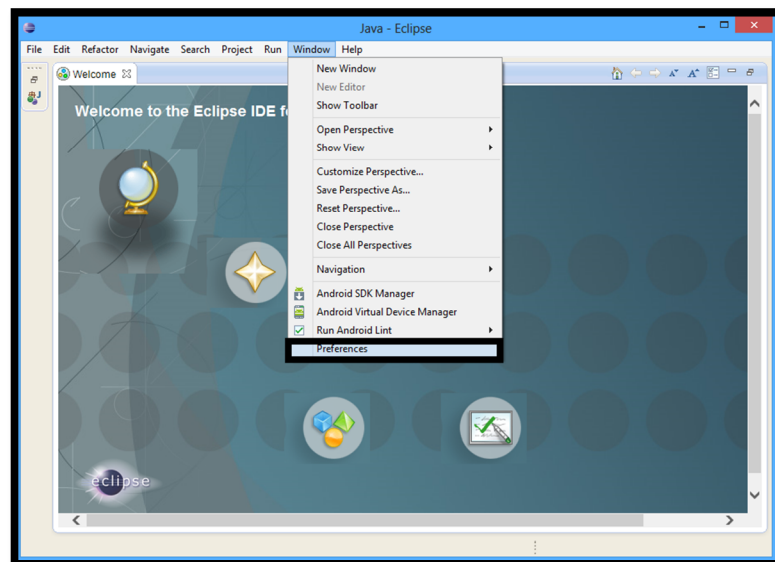


Figura 26. Opción preferencias del menu window

10. Seleccionar Android en la lista de menús que aparece al lado izquierdo de la ventana, e indicar la localización donde se instaló el SDK de Android (android-sdk).

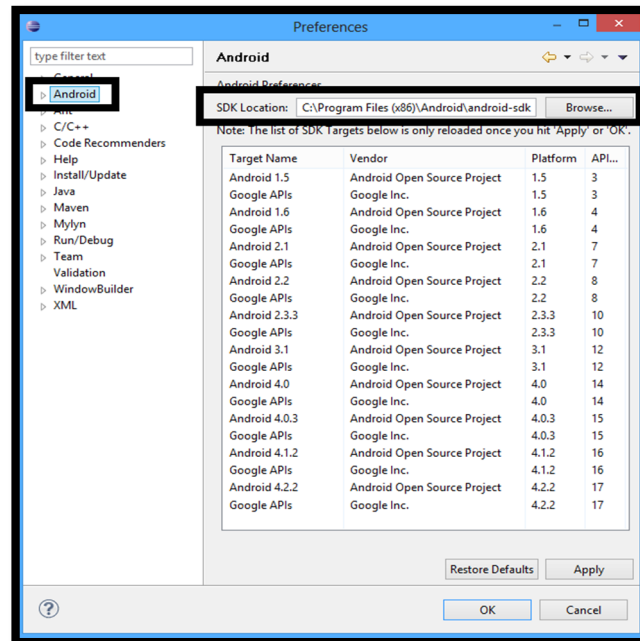


Figura 27. Localización del SDK de Android

## ANEXO C IMPORTAR LAS LIBRERÍAS Y LOS EJEMPLOS IOIO AL ENTORNO DE DESARROLLO ANDROID.

1. Descargar la versión recomendada de las librerías y ejemplos IOIO según la versión de la placa del siguiente link, en este caso la versión V1.

<https://github.com/ytai/ioio/wiki/Downloads>

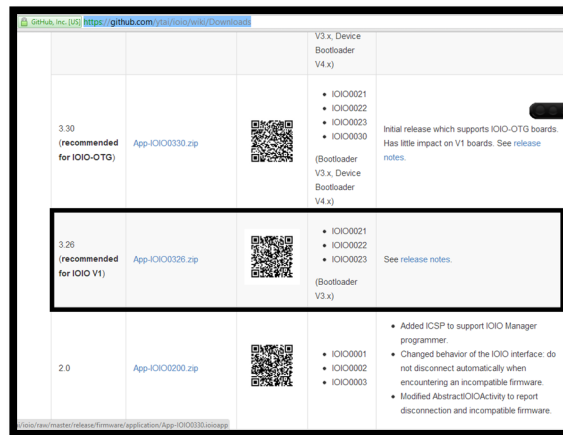


Figura 28. Paquete que contiene las librerías IOIO

2. Importar el archivo .zip que contiene las librerías. Para esto en eclipse dar a File (archivo), import (importar), en las carpetas escoger General, Existing Projects into Workspace (Proyecto existente en el área de trabajo), y next (siguiente).

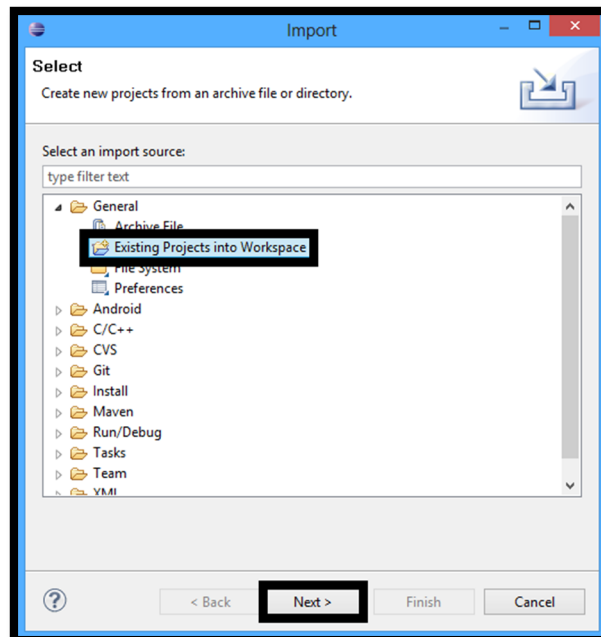


Figura 29. Vista de importación de un proyecto



3. Seleccionar la dirección donde se descargó el paquete con las librerías y ejemplos IOIO y clic en Finish (finalizar).

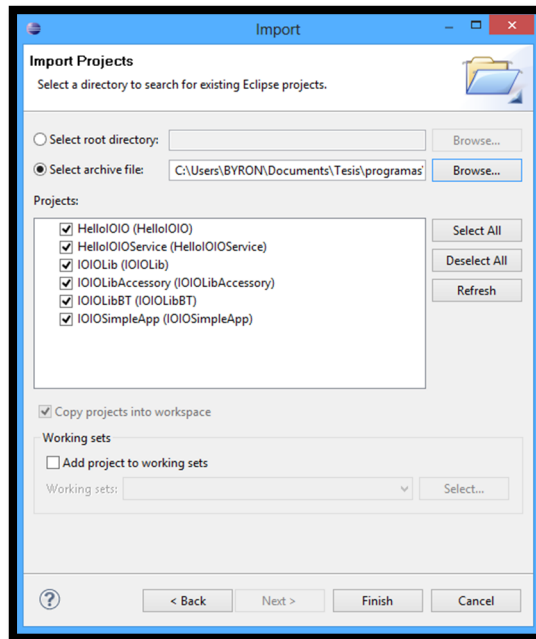


Figura 30. Dirección del paquete que contiene las librerías y ejemplos IOIO

4. Al finalizar la importación de las librerías se podrá verlos en el explorador de paquetes de Eclipse.

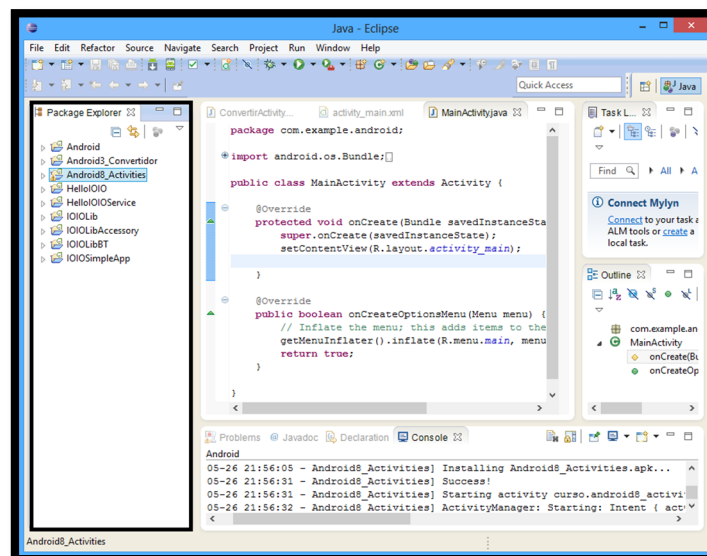


Figura 31. Explorador de proyectos

## **ANEXO D PROPUESTA DE PRACTICAS DIRIGAS A LOS ESTUDIANTES.**

1. Desarrollar un juego de luces secuenciales de 8 led's de izquierda a derecha, conectados a los pines de salida digital no tolerantes a 5V de la IOIO, con un retardo de 500ms en cada transición
2. Crear una aplicación que permita verificar los pulsos provenientes de tres pulsadores, adelantar, retroceder y encerrar respectivamente. Para esta aplicación hacer uso de la entrada digital a 5V.
3. Elaborar un contador decimal 0-99, con alarma sonora en un intervalo de diez números
4. Crear un letrero dinámico de izquierda a derecha con la frase "HOLA lolo" mediante la matriz de led's.
5. Elaborar un medidor de voltaje y temperatura, mediante el potenciómetro y el sensor de temperatura que forman parte del módulo análogo digital del entrenador.
6. Elaborar una aplicación que controle la luminosidad de un led conectado a la IOIO mediante PWM.
7. Desarrollar un lector de temperatura mediante un dispositivo que trabaja bajo el protocolo de comunicación I2C.
8. Desarrollar una aplicación que permita leer la temperatura de un sensor LM35 de forma inalámbrica mediante Bluetooth.
9. Realizar un medidor de distancia mediante una red punto a punto con módulos ZigBee, el transmisor envía la distancia del sensor, el receptor recibe la información de la distancia y en caso de que la distancia sea menor a 5m se emite una alarma visual.
10. Realizar una aplicación que reciba la información de ubicación del GPS y grafique el punto recibido en un mapa.
11. Desarrollar una aplicación que permita monitorear remotamente desde la Pc la humedad de suelo mediante un módulo Wi-Fi; en caso que se necesite riego enviar una alarma visual de una bombilla a 110V.

## ANEXO E ENTRENADOR PARA APLICACIONES ELECTRÓNICAS CON TERMINALES MÓVILES BASADOS EN SISTEMA OPERATIVO ANDROID

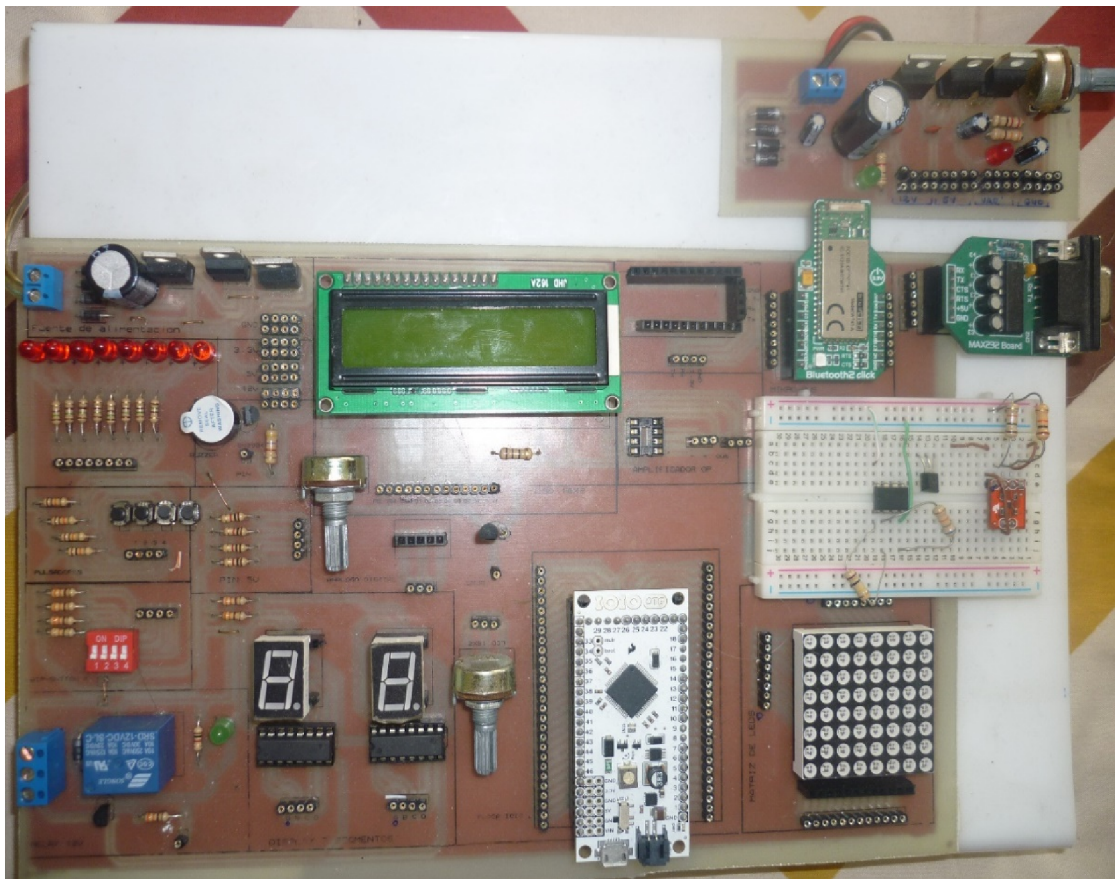


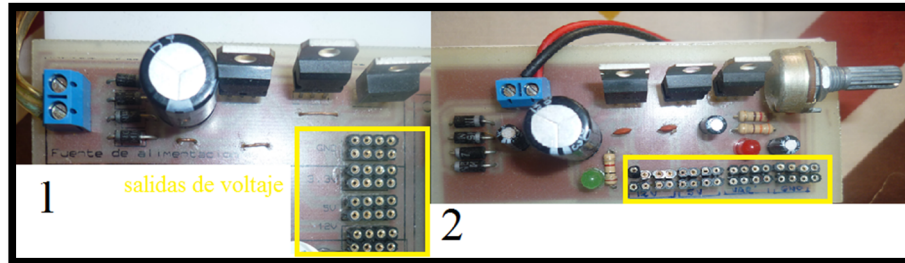
Figura 32. Entrenador Android

El entrenador electrónico Android es una placa diseñada y fabricada para ser usada en el laboratorio de la carrera de Ingeniería en Electrónica y Redes de Comunicación como equipo con el que los estudiantes podrán realizar las distintas prácticas electrónicas con dispositivos móviles basados en el sistema operativo Android.

El entrenador está constituido por diversos bloques que son: fuentes de alimentación, módulo de entrada y salida digital, conversión análogo-digital, comunicación serial, comunicación ZigBee y Wi-fi, comunicación Bluetooth, Gps, matriz de led's y protoboard.

- **Fuente de alimentación.**

El entrenador Android posee dos fuentes de alimentación totalmente independientes una específicamente destinada a alimentar a la circuitería interna de la placa y la otra orientada a alimentar a los distintos módulos adicionales del entrenador.



**Figura 33.** Fuentes de alimentación del entrenador

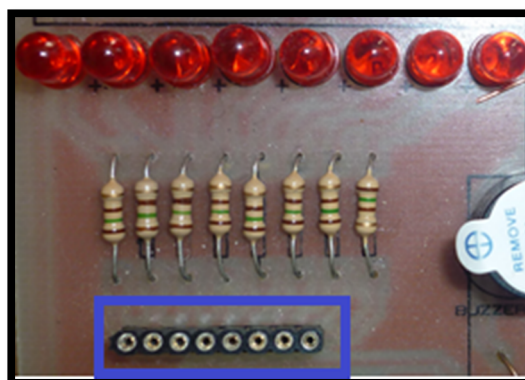
La fuente número uno es la que alimenta a los periféricos internos de la placa, esta fuente tiene salidas de voltaje que pueden ser usados para alimentar algún dispositivo electrónico; las salidas que tiene esta fuente son: 12VDC, 5VDC y 3.3VDC.

La fuente número dos ofrece varias salidas de voltaje para alimentar a los módulos y circuitos adicionales a la placa principal del entrenador, esta fuente cuenta con salidas de voltaje de 12VDC, 5VDC y voltaje variable de 0-12VDC controlado por el potenciómetro.

- **Entrada y salida digital**

**Led's**

El entrenador cuenta con 8 led's rojos, para poder energizar a los mismos es necesario colocar un voltaje de 3.3V o 5V en los zócalos de conexión de cada uno de ellos como se indica en la siguiente figura.



**Figura 34.** Led's del entrenador electrónico

## Pulsadores

El entrenador posee 4 pulsadores normalmente abiertos, que al activarlos enviarán un 1L (5VDC) al circuito al que estén conectados independientemente como se puede observar en la siguiente figura.



Figura 35. Pulsadores del entrenador

## Dipswitch

La placa de entrenamiento posee un Dip-switch de cuatro posiciones al momento de cambiar de posición este enviará la señal respectiva al circuito al cual está conectada la salida.

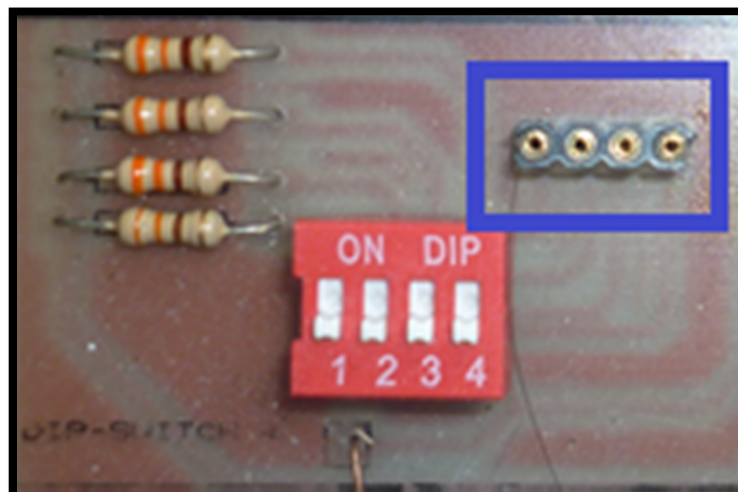


Figura 36. Dip-switch entrenador Android

## Relé

El entrenador cuenta con un relé que sirve de interfaz para alimentar circuitos con más de 12VDC o circuitos de corriente alterna, este relé ya está configurado internamente solo hace falta conectar el pin que funcionará como actuador y las salidas respectivas. Además cuenta con un led verde que indica cuando el relé está activado.

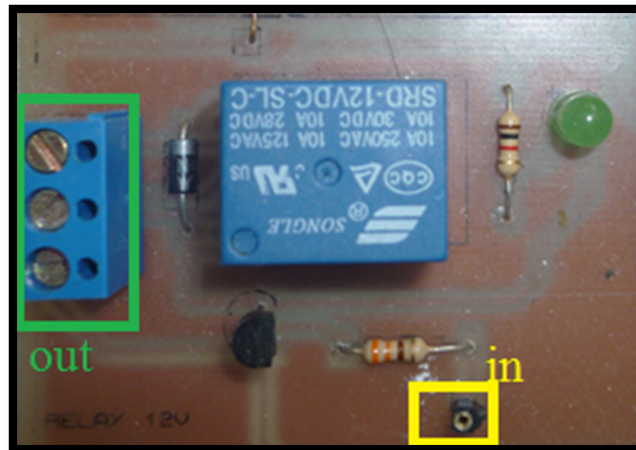


Figura 37. Relé del entrenador Android

## Buzzer.

El entrenador electrónico Android cuenta con un buzzer o zumbador electrónico que para poder usarlo solo hace falta activarlo mediante la entrada del mismo y esto ocasionará que emita el sonido.

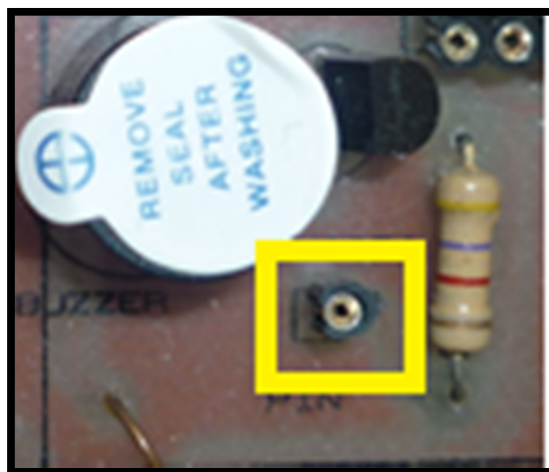


Figura 38. Buzzer del entrenador electrónico Android



## Display 7 segmentos

El entrenador cuenta con dos display 7 segmentos ánodo común cada uno con su respectivo decodificador 74LS47, la configuración y conexión ya está realizada internamente, lo único que el usuario debe hacer es conectar las entradas a los pines respectivos siendo la letra D el bit menos significativo (A0) y la letra A el bit más significativo (A3) como se observa en la siguiente figura.

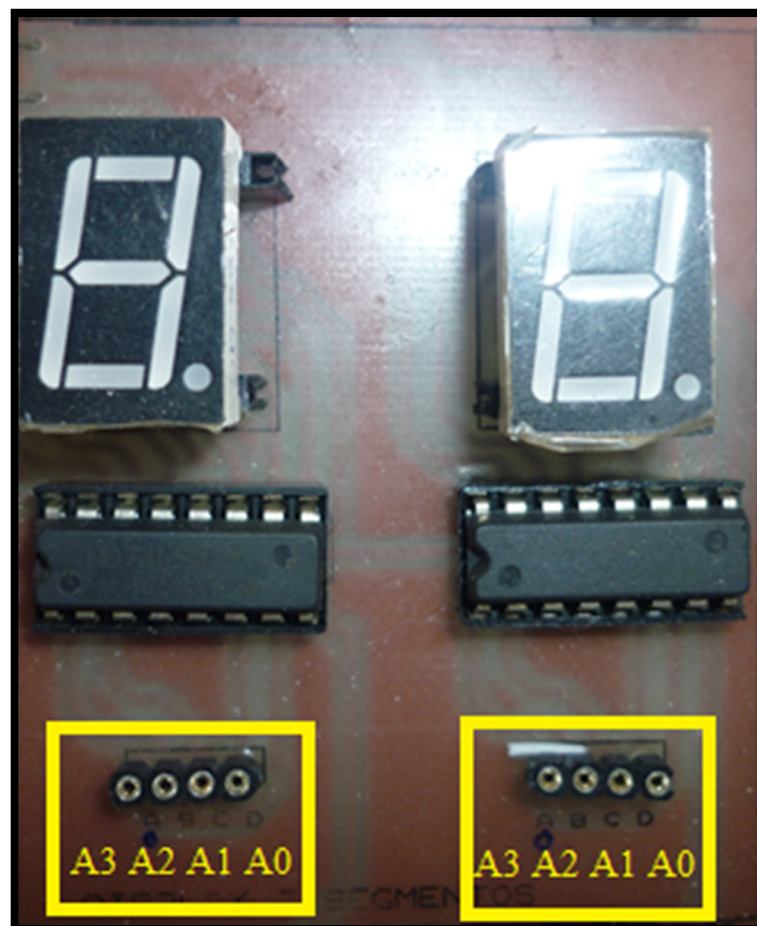


Figura 39. Display's 7 segmentos entrenador Android

## Pantalla LCD

La placa de entrenamiento Android cuenta con una pantalla Lcd de 16x2 configurada y lista para que los estudiantes puedan trabajar con ella. Esta pantalla tiene sus propios zócalos de conexión y solo hace falta conectar los pines correspondientes de cada zócalo y así comenzar con la utilización de dicho dispositivo. A demás tiene un potenciómetro de donde se puede ajustar el contraste de la pantalla.



**Figura 40.** LCD del entrenador Android

- **Módulo conversor análogo-digital.**

Para esta sección el entrenador cuenta con un sensor de temperatura LM-35, un potenciómetro y un zócalo donde se puede conectar un potenciómetro adicional. Los potenciómetros necesitan ser alimentados y configurados mientras que el sensor de temperatura solo hace falta conectar al controlador o a la IOIO la salida analógica y hacer uso del mismo.



**Figura 41.** Sección conversor análogo-digital del entrenador



- **Comunicación serial**

El módulo de comunicación serial está compuesto por el módulo Max232 Board este dispositivo cuenta con su propio zócalo de conexión y necesita ser alimentado a 5VDC entre sus pines para ser usado tiene pines Tx y Rx que permitirán la comunicación con otro equipo electrónico mediante comunicación serial.

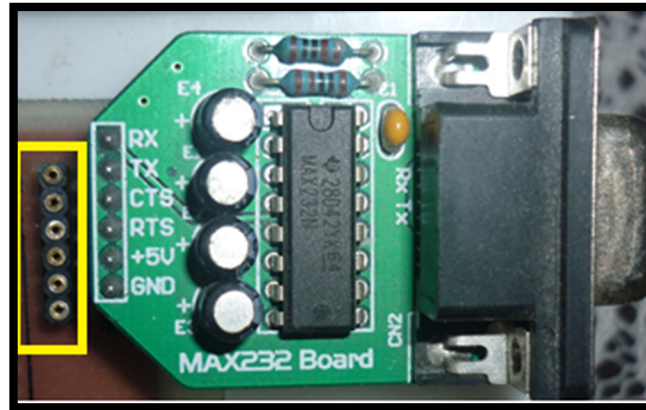


Figura 42. Módulo de comunicioón serial entrenador Android

- **Comunicación Bluetooth y GPS**

El módulo Bluetooth y GPS comparten el mismo zócalo de conexión, para trabajar con estos equipos es necesario alimentar de igual forma los módulos mediante la fuente adicional que trae consigo el entrenador y los pines respectivos de cada uno de los módulos que se vaya a conectar ya que el orden de los pines no es el mismo en los dos casos. De igual forma los dos módulos disponen de pines TX y RX para su respectiva comunicación.

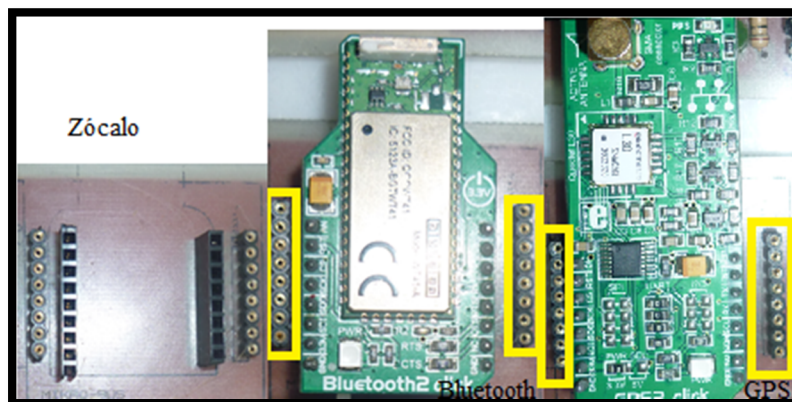


Figura 43. Módulo Bluetooth y GPS entrenador Android

- **Módulo Xbee y Wi-fi**

Los módulos Xbee y Wi-fly de igual manera comparten un mismo zócalo en común este zócalo tienes sus propios pines de conexión, necesita ser alimentado mediante la fuente adicional del entrenador y de igual forma dispone de pines de Tx y Rx que en este caso son los mismos para los dos módulos tanto Xbee como Wi-fly.

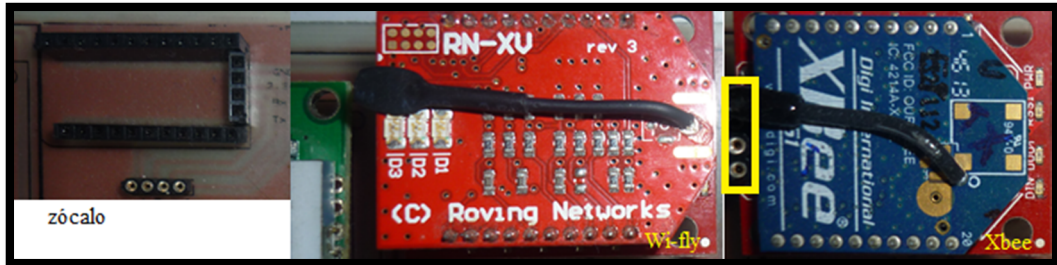


Figura 44. Módulo Wi-Fly y Xbee del entrenador

- **Matriz de led's**

El entrenador cuenta con una matriz de led's de ocho filas por ocho columnas esta tiene su propio zócalo de conexión y sus pines están ordenados por filas y por columnas con el propósito de facilitar la conexión y el trabajo con la misma.

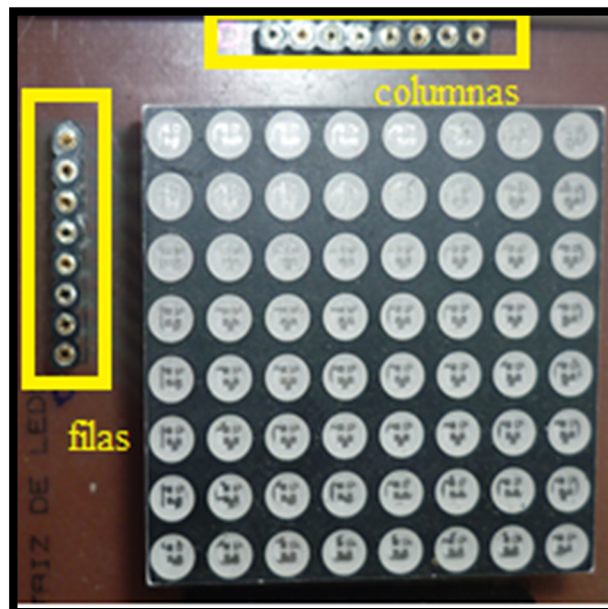
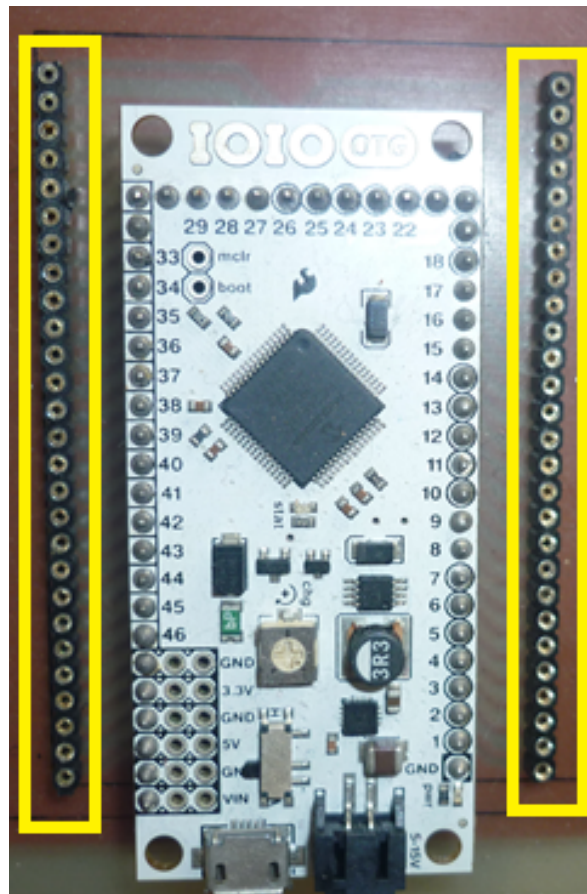


Figura 45. Matriz de led's del entrenador Android

- **Placa IOIO**

La placa IOIO está ubicada en la parte central del entrenador cuenta con su zócalo respectivo de donde se desprende cada uno de los pines en el mismo orden en que están enumerados en la placa principal de donde el estudiante puede hacer uso de ellos a su conveniencia.



**Figura 114.** Placa IOIO