ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA ELECTRÓNICA E INSTRUMENTACIÓN

PROYECTO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA E INSTRUMENTACIÓN

AUTORES: CHICAISA AIMACAÑA, JORGE DAVID

NIETO CONDO, RAÚL EDUARDO

TEMA: DISEÑO E IMPLEMENTACIÓN DE INSTRUMENTOS VIRTUALES UTILIZADOS EN LA MEDICIÓN DE VARIABLES ELÉCTRICAS CON DISPOSITIVOS MÓVILES BAJO LA PLATAFORMA ANDROID, PARA EL LABORATORIO DE CIRCUITOS ELECTRÓNICOS

DIRECTOR: ING. BUCHELI, JOSÉ

CODIRECTOR: ING. RIVAS, DAVID

LATACUNGA, MAYO 2014

UNIVERSIDAD DE LA FUERZAS ARMANDAS - ESPE

EXTENSIÓN LATACUNGA

INGENIERÍA ELECTRÓNICA E INSTRUMENTACIÓN

DECLARACIÓN DE RESPONSABILIDAD

Nosotros: Jorge David Chicaisa Aimacaña

Raúl Eduardo Nieto Condo

DECLARAMOS QUE:

El proyecto de grado denominado "DISEÑO E IMPLEMENTACIÓN DE INSTRUMENTOS VIRTUALES UTILIZADOS EN LA MEDICIÓN DE VARIABLES ELÉCTRICAS CON DISPOSITIVOS MÓVILES BAJO LA PLATAFORMA ANDROID, PARA EL LABORATORIO DE CIRCUITOS ELECTRÓNICOS", ha sido desarrollado con base a una investigación exhaustiva, respetando derechos intelectuales de terceros, conforme las citas que constan al pie de las páginas correspondiente, cuyas fuentes se incorporan en la bibliografía.

Consecuentemente este trabajo es de nuestra autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance científico del proyecto de grado en mención.

Latacunga, Mayo del 2014

Jorge David Chicaisa Aimacaña	Raúl Eduardo Nieto Condo
C.I: 0503251548	C.I: 1804249959

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

EXTENSIÓN LATACUNGA

INGENIERÍA ELECTRÓNICA E INSTRUMENTACIÓN

CERTIFICADO

Ing. José Bucheli (Director)

Ing. David Rivas (Codirector)

CERTIFICAN

Que el trabajo titulado "DISEÑO E IMPLEMENTACIÓN DE INSTRUMENTOS VIRTUALES UTILIZADOS EN LA MEDICIÓN DE VARIABLES ELÉCTRICAS CON DISPOSITIVOS MÓVILES BAJO LA PLATAFORMA ANDROID, PARA EL LABORATORIO DE CIRCUITOS ELECTRÓNICOS" realizado por los Señores Jorge David Chicaisa Aimacaña y Raúl Eduardo Nieto Condo, ha sido guiado y revisado periódicamente y cumple normas estatuarias establecidas de la Universidad de las Fuerzas Armadas-ESPE.

Debido a que constituye un trabajo de excelente contenido científico y aplicable para el desarrollo profesional, **SI** recomiendan su aplicación.

El mencionado trabajo consta de un documento empastado y un disco compacto el cual contiene los archivos en formato portátil de Acrobat (pdf). Autorizan a los Señores Jorge David Chicaisa Aimacaña y Raúl Eduardo Nieto Condo que lo entregue al Ing. José Bucheli Andrade, en su calidad de Director de la Carrera.

Latacunga, Mayo del 2014

Ing. José Bucheli	Ing. David Rivas
DIRECTOR	**CODIRECTOR**

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

EXTENSIÓN LATACUNGA

INGENIERÍA ELECTRÓNICA E INSTRUMENTACIÓN

AUTORIZACIÓN

Nosotros: Jorge David Chicaisa Aimacaña

Raúl Eduardo Nieto Condo

Autorizamos a la UNIVERSIDAD DE LA FUERZAS ARMANDAS-ESPE, la publicación, en la biblioteca virtual de la Institución del trabajo "DISEÑO E IMPLEMENTACIÓN DE INSTRUMENTOS VIRTUALES UTILIZADOS EN LA MEDICIÓN DE VARIABLES ELÉCTRICAS CON DISPOSITIVOS MÓVILES BAJO LA PLATAFORMA ANDROID, PARA EL LABORATORIO DE CIRCUITOS ELECTRÓNICOS", cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y autoría.

Latacunga, Mayo del 2014

Jorge David Chicaisa Aimacaña	Raúl Eduardo Nieto Condo
C.I: 0503251548	C.I: 1804249959

DEDICATORIA

Este título se lo dedico a DIOS por darme la oportunidad de vivir y haber llegado hasta donde estoy, por darme fuerza en momentos de decepción y por darme motivos para sonreír. Se lo dedico a mi padre, José, por ser ejemplo de perseverancia, honestidad y sacrificio, además por haber sido el pedestal más sólido sobre el cual me he apoyado todo el tiempo en que he perseguido este sueño de ser un profesional competente; a mi madre, Carmen, por apoyarme todo el tiempo y por enseñarme a cultivar las virtudes necesarias para enfrentarme al desafío que representa ser una persona de bien, dispuesta a colaborar con la comunidad; a mis hermanos y hermanas que siempre han estado dándome palabras de aliento y de cierta forma ayudándome a mirar claros mis objetivos.

A todas estas personas les dedico cada uno de mis logros, diciéndoles simplemente que las amo y que todo lo hecho ha sido por y gracias a Uds.

Jorge David Chicaisa Aimacaña

Dedico este proyecto de tesis principalmente a Dios y a la Virgen Santísima, por darme la vida y permitirme concluir y ascender un peldaño más en mi vida profesional. A mis padres, Raúl y Zoila por ser el pilar más importante y por demostrarme siempre su amor y apoyo incondicional. A mis hermanas Jazmin y Valeria por ser quienes me ofrecieron su mano cuando sentía desmayar.

A todas y cada una de las personas que ayudaron directa e indirectamente en la culminación de este proyecto.

Raúl Eduardo Nieto Condo

AGRADECIMIENTO

A mis padres, porque siempre me han apoyado en todas las decisiones que he tomado y gracias a sus consejos he podido alcanzar este logro tan importante en mi vida.

Al Ing. José Bucheli, director de tesis y al Ing. David Rivas, codirector de tesis, por sus valiosas guías y acertada dirección para la realización de este proyecto de grado.

A la Ing. Jacqueline Llanos, por el apoyo científico y moral que me supo brindar durante la realización de este proyecto.

Jorge David Chicaisa Aimacaña

A mi familia, por brindarme su amor, paciencia, y sobretodo enseñarme a luchar ante cualquier obstáculo y no rendirme ante las adversidades de la vida.

A los Ingenieros José Bucheli y David Rivas, director y codirector de tesis respectivamente, por sus valiosas guías y acertadas direcciones para la realización de este proyecto de grado, considerándolos más que maestros unos grandes amigos.

A la Facultad de Ingeniería Electrónica e Instrumentación por los conocimientos impartidos durante mis años de estudio

A mi compañero Jorge Chicaisa, un soporte grande para la culminación de este proyecto, así como a mis compañeros de formación académica, Juan Ruiz, Rafael Pérez, Diego Paredes, Marcelo Guevara, Belén Guano, Diana Romero, gracias por compartir buenos y malos momentos.

Raúl Eduardo Nieto Condo

ÍNDICE DE CONTENIDOS

DECLARACIÓN DE RESPONSABILIDAD ..... **i**

CERTIFICADO ..... **ii**

AUTORIZACIÓN ..... **iii**

DEDICATORIA ..... **iv**

AGRADECIMIENTO ..... **v**

ÍNDICE DE CONTENIDOS ..... **vi**

LISTADO DE TABLAS ..... **x**

LISTADO DE FIGURAS ..... **xi**

LISTADO DE ANEXOS ..... **xiv**

RESUMEN ..... **xv**

ABSTRACT ..... **xvi**

FUNDAMENTOS TEÓRICOS **1**

1.1 DESCRIPCIÓN DEL PROBLEMA 1

1.2 JUSTIFICACIÓN E IMPORTANCIA DEL PROBLEMA 2

1.3 VARIABLES ELÉCTRICAS 3

1.3.1 CORRIENTE ELÉCTRICA 3

1.3.2 TENSIÓN ELÉCTRICA 4

1.3.3 POTENCIA ELÉCTRICA 6

1.3.4 RESISTENCIA ELÉCTRICA 6

1.4 INSTRUMENTACIÓN VIRTUAL 7

1.4.1 INTRODUCCIÓN 7

1.4.2 INSTRUMENTO VIRTUAL 8

1.4.3 INSTRUMENTACIÓN VIRTUAL FRENTE A LA CONVENCIONAL 9

1.5 ADQUISICIÓN DE DATOS MEDIANTE MICROCONTROLADORES 10

1.5.1 MICROCONTROLADORES ARM DE 32 BITS 10

1.5.2 ARQUITECTURA ARM 11

1.5.2.1 CARACTERÍSTICAS 14

1.5.2.2 VENTAJAS 15

1.5.2.3 NÚCLEOS ARM PARA MICROCONTROLADORES 16

1.6 ARDUINO 17

1.6.1 WIRING 18

1.6.2 PROCESSING 18

1.6.3 ARDUINO DUE 18

1.6.3.1 BENEFICIOS DEL NÚCLEO ARM 19

1.6.3.2 CARACTERÍSTICAS DE LA TARJETA ARDUINO DUE 20

1.7 ESTANDAR BLUETOOTH (IEEE 802.15.1) 21

1.7.1 ORIGEN TECNOLOGÍA BLUETOOTH 21

1.7.2 FUNCIONAMIENTO 22

1.7.3 TOPOLOGÍA BLUETOOTH 23

1.7.4 ARQUITECTURA DEL HARDWARE 24

1.7.5 ARQUITECTURA DE SOFTWARE 25

1.7.6 VENTAJAS 27

1.7.7 ESPECIFICACIONES 27

1.8 DESCRIPCIÓN DE LA PLATAFORMA ANDROID 29

1.8.1 INTRODUCCIÓN 29

1.8.2 ARQUITECTURA 30

1.8.3 CARACTERÍSTICAS PRINCIPALES 32

1.8.4 INTERFAZ DE PROGRAMACIÓN DE APLICACIONES (API) 34

1.8.5 VERSIONES DEL SISTEMA OPERATIVO ANDROID 36

1.8.6 COMPARACIÓN ENTRE SISTEMAS OPERATIVOS MÓVILES 39

ANÁLISIS Y DISEÑO DE INSTRUMENTOS VIRTUALES 40

2.1 DISEÑO DEL HARDWARE **..... 40**

2.1.1 HARDWARE DEL INSTRUMENTO VIRTUAL 40

2.1.2 SISTEMA DE ADQUISICIÓN DE DATOS 41

2.1.2.1 ACONDICIONAMIENTO DE SEÑALES 42

2.1.2.1.1 ACONDICIONAMIENTO DE VOLTAJE 44

2.1.2.1.2 ACONDICIONAMIENTO DE CORRIENTE 52

2.1.2.1.3 MEDICIÓN DE LA RESISTENCIA 55

2.1.2.2 CONVERSIÓN ANÁLOGA DIGITAL 56

2.1.2.5.1 MUESTREO 57

2.1.2.5.2 CUANTIZACIÓN 58

2.1.2.5.3 CODIFICACIÓN 58

2.1.3 COMUNICACIÓN SERIAL 58

2.1.3.1 MODULO BLUETOOTH HC-06 58

2.2 DISEÑO DEL SOFTWARE **..... 60**

2.2.1 ENTORNO DE DESARROLLO PARA ARDUINO 61

2.2.2 LIBRERÍAS 62

2.2.2.1 AnalogReadResolution ( ) 62

2.2.2.2 AnalogWriteResolution ( ) 63

2.2.3 PROGRAMACIÓN DE LA TARJETA ARDUINO DUE 64

2.2.3.1 MULTÍMETRO 65

2.2.3.1.1 DIAGRAMA DE FLUJO DEL PROGRAMA DEL MULTÍMETRO 65

2.2.3.2 OSCILOSCOPIO 69

2.2.3.2.1 DIAGRAMA DE FLUJO DEL PROGRAMA DEL OSCILOSCOPIO 69

2.2.4 INSTRUMENTOS VIRTUALES PARA LA PLATAFORMA ANDROID 70

2.2.4.1 ECLIPSE COMO ENTORNO DE DESARROLLO ANDROID 70

2.2.4.2 CONFIGURACIÓN DE ECLIPSE PARA EL DESARROLLO DE APLICACIONES ANDROID 71
2.2.4.3 CREACIÓN DE UN NUEVO PROYECTO ANDROID 72
2.2.4.4 DIAGRAMA DE FLUJO DE LA APLICACIÓN EN ANDROID 77

PRUEBAS Y OPERACIÓN DEL SISTEMA 84

3.1 GENERALIDADES **84**

3.2 PRUEBAS REALIZADAS AL MULTÍMETRO IMPLEMENTADO **84**

3.3 PRUEBAS REALIZADAS AL OSCILOSCOPIO IMPLEMENTADO **92**

CONCLUSIONES Y RECOMENDACIONES 94

4.1 CONCLUSIONES **94**

4.2 RECOMENDACIONES **97**

BIBLIOGRAFÍA 98

LINKGRAFÍA 99

ANEXOS 101

LISTADO DE TABLAS

CAPÍTULO 1

Tabla 1.1: Núcleos ARM para Microcontroladores 16
Tabla 1.2: Características de la Tarjeta Arduino Due 20
Tabla 1.3: Características de tecnologías de corto enlace 22
Tabla 1.4: Clases Bluetooth 28
Tabla 1.5: Ancho de Banda según las versiones Bluetooth 28
Tabla 1.6: Características principales plataforma Android 33
Tabla 1.7: Nivel API en función de la plataforma Android 35
Tabla 1.8: Versiones del sistema operativo Android 36
Tabla 1.9: Comparativa de sistemas operativos para Smartphone's 39

CAPÍTULO 2

Tabla 2.1: Características modulo Bluetooth HC-06 59
Tabla 2.2: Funciones comunes de la barra de herramientas del entorno de desarrollo para Arduino 62

CAPÍTULO 3

Tabla 3.1: Comparación de las mediciones realizadas para el voltaje alterno. 86
Tabla 3.2: Comparación de las mediciones realizadas para el voltaje continuo. 87
Tabla 3.3: Comparación de las medidas de corriente alterna realizadas. 88
Tabla 3.4: Comparación de las medidas de corriente continua realizadas. 89
Tabla 3.5: Comparación de las mediciones de resistencia realizadas 90
Tabla 3.6: Comparación de las señales mostradas en el osciloscopio Agilent DSO-X 2014A y las señales mostradas en la aplicación Android 92

LISTADO DE FIGURAS

CAPÍTULO 1

Figura 1.1: Diagrama de bloques del proyecto a ser implementado 2
Figura 1.2: Corriente contra flujo de electrones 4
Figura 1.3: Diferencia de cargas eléctricas entre dos cuerpos A y B, referenciados a masa (0V) 5
Figura 1.4: Diagrama de bloques de un Sistema basado en instrumentación virtual 8
Figura 1.5: Microcontrolador ARM 11
Figura 1.6: Estructura de un Microcontrolador ARM 12
Figura 1.7: Organización interna del procesador ARM 14
Figura 1.8: Tarjeta de entrenamiento Arduino Due 19
Figura 1.9: Modelo de conectividad Bluetooth 21
Figura 1.10: Red Dispersa 24
Figura 1.11: Arquitectura Hardware Bluetooth 25
Figura 1.12: Arquitectura Software Bluetooth 26
Figura 1.13: Logo principal de Android 30
Figura 1.14: Sistema de capas de Android 31

CAPÍTULO 2

Figura 2.1: Diagrama de bloques del hardware 40
Figura 2.2: Diagrama de bloques de un sistema de adquisición de datos 42
Figura 2.3: Acondicionamiento de señales 43
Figura 2.4: Diagrama esquemático del acondicionamiento de la señal de voltaje alterno y continuo 44
Figura 2.5: Divisor de Tensión 45
Figura 2.6: Divisor de tensión aplicado al voltaje AC de entrada 46
Figura 2.7: Divisor de tensión aplicado al voltaje continuo de entrada 47
Figura 2.8: Seguidor de tensión 48
Figura 2.9: Símbolo eléctrico y terminales de un amplificador operacional 49
Figura 2.10: Amplificador Operacional LM741 49
Figura 2.11: Extractor de valor medio absoluto MAV 51
Figura 2.12: Placa de acondicionamiento de las señales de voltaje alterno y continuo 51
Figura 2.13: Diagrama esquemático del acondicionamiento de la señal de corriente continua. 52

Figura 2.14: Placa final del acondicionamiento de la señal de corriente continua. 53
Figura 2.15: Diagrama esquemático del acondicionamiento de la señal de corriente alterna. 54
Figura 2.16: Placa final del acondicionamiento de la señal de corriente alterna 54
Figura 2.17: Diagrama esquemático del circuito RC implementado 55
Figura 2.18: Circuito RC implementado en la placa final 56
Figura 2.19: Proceso de la conversión análoga digital 56
Figura 2.20: Modulo Bluetooth HC-06 59
Figura 2.21: Conexión física entre tarjeta Arduino Due y módulo Bluetooth HC-06 60
Figura 2.22: Entorno de desarrollo para Arduino 61
Figura 2.23: Diagrama de flujo de la programación del Multímetro en la Tarjeta Arduino Due 66
Figura 2.24: Diagrama de flujo de la programación del Multímetro en la Tarjeta Arduino Due 67
Figura 2.25: Diagrama de flujo de la programación del Multímetro en la Tarjeta Arduino Due 68
Figura 2.26: Diagrama de flujo de la programación del osciloscopio en la tarjeta Arduino Due 69
Figura 2.27: Creación de un nuevo proyecto Android 73
Figura 2.28: Configuración de un nuevo proyecto Android 74
Figura 2.29: Configuración icono de la aplicación 75
Figura 2.30: Icono del proyecto para la aplicación Android 75
Figura 2.31: Creación de una actividad para un nuevo proyecto Android 76
Figura 2.32: Nueva actividad en blanco para el proyecto android 77
Figura 2.33: Diagrama de flujo de la Activity (principal) 78
Figura 2.34: Layout de la Activity (Principal) 78
Figura 2.35: Diagrama de flujo de la Activity (Ayuda) 79
Figura 2.36: Layout de la Activity (Ayuda) 79
Figura 2.37: Diagrama de flujo de la Activity (Multímetro) 80
Figura 2.38: Layout de la Activity (Multímetro) 81
Figura 2.39: Diagrama de flujo de la Activity (Osciloscopio) 82
Figura 2.40: Layout de la Activity (Osciloscopio) 83

CAPÍTULO 3

Figura 3.1: Fuente de Alimentación de LabVolt. 85

Figura 3.2: Curva de respuesta del voltimetro AC diseñado, respecto a un patrón... 87
Figura 3.3: Curva de respuesta del Voltímetro DC diseñado, respecto a un atrón.... 88
Figura 3.4: Curva de respuesta del Amperímetro AC diseñado, respecto a un patrón. 89
Figura 3.5: Curva de respuesta del Amperímetro DC diseñado, respecto a un patrón. 90
Figura 3.6: Curva de respuesta del óhmetro diseñado respecto a un patrón..... 91
Figura 3.7: Curva de respuesta de un osciloscopio diseñado respecto a un patrón 93

LISTADO DE ANEXOS

A. Glosario de Términos

B. Circuitos Esquemáticos

C. Circuitos Impresos

D. Programación ANDROID

E. Programación ARDUINO

F. Certificados de Calibración

G. Manual de Usuario

RESUMEN

El presente proyecto está orientado al estudio y desarrollo de un dispositivo que permita crear productos comercialmente rentables, utilizando dispositivos móviles que trabajen con software libre, en este proyecto se analiza y crea las bases para el desarrollo de instrumentos virtuales utilizados en la medición de variables eléctricas, sobre la plataforma de código abierto Android. Para el desarrollo del dispositivo se utiliza un microcontrolador de 32 bits, que permite realizar el procesamiento digital de las variables eléctricas, además gestiona la comunicación con cualquier dispositivo que trabaje con Android, a través de un protocolo de comunicación inalámbrica, como lo es Bluetooth. En cuanto a su desarrollo, el proyecto comenzará analizando, paso a paso, los distintos procesos de su evolución, partiendo de fundamentos teóricos, conceptos básicos. A partir del segundo capítulo, se presenta el diseño del hardware y software necesario para obtener los instrumentos de un laboratorio de circuitos electrónicos, de forma virtual en cualquier dispositivo que trabaje con el sistema operativo Android, describiendo sus ventajas, funcionalidades y la manera en que se deberá desarrollar el dispositivo. Finalmente, una vez desarrollado el equipo, se presenta el producto como concepto comercial. Desarrollando la presentación propia del producto, su funcionamiento, ventajas y características, su valor distintivo frente a otros equipos de la competencia. El análisis de mercado del producto, permitirá posicionarlo en el nicho de negocios esperado, considerando la competencia y estrategia de venta, tomando en cuenta su análisis de costos y proyección de producción.

ABSTRACT

This project aims to study and develop a device that allows to create commercially viable products using mobile devices to work with free software, this project analyzes and creates the basis for the development of virtual instruments used in measuring electrical variables on open source Android platform. For the development of the device 32-bit microcontroller, which allows the processing of digital electrical variables used also manages communication with any device that works with Android, through a wireless communication protocol, such as Bluetooth. In terms of development, the project will start by analyzing, step by step, the various processes of evolution, based on theoretical foundations, basic concepts. From the second chapter, the design of the hardware and software required for laboratory instruments electronic circuits on virtually any device that works with the Android operating system, is presented describing its advantages, features and how must develop the device. Finally, once the team developed the product as a business concept is presented. Developing own product presentation, operation, advantages and features, its distinctive value against other teams in the competition. The analysis of the product, will place it in the business expected, considering the competition and sales strategy, taking into account their cost analysis and projections of production.

CAPÍTULO 1

FUNDAMENTOS TEÓRICOS

1.1 DESCRIPCIÓN DEL PROBLEMA

El presente proyecto tiene como propósito desarrollar instrumentos virtuales para la medición de variables eléctricas, que son muy necesarios para toda persona que se dedica al estudio de la electrónica, tales como: un óhmetro, un osciloscopio, un voltímetro y un amperímetro para señales de AC y DC, estas aplicaciones se ejecutarán sobre la plataforma ANDROID, debido a que esta plataforma es de arquitectura abierta, y permite desarrollar diversas aplicaciones, sin tener que pagar por licencias.

Además se implementará un dispositivo electrónico portátil y de bajo costo que permita adquirir las señales a medir, y gestione la comunicación con el dispositivo móvil, que puede ser un Smartphone o una Tablet. Para esto se utilizará una tarjeta de adquisición de datos con un microprocesador de 32 bits.

En el Figura 1.1 se puede observar el diagrama de bloques del proyecto a implementar.

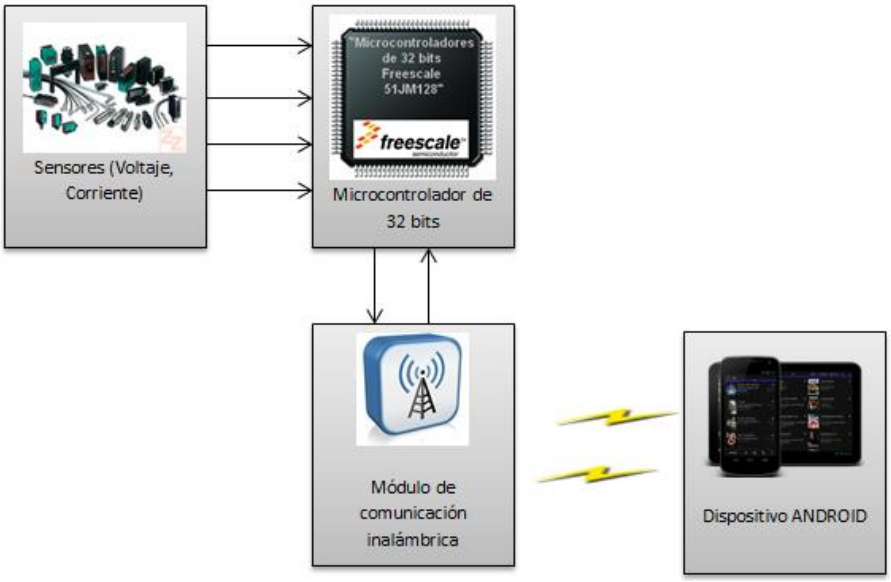


Figura 1.1: Diagrama de bloques del proyecto a ser implementado

1.2 JUSTIFICACIÓN E IMPORTANCIA DEL PROBLEMA

El desarrollo de este proyecto es muy importante debido a que la utilización de dispositivos móviles en los últimos años han ido en aumento, no solo los teléfonos celulares en sus modalidades de teléfono y Smartphone se encuentran presentes en el diario vivir de las personas; toda una cantidad de dispositivos móviles para múltiples propósitos se han vuelto disponibles, desde los ya conocidos PDA, hasta dispositivos increíblemente especializados como por ejemplo los navegadores basados en GPS, esto se debe a que tienen muchas ventajas tales como: movilidad, conectividad, menor tamaño, entre otros.

Gracias a las ventajas que estos presentan, se los puede utilizar para diversas aplicaciones. En este caso se los utilizará como instrumentos de un laboratorio de electrónica básico.

El desarrollo de este proyecto es muy importante por:

- En Ecuador los avances tecnológicos no se han desarrollado de forma

creciente en los últimos tiempos, por lo que es conveniente fomentar la investigación en nuevos dispositivos que permitan realizar aplicaciones de uso común que permitan cubrir una necesidad en la sociedad como lo es la plataforma Android, en conjunto con los microcontroladores de 32 bits.

- Al implementar instrumentos virtuales para la medición de variables eléctricas con transmisión de datos a través del protocolo Bluetooth, permitirá realizar un monitoreo de dichas variables en cualquier dispositivo que trabaje con el sistema operativo Android, de forma remota.
- El costo de los instrumentos físicos para la medición de variables eléctricas es alto ya que poseen Software y Hardware propietario, es por ese motivo que a través de este proyecto se pretende realizar un dispositivo de bajo costo ya que se trabajará con plataformas de código libre.
- La implementación de este proyecto permitirá tener en un solo dispositivo móvil los diferentes instrumentos necesarios para la medición de variables eléctricas, con tan solo ejecutar la aplicación correspondiente al instrumento que se requiera utilizar.

1.3 VARIABLES ELÉCTRICAS

1.3.1 CORRIENTE ELÉCTRICA

La corriente eléctrica o intensidad eléctrica es el flujo de carga eléctrica por unidad de tiempo que recorre un material, está definida por convenio en dirección contraria al desplazamiento de los electrones (Figura 1.2). (MOLINA, 2012)

Cuando de un punto a otro de un conductor se desplaza una o más cargas eléctricas diremos que circula por él una corriente eléctrica. Si la carga se transfiere a una velocidad de 1 culombio por segundo (C/s) la corriente por el conductor tiene una intensidad de 1 amperio (A), es decir, 1 A = 1 C/s. (EDMINISTER, 1965)

Mientras mayor es la cantidad de carga que fluye a través de un conductor por unidad de tiempo, mayor es la corriente; en forma de ecuación:

$$I = \frac{Q}{t} \tag{1.1}$$

Dónde: I = corriente en amperios (A),

Q = carga en coulombs (C),

t = tiempo en segundos (s).

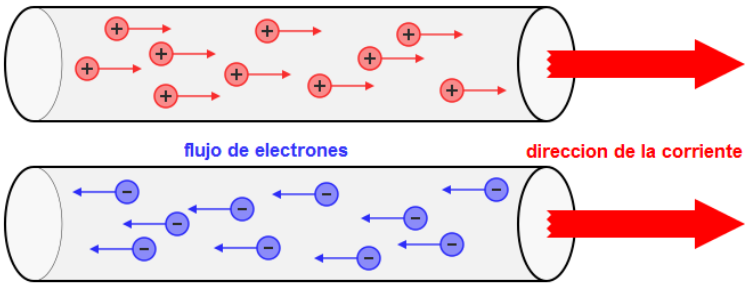


Figura 1.2: Corriente contra flujo de electrones

1.3.2 TENSIÓN ELÉCTRICA

La tensión eléctrica o diferencia de potencial es, por definición, el trabajo necesario para desplazar la unidad de carga eléctrica positiva de un punto al otro en contra o a favor de las fuerzas del campo. (EDMINISTER, 1965)

Si dos puntos que tienen una diferencia de potencial como se muestra en la Figura 1.3 se unen mediante un conductor, se producirá un flujo de electrones. Parte de la

carga que crea el punto de mayor potencial se trasladará a través del conductor al punto de menor potencial y, en ausencia de una fuente externa (generador), esta corriente cesará cuando ambos puntos igualen su potencial eléctrico. (MOLINA, 2012)

La diferencia de potencial, o el voltaje transversal, entre dos puntos cualesquiera de un sistema eléctrico queda determinado por:

$$v = \frac{W}{Q} \tag{1.2}$$

Dónde: V = diferencia de potencial en volts,

W = energía disipada o absorbida en joules (J),

Q = carga medida en coulombs.

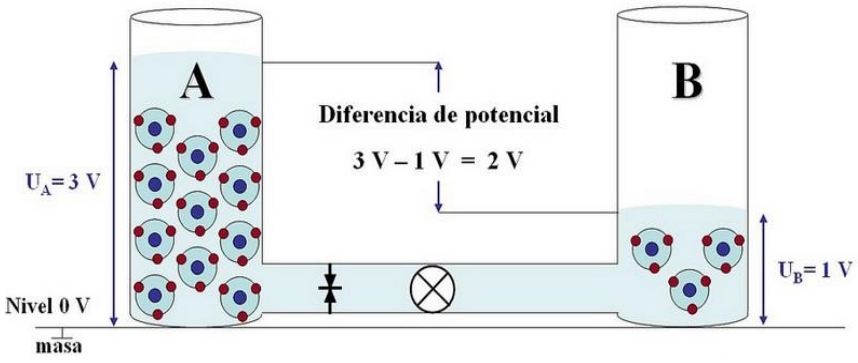


Figura 1.3: Diferencia de cargas eléctricas entre dos cuerpos A y B, referenciados a masa (0V)

1.3.3 POTENCIA ELÉCTRICA

La potencia eléctrica es la relación de paso de energía de un flujo por unidad de tiempo; es decir, la cantidad de energía entregada o absorbida por un elemento en un tiempo determinado. (ROBERT & LOUIS, 1997)

La potencia es la tasa temporal de gasto o de absorción de energía, por lo tanto, la energía y la potencia están relacionadas por la siguiente ecuación:

$$P = \frac{W}{t} \tag{1.3}$$

Dónde: P = potencia en watts,

W = energía en joules (J),

t = tiempo en segundos.

1.3.4 RESISTENCIA ELÉCTRICA

Los cuerpos aislantes de la electricidad se caracterizan por impedir el paso de la corriente eléctrica a través de ellos. Este fenómeno se debe a que los electrones se encuentran ligados fuertemente a sus átomos y para arrancarlos es necesario aplicar mucha energía, produciéndose una especie de rozamiento que se transforma en calor, estos choques son menores en los buenos conductores que en los malos, esta cualidad que tienen los materiales de denomina resistencia eléctrica. (ALCALDE, 2003)

Mientras mayor es la resistencia, menor es la corriente y viceversa, esto nos indica la ley más fundamental de los circuitos eléctricos: la *ley de Ohm.*

$$I = \frac{E}{R} \tag{1.4}$$

Dónde: I = Amperios (A),

E = Volts (V),

R = ohms (Ω).

1.4 INSTRUMENTACIÓN VIRTUAL

1.4.1 INTRODUCCIÓN

Un dispositivo virtual es un objeto que posee la capacidad de producir un efecto, sin estar presente físicamente. Es por eso que la denominada Instrumentación Virtual ha revolucionado el mercado de la Instrumentación principalmente porque no es necesario disponer físicamente de los Instrumentos para realizar aplicaciones. (VELÁSQUEZ, 2010)

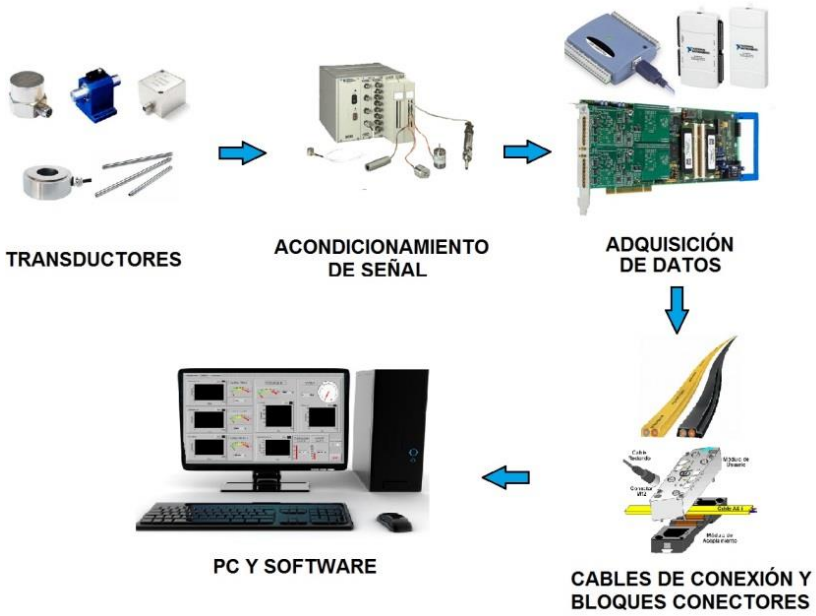


Figura 1.4: Diagrama de bloques de un Sistema basado en instrumentación virtual

En la Figura 1.4 se puede observar el diagrama de bloques de cómo está conformado un sistema basado en instrumentación virtual. El concepto de instrumentación virtual nace a partir del uso de la computadora personal, como forma de reemplazar equipos físicos por software, permite a los usuarios interactuar con la computadora como si estuviesen utilizando un instrumento real. El usuario manipula un instrumento que no es real, se ejecuta en una computadora, tiene sus características definidas por software pero realiza las mismas funciones que un equipo real. (DÍAZ & ANTONIO, 2006)

1.4.2 INSTRUMENTO VIRTUAL

La industria de la instrumentación está sufriendo importantes cambios como resultado de la revolución de los computadores personales. Estos cambios están ocurriendo tanto en el componente hardware como en el software.

Un instrumento virtual puede realizar las tres funciones básicas de un instrumento convencional: adquisición, análisis y presentación de datos. Sin embargo, el instrumento virtual permite personalizar el instrumento y agregarle mucha más funcionalidad sin incurrir en costos adicionales. (GONZALES, 2006)

Por tanto un instrumento virtual consta de un elemento hardware y otro software. El primero está constituido por una computadora personal, que realiza el procesamiento y visualización de los datos y un sistema específico, cuya función es la de adquirir los datos. Este último elemento, generalmente, está constituido por una placa de adquisición de datos, que se inserta en una ranura de expansión de la tarjeta principal de la computadora. El componente software consiste en un programa que se ejecute en la computadora y que controla los elementos hardware, procesa y visualiza los datos de tal modo que los usuarios pueden interactuar con el computador como si fuese un instrumento electrónico tradicional.

1.4.3 INSTRUMENTACIÓN VIRTUAL FRENTE A LA CONVENCIONAL

Cuando se compara la instrumentación virtual y la instrumentación convencional se compara generalmente el proceso de sensado. El proceso de medida o sensado consiste en la asignación de números a las propiedades de los objetos o acontecimientos del mundo real, de forma que se obtenga una descripción de los mismos.

Desde el punto de vista de la instrumentación tradicional o convencional un instrumento de sensado es un dispositivo capaz de recoger señales y campo y proporcionar medidas hacia un dispositivo controlador. Desde el punto de vista de la instrumentación virtual, la computadora mediante el hardware necesario recoge dichas señales de campo y las procesa; por tanto la computadora se convierte en el dispositivo para mediar y controlar un proceso.

Ventajas de la instrumentación virtual frente a la instrumentación convencional:

- La interfaz y el instrumento es definido por el usuario
- Funcionalidad ilimitada, orientado a aplicaciones, conectividad amplia.
- El software es la clave del sistema.
- Bajo costo/función, variedad de funciones.
- Es reusable y se puede aplicar tantas veces se requiera
- Arquitectura “abierta”
- Rápida incorporación de nuevas tecnologías, gracias a la plataforma PC.
- Altas economías de escala, bajos costos de mantenimiento
- Intercambio de información con otras aplicaciones de Windows, Linux u otro sistema operativo
- Múltiples capturas desde un solo punto, y posibilidad de envío a múltiples puntos locales o remotos

1.5 ADQUISICIÓN DE DATOS MEDIANTE MICROCONTROLADORES

Hoy en día el uso de microcontroladores para realizar adquisición de datos se ha vuelto un tema frecuente, debido a la fiabilidad que ha presentan en cuanto a resultados.

1.5.1 MICROCONTROLADORES ARM DE 32 BITS

La mayoría de los fabricantes de semiconductores ofrecen Microcontroladores

basados en tecnología ARM (Figura 1.5). Estos microcontroladores ofrecen una amplia gama de periféricos y una inmejorable relación precio-prestaciones. (ELECTRÓNICA, 2005)

La arquitectura ARM (Advanced RISC Machine) fue creada por Acorn Computer Group, como el primer procesador RISC con gran impacto comercial en el mundo. La filosofía RISC (Reduced Instruction Set Computer) busca realizar un proceso de la manera menos complicado posible. Ésta arquitectura es una alternativa muy importante en el diseño de procesadores, porque permite usar un número menor de instrucciones, alberga menos espacio de memoria y debido a su procesador de 32 bits puede desempeñar muchas más aplicaciones que otras arquitecturas.

La arquitectura ARM tiene un conjunto de instrucciones simple pero eficiente que permite un tamaño de silicio compacto y ofrece alta velocidad de ejecución a bajo consumo.

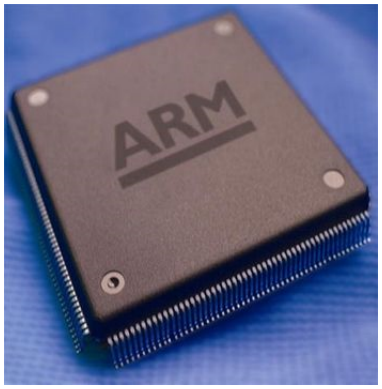


Figura 1.5: Microcontrolador ARM[1]

1.5.2 ARQUITECTURA ARM

La arquitectura ARM ha sido utilizada en numerosos diseños y aplicaciones

[1] http://tecnoholic.blogspot.com/2012_03_01_archive.html

específicas para productos estándar que pueden encontrarse actualmente en prácticamente todos los teléfonos móviles y la mayoría de los MP3, PDA's, cámaras y sistemas de Navegación. Además de ser utilizada en muchos productos de automoción y aplicaciones médicas e industriales.

Ésta arquitectura ha sido utilizada para el diseño de Microcontroladores estándar durante algún tiempo, en la Figura1.6 se muestra la estructura de un microcontrolador ARM. (ELECTRÓNICA, 2005)

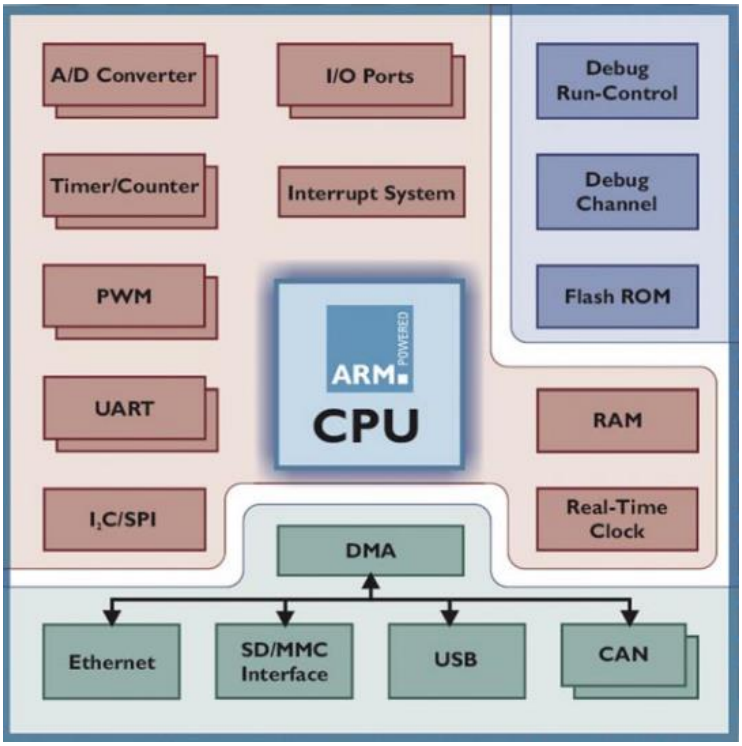


Figura 1.6: Estructura de un Microcontrolador ARM[2]

La arquitectura ARM es una implementación RISC con los siguientes comandos:

- Con los comandos originales ARM todas las instrucciones son 32-bit. La mayoría de las instrucciones pueden implementarse condicionalmente,

[2] http://www.redeweb.com/_txt/646/66.pdf

evitando saltos Branch asociados con las sentencias IT/THEN/ELSE.

- Con la introducción del núcleo ARM7TDMI se desarrolló el juego de instrucciones THUMB, de 16-bits. Las instrucciones Thumb son una abreviación de las instrucciones ARM 32-bits más frecuentemente utilizadas. Aunque se requiere un número ligeramente superior de instrucciones para escribir un programa, el tamaño del código se reduce en un 30-40% aproximadamente.
- TUMB2 complementa Thumb con la mayoría de las instrucciones ARM más populares y además permite ejecución condicional. Adicionalmente se ha optimizado el juego de instrucciones para compiladores de alto nivel (C/C++) y exigencias comunes en sistemas embebidos como la manipulación de bits y división de hardware, Thumb2 combina el comportamiento aritmético de ARM con la densidad de código del juego de instrucciones Thumb.

Algunos núcleos ARM incorporan un conjunto adicional de instrucciones para algoritmos DSP. Hay Microcontroladores que incluyen coprocesador VFP (Vector Floating Point), para acelerar la aritmética en punto flotante de acuerdo con el estándar IEEE754.

La continuidad del juego de instrucciones entre ambos conjuntos, ARM y THUMB, facilita una proposición única dentro de la industria del microcontrolador. Un usuario puede cualificar y reutilizar librerías de software validadas de un proyecto a otro en el mismo entorno. Esta reutilización puede reducir notoriamente los costes de mantenimiento a lo largo de múltiples proyectos.

En la Figura 1.7 se observa como está constituida la organización interna del procesador ARM

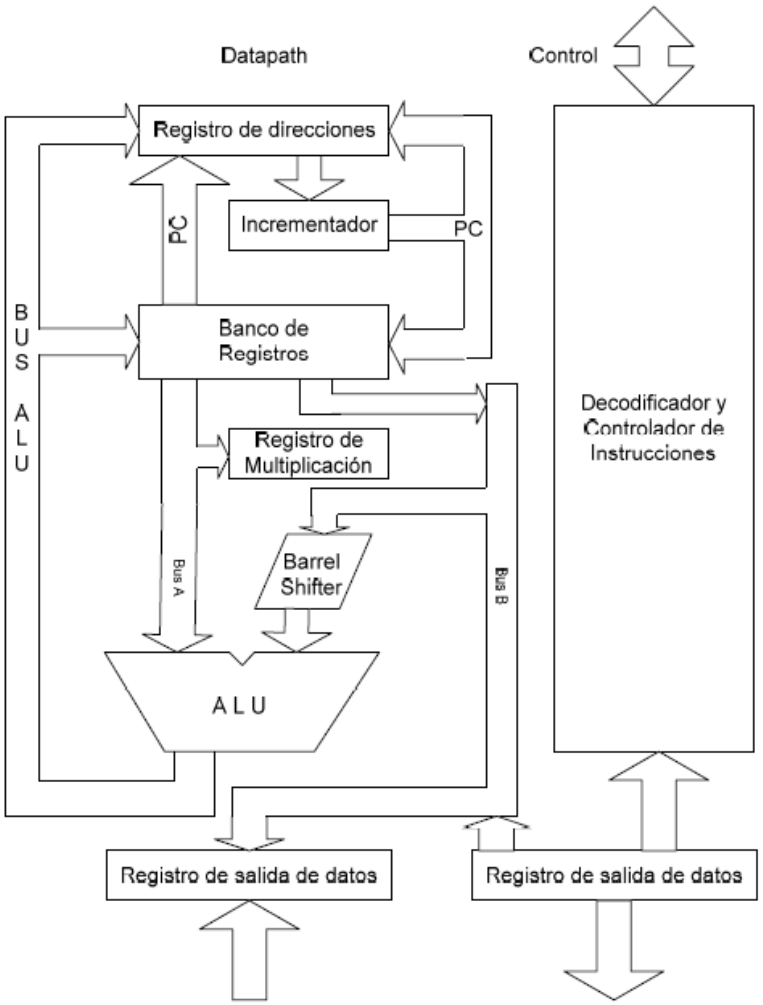


Figura 1.7: Organización interna del procesador ARM[3]

1.5.2.1 CARACTERÍSTICAS

Entre las características más relevantes de los microcontroladores ARM se tiene:

- Posee una arquitectura RISC de 32 bits
- Tiene un rendimiento sobre los 130 MIPs con un procesador típico de 0.13 um
- Diseño pequeño y tiene un consumo muy bajo de potencia

[3] http://www.redeweb.com/_txt/646/66.pdf

- Programación de alto nivel, comparable con los microcontroladores de 16 bits
- Amplio sistema operativo compatible con Windows CE, Palm OS, Symbian OS y Linux
- Posee un amplio número de herramientas de desarrollo para seleccionar
- Simulación acorde para los principales entornos de desarrollo EDA
- Excelente depuración de errores para los diseños SoC
- Migración y soporte a través de nuevos procesos tecnológicos
- Su código es compatible con los procesadores ARM9, ARM9E, ARM10

1.5.2.2 VENTAJAS

Estos microcontroladores presentan múltiples ventajas, respecto a los microcontroladores de 8 y 16 bits, que se encuentran comúnmente en el mercado, entre las ventajas más destacadas tenemos.

- Aumento de las prestaciones, permite realizar tareas más rápidas y de mejor calidad.
- Aumento de la complejidad de las aplicaciones, permiten diseñar interfaces gráficas, interfaces inalámbricas.
- Soportan distintos tipos de conectividad.
- Permiten la reducción de los costos, debido al uso masivo de este tipo de microcontroladores en diversas aplicaciones.

1.5.2.3 NÚCLEOS ARM PARA MICROCONTROLADORES

En la Tabla 1.1 se detalla algunos de los núcleos ARM para microcontroladores, con sus respectivas características.

Tabla 1.1: Núcleos ARM para Microcontroladores[4]

ARM Core	ISA Extensions	Memory Interface	CPU ClocK	Características
ARM7TDMI	ARM/Thumb	-	Hasta 100MHz	-
ARM720T	ARM/Thumb	MMU, Cache	Hasta 100 MHz	-
ARM920T	ARM/Thumb	MMU, Cache	Hasta 250 MHz	-
ARM926EJ-S	ARM/Thumb	MMU, Cache, TCM	Hasta 250 MHz	DSP Extensions + JAVA Engine
ARM946E-S	ARM/Thumb	MMU, Cache, TCM	Hasta 250 MHz	DSP Extensions
ARM966E-S	ARM/Thumb	TCM	Hasta 250 MHz	DSP Extensions
ARM1126J-S	ARM/Thumb	MMU, Cache		Float P. Co-Proc.
Cortex-M1	Thumb	---	Hasta 100 MHz	SofCore for FPGA
Cortex-M3	Thumb2	MPU	Hasta 150 MHz	CPU Core integrating InterruptControl Syst.
Cortex-R4(F)	ARM/Thumb2	MPU, TCM, Cache	Hasta 400 MHz	Interface for Dual-Core Safety Architect.

- *Cache:* Acelera la velocidad de acceso a memoria, almacenando la información accedida recientemente de memoria más lenta en RAM de acceso más rápido.
- *MMU (Memory Management Unit):* Permite memoria virtual y es generalmente requerida para plataformas con sistemas operativos como Windows CE o Linux.

[4] http://www.redeweb.com/_txt/646/66.pdf

- *MPU (Memory Protection Unit):* Administra el acceso a ciertas áreas de almacenamiento para aplicaciones de seguridad.
- *TCM (Tightly Coupled Memory):* La memoria (RAM, Flash) está conectada directamente con las CPU para ofrecer un acceso más rápido.

1.6 ARDUINO

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Se creó para artistas, diseñadores, aficionados y cualquiera interesado en crear entornos u objetos interactivos. (ARDUINO, 2010)

Arduino puede tomar información del entorno a través de sus pines de entrada de toda una gama de sensores y puede afectar aquello que le rodea controlando luces, motores y otros actuadores. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un ordenador, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software (p.ej. Flash, Processing, MaxMSP).

Las placas pueden ser hechas a mano o compradas montadas de fábrica; el software puede ser descargado de forma gratuita. Los ficheros de diseño de referencia (CAD) están disponibles bajo una licencia abierta, pudiendo adaptarse a cualquier necesidad. (HERRADOR ENRIQUEZ, 2009)

1.6.1 WIRING

Es un marco de programación de código abierto para los microcontroladores.

Permite la escritura de software multiplataforma para controlar los dispositivos conectados a una amplia gama de placas de microcontrolador para crear todo tipo de codificación creativa, objetos interactivos, espacios o experiencias físicas. (BARRIGA, 2013)

1.6.2 PROCESSING

Es un lenguaje de programación, entorno de desarrollo, y comunidad en línea, de código abierto para las personas que desean crear imágenes, animaciones e interacciones. Inicialmente creado para servir como un cuaderno de bocetos de software y para enseñar los fundamentos de programación de computadoras dentro de un contexto visual, "processing" evolucionó hasta convertirse en una herramienta de desarrollo para los profesionales. (BEN FRY, 2004)

1.6.3 ARDUINO DUE

El Arduino Due es una placa electrónica basada en el Atmel SAM3X8E CPU ARM Cortex-M3. Es la primera placa Arduino basado en un microcontrolador núcleo ARM de 32 bits. Lleva 54 entradas / salidas digitales (de las cuales 12 se puede utilizar como salidas PWM), 12 entradas analógicas y 4 puertos UART (puertos seriales), un reloj de 84 MHz, un puerto USB OTG, 2 conversores DAC (digital/analógico), 2 TWI, un conector de alimentación, un conector SPI, un

conector JTAG, un botón de reset y un botón de borrado, la Figura1.8 muestra la vista frontal de la tarjeta.

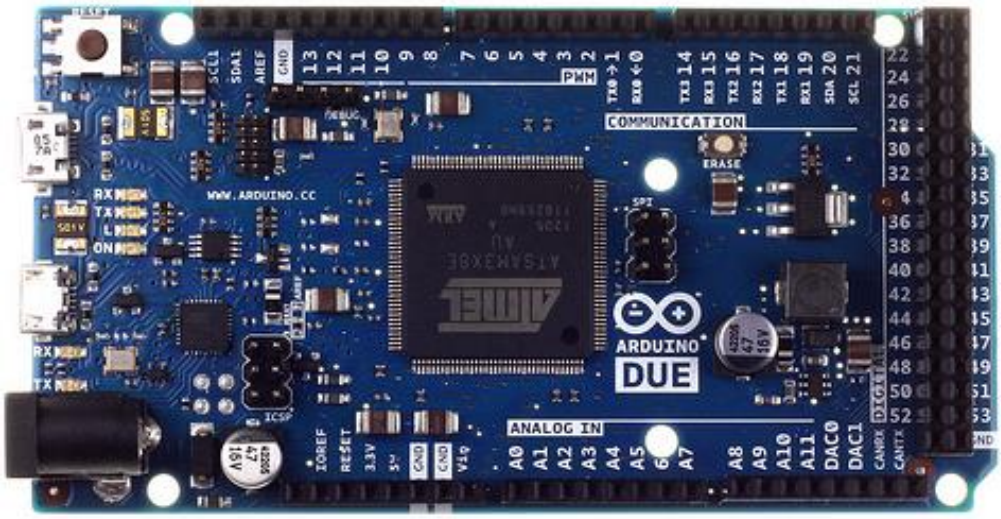


Figura 1.8: Tarjeta de entrenamiento Arduino Due[5]

1.6.3.1 BENEFICIOS DEL NÚCLEO ARM

La tarjeta Arduino Due tiene un núcleo de 32 bits que puede superar las típicas placas con microcontroladores de 8 bits. Las diferencias más significativas son:

- Un núcleo de 32 bits, que permite operaciones con datos de 4 bytes de ancho en un solo ciclo de reloj de la CPU
- Reloj de la CPU a 84Mhz
- 96 KBytes de SRAM
- 512 KBytes de memoria Flash para código
- Un controlador DMA, que puede aliviar a la CPU cuando se hacen tareas intensivas con memoria

[5] http://www.redeweb.com/_txt/646/66.pdf

1.6.3.2 CARACTERÍSTICAS DE LA TARJETA ARDUINO DUE

La Tabla 1.2 muestra las características principales de la tarjeta Arduino Due

Tabla 1.2: Características de la Tarjeta Arduino Due

Microcontroladores	AT91SAM3X8E
Voltaje de operación	3.3V
Voltaje de entrada	7-12 V
Límites de voltaje	6-16 V
Pines I/O digital	54 (de los cuales 12 proporcionan salida PWM)
Entradas analógicas	12
Salidas analógicas	2 (DAC)
Salida máxima I/O	130 mA
Corriente máx. por pin 3.3V	800 mA
Corriente máx. por pin 5V	800 mA
Memoria Flash	512 KB disponibles para las aplicaciones de usuario
SRAM	96 KB (64+32 KB)
Frecuencia de Reloj	84 MHz

El SAM3X tiene 512KB de memoria en dos bloques de 256KB para almacenar código. El bootloader viene precargado de ATMEL y está almacenado en la memoria dedicada ROM. La SRAM disponible es de 96Kb y dos bloques de 64 KB y 32 KB. Toda la memoria disponible (Flash, RAM y ROM) puede accederse directamente como una dirección de memoria plana.

1.7 ESTANDAR BLUETOOTH (IEEE 802.15.1)

1.7.1 ORIGEN TECNOLOGÍA BLUETOOTH

Bluetooth empieza a concebirse en Ericsson Mobile Communication AB (Suecia) en 1994 como el efecto colateral de un proyecto sobre enlaces de comunicadores múltiples conectados a la red celular mediante teléfonos.

Es un estándar abierto para conectividad inalámbrica con el principal apoyo de las PC y de las industrias de la telefonía celular, originalmente fue creada con el fin de definir una especificación global y abierta para la comunicación inalámbrica de voz y datos a corta distancia utilizando radiofrecuencia, un modelo de conectividad de Bluetooth se puede observar en la Figura 1.9. Ésta tecnología inalámbrica es un estándar de comunicaciones cuyo objetivo es, básicamente, eliminar los cables entre dispositivos móviles y PCs. (SPARACINO, 2003)

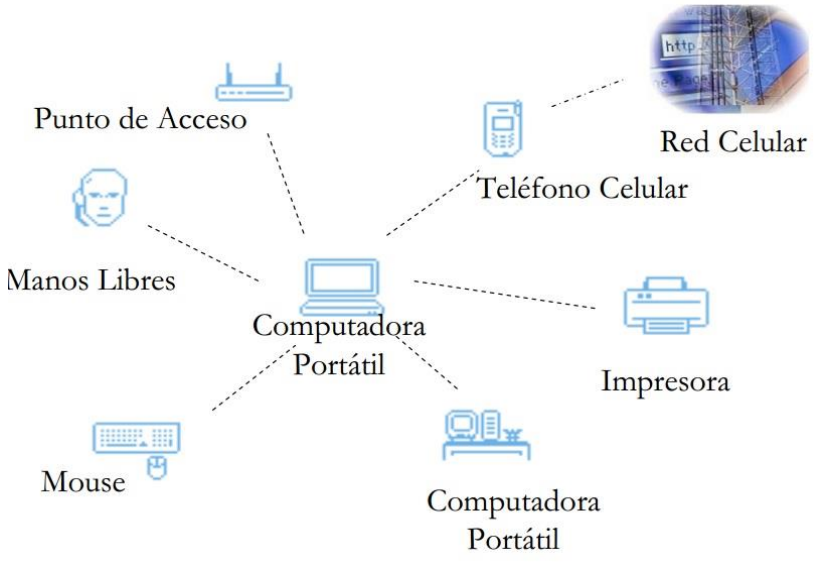


Figura 1.9: Modelo de conectividad Bluetooth

1.7.2 FUNCIONAMIENTO

Trabaja en dos capas del modelo OSI que son la de enlace y aplicación, incluye un transceiver que trasmite y recibe a una frecuencia de 2.4 Ghz. Las conexiones que se realizan son de uno a uno con un rango máximo de 10 metros, si se deseara implementar la distancia se tendría que utilizar repetidores los cuales nos ayudarían a abarcar una distancia de 100 metros.

Bluetooth por cuestiones de seguridad cuanta con mecanismos de encriptación de 64 bits y autentificación para controlar la conexión y evitar que dispositivos puedan acceder a los datos o realizar su modificación.

Durante la transferencia de datos el canal de comunicaciones permanece abierto y no requiere la intervención directa del usuario cada vez que se desea transferir voz o datos de un dispositivo a otro. La velocidad máxima que se alcanza durante la transferencia es de 700 kb/seg y consume un 97% menos que un teléfono móvil.

En la actualidad existen varias tecnologías de corto alcance, en la Tabla 1.3 se muestra una comparación de las características principales de la tecnología Bluetooth sobre las demás.

Tabla 1.3: Características de tecnologías de corto enlace[6]

Tecnología	Frecuencia	Tasa de Tx	Alcance	Seguridad	Aplicación
ANT+	2.4GHz	1Mbps	< 10 m	-	Monitoreo: Salud y Deportes
Bluetooth	2.4GHz	1 - 54Mbps	10 – 100 m	Alta con encriptación	Aplicaciones de Audio
Infrared (IrDA)	800 – 1000um	9600 bps	1 m	-	Mando a distancia
NFC (Near field communication)	13.56 MHz	106 – 848 kbps	< 30 cm	Alta	Acceso a pagos
RFID (Radio Frequency	125 KHz, 13.56	424 kbps	< 1 m	>Vulnerable	Rastreo, inventarios

[6] http://electronicdesign.com/communications/fundamentals-short-range-wireless-technology

IDentification)	MHz, 902 a 928 MHz				
6LoWPAM (IPv6 over Low power Wireless Personal Area Networks)	2.4GHz	-	< 10 m	Alta	Monitoreo y control vía Internet
UWB (Ultrawideband)	3.1 - 10.6 GHz	53 - 480 Mbps	< 100 m	Alta	Transferencia de video
Wi-Fi	2.5 – 5GHz	54 – 300 Mbps	< 100 m	Alta	Redes locales, acceso a internet
ZigBee	2.4 GHz	250 Kbps	< 10 m	Mediante autentificación y encriptación	Monitoreo y control del hogar (Domótica), e Industria
Z-Ware	908.42 MHz	9600 bps – 40 Kbps	< 30 m	-	Monitoreo y control del hogar (Domótica)

Cada una de estas tecnologías fueron creadas con el propósito de cubrir las necesidades de otras, algunas se desarrollaron para aplicaciones específicas, sin embargo otras son flexibles y genéricas.

1.7.3 TOPOLOGÍA BLUETOOTH

Cuando un dispositivo Bluetooth está dentro del radio de cobertura de otro, puede establecer un enlace entre ellos. Hasta ocho unidades Bluetooth pueden comunicarse entre ellas y formar una Piconet. La unión de varias Piconet se denomina Scatternet o Red Dispersa (Figura 1.10).

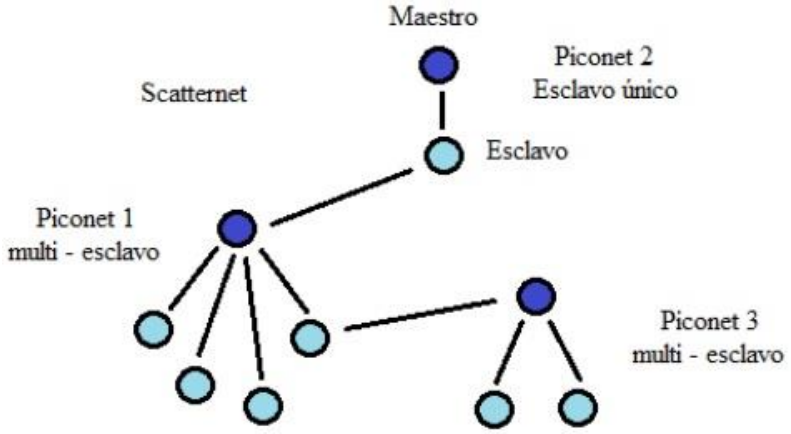


Figura 1.10: Red Dispersa[7]

Los dispositivos que están dentro de una piconet pueden ser: maestro o esclavo. En una piconet solo puede haber un maestro, quien normalmente inicia la conexión, el resto de unidades son los esclavos. Cualquier dispositivo puede hacer las funciones de maestro y esclavo. (SPARACINO, 2003)

El maestro es el dispositivo de una piconet cuyo reloj y patrón de saltos se utilizan para sincronizar a todos los dispositivos esclavos.

La topología Bluetooth se puede describir como una estructura de piconets múltiples. Dado que la especificación Bluetooth soporta conexiones punto a punto y punto a multipunto, se pueden establecer y enlazar varias piconets en forma de Scatternet.

1.7.4 ARQUITECTURA DEL HARDWARE

El hardware que compone el dispositivo Bluetooth está compuesto por dos partes. Un dispositivo de radio, encargado de modular y transmitir la señal; y un

[7] http://campus.almagro.ort.edu.ar/informatica/2008/bm5b/articulo/2563/grupo-1-bluetooth-por-maii-

controlador digital. El controlador digital está compuesto por un CPU, por un procesador de señales (DSP – Digital Signal Processor) llamado Link Controller (controlador de Enlace) y de los interfaces con el dispositivo anfitrión.

El LC o Link Controller está encargado de hacer el procesamiento de la banda base y del manejo de los protocolos para corrección de errores de la capa física. Además, se encarga de las funciones de transferencia (tanto asíncrona como síncrona), codificación de Audio y encripción de datos.

El CPU del dispositivo se encarga de atender las instrucciones relacionadas con Bluetooth del dispositivo anfitrión, para así simplificar su operación. Para ello corre un software denominado Link Manager que tiene la función de comunicarse con otros dispositivos por medio del protocolo LMP (Link Management Protocol).

En la Figura 1.11 se muestra la arquitectura del hardware del estándar Bluetooth.

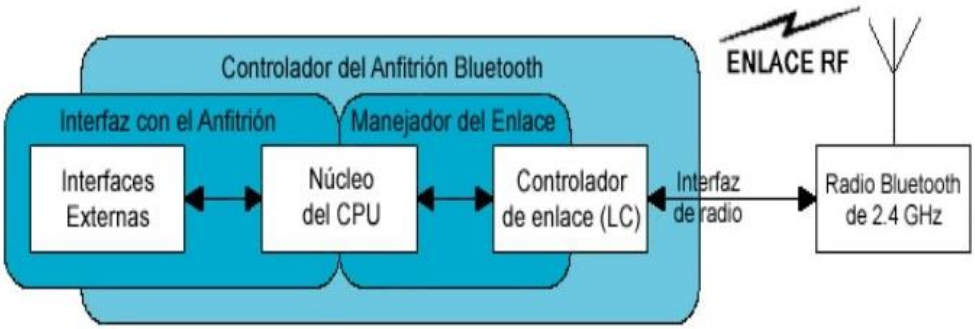


Figura 1.11: Arquitectura Hardware Bluetooth[8]

1.7.5 ARQUITECTURA DE SOFTWARE

Buscando ampliar la compatibilidad de los dispositivos Bluetooth, los dispositivos que se apeguen al estándar utilizan como interfaz entre el dispositivo anfitrión (laptop, teléfono celular, etc.) y el dispositivo Bluetooth como tal (chip Bluetooth)

[8] http://artemisa.unicauca.edu.co/~dabravo/bluetooth/quees.htm

una interfaz denominada HCI (Host Controller Interface). En la Figura 1.12 se puede apreciar la arquitectura de software del estándar Bluetooth.

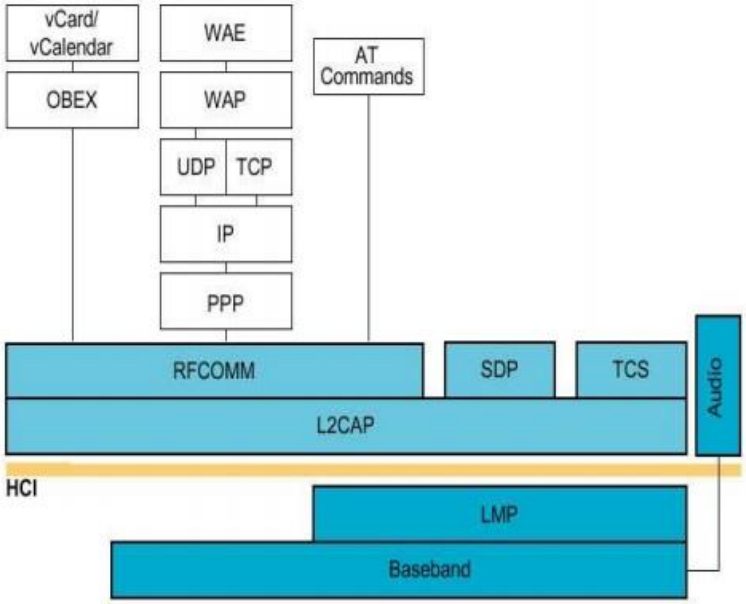


Figura 1.12: Arquitectura Software Bluetooth[9]

Los protocolos de alto nivel como el SDP (Protocolo utilizado para encontrar otros dispositivos Bluetooth dentro del rango de comunicación, encargado, también, de detectar la función de los dispositivos en rango), RFCOMM (Protocolo utilizado para emular conexiones de puerto serial) y TCS (Protocolo de control telefonía) interactúan con el controlador de banda base a través del Protocolo L2CAP (Logical Link Control and Adaptation Protocol). El protocolo L2CAP se encarga de la segmentación y reensamblaje de los paquetes para poder enviar paquetes de mayor tamaño a través de la conexión Bluetooth.

[9] http://artemisa.unicauca.edu.co/~dabravo/bluetooth/quees.htm

1.7.6 VENTAJAS

Una de sus principales ventajas sobre otros sistemas de comunicaciones inalámbricas es su radio de acción, bastante superior al de los dispositivos que funcionan por infrarrojos, permitiendo la comunicación incluso cuando los aparatos están separados por objetos o paredes. (MULLER, 2002)

Además, de que el estándar construye áreas personales de comunicaciones inalámbricas de forma fácil y sencilla, tan simple como prender una luz, mediante Bluetooth, todos los dispositivos digitales se comunicarán de forma espontánea, para ofrecer tres importantes ventajas:

- *Puntos de Acceso de Voz y Datos:* la tecnología Bluetooth simplifica el acceso a otras redes. Mediante una conexión Bluetooth se pueden identificar diferentes redes y conectarse a ellas. Por ejemplo, un usuario puede conectarse fácil y rápidamente a Internet a través de su teléfono celular y mediante cualquier dispositivo destinado a las conexiones por cable que sea compatible con Bluetooth.
- *Sustituto de todo tipo de Cables:* elimina la necesidad de los molestos cables anexos a cualquier aparato.
- *Redes personales específicas:* Todos los dispositivos que soportan el estándar pueden establecer intercambios de información de forma automática y funcionar simultáneamente con cualquier otro aparato.

1.7.7 ESPECIFICACIONES

Los dispositivos que incorporan esta tecnología pueden comunicarse entre ellos, y no hace falta que estén alineados, ni en la misma habitación siempre que se encuentre dentro de su alcance. Se clasifican en función de su potencia de

transmisión, siendo totalmente compatibles entre los diferentes tipos, la clasificación se muestra en la Tabla 1.4

Tabla 1.4: Clases Bluetooth[10]

Clase	Potencia Máxima Permitida (mW)	Potencia Máxima Permitida (dBm)	Alcance (Aproximado)
Clase1	100 mW	20 dBm	100 m
Clase2	2.5 mW	4 dBm	10 m
Clase3	1 mW	0 dBm	1 m

En la mayoría de los casos, la cobertura efectiva de un dispositivo de clase 2 se extiende cuando se conecta a un transceptor de clase 1. Lo que es debido a la mayor sensibilidad y potencia de transmisión del dispositivo de clase 1, es decir, la mayor potencia de transmisión del dispositivo de clase 1 permite que la señal llegue con energía suficiente hasta el de clase 2. Por otra parte al tener el dispositivo de clase 1 mayor sensibilidad permite recibir la señal del otro aunque esta sea más débil.

También los podemos clasificar en función de su ancho de banda como se muestra en la Tabla 1.5

Tabla 1.5: Ancho de Banda según las versiones Bluetooth

Versión	Ancho de banda
Versión 1.2	1 Mbit/s
Versión 2.0 + EDR	3 Mbit/s
Versión 3.0 + HS	24 Mbit/s
Versión 4.0	24 Mbit/s

[10] http://www.ecured.cu/index.php/Bluetooth

1.8 DESCRIPCIÓN DE LA PLATAFORMA ANDROID

1.8.1 INTRODUCCIÓN

Android Fue desarrollado por "Android Inc.", empresa que en julio de 2005 fue comprada por Google, aunque no fue hasta 2008 cuando se popularizó, gracias a la unión al proyecto de "Open Handset Alliance", un consorcio creado a finales de 2007 liderado por google con 34 miembros más, incluyendo fabricantes de dispositivos móviles, desarrolladores de aplicaciones, hardware, software y telecomunicaciones, que decidieron promocionar el software libre, google pretende dejar que las empresas fabricantes pertenecientes a la alianza, entre las que se encuentra a HTC, LG, Nvidia, Texas instruments o Samsung, fabriquen los terminales que incorporaren este nuevo sistema operativo.

Es así como nace Android, un sistema operativo y una plataforma software, basado en Linux para teléfonos móviles, tablet's, así como también netbooks, reproductores de música e incluso PC's, aunque en estos elementos aún, es poco habitual Android permite programar en un entorno de trabajo (framework) de Java, aplicaciones sobre una máquina virtual Dalvik (una variación de la máquina de Java con compilación en tiempo de ejecución). Además, lo que le diferencia de otros sistemas operativos, es que cualquier persona que sepa programar puede crear nuevas aplicaciones, widgets, o incluso, modificar el propio sistema operativo, dado que Android es de código libre, por lo que sabiendo programar en lenguaje Java, va a ser muy fácil comenzar a programar en esta plataforma.

Google tiene una estrategia muy bien definida quiere dominar el mercado del software, no del hardware. A sus múltiples aplicaciones gratuitas orientadas a Internet se van sumando otras nuevas destinadas a hacer la competencia a los tradicionales desarrolladores de aplicaciones informáticas y ahora entra con la misma fórmula al mercado de los teléfonos y smartphone's. (SORIANO, 2012)

Figura 1.13: Logo principal de Android[11]

1.8.2 ARQUITECTURA

Dado que Android está basado en el núcleo de Linux, tiene acceso a sus recursos, pudiendo gestionarlo, gracias a que se encuentra en una capa por encima del Kernel, accediendo así a recursos como los controladores de pantalla, cámara, memoria flash, etc.

[11] http://www.android.com/

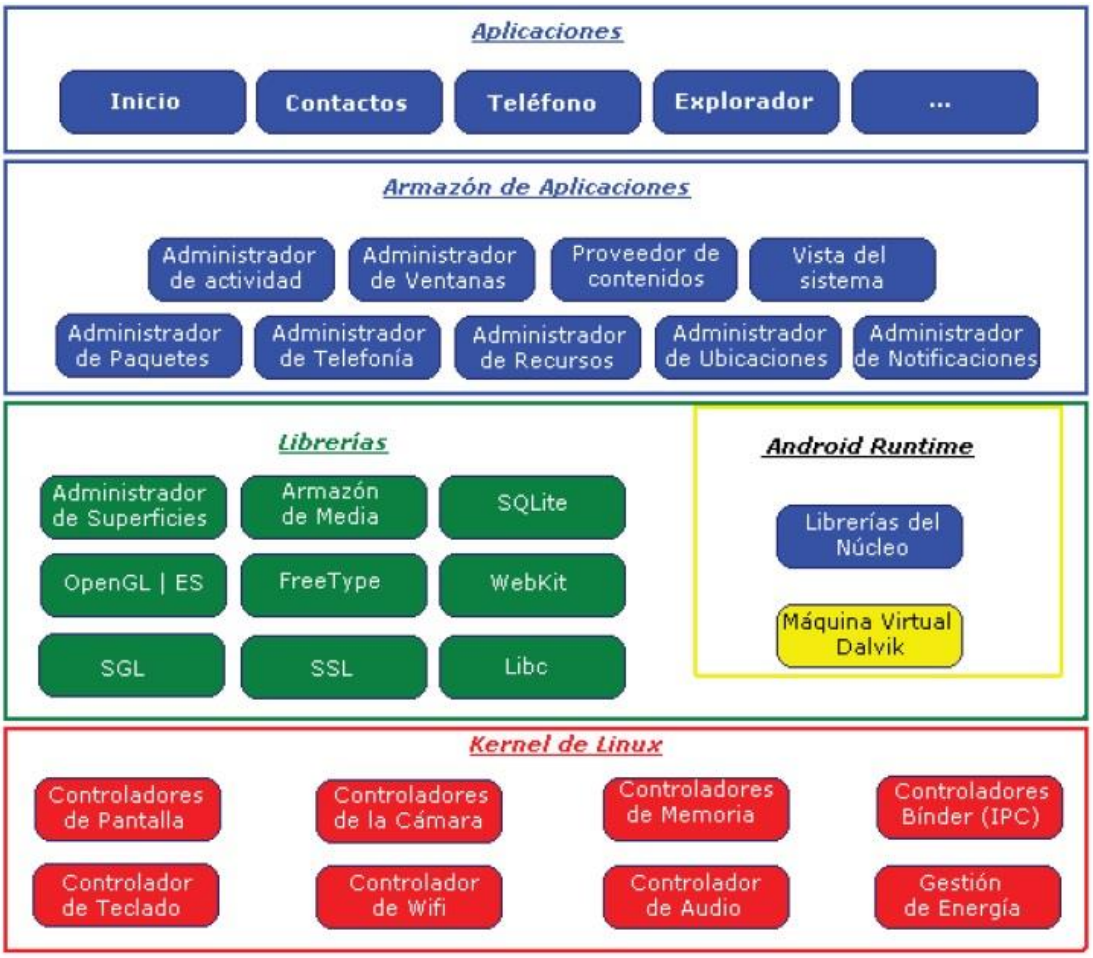


Figura 1.14: Sistema de capas de Android

En la Figura 1.14, se muestran las capas que conforman el sistema operativo Android, las que forman parte del propio Kernel de Linux, donde Android puede acceder a diferentes controladores; las librerías creadas para el desarrollo de aplicaciones Android, la capa de organización de los diferentes administradores de recursos, la capa de aplicaciones a las que tiene acceso. (MANUEL BÁEZ).

Android está formado por una capa de aplicaciones bien definida, donde corren todos los programas de alto nivel, amigable con las personas, siendo cada vez más intuitiva con el ser humano.

Luego tiene un soporte de aplicaciones, que sería el menú del teléfono propiamente dicho, siendo éste un software de alto nivel más, que maneja a los

otros programas, algunos objetivos de esta capa serían por ejemplo: ¿cómo elegir entre un software y otro?, ¿cuál elegir para ejecutar?, ¿qué opciones se podrían aplicar a las aplicaciones preferidas?, ¿cuándo haya baja o alta señal tener distintas opciones?, etc.

Debajo del menú o armazón de aplicaciones se encuentran las librerías que sirven a las aplicaciones para que se puedan ejecutar en conjunto con la máquina virtual Delvik que se encarga se abstraer el hardware y no tener que preocuparse mucho por lo que pasa debajo de la línea de software puro.

A parte se conectan con el manejador del núcleo de Linux para tener una coherencia con la capa subyacente que sería el Kernel de Linux, que ya es de bajo nivel y se conecta directamente con el hardware del teléfono, como se muestra en la Figura 1.15 los distintos drivers existentes en el Handset, la batería, la pantalla, el teclado, etc.

Las capas del sistema operativo Android determinan su capacidad y funcionalidad en un dispositivo móvil, las aplicaciones se encuentran basadas en un interfaz de programación de aplicaciones API (Application Programming Interface).

1.8.3 CARACTERÍSTICAS PRINCIPALES

Se trata de un sistema operativo abierto multitarea, permite a los desarrolladores acceder a todas las funcionalidades del dispositivo.

Todas las aplicaciones son reemplazables, el sistema operativo no diferencia entre las aplicaciones básicas del teléfono y las aplicaciones de terceros. Cualquier aplicación puede ser reemplazada libremente, incluso las que trae por defecto el sistema operativo como el navegador Web integrado basado en el motor *WebKit.*

Las características principales del sistema operativo Android se describen con detalles en la Tabla 1.6.

Tabla 1.6: Características principales plataforma Android

APLICACIÓN	CARACTERÍSTICAS
Diseño del Dispositivo	La plataforma es adaptable a pantallas más grandes, VGA, biblioteca de gráficos 2D, biblioteca de gráficos 3D basada en las especificaciones de la OpenGL ES 2.0 y diseño de teléfonos tradicionales.
Almacenamiento	SQLite, una base de datos liviana, que es usada para propósitos de almacenamiento de datos.
Conectividad	Android soporta las siguientes tecnologías de conectividad: GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi y Wimax.
Mensajería	SMS y MMS son formas de mensajería, incluyendo mensajería de texto y ahora la Android Cloud toDeviceMessaging Framework (C2DM) es parte del servicio de PushMessaging de Android.
Navegador Web	El navegador web incluido en Android está basado en el motor de renderizado de código abierto WebKit, emparejado con el motor JavaScript V8 de Google Chrome. El navegador obtiene una puntuación de 93/100 en el test Acid3.
Soporte Java	Aunque la mayoría de las aplicaciones están escritas en Java, no hay una máquina virtual Java en la plataforma. El bytecode Java no es ejecutado, sino que primero se compila en un ejecutable Dalvik y corre en la Máquina Virtual Dalvik que es una máquina virtual especializada, diseñada específicamente para Android y optimizada para dispositivos móviles que funcionan con batería y que tienen memoria y procesador limitados.
Soporte Multimedia	Android soporta los siguientes formatos multimedia: WebM, H.263, H.264,(en 3GP o MP4), MPEG-4 SP, AMR,AMR-WB, ACC, HE-ACC, MP3, MIDI, OggVorbis, WAV, JPEG, PNG, GIF y BMP
Soporte para streaming	Streaming RTP/RTSP (3GPP PSS, ISMA), descarga progresiva de HTML (HTML5 <video>tag). Adobe Flash Streaming (RTMP) es soportado mediante el Adobe Flash Player. Se planea el soporte de Microsoft SmoothStreaming con el port de Silverlight a Android. Adobe Flash HTTP DynamicStreaming estará disponible mediante una actualización de Adobe Flash Player
Soporte para Hardware adicional	Android soporta cámaras de fotos, de vídeo, pantallas táctiles, GPS, acelerómetros, giroscopios, magnetómetros, sensores de proximidad y de presión, termómetro, aceleración 2D y 3D.
Entorno de desarrollo	Incluye un emulador de dispositivos, herramientas para depuración de memoria y análisis del rendimiento del software. El entorno de desarrollo integrado es Eclipse.

Market	El Android Market es un catálogo de aplicaciones gratuitas o de pago en el que pueden ser descargadas e instaladas en dispositivos Android sin la necesidad de un PC.
Multi-táctil	Android tiene soporte nativo para pantallas multi-táctiles que inicialmente hicieron su aparición en dispositivos como el HTC Hero. La funcionalidad fue originalmente desactivada a nivel de kernel (posiblemente para evitar infringir patentes de otras compañías). Más tarde, Google publicó una actualización para el NexusOne y el Motorola Droid que activa el soporte para pantallas multi-táctiles de forma nativa.
Bluetooth	El soporte para A2DP y AVRCP fue agregado en la versión 1.5; el envío de archivos (OPP "Object Push Profile"") y la exploración del directorio telefónico fueron agregados en la versión 2.0; y el marcado por voz junto con el envío de contactos entre teléfonos lo fueron en la versión 2.2.
Videollamada	Android soporta videollamada a través de Google Talk desde su versión HoneyComb.
Multitarea	Multitarea real de aplicaciones está disponible, es decir, las aplicaciones que no estén ejecutándose en primer plano reciben ciclos de reloj, a diferencia de otros sistemas de la competencia en la que la multitarea es congelada.

1.8.4 INTERFAZ DE PROGRAMACIÓN DE APLICACIONES (API)

Es el conjunto de funciones, procedimientos o métodos, que en la programación orientada a objetos, ofrece la posibilidad de abstracción de muchas líneas de comandos en una sola biblioteca.

El API es utilizado por otro software de programación, como una capa de abstracción entre el lenguaje de programación y el lenguaje propio del sistema operativo, generalmente denominadas "librerías".

Nivel API es un valor entero que identifica de forma exclusiva la revisión del Framework de la API que ofrece una versión de la plataforma Android. Nivel de API permite al sistema determinar correctamente si una aplicación es compatible con el sistema, antes de instalar la aplicación.

Las API abren distintos tipos de diálogos con el proveedor para obtener o actualizar información en el mismo, entre ellos:

- Acceso a bases de datos
- Comunicación cliente/servidor
- Comunicación peer-to-peer
- Comunicación en tiempo real
- Procesamiento de transacciones

La Tabla 1.7 especifica el nivel de API con el apoyo de cada versión de la plataforma Android:

Tabla 1.7: Nivel API en función de la plataforma Android

Versión de la plataforma Android	Nivel API	Código de la Versión
1.0	1	BASE
1.1	2	BASE_1_1
1.5x	3	CUPCAKE
1.6x	4	DONUT
2.0	5	ECLAIR
2.0.1	6	ECLAIR_0_1
2.1.x	7	ECLAIR_MR1
2.2.x	8	FROYO
2.3 **2.3.1** **2.3.2**	9	GINGERBREAD
2.3.3 **2.3.4**	10	GINGERBREAD_MR1
3.0.x	11	HONEYCOMB
3.1.x	12	HONEYCOMB_MR1
3.2	13	HONEYCOMB_MR2
4.0 **4.0.1** **4.0.2**	14	ICE_CREAM_SANDWICH
4.0.3	15	ICE_CREAM_SANDWICH_MR1
4.1	16	JELLY BEAN
4.2	17	JELLY BEAN_MR1
4.3	18	JELLY BEAN_MR2
4.4	19	KITKAT

1.8.5 VERSIONES DEL SISTEMA OPERATIVO ANDROID

Android desde su liberación inicial ha visto numerosas actualizaciones, las mismas que al sistema operativo básicamente arreglan bugs y agregan nuevas funciones. Generalmente cada actualización del sistema operativo Android es desarrollada bajo un nombre en código de un elemento relacionado con postres desde la versión 1.5 (MINERVA, 2013)

Los siguientes nombres código del sistema operativo Android que se muestran en la Tabla 1.8 están en orden cronológico y alfabético:

Tabla 1.8: Versiones del sistema operativo Android

VERSIONES / NOMBRE	FECHA DE LIBERACION	LOGO
1.0 (Aple pie / Tarta de Manzana)	-Liberado el 23 de septiembre de 2008 -Android Market de descarga de aplicaciones y actualizaciones a través de la aplicación de mercado. -Google Calendar. -El 9 de febrero de 2009se presento una versión mejorada v1.1	
1.5 (Cupcake / Magdalena Glaseada) Basado en el kernel de Linux 2.6.27	Fue liberado el 30 de abril de 2009 -Posibilidad de grabar y reproducir videos a través del modo camcorder. -Un nuevo teclado con predicción de texto. -Soporte para Bluetooth A2DP y AVRCP.	
1.6 (Donut / Rosquilla) Basado en el kernel de Linux 2.6.29	Fue liberado el 15 de septiembre de 2009 -Interfaz integrada de cámara, filmadora y galería, soporte para resoluciones de pantalla WVGA -Navegación gratuita turn-by-turn de Googl	

2.0 / 2.1 (Eclair / pastel francés Basado en el kernel de Linux 2.6.29	Fue liberado el 26 de octubre de 2009 -Nuevo interfaz de usuario en el navegador y soporte para HTML5 -Mejoras en Google Maps 3.1.2 -Soporte para Microsoft Exchange. -MotionEent mejorado para captura de eventos multitouch. -Teclado virtual mejorado. -Bluetooth 2.1	
2.2 (Froyo /abreviatura de "frozen yogurt" yogurt helado) Basado en el kernel de Linux 2.6.32	Fue liberado el 20 de mayo de 2010 -Mejoras en la velocidad de las aplicaciones, gracias a la implementación de JIT. -Integración del motor JavaScript V8 -Soporte mejorado de Microsoft Exchange. -Funcionalidad de Wi-Fi hotspot y tethering por USB -Permite desactivar el tráfico de datos a través de la red del operador	
2.3 (Gingerbread / pan de jenjibre) Basado en el kernel de Linux 2.6.35.7 Actual en Smart	-Fue liberado el 6 de diciembre de 2010 -Soporte nativo para telefonía VoIP SIP -Soporte para reproducción de videos WebM/VP8 y decodificación de audio AAC -Soporte para Near Field Communication. -Teclado multi-táctil rediseñado. -Soporte mejorado para desarrollo de código nativo.	
3.0 / 3.1 / 3.2 (Honeycomb / Panal de miel)	-Fue liberado el 22 de febrero de 2011 -Escritorio 3D con widgets rediseñados - Sistema multitarea mejorado -Mejoras en el navegador web predeterminado. -Soporte para video chat mediante Google Talk. -Los widgets pueden redimensionarse de forma manual.	

4.0 (Ice CreamSandwic H / sandwith de helado)	-Fue liberado el 19 de octubre de 2011 -Opción de utilizar los botones virtuales en la interfaz de usuario, en lugar de los botones táctiles capacitivos. -Añade la posibilidad de finalizar una tarea simplemente desplazándola fuera de la lista. -AndroidBeam es la nueva característica que nos permitirá compartir contenido entre teléfonos. Vía NFC (Near Field Communication). -Un único y nuevo framework para las aplicaciones. -Soporte nativo para el uso de Stylus (lápiz táctil).	
4.1 / 4.1.2 / 4.2 / 4.2.1 (Jelly Bean / Judia)	Fue liberado el 9 de julio del 2012. -Una de las mejoras más importantes es el llamado Proyecto Mantequilla. Se trata de un esfuerzo por mejorar dramáticamente la experiencia de usuario por medio de una interfaz más responsiva. Las animaciones son más fluidas y la respuesta al tacto es notoriamente más veloz, también existen mejores en cuanto al rediseño de la pantalla principal.	
4.4 (KitKat)	Mejora su rendimiento para que funcione en los dispositivos de menos prestaciones, añade importantes nuevas funcionalidades tanto para el usuario como para los desarrolladores. -Solución de la fragmentación entre versiones, solamente requiere para su funcionamiento 512 MB de memoria RAM. -Brinda apoyo para las transacciones seguras a través de NFC, basadas en el Emulación Card Host (HCE). -Es una nueva forma de crear tutoriales, ensayos de materiales, vídeos de marketing, etc.	

1.8.6 COMPARACIÓN ENTRE SISTEMAS OPERATIVOS MÓVILES

En la tabla 1.9 se muestra una breve comparación entre los sistemas operativos para smartphone's con sus principales características:

Tabla 1.9: Comparativa de sistemas operativos para Smartphone's

	Android	**Blackberry OS**	**IOS**	**Windows Phone**	**Firefox OS**
Familia del SO	Linux	Mobile OS	Darwin	Windows NT	Linux
Arquitectura Soportada	ARM, MIPS, X86	ARM	ARM	ARM	ARM
Programado en:	C, C++, Java	C, C++	C, C++, Objetive-C	XNA, .NET, C, C++, VB.NET	C++, JavaScript, $HTML_{5,}$ CSS
Servicio Remoto de localización GPS	No de serie.	No de serie.	SI	SI	NO Disponib.
Tipo de interfaz	Iconos y widgets	Baldosas e iconos	Iconos	Baldosas Animadas	Iconos
Sincronización con la nube	Google Drive	No de serie.	iCLoud	SkyDrive	SI
Tienda de aplicaciones	Google Play	App World.	App Store	Windows Phone Store	Firefox Market Place
Mensajería	Google Talk	BBM	iMessage	Skype	No de serie

CAPÍTULO 2

ANÁLISIS Y DISEÑO DE INSTRUMENTOS VIRTUALES

2.1 DISEÑO DEL HARDWARE

2.1.1 HARDWARE DEL INSTRUMENTO VIRTUAL

El hardware implementado para el proyecto se lo ha dividido en varias partes. En la Figura 2.1 se muestra el diagrama de bloques del hardware del proyecto.

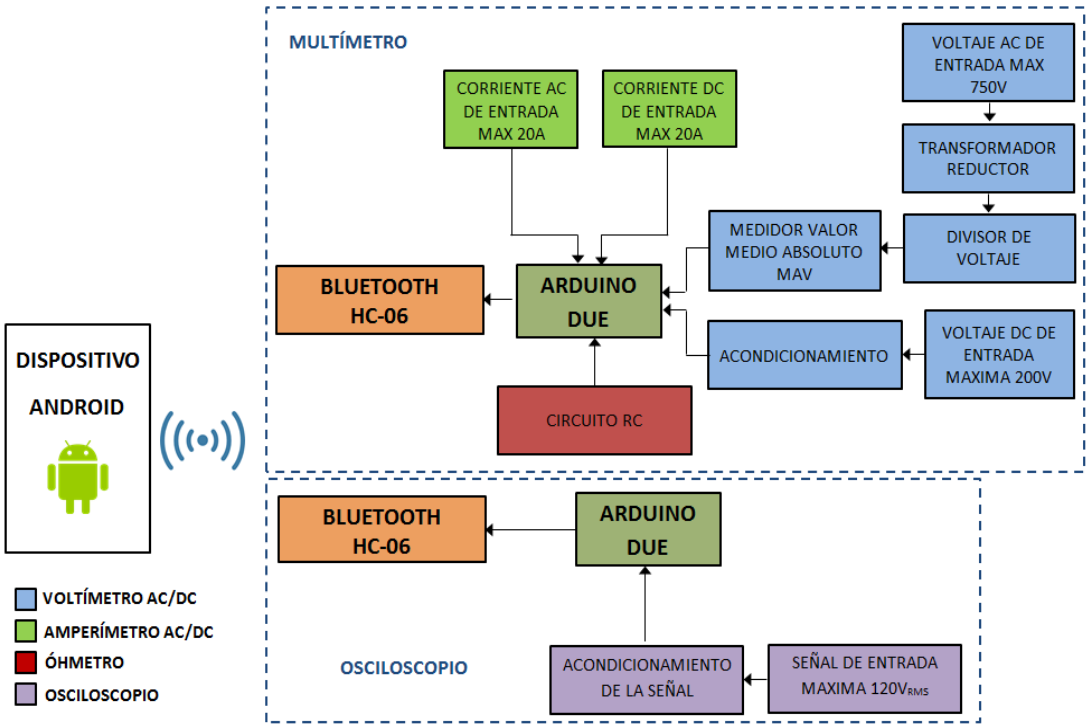


Figura 2.1: Diagrama de bloques del hardware

El proyecto consta de dos etapas: la primera integra las funciones básicas de un multímetro y la segunda las funciones básicas de un osciloscopio.

1) **Multímetro:** El núcleo del multímetro se basa en una tarjeta Arduino Due, la que se encarga de realizar la lectura de las señales acondicionadas tanto de Voltaje AC/DC (para el voltímetro) como de Corriente AC/DC (para el amperímetro), así también como la señal de un circuito RC (Resistivo-Capacitivo para el óhmetro), y gestiona la comunicación con el dispositivo Android a través de modulo Bluetooth HC-06.
2) **Osciloscopio:** Realiza el muestreo de la señal de entrada utilizando una tarjeta Arduino Due y esta a su vez se comunica con la tarjeta Arduino Due del multímetro para obtener el dato de frecuencia de la señal de entrada, y envía los datos tanto de frecuencia como de la señal muestreada por medio del módulo bluetooth HC-06 específico para esta opción hacia el dispositivo Android para visualizar de forma gráfica.

2.1.2 SISTEMA DE ADQUISICIÓN DE DATOS

Un sistema de adquisición de datos está diseñado fundamentalmente para la medida y el análisis de una o varias magnitudes físicas características de un sistema real, y convertirlas en datos para llevarlas a un dispositivo donde se pueda procesar y realizar la presentación de éstas señales.

Hoy en día la mayor parte de sistemas de adquisición de datos son basados en computadoras o en algún tipo de microprocesador, el mismo que tiene un conversor Análogo-Digital. En la Figura 2.2 se muestra el diagrama de bloques que conforma un sistema de adquisición de datos.

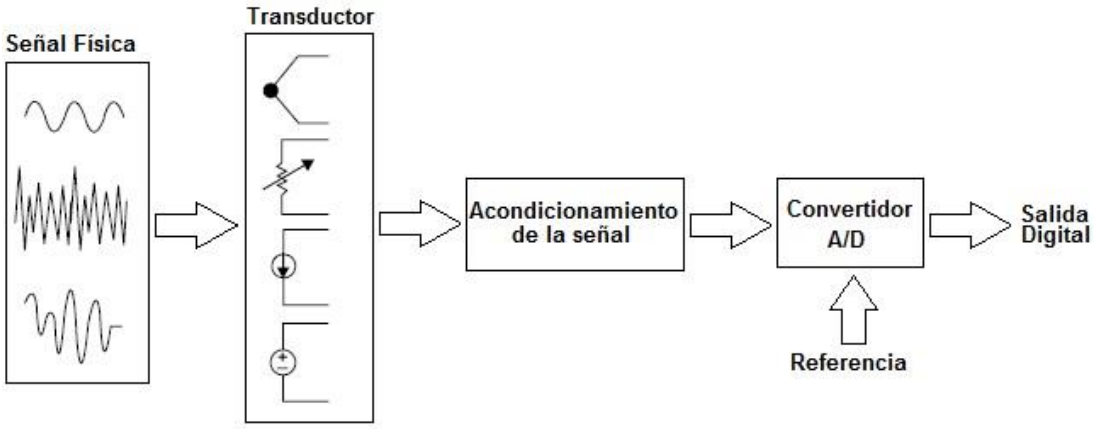


Figura 2.2: Diagrama de bloques de un sistema de adquisición de datos

En el presente proyecto se trabaja con señales eléctricas por lo que el sistema de adquisición de datos quedaría reducido al acondicionamiento de señal y conversión A/D (Análoga/Digital); mediante el acondicionamiento de las señales se proporciona una salida utilizable para la tarjeta Arduino Due en respuesta a las magnitudes que se van a medir.

2.1.2.1 ACONDICIONAMIENTO DE SEÑALES

El acondicionamiento de una señal consiste en la manipulación de dicha señal, con los dispositivos adecuados, para obtener rangos de voltajes o corrientes adecuados a las características del diseño, la señal de salida de un sistema de medición en general se debe procesar de una forma adecuada para la siguiente etapa de la operación. Por ejemplo: si la señal de salida es demasiado pequeña, sería necesario amplificarla; podría contener interferencias que eliminar; ser no lineal y requerir su linealización; ser análoga y requerir su digitalización; ser digital y convertirla en análoga; ser un cambio de voltaje y convertirla a un

cambio en corriente de magnitud adecuada, etc. En la Figura 2.3 se aprecia el esquema en bloques para el acondicionamiento de una señal.

Figura 2.3: Acondicionamiento de señales

El acondicionamiento de señales resulta conveniente al momento de realizar la instrumentación del proyecto, ya que al tener una señal con rangos de voltaje más amplios, se puede obtener mayor resolución en la medición, presentando mayor inmunidad al ruido.

Nuestro sistema de instrumentación se pueden clasificar en dos clases principales: analógicos y digitales.

- Los *sistemas analógicos* son un conjunto de dispositivos interconectados que tiene la capacidad de generar, transmitir, procesar o almacenar señales analógicas. Un sistema analógico contiene dispositivos que manipulan cantidades físicas representadas en forma analógica. En un sistema de este tipo, las cantidades varían sobre un intervalo continuo de valores.
- Los *sistemas digitales* es cualquier dispositivo destinado a la generación, transmisión, procesamiento o almacenamiento de señales digitales. También un sistema digital es una combinación de dispositivos diseñado para manipular cantidades físicas o información que estén representadas en forma digital; es decir, que sólo puedan tomar valores discretos.

2.1.2.1.1 ACONDICIONAMIENTO DE VOLTAJE

Para el desarrollo del proyecto se realizó el acondicionamiento de señales de voltaje tanto en alterna como continua, en la Figura 2.4 se muestra el circuito de acondicionamiento para que estas señales puedan ingresar al conversor análogo-digital de la tarjeta Arduino Due, utilizando divisores de tensión, seguidores de tensión, y el extractor de valor medio absoluto (MAV).

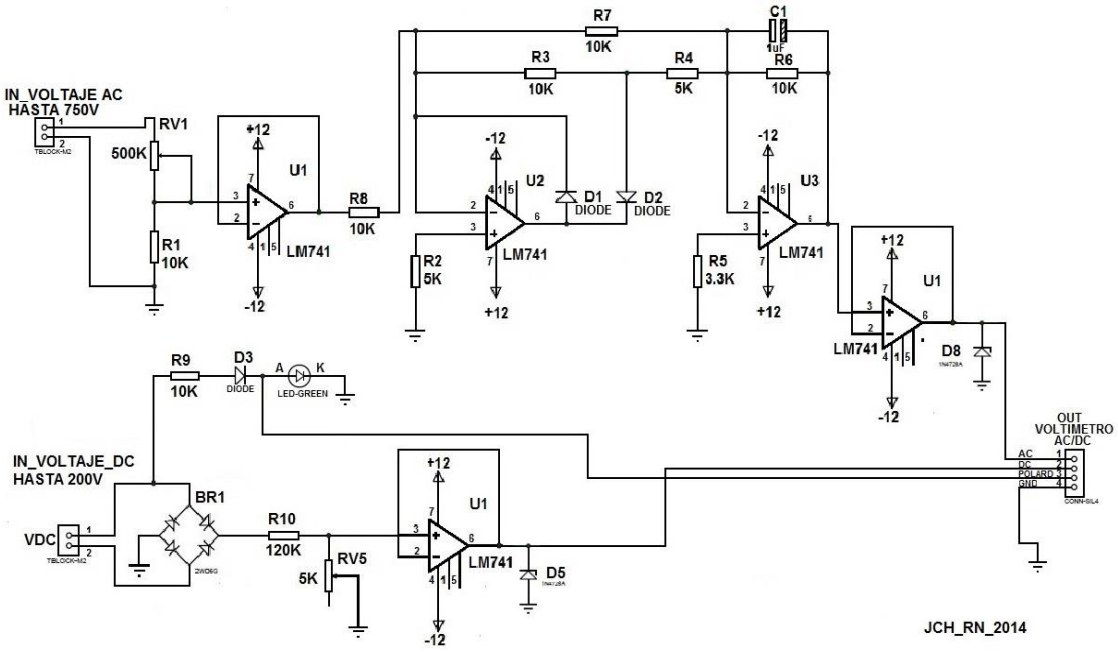


Figura 2.4: Diagrama esquemático del acondicionamiento de la señal de voltaje alterno y continuo

El *divisor de tensión* consiste en una fuente de voltaje que proporciona la energía y dos resistencia R1 y R2. Cualquier combinación de R1 con R2 formará un divisor de tensión como se puede observar en la Figura 2.5. Se utilizan en casos en que los voltajes son demasiados grandes y donde existe la necesidad de dividir tales voltajes, con la finalidad de obtener una tensión menor a partir de una mayor.

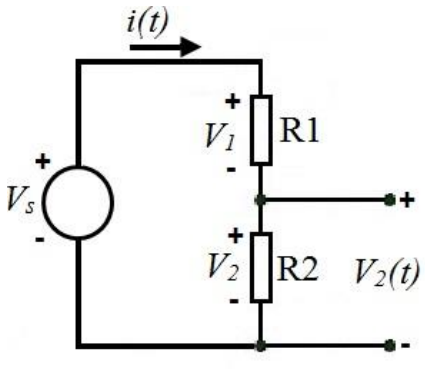


Figura 2.5: Divisor de Tensión

Ecuación del divisor de voltaje:

$$V_s(t) = v_1(t) + V_2(t) = i(t) * (R_1 + R_2) \rightarrow i(t) = \frac{V_s(t)}{R_1+R_2}$$

$$V_2(t) = i(t) * R_2 = \frac{R_2}{R_1+R_2} * V_s(t)$$

$$\frac{V_2(t)}{V_s(t)} = \frac{R_2}{R_1+R_2} \quad (2.1)$$

El voltaje V_s (t) se divide en los voltajes que caen en las resistencias R1 y R2. Esta fórmula sólo es válida si la salida V_2 (t) está en circuito abierto (no circula corriente por los terminales donde se mide V_2 (t)).

Para la implementación del multímetro se han diseñado dos divisores de tensión, los cálculos se muestran a continuación:

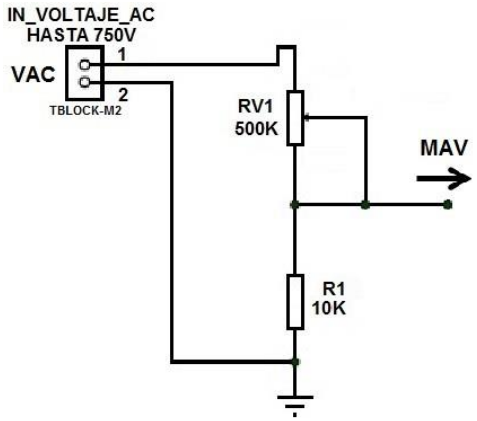


Figura 2.6: Divisor de tensión aplicado al voltaje AC de entrada

Cálculo del divisor de tensión de la Figura 2.6.

$$V_{R1} = \frac{R_1}{RV_1 + R_1} * VAC$$

$$V_{R1} = \frac{10K\Omega}{500K\Omega + 10K\Omega} * 750V$$

$$V_{R1} = 14.7V\ (V_{MAX}) \qquad (2.2)$$

El divisor de tensión de la Figura 2.6 se lo implementa en dos ocasiones al desarrollar el Osciloscopio (voltaje V_{R13} y voltaje V_{R21}, ver circuito esquemático en ANEXO B OSCILOSCOPIO)

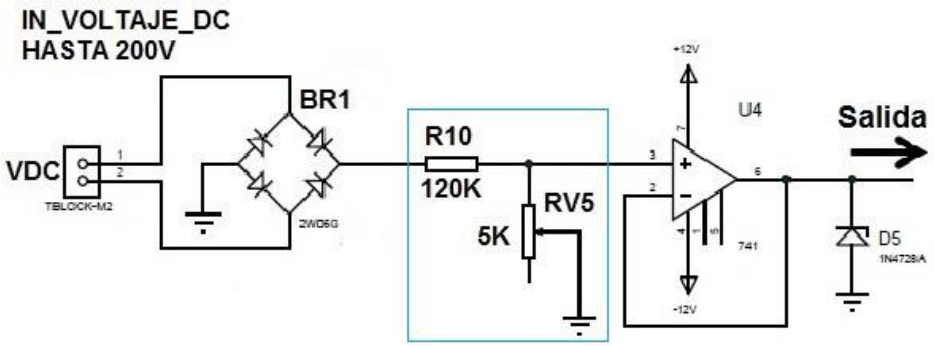


Figura 2.7: Divisor de tensión aplicado al voltaje continuo de entrada

Cálculo del divisor de tensión de la Figura 2.7, el potenciómetro RV5 se fijó en el valor de 2KΩ, así de esta manera tener un voltaje de salida menor a 3.3V que requiere la tarjeta Arduino Due para trabajar.

$$V_{R1} = \frac{RV_5}{R_{10}+RV_5} * V_{BR1}$$

$$V_{R1} = \frac{2K\Omega}{120K\Omega + 2K\Omega} * 198.6V$$

$$V_{R1} = 3.25\ (V_{max}) \tag{2.3}$$

El *seguidor de tensión* es aquel circuito que proporciona a la salida la misma tensión que a la entrada, independientemente de la carga que se le acopla. Esta aplicación es importante en la amplificación de señales que teniendo un buen nivel de tensión son de muy baja potencia y por tanto se atenúan en el caso de conectarlas a amplificadores de mediana o baja impedancia de entrada.

Para implementar el seguidor de tensión en el presente proyecto se utiliza el amplificador operacional LM741, así como se muestra en la Figura 2.8.

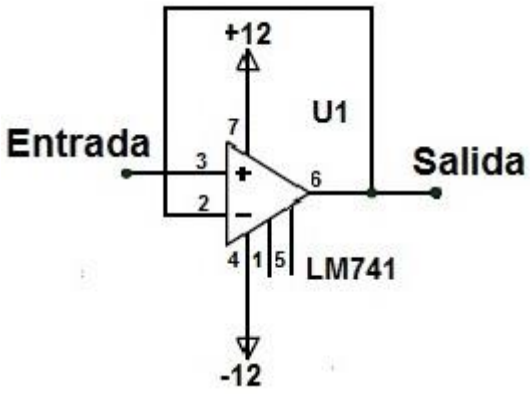


Figura 2.8: Seguidor de tensión

La mayor parte del control y medida de los procesos industriales se realiza mediante circuitos electrónicos, siendo el amplificador operacional un módulo básico de dichos circuitos de control. Aunque cada vez más, el procesado de la información y la toma de decisiones del sistema se realiza con circuitos digitales o sistemas basados en microprocesadores, la conversión de las variables medidas (temperatura, presión, velocidad, etc.) en variables eléctricas: corriente o tensión (en los sensores), o la conversión inversa (en los actuadores analógicos), requiere de circuitos analógicos, donde el amplificador operacional juega un papel fundamental.

Un amplificador operacional es un amplificador diferencial. Desde el punto de vista de una señal, tiene tres terminales: dos terminales de entrada y un terminal de salida. La Figura 2.9 muestra el símbolo que representa el Amplificador Operacional.

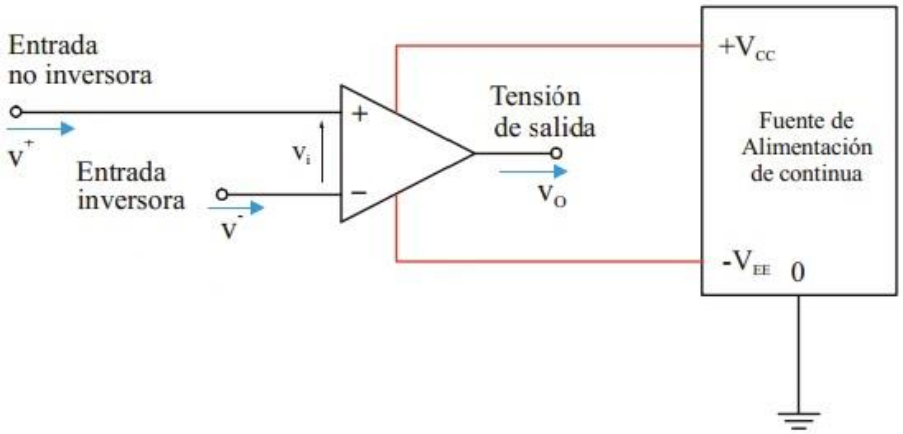


Figura 2.9: Símbolo eléctrico y terminales de un amplificador operacional[12]

En la Figura 2.10 se observa la distribución de pines del Amplificador Operacional utilizado para los acondicionamientos de las señales.

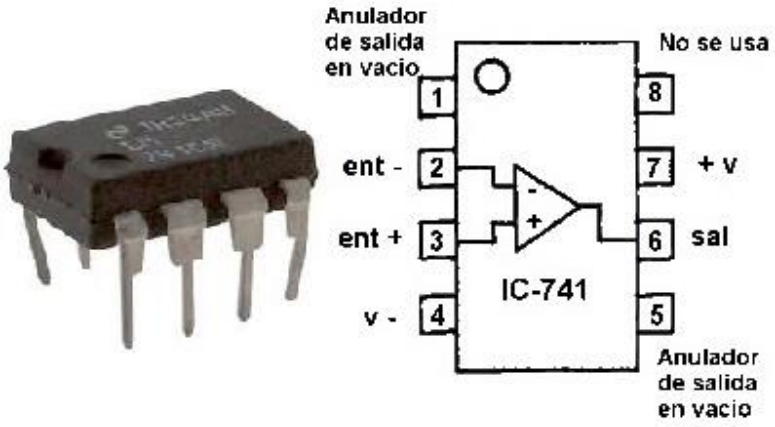


Figura 2.10: Amplificador Operacional LM741[13]

La serie LM741 son amplificadores operacionales de propósito general que ofrecen un mejor rendimiento frente a los estándares industriales. Los amplificadores ofrecen muchas características que hacen que su utilización sea

[12] http://cvb.ehu.es/open_course_ware/castellano/tecnicas/electro_gen/teoria/tema-6-teoria.pdf
[13] http://www.datasheetcatalog.net/es/datasheets_pdf/L/M/7/4/LM741.shtml

casi infalible: Protección de sobrecarga en la entrada y la salida, su salida no queda con tensión cuando se excede el rango en modo común, ausencia de oscilaciones.

Entre las características más importantes que posee este circuito integrado, se pueden destacar:

- Alta impedancia (resistencia) de entrada: del orden de 1 MΩ, lo cual implica que la intensidad de corriente por los terminales de entrada será despreciable.
- Baja impedancia de salida: del orden de 150Ω, pudiendo atacar cualquier carga (circuito) sin que su funcionamiento se modifique dependiendo del valor de ésta.
- Tensión máxima de alimentación: ± Vcc = ± 18 V. Implica que la tensión de salida nunca podrá superar a la de alimentación.
- Alta ganancia de tensión en lazo abierto (sin conectar ningún componente entre la salida y cualquiera de las entradas) con pequeños valores de tensión en los terminales de entrada se consiguen grandes tensiones de salida.

El *extractor de valor medio absoluto (VMA o MAV) de la función* es una técnica para efectuar la conversión se señales de voltaje alterno en señales de voltaje continuo constantes.

En muchas ocasiones cuando se monitorean, procesan y controlan señales alternas por medio de una computadora o un circuito electrónico, es conveniente atenuarlas a un nivel de voltaje seguro para los circuitos electrónicos o las tarjetas de lectura de la computadora, y transformarlas en niveles de voltaje continuo, que sean compatibles con la tecnología de los circuitos y dispositivos de lectura y procesamiento.

En la Figura 2.11 se puede ver el MAV utilizado en el proyecto para la conversión de señal de voltaje alterno a voltaje continuo.

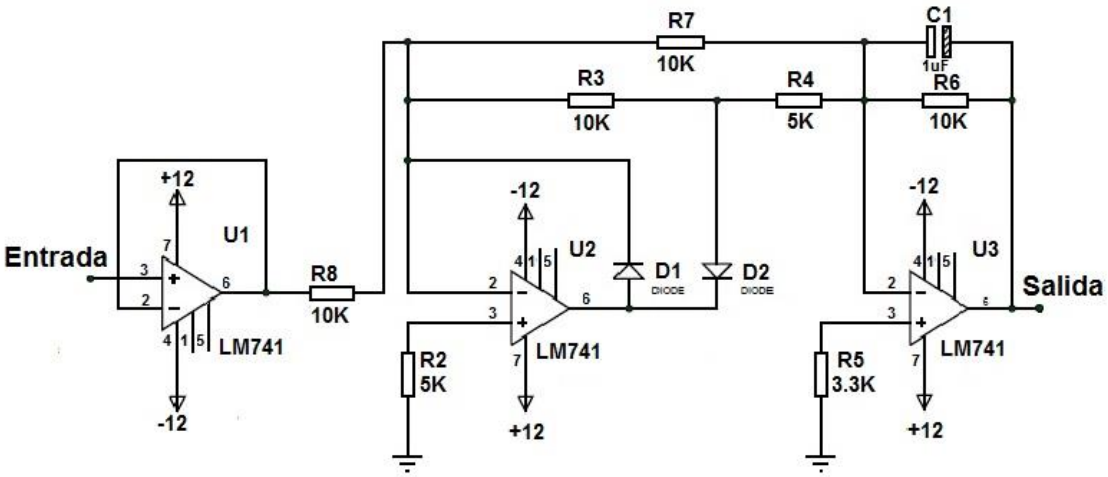


Figura 2.11: Extractor de valor medio absoluto MAV

En la Figura 2.12 se muestra la placa final de los acondicionamientos de señales de voltaje alterno y continuo

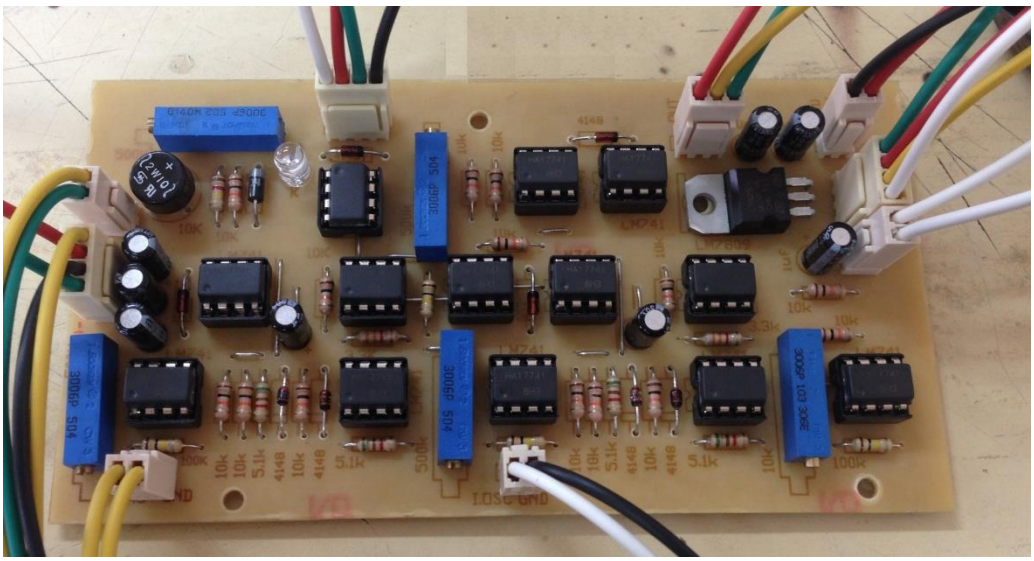


Figura 2.12: Placa de acondicionamiento de las señales de voltaje alterno y continuo

2.1.2.1.2 ACONDICIONAMIENTO DE CORRIENTE

Para el desarrollo del proyecto se realizó el acondicionamiento de corrientes tanto en alterna como continua.

Para el acondicionamiento de la señal de corriente continua se realizó utilizando el método invasivo el cual nos permite obtener un voltaje proporcional a la corriente que circula a través de una resistencia cuyo valor debe ser bien bajo colocada en serie al circuito que se requiere medir.

En las Figuras 2.13 y 2.14 se muestra el diagrama esquemático del acondicionamiento de la señal de corriente continua y la placa final respectivamente, implementado en el proyecto.

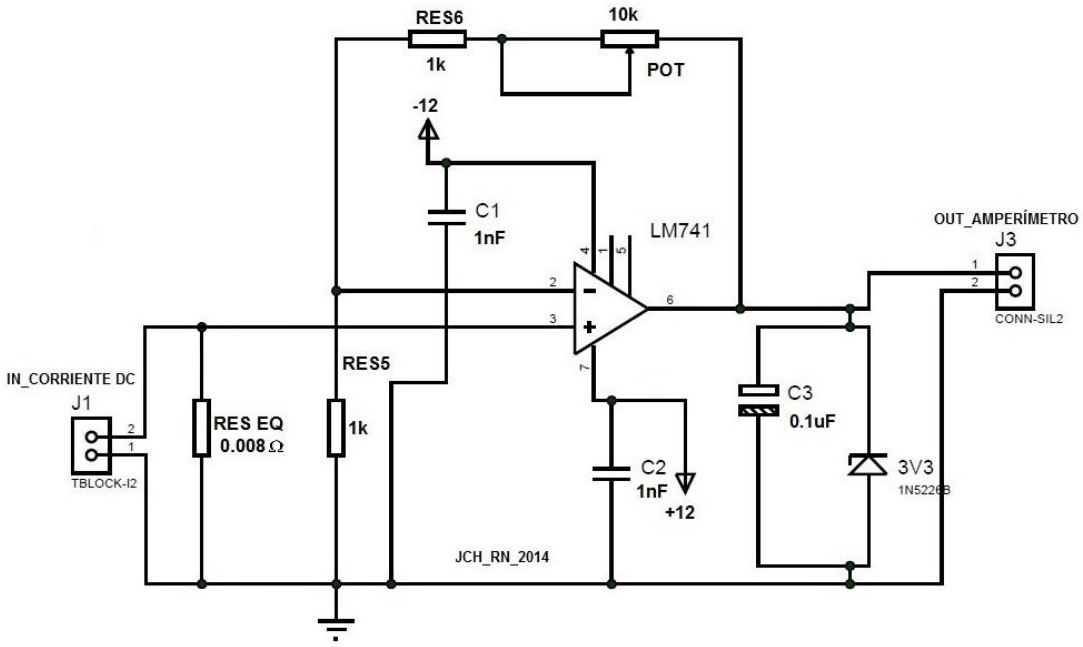


Figura 2. 13: Diagrama esquemático del acondicionamiento de la señal de corriente continua.

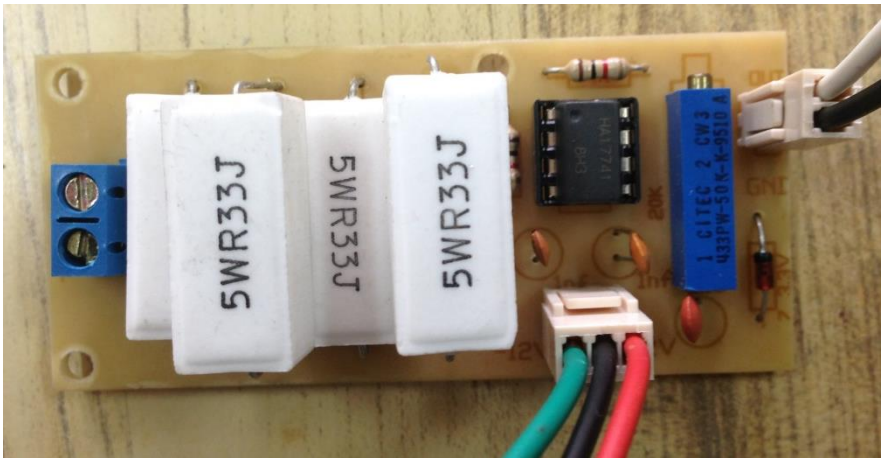


Figura 2.14: Placa final del acondicionamiento de la señal de corriente continua.

La medición de corriente alterna se realizó de la misma forma que la señal de corriente continua, con una modificación en el circuito de acondicionamiento, al cual se le aumentó una etapa de rectificación, para convertir la señal de corriente alterna en señal de corriente continua.

En las Figuras 2.15 y 2.16 se muestra el diagrama esquemático del acondicionamiento de la señal de corriente alterna y la placa final respectivamente, implementado en el proyecto.

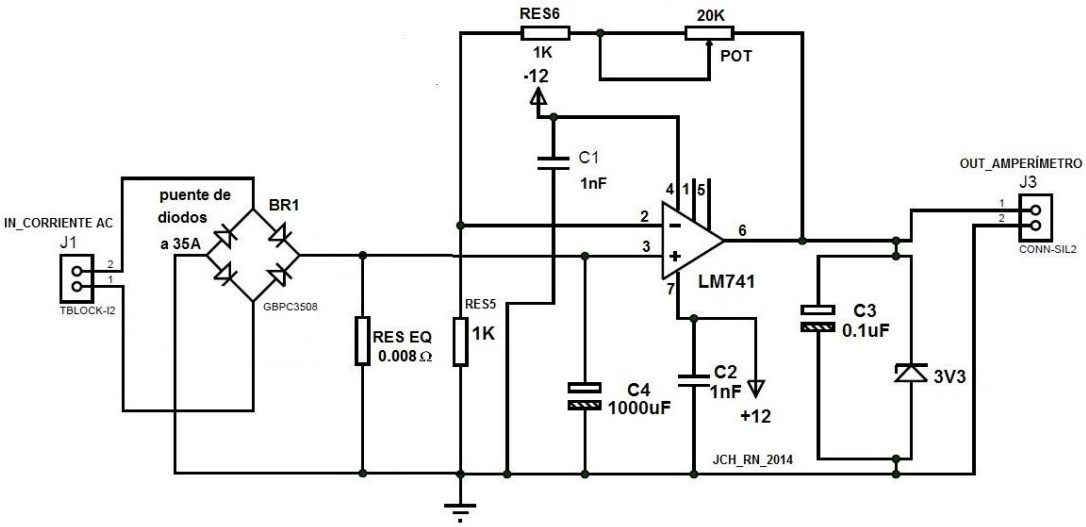


Figura 2.15: Diagrama esquemático del acondicionamiento de la señal de corriente alterna.

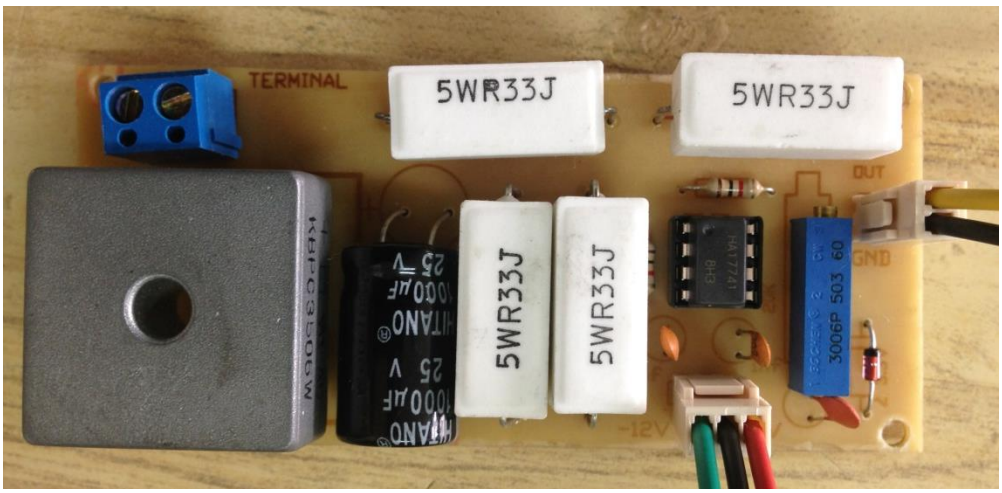


Figura 2.16: Placa final del acondicionamiento de la señal de corriente alterna

2.1.2.1.3 MEDICIÓN DE LA RESISTENCIA

La medición de la resistencia se la realizó mediante el principio de carga y descarga de un capacitor en un circuito RC.

Un circuito RC es un circuito con una resistencia y un capacitor en serie, donde las corrientes, voltajes y potencias cambian en el tiempo, pero para ello se debe cargar o descargar dicho capacitor.

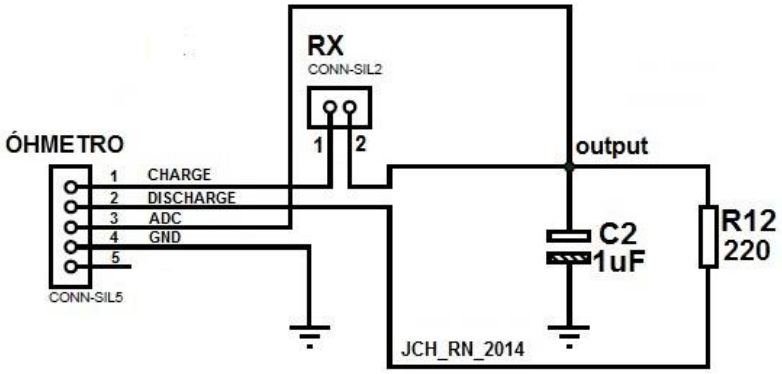


Figura 2.17: Diagrama esquemático del circuito RC implementado

El capacitor C2 se cargará a través de la resistencia RX, una vez que se haya ingresado a la aplicación del multímetro en el dispositivo Android y seleccionado la opción de *medición de resistencia.*

El valor de la resistencia RX es calculado mediante la ecuación:

$$TC = R \times C \tag{2.4}$$

Dónde: TC= Constante de tiempo en segundos

R= Resistencia en ohmios

C= Condensador en faradios

La constante de tiempo nos indica el tiempo que el condensador tarda en adquirir el 63.2% de la carga final de equilibrio.

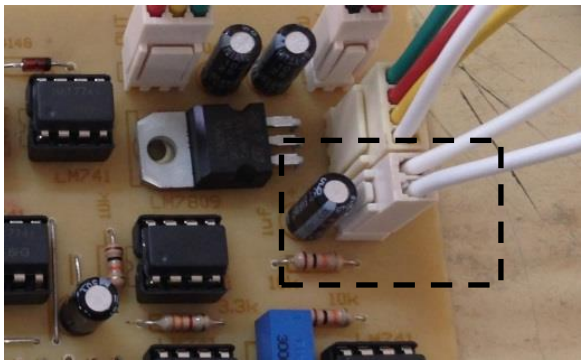


Figura 2.18: Circuito RC implementado en la placa final

En las Figuras 2.17 y 2.18 se muestra el diagrama esquemático del circuito RC y la placa final respectivamente, implementado en el proyecto.

2.1.2.2 CONVERSIÓN ANÁLOGA DIGITAL

La conversión análoga digital o digitalización, consiste básicamente en realizar de forma periódica medidas de la amplitud de la señal de entrada y traducirlas a un lenguaje numérico.

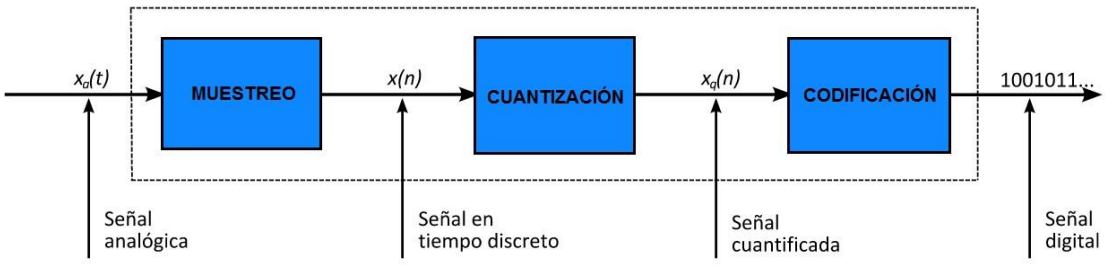


Figura 2.19: Proceso de la conversión análoga digital

La tarjeta Arduino Due abre las posibilidades de programación en particular para el procesamiento Analógico / Digital, la razón del muestreo teórico va desde los 15 Ksps hasta los 1000 ksps (kilomuestras por segundo), tiene 12 entradas análogas conectadas a un conversor análogo digital de 12 bits. Esto significa que convertirá tensiones entre 0 y 3.3V que soporta los pines de entrada, a un número entero entre 0 y 4095.

El conversor análogo digita (ADC) recibe señales eléctricas con las que se trabaja en el proyecto y a continuación las convierte en señales digitales, codificadas en valores numéricos binarios, "ceros" y "unos", para realizar esta tarea tiene que efectuar los siguientes procesos:

- Muestreo
- Cuantización
- Codificación

2.1.2.5.1 MUESTREO

Para convertir una señal analógica en digital, el primer paso consiste en realizar un muestreo (sampling) de ésta, o lo que es iguala, tomar diferentes muestras de tensiones o voltajes en diferentes puntos de la onda senoidal. La frecuencia a la que se realiza el muestreo se denomina razón, tasa o también frecuencia de muestreo y se mide en kilohertz (kHz).

Para garantizar la toma de muestras y la conversión de forma correcta se debe considerar la velocidad de muestreo, para lo cual el teorema de Nyquist, establece que la frecuencia de muestreo *fs*, debe ser como mínimo el doble que el ancho de banda de la señal muestreada. Sino ocurre esta situación, se tiene lugar el fenómeno denominado *aliasing.*

$$fs > 2 * fm \tag{2.5}$$

2.1.2.5.2 CUANTIZACIÓN

La cuantización lo que hace es convertir una sucesión de muestras de amplitud continua en una sucesión de valores discretos preestablecidos según el código utilizado, por tanto, representa el componente de muestreo de los diferentes niveles o variaciones de amplitud que contiene la señal analógica original, que permite medirlos y asignarles sus correspondientes valores en el sistema numérico decimal, antes de convertir esos valores en sistema numérico binario, por lo que se puede objetar que el muestreo representa el tiempo de captura de una señal y la cuantización es el componente de la amplitud del muestreo.

2.1.2.5.3 CODIFICACIÓN

Los valores de las tomas de los diferentes niveles o variaciones de amplitud se representan numéricamente por medio de códigos y estándares previamente establecidos, lo más común es codificar la señal digital en código numérico binario, esta representación se conoce como codificación.

2.1.3 COMUNICACIÓN SERIAL

2.1.3.1 MÓDULO BLUETOOTH HC-06

Para el desarrollo del proyecto se utiliza comunicación serial la cual será realizada mediante el protocolo de comunicación inalámbrica Bluetooth, para lo que se utiliza el módulo HC-06, el mismo que funciona solo como esclavo, el módulo se lo encuentra en el mercado como se indica en la Figura 2.20.

Figura 2.20: Módulo Bluetooth HC-06[14]

El módulo Bluetooth HC-06 utiliza el protocolo UART RS 232 serial, una de las principales ventajas de éste módulo, además de su pequeño tamaño y sus buenas características de transmisión y recepción que brindan un alcance muy amplio (por tratarse de un sistema local Bluetooth), es el bajo consumo de corriente que posee tanto en funcionamiento, como en modo de espera, es decir, alimentado con energía, o sin conexión o enlace, en nuestro caso con un dispositivo Android. En la Tabla 2.1 se muestra las características del módulo bluetooth.

Tabla 2.1: Características módulo Bluetooth HC-06

Frecuencia	2.4 GHz ISM Band
Bluetooth	V2.0+EDR
Potencia de salida de clase	Clase 2
Rango de baudios ajustable	1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
Default	Esclavo, 9600 Baud Rate, N, 8, 1.
Distancia	10 metros
Voltaje de funcionamiento	3.3V
Interfaz de Host	USB UART
Interfaz de audio	Ninguno
Protocolo	RFCOMM conocido también como emulador de puerto serial
Dimensiones	27x13x2.2 mm, peso: 1 gramo

[14] http://blog.make-a-tronik.com/tutorial-modulo-bluetooth-hc-06-parte-1/

Otra característica interesante de este módulo es que una vez que ha realizado un enlace con otro dispositivo es capaz de recordarlo en su memoria y no solicita validación alguna ("1234" por defecto), pero si se activa el pin "KEY" hacia la tensión de alimentación, esta información se elimina y el módulo HC-06 solicitará nuevamente la validación del enlace.

Otro detalle particular es que su tensión de alimentación es de 3,3V y su bajo consumo (8mA en transmisión/recepción activa) lo transforman en un dispositivo ideal para trabajar con la tarjeta Arduino Due de la misma tensión, logrando de este modo trabajar con la misma fuente de alimentación, demostrando características excepcionales en la implementación de éste proyecto. En la Figura 2.21 se observa la conexión de la tarjeta Arduino Due con el módulo Bluetooth HC-06.

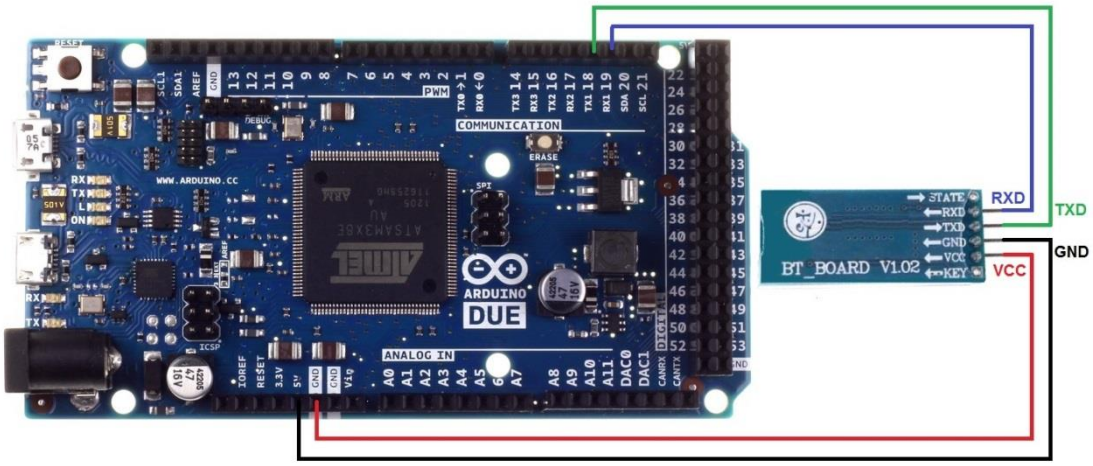


Figura 2.21: Conexión física entre tarjeta Arduino Due y módulo Bluetooth HC-06

2.2 DISEÑO DEL SOFTWARE

Una vez revisado el diseño del hardware del proyecto, se explica cómo se diseñó el software tanto en Arduino como en Android.

2.2.1 ENTORNO DE DESARROLLO PARA ARDUINO

El entorno de desarrollo Arduino (Figura 2.22), está constituido por un editor de texto para escribir el código, un área de mensajes, una consola de texto, una barra de herramientas con botones para las funciones comunes, y una serie de menús. Permite la conexión con el hardware de Arduino para cargar los programas y comunicarse con ellos.

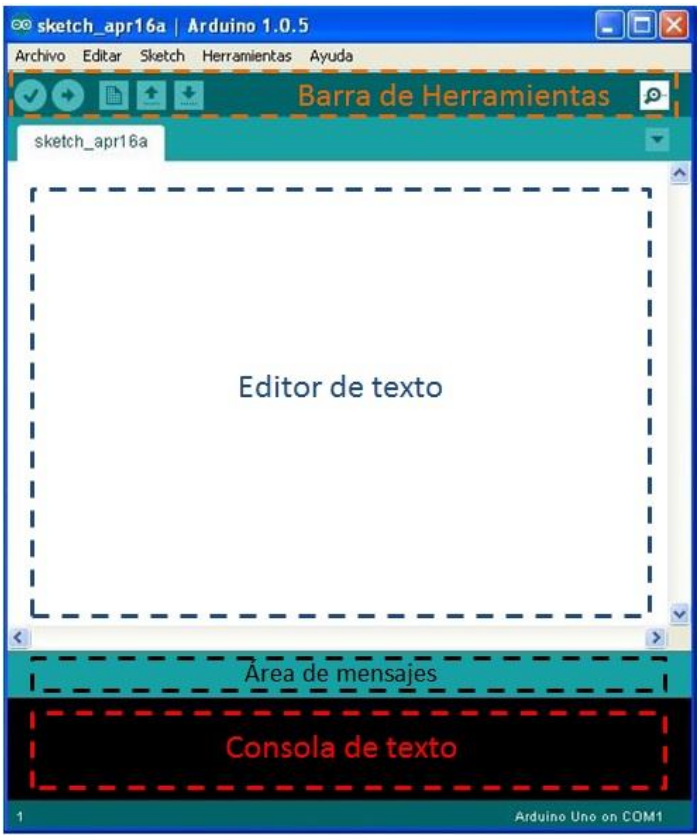


Figura 2.22: Entorno de desarrollo para Arduino

Arduino utiliza para escribir el software lo que denomina "sketch" (programa). Estos programas son escritos en el editor de texto. Existe la posibilidad de cortar/pegar y buscar/remplazar texto. En el área de mensajes se muestra información mientras se cargan los programas y también muestra errores. La consola muestra el texto de salida para el entorno de Arduino incluyendo los

mensajes de error completos y otras informaciones. La barra de herramientas permite verificar el proceso de carga, creación, apertura y guardado de programas, y la monitorización serie, en la Tabla 2.2 se puede observar cada función que cumplen sus herramientas.

Tabla 2.2: Funciones comunes de la barra de herramientas del entorno de desarrollo para Arduino

Herramientas	Función
	Verificar Chequea el código en busca de errores.
	Cargar Compila el código y lo graba en la placa E/S de Arduino.
	Nuevo Crea un nuevo sketch
	Abrir Presenta un menú de todos los programas sketch de su "sketchbook", (librería de sketch). Un click sobre uno de ellos lo abrirá en la ventana actual
	Guardar Guarda el programa Sketch
	Monitor Serial Inicia la monitorización serie

2.2.2 LIBRERÍAS

2.2.2.1 AnalogReadResolution( )

Esta librería es una extensión de la API analógica para la tarjeta Arduino Due. Define el tamaño (en bits) del valor devuelto por analogRead( ). Por defecto es 10

bits (devuelve valores entre 0-1023) para la compatibilidad con placas basadas en AVR.

La tarjeta Arduino Due tiene capacidades de ADC de 12 bits que puede acceder al cambiar la resolución a 12. Esto devolverá los valores de analogRead( ) entre 0 y 4095.

- *Sintaxis:* analogReadResolution(bits)
- *Parámetros:* bits: determina la resolución (en bits) del valor devuelto por la función analogRead ( ). Se puede configurar resoluciones superiores a 12 pero los valores devueltos por el analogRead( ) sufrirá una aproximación.
- *Devoluciones:* Ninguno
- *Nota:* Si establece el valor de AnalogReadResolution( ) a un valor más alto que las capacidades de la tarjeta, la Arduino sólo informará en su mayor resolución llenando los bits extras con ceros. Por ejemplo: el uso de la tarjeta Arduino Due con analogReadResolution(16) le dará un número de 16 bits, aproximada con los primeros 12 bits que contienen la verdadera lectura ADC y los últimos 4 bits rellena con ceros . Si establece el valor de AnalogReadResolution ( ) a un valor inferior que las capacidades de la tarjeta, el extra de bits menos significativos leídos de la ADC se descartan.

2.2.2.2 AnalogWriteResolution( )

Esta librería es una extensión de la API analógica para la tarjeta Arduino Due. Establece la resolución de la función analogWrite( ). Por defecto es 8 bits (valores entre 0-255) para la compatibilidad con placas basadas en AVR.

La tarjeta Arduino Due tiene las siguientes capacidades de hardware: 12 pines que viene por defecto a 8 bit para PWM, al igual que las tarjetas basadas en AVR. Estos pueden ser cambiados a una resolución de 12 bits; tiene 2 pines con DAC (Digital-to-Analog Converter / Conversor Digital - Análogo) de 12 bits.

Al establecer la resolución de escritura a 12 bits, se puede utilizar analogWrite( ) con valores entre 0 y 4095 para aprovechar la máxima resolución del conversor o para ajustar la señal de PWM sin volcarse.

- *Sintaxis:* analogWriteResolution(bits)
- *Parámetros:* bits: determina la resolución (en bits) de los valores utilizados en la función analogWrite( ). El valor puede variar de 1 a 32 bits. Si selecciona una resolución más alta o más baja que las capacidades del hardware de la tarjeta, el valor utilizado en analogWrite ( ) será truncado si es demasiado alto o rellenado con ceros si es demasiado bajo.
- *Devoluciones:* Ninguno
- *Nota:* Si establece el valor de analogWriteResolution( ) a un valor más alto que las capacidades de la tarjeta, ésta descartara los bits adicionales. Por ejemplo: usando la tarjeta Arduino Due con analogWriteResolution(16) en un pin DAC a 12 bits, sólo los primeros 12 bits de los valores pasados al analogWrite( ) se utilizarán y los últimos 4 bits se descartarán. Si se establece el valor analogWriteResolution( ) a un valor inferior a las capacidades de la tarjeta, las partes que faltan se rellenan con ceros para completar el tamaño requerido del hardware. Por ejemplo: el uso de la tarjeta Arduino Due con analogWriteResolution(8) en un pin DAC de 12 bits, la tarjeta Arduino se sumará 4 bits cero al valor de 8 bits utilizado en analogWrite( ) para obtener los 12 bits necesarios.

2.2.3 PROGRAMACIÓN DE LA TARJETA ARDUINO DUE

Para cumplir con el objetivo planteado en el proyecto se utilizó dos tarjetas Arduino Due, cada una con especificaciones diferentes:

a) Multímetro,
b) Osciloscopio

2.2.3.1 MULTÍMETRO

Este programa realiza las siguientes actividades:

- Muestrea las señales de entradas eléctricas como voltaje, corriente, resistencia.
- Analiza la señal de entrada del osciloscopio, calcula su frecuencia y envía una señal de voltaje proporcional a través del conversor digital/análogo hacia la tarjeta que se encarga de gestionar las funciones del osciloscopio.
- Acondiciona digitalmente la señal de voltaje alterna a un rango de 0-750V
- Acondiciona digitalmente la señal de voltaje continuo a un rango de 0-200V
- Acondiciona digitalmente la señal de corriente alterna a un rango de 0-20A
- Acondiciona digitalmente la señal de corriente continua a un rango de 0-20A
- Mide el tiempo en que se demora en cargar un capacitor de 1uF para calcular la resistencia que se conecta en serie a dicho capacitor.

2.2.3.1.1 DIAGRAMA DE FLUJO DEL PROGRAMA DEL MULTÍMETRO

El diagrama de flujo del programa del multímetro se muestra en las Figuras 2.23, 2.24 y 2.25

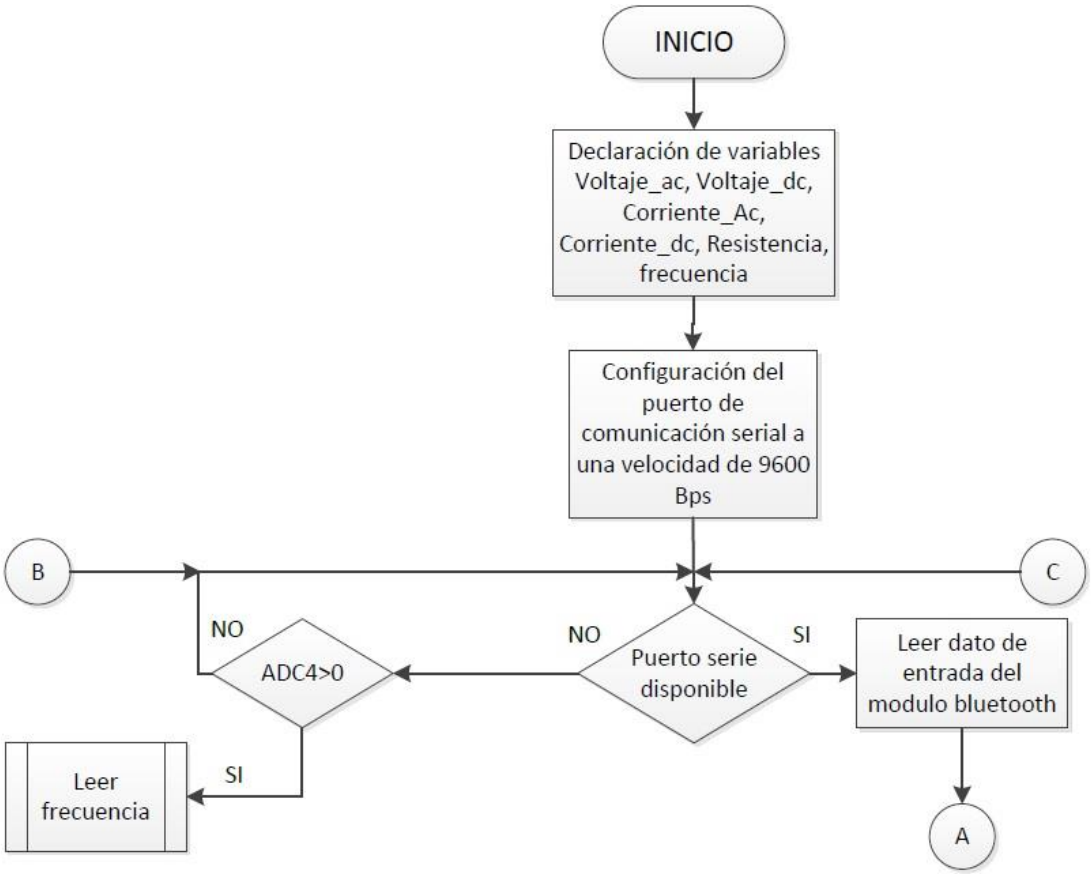


Figura 2.23: Diagrama de flujo de la programación del Multímetro en la Tarjeta Arduino Due

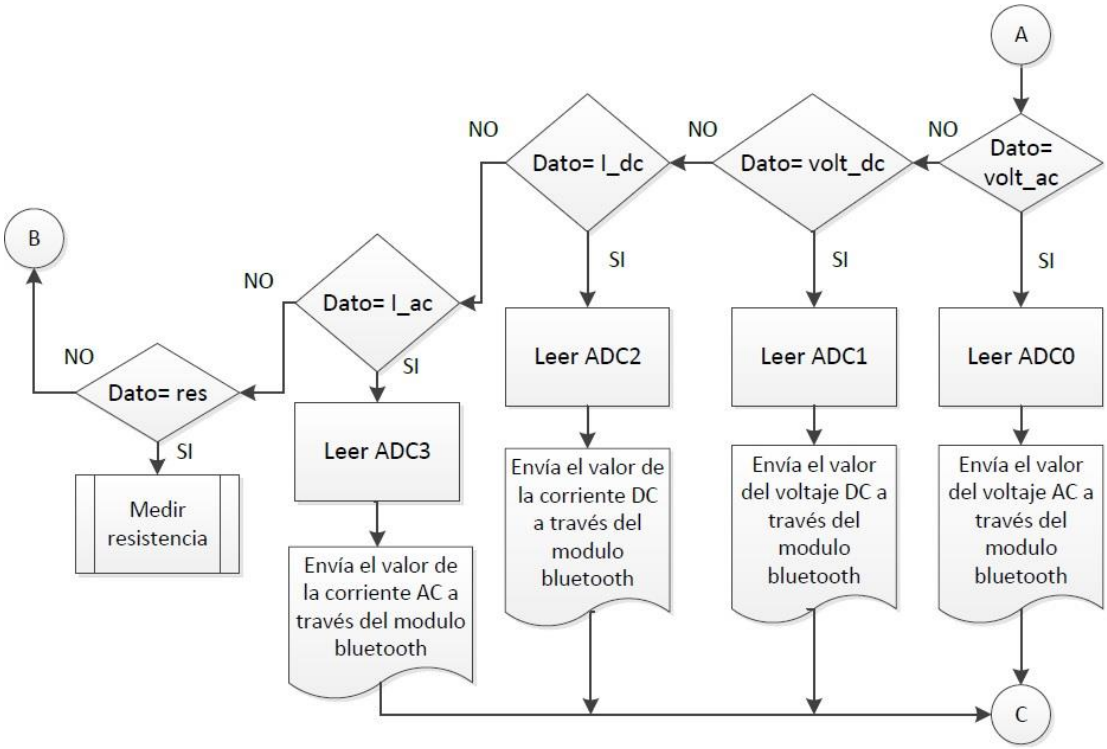


Figura 2.24: Diagrama de flujo de la programación del Multímetro en la Tarjeta Arduino Due

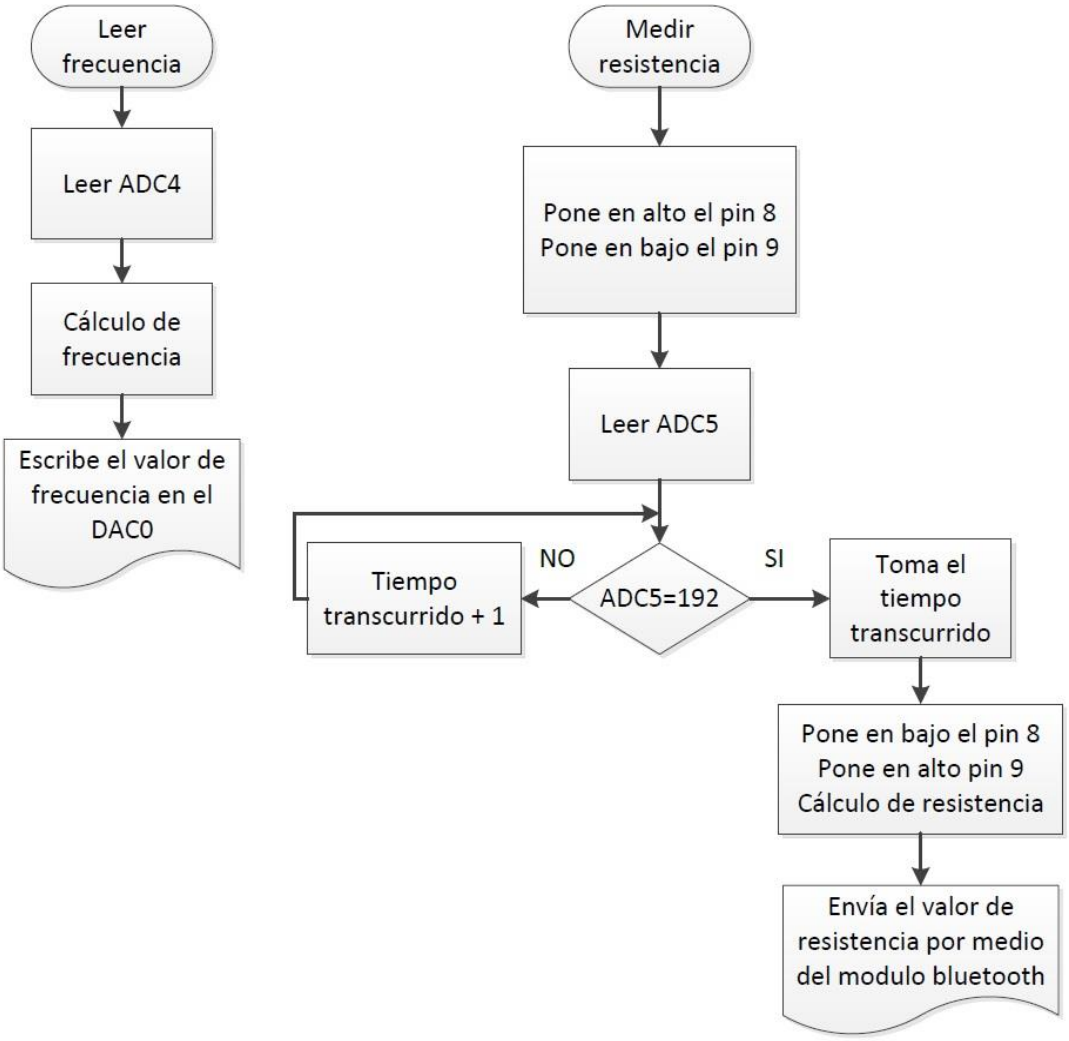


Figura 2.25: Diagrama de flujo de la programación del Multímetro en la Tarjeta Arduino Due

En el ANEXO E se muestra el código de programación del multímetro en la Tarjeta Arduino Due con sus instrucciones propiamente detalladas.

2.2.3.2 OSCILOSCOPIO

Este programa realiza las siguientes actividades:

- Muestrea la señal de entrada a 100Ksps
- Lee el dato de frecuencia que es enviado por la tarjeta que gestiona las funciones de multímetro
- Calcula el voltaje V_{RMS} de la señal de entrada

2.2.3.2.1 DIAGRAMA DE FLUJO DEL PROGRAMA DEL OSCILOSCOPIO

El diagrama de flujo del programa del osciloscopio se muestra en las Figuras 2.26

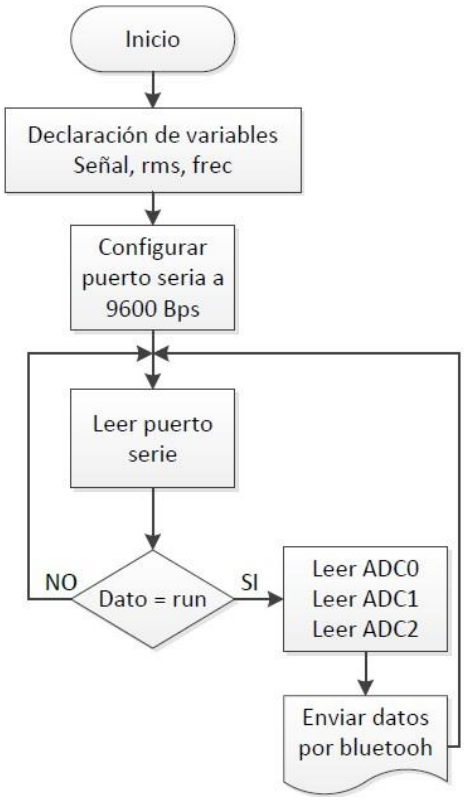


Figura 2.26: Diagrama de flujo de la programación del osciloscopio en la tarjeta Arduino Due

En el ANEXO E se muestra el código de programación del osciloscopio en la Tarjeta Arduino Due con sus instrucciones propiamente detalladas.

2.2.4 INSTRUMENTOS VIRTUALES PARA LA PLATAFORMA ANDROID

Android proporciona acceso a una amplia gama de bibliotecas y herramientas que pueden ser utilizadas para construir una gran variedad de aplicaciones, la aplicación que se desarrolla en este proyecto se trata de Instrumentos Virtuales (Voltímetro AC/DC, Amperímetro AC/DC, Óhmetro, Osciloscopio), esta aplicación puede correr en cualquier dispositivo Android, la aplicación de éste proyecto se la realiza en particular para el laboratorio de circuitos electrónicos de la Universidad de las Fuerzas Armadas ESPE - Extensión Latacunga.

El proyecto se familiariza con la plataforma Android al tener un ambiente de desarrollo integrado, que permita la conexión entre las librerías de Android con el lenguaje de programación JAVA, para facilitar al momento del desarrollo la detección de errores en el momento de la compilación del código.

2.2.4.1 ECLIPSE COMO ENTORNO DE DESARROLLO ANDROID

Para programar en Android se necesita preparar el entorno de trabajo, es necesario disponer de una versión de eclipse actualizada para poder realizar el proyecto, como segunda instancia se necesita el kit de desarrollo de software para Android o Android SDK.

Para nuestro proyecto se utilizó el paquete ADT Bundle, el mismo que ofrece todo lo necesario para comenzar a desarrollar aplicaciones, incluyendo una versión del IDE de Eclipse con una función de ADT (Android Developer Tools) para agilizar el desarrollo de su aplicación Android, brinda un entorno de desarrollo ya completo e integrado sin necesidad de hacer ninguna configuración previa o posterior a su instalación, básicamente incluye todo lo que necesitamos para comenzar a desarrollar la aplicación en Android.

2.2.4.2 CONFIGURACIÓN DE ECLIPSE PARA EL DESARROLLO DE APLICACIONES ANDROID

El SDK de Android proporciona las bibliotecas API y las herramientas de desarrollo necesarias para crear, probar y depurar aplicaciones para Android. Se lo puede encontrar en la página web: http://developer.android.com/sdk/index.html

Al realizar la descarga, el paquete incluye todo lo necesario para comenzar a realizar aplicaciones:

- Versión de Eclipse con el ADT-plugin ya integrado
- Componentes de la librería Android SDK
- Android Platform-tools
- La imagen más reciente del sistema Android para el emulador

Para comenzar con el desarrollo de la aplicación se debe tener en cuenta que se debe contar con la actualización del Java Development Kit o (JDK), éste es un software que provee herramientas de desarrollo para la creación de programas en Java

Una vez que el IDE ya está cargado con el Android Developer Tools plugin y SDK se puede comenzar a desarrollar el proyecto.

2.2.4.3 CREACIÓN DE UN NUEVO PROYECTO ANDROID

Un proyecto Android contiene todos los archivos que componen el código fuente de su aplicación Android. Las herramientas del SDK de Android hacen que sea fácil para iniciar un nuevo proyecto con un conjunto de directorios y archivos de proyecto por defecto.

Para la creación de un nuevo proyecto Android se debe seguir los siguientes pasos:

1. Hacer clic en Nuevo en la barra de herramientas, seleccionar la carpeta Android, y escoger "Proyecto de aplicación de Android", y hacer clic en Siguiente, aparece una ventana como se ilustra en la Figura 2.27, rellenar el formulario que a continuación se detalla y dar clic en siguiente:
    - *Nombre de la Aplicación*: Es el nombre de la aplicación que aparece a los usuarios. Para este proyecto el nombre es "Instrumentos_Virtuales"
    - *Nombre del Proyecto*: Es el nombre del directorio del proyecto y el nombre visible en Eclipse
    - *Nombre del paquete*: Es el espacio del nombre del paquete para la aplicación, éste nombre debe ser único en todos los paquetes instalados en el sistema Android.
    - *SDK Mínimo requerido:* Es la versión más baja de Android que admite la aplicación, indican usando el nivel de API.
    - *SDK Target:* Indica la versión más alta de Android (también mediante el nivel de API). A medida que se disponga de nuevas versiones de Android, se debe probar la aplicación en la nueva versión y actualizar este valor para que coincida con el último nivel de API con el fin de aprovechar las nuevas características de la plataforma.

- *Compilar con:* Es la versión de la plataforma con cual se compila la aplicación.
- *Tema:* Especifica el estilo de interfaz de usuario de Android que tendrá la aplicación.

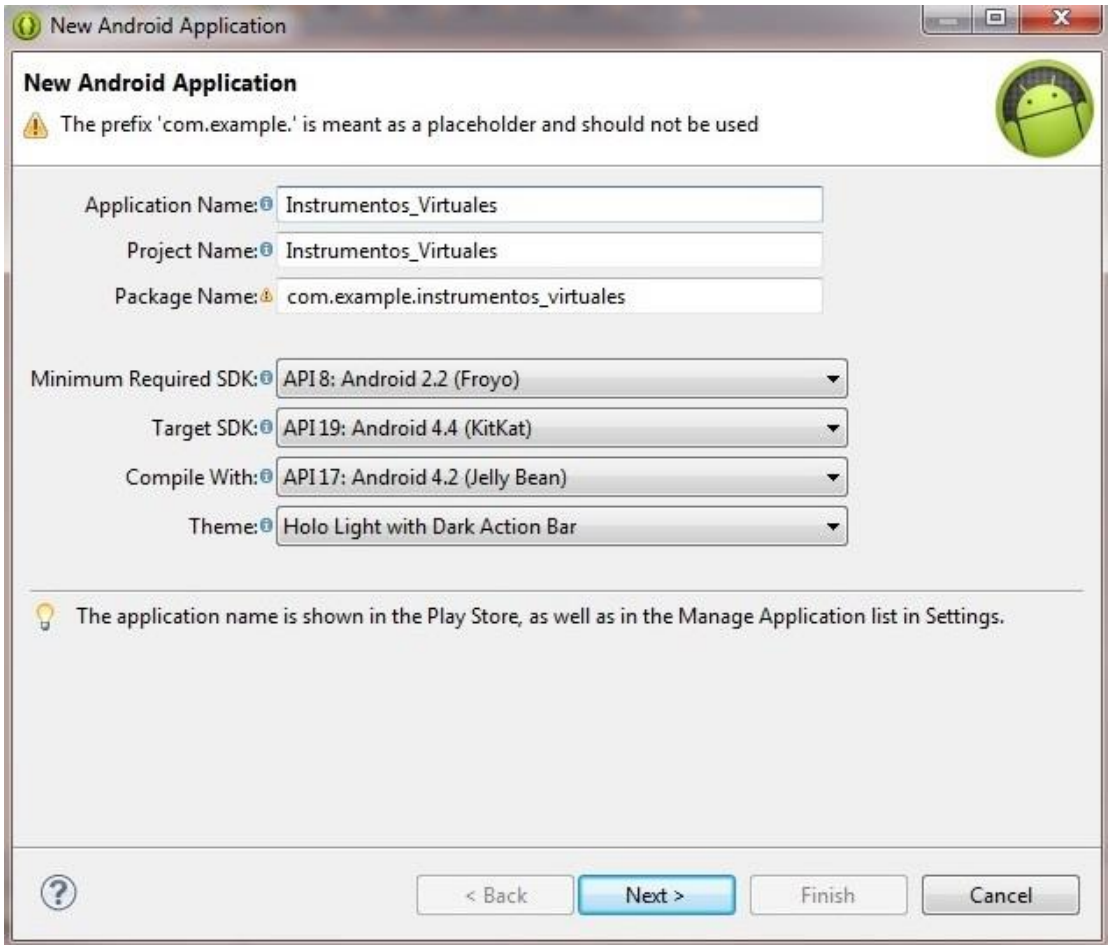


Figura 2.27: Creación de un nuevo proyecto Android

2. En la siguiente pantalla para configurar el proyecto, dejar las selecciones predeterminadas como se ilustra en la Figura 2.28 y hacer clic en siguiente.

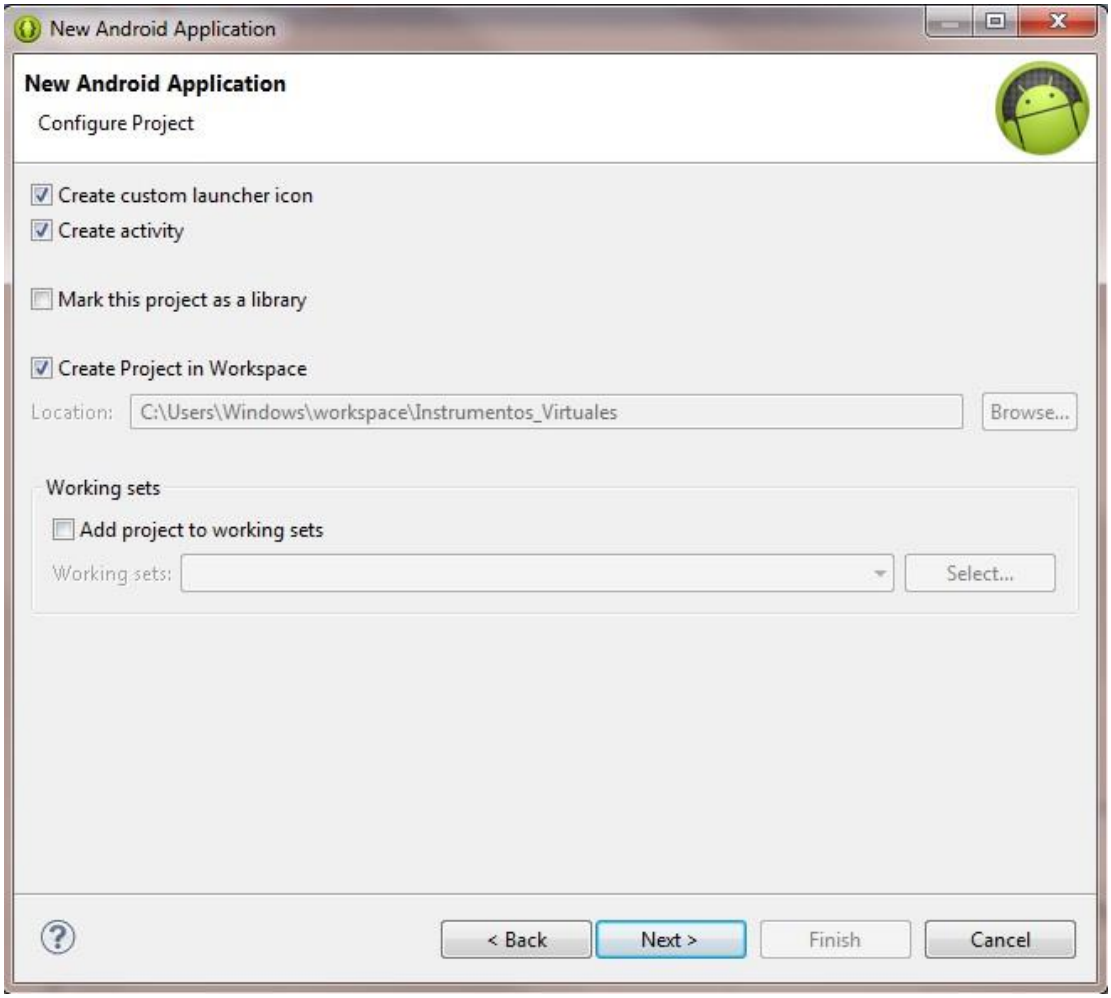


Figura 2.28: Configuración de un nuevo proyecto Android

3. En la siguiente pantalla que se muestra en la Figura 2.29 se puede crear un icono para la aplicación, personalizándolo de varias maneras. Antes de publicar la aplicación, asegurarse que el icono cumpla con las especificaciones definidas en la iconografía guía de diseño y hacer clic en siguiente. El icono creado para el proyecto se muestra en la Figura 2.30

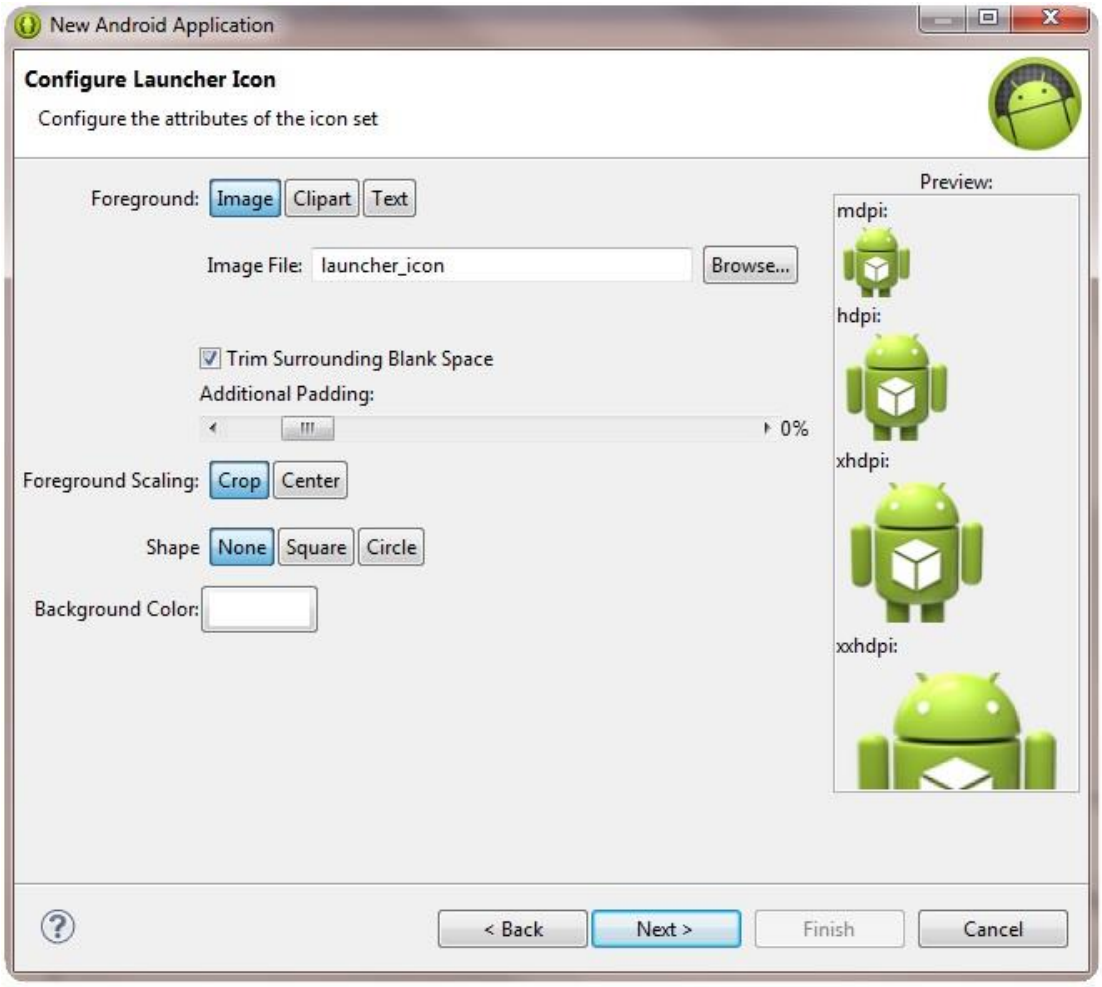


Figura 2.29: Configuración icono de la aplicación

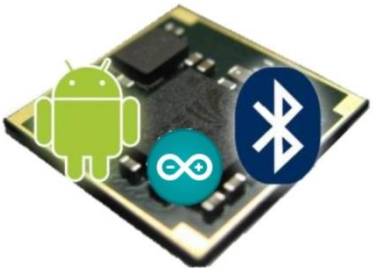


Figura 2.30: Icono del proyecto para la aplicación Android

4. En la Figura 2.31 se muestra la pantalla en la que se puede seleccionar una plantilla de actividad, desde la cual se comienza a construir la aplicación; para este proyecto se seleccionó *BlankActivity*, a continuación dar clic en siguiente, configurar la actividad en blanco, dejar los detalles de la actividad en su estado predeterminado y hacer click en Finalizar (Figura 2.32).

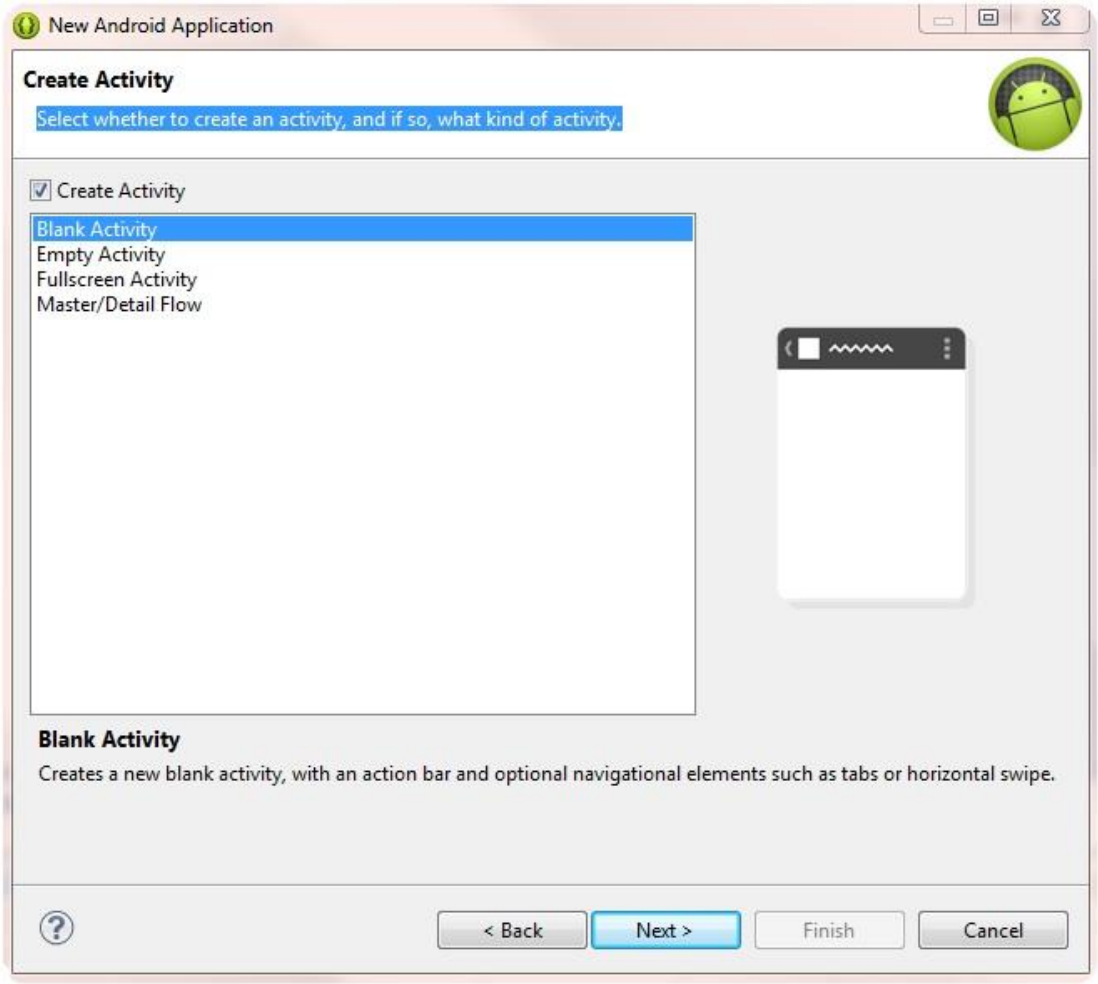


Figura 2.31: Creación de una actividad para un nuevo proyecto Android

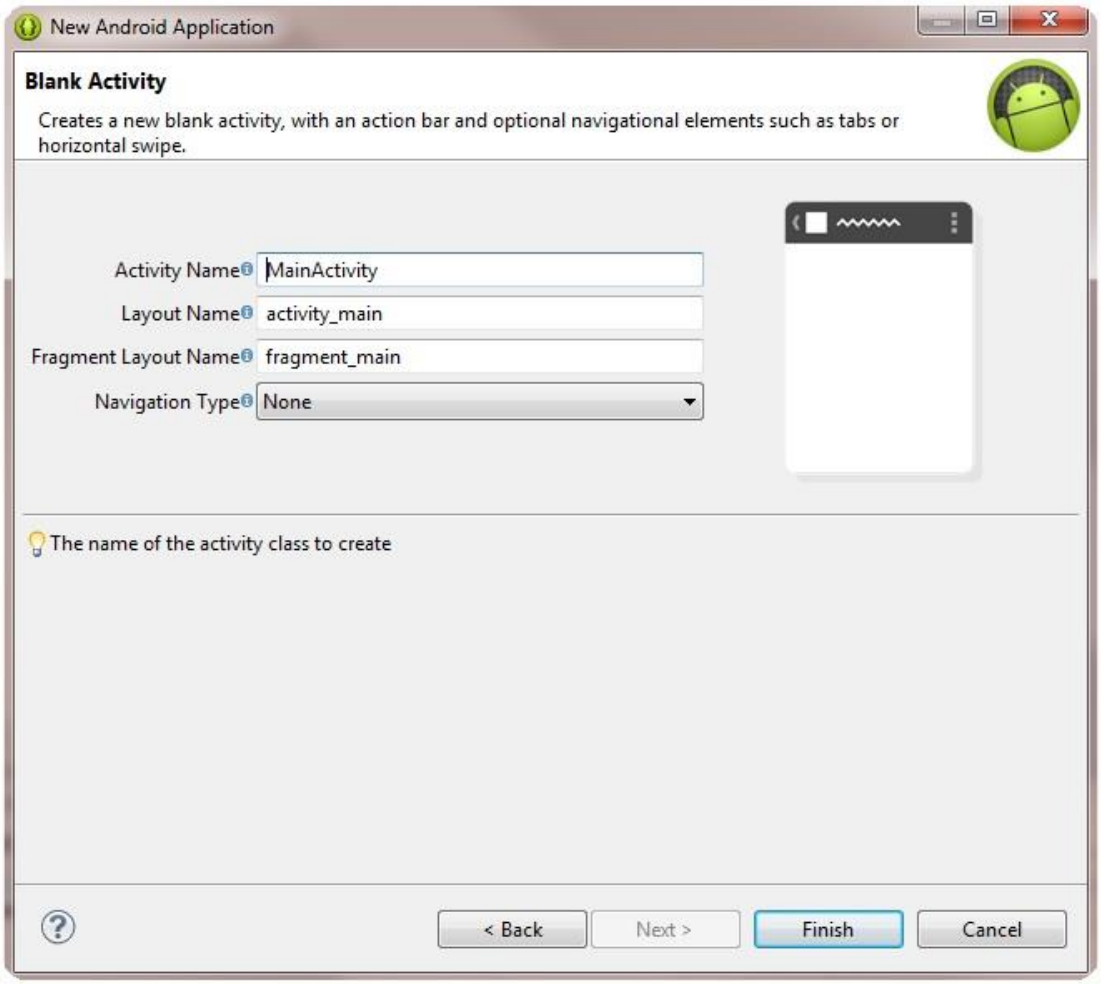


Figura 2.32: Nueva actividad en blanco para el proyecto Android

2.2.4.4 DIAGRAMA DE FLUJO DE LA APLICACIÓN EN ANDROID

El diagrama de flujo del código de la aplicación realizada en la plataforma Android se puede observar en la Figura 2.33, 2.35, 2.37 y 2.39.

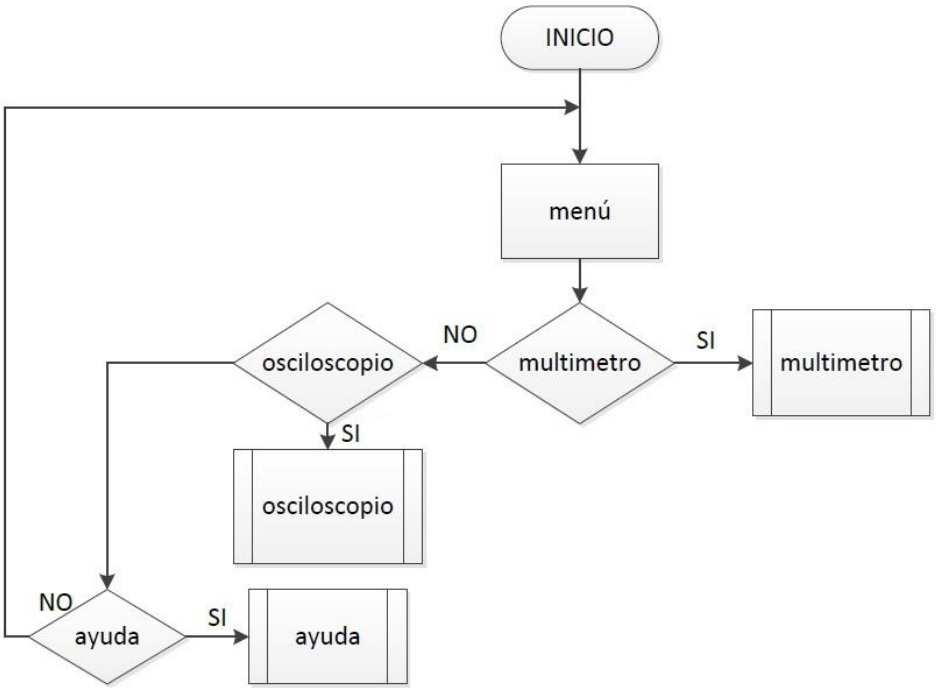


Figura 2.33: Diagrama de flujo de la Activity (Principal)

Figura 2.34: Layout de la Activity (Principal)

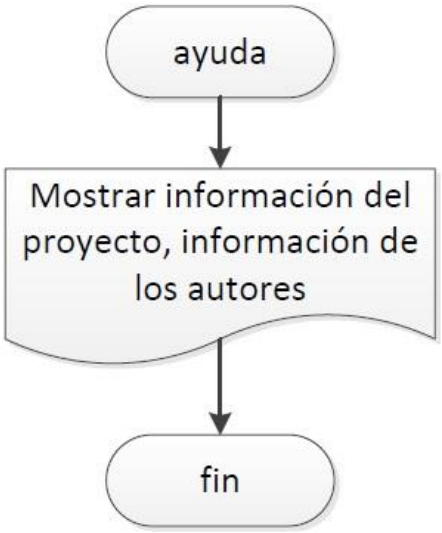


Figura 2.35: Diagrama de flujo de la Activity (Ayuda)

Figura 2.36: Layout de la Activity (Ayuda)

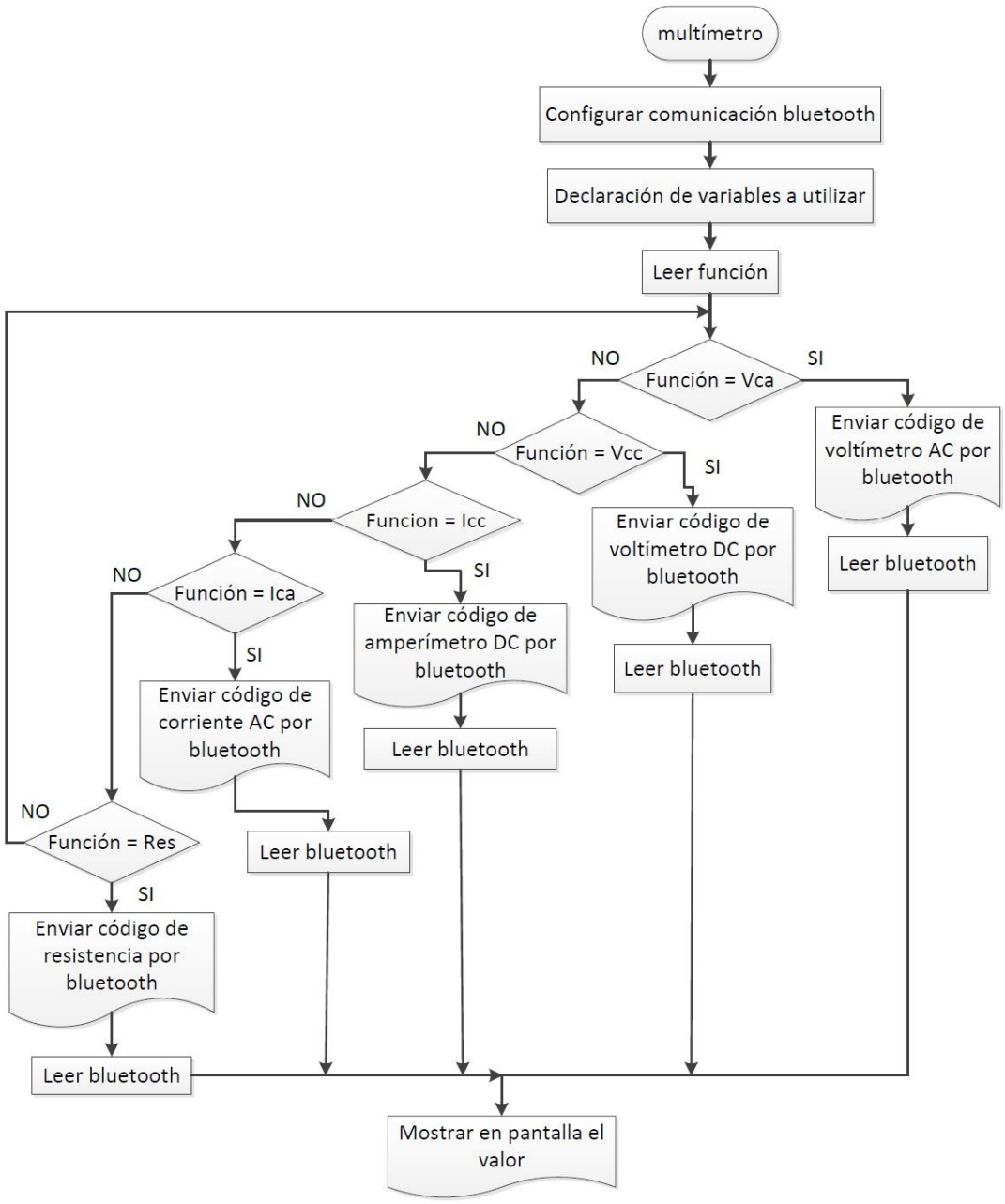


Figura 2.37: Diagrama de flujo de la Activity (Multímetro)

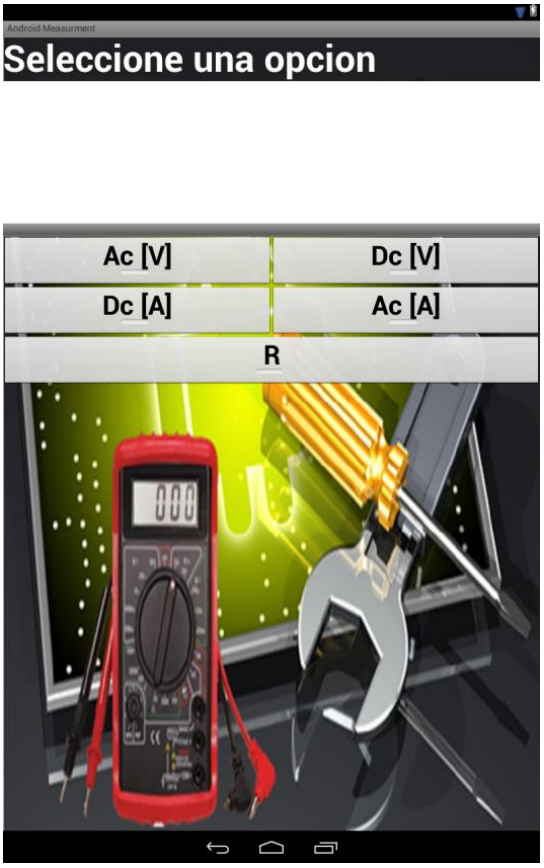


Figura 2.38: Layout de la Activity (Multímetro)

En las Figuras 2.34, 2.36, 2.38 y 2.40 se muestran los Layout’s de las actividades realizadas en el proyecto.

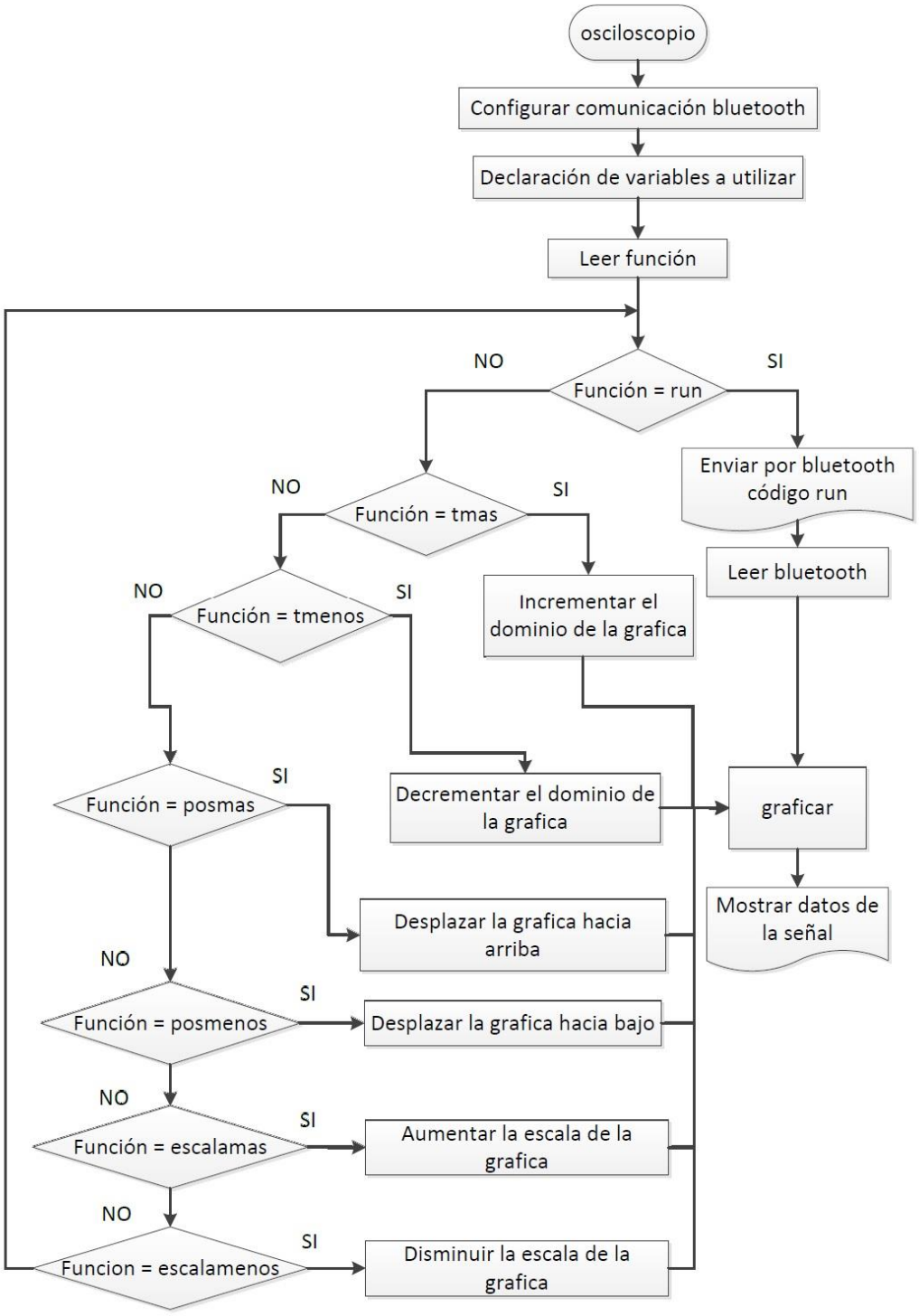


Figura 2.39: Diagrama de flujo de la Activity (Osciloscopio)

Figura 2.40: Layout de la Activity (Osciloscopio)

CAPÍTULO 3

PRUEBAS Y OPERACIÓN DEL SISTEMA

3.1 GENERALIDADES

En este capítulo se realizó el análisis del comportamiento práctico del dispositivo, en el laboratorio de circuitos electrónicos de la Universidad de las Fuerzas Armadas - ESPE Extensión Latacunga. Las pruebas realizadas al prototipo ayudaron a conocer los errores que se produjeron durante el funcionamiento para mejorar el sistema y realizar las respectivas correcciones. En tanto que los resultados de las pruebas ayudaron a la determinación de importantes conclusiones.

3.2 PRUEBAS REALIZADAS AL MULTÍMETRO IMPLEMENTADO

Una vez realizado el respectivo acondicionamiento de las variables eléctricas y la adquisición de las mismas se efectuó las pruebas para determinar el grado de confiabilidad que brinda el prototipo.

Para las pruebas de voltaje se utilizó la fuente de alimentación de LabVolt (Figura 3.1), disponible en el laboratorio de control eléctrico de la Universidad De Las Fuerzas Armadas - ESPE Extensión Latacunga, la cual es capaz de suministrar un voltaje AC variable de 0 a 208V y un voltaje DC entre 0 y 120V. Para las pruebas de corriente se conectó una carga resistiva tanto a la fuente de alimentación variable AC como DC, logrando una variación de corriente entre 0 y 8 Amperios para los dos casos.

Figura 3.1: Fuente de Alimentación de LabVolt.

Para el efecto de las pruebas del óhmetro se utilizó una década de resistencias disponible en el laboratorio de circuitos electrónicos de la Universidad de Las Fuerzas Armadas – ESPE Extensión Latacunga, la cual tiene un rango de operación de 0 a 9 MΩ.

El instrumento patrón utilizado para estas pruebas fue el multímetro Agilent 34410A. En las tablas 3.1, 3.2, 3.3 y 3.4 se indican los resultados de las pruebas realizadas al dispositivo implementado.

Para el cálculo del error se utilizó la siguiente fórmula:

$$Error\ (\%) = \frac{V_P - V_{IBP}}{V_P} * 100 \qquad (3.1)$$

Dónde:

V_P: Valor indicado por el instrumento patrón

V_{IBP}: Valor indicado por el instrumento bajo prueba

Tabla 3.1: Comparación de las mediciones realizadas para el voltaje alterno.

PATRÓN (V)	IBP (V)	ERROR (%)
0	0	0,00
5	4,8	4,00
10	9,4	6,00
15	14,8	1,33
20	19,2	4,00
30	27,3	9,00
40	38,5	3,75
50	49,67	0,66
60	60,06	0,10
70	70,6	0,86
80	80,02	0,02
90	90,4	0,44
100	101,1	1,10
110	111,1	1,00
120	121,1	0,92
130	129,2	0,62
140	139,7	0,21
150	147,9	1,40
160	159	0,63
170	169	0,59
180	181	0,56
190	189,6	0,21
200	199,4	0,30
Error promedio		±1,64

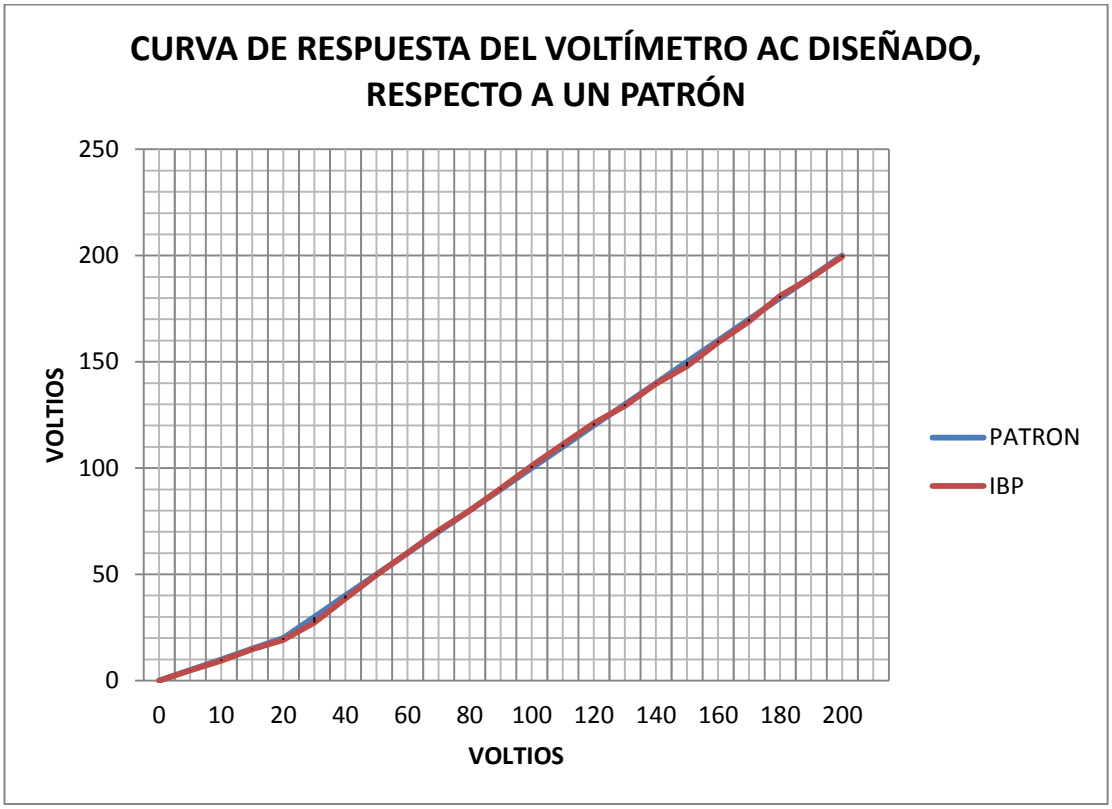


Figura 3.2: Curva de respuesta del voltímetro AC diseñado, respecto a un patrón.

Tabla 3.2: Comparación de las mediciones realizadas para el voltaje continuo.

PATRÓN (V)	IBP (V)	ERROR (%)
0	0	0,00
10	9,9	1,00
15,1	15,16	0,40
30	30,3	1,00
40	39,6	1,00
50	50,01	0,02
60	60	0,00
70	69,6	0,57
90	90,5	0,56
100	99,9	0,10
110	109,3	0,64
120	121	0,83
130	129,6	0,31
139,4	139,8	0,29
Error promedio		±0,74

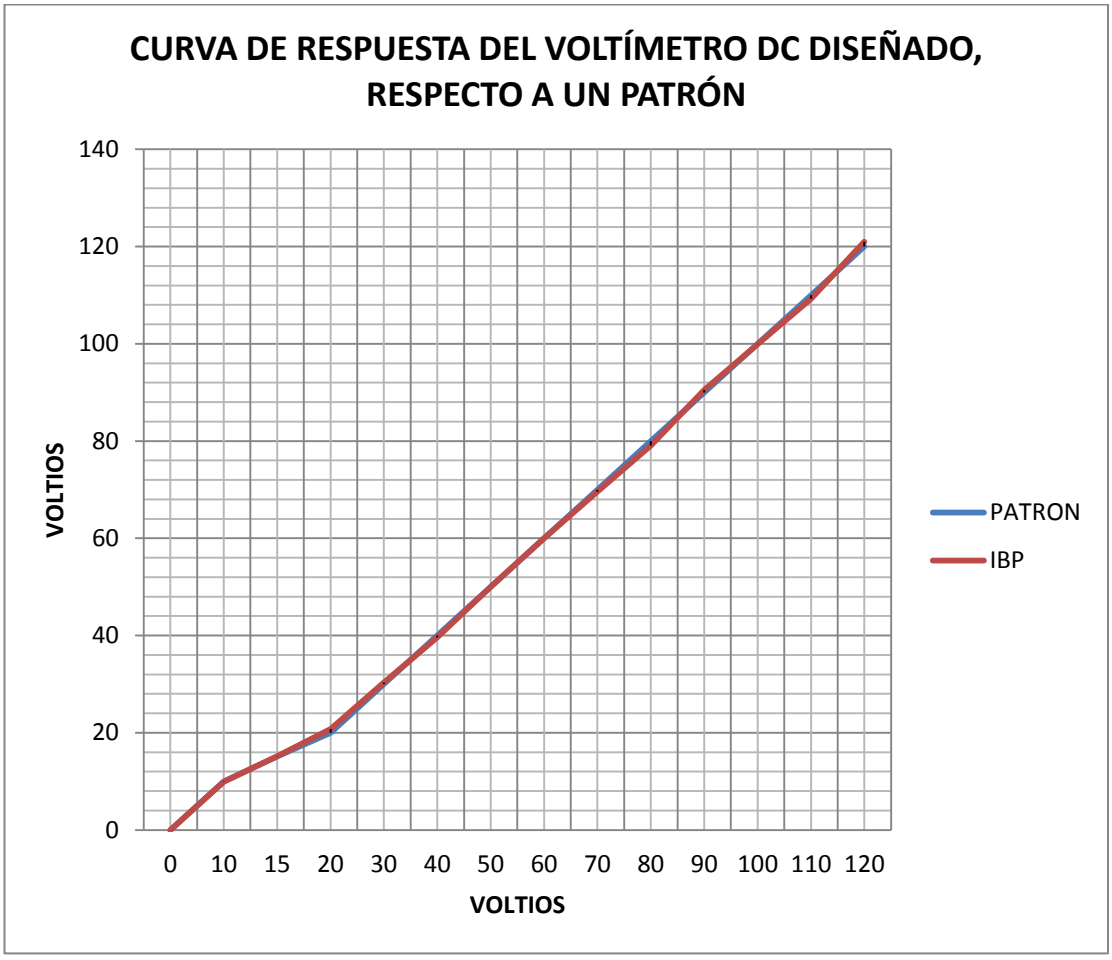


Figura 3.3: Curva de respuesta del Voltímetro DC diseñado, respecto a un patrón

Tabla 3.3: Comparación de las medidas de corriente alterna realizadas.

PATRÓN (A)	IBP (A)	ERROR (%)
0	0	0,00
1	0,93	7,00
2	1,98	1,00
3	2,94	2,00
4	4,1	2,50
5,02	5,05	0,60
6,01	5,88	2,16
7,35	7,42	0,95
Error promedio		±2,03

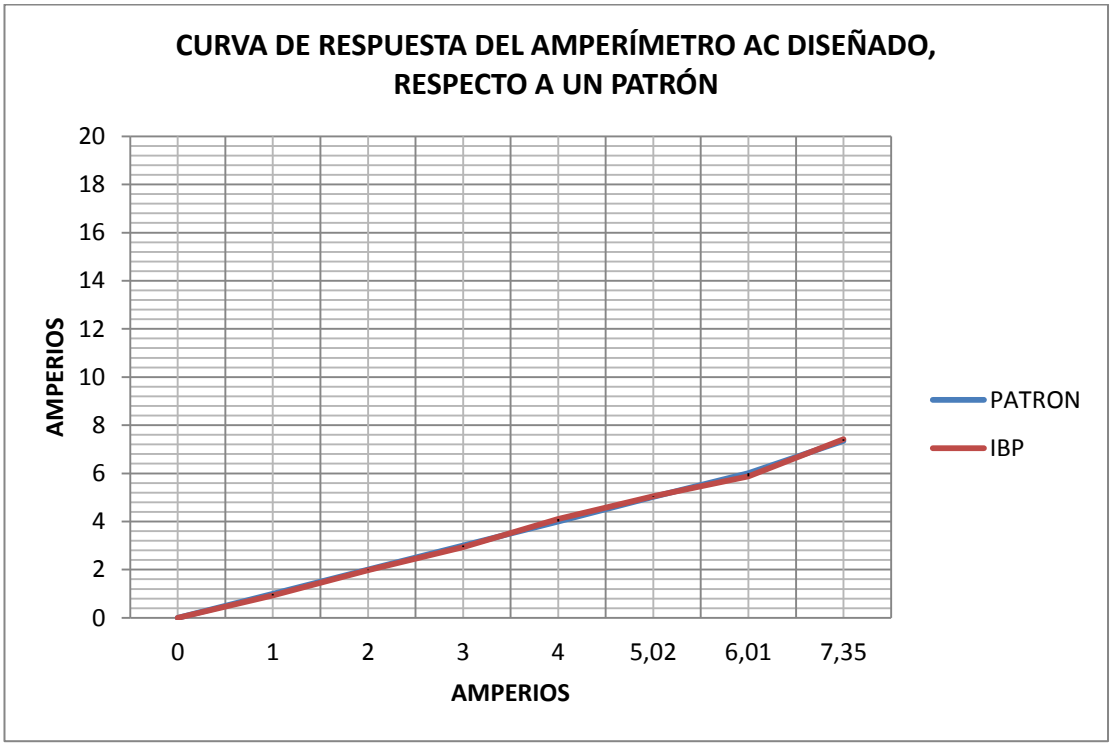


Figura 3.4: Curva de respuesta del Amperímetro AC diseñado, respecto a un patrón.

Tabla 3.4: Comparación de las medidas de corriente continua realizadas.

PATRÓN (A)	IBP (A)	ERROR (%)
0	0	0,00
1	1	0,00
2	2	0,00
3	3	0,00
4	4	0,00
5	5,01	0,20
6	6,01	0,17
7	7,01	0,14
8	8,1	1,25
8,7	8,8	1,15
Error promedio		±0,29

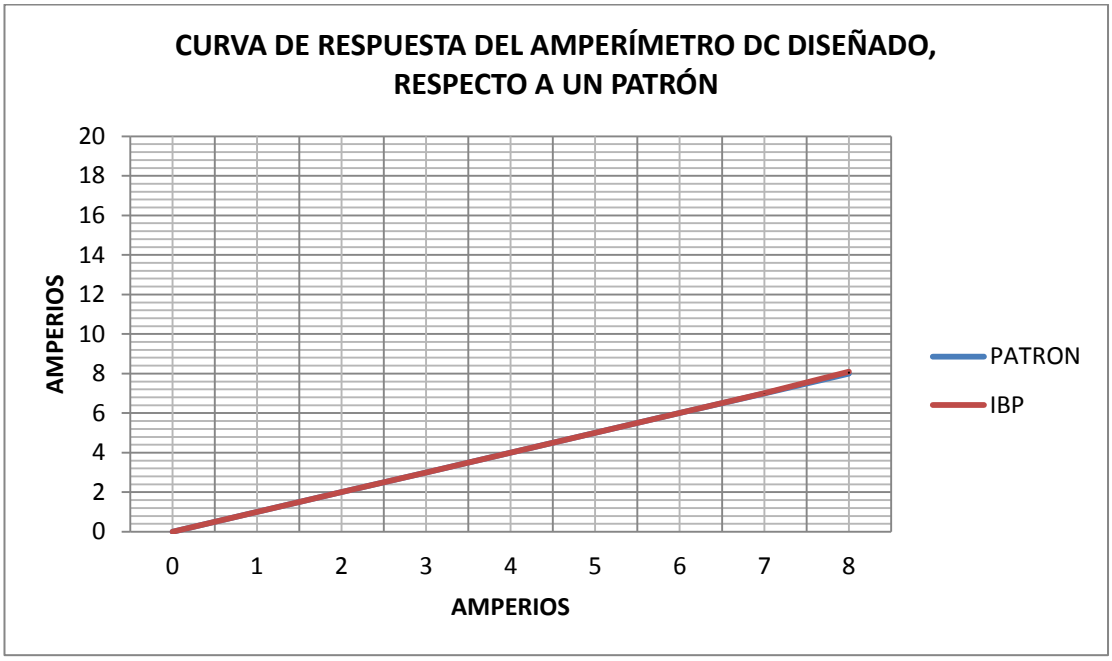


Figura 3.5: Curva de respuesta del Amperímetro DC diseñado, respecto a un patrón.

Tabla 3.5: Comparación de las mediciones de resistencia realizadas

PATRÓN (Ω)	IBP (Ω)	ERROR (%)
100	100	0,00
200	200	0,00
300	300	0,00
400	420	5,00
500	500	0,00
1000	990	1,00
1500	1510	0,67
1800	1830	1,67
2000	2030	1,50
2200	2230	1,36
2500	2550	2,00
3000	3030	1,00
3300	3350	1,52
4700	4780	1,70
5000	5100	2,00
6000	6140	2,33

7000	7180	2,57
8000	8180	2,25
9000	9200	2,22
10000	10260	2,60
11000	11260	2,36
15000	15400	2,67
30000	30900	3,00
100000	99670	0,33
200000	207000	3,50
300000	304000	1,33
500000	505000	1,00
900000	920000	2,22
1000000	1000400	0,04
2000000	2010000	0,50
Error promedio		±1,61

Figura 3.6: Curva de respuesta del óhmetro diseñado respecto a un patrón

3.3 PRUEBAS REALIZADAS AL OSCILOSCOPIO IMPLEMENTADO

Para el efecto de la prueba de funcionamiento del osciloscopio implementado se utilizó un generador de funciones Tektronix CFG253 y un osciloscopio Agilent DSO-X 2014A, existentes en el laboratorio de circuitos electrónicos de la Universidad de las Fuerzas Armadas - ESPE Extensión Latacunga. La forma en la que se llevó a cabo esta prueba fue mediante la comparación de las mediciones del dispositivo implementado y las señales obtenidas por parte del generador de funciones anteriormente mencionado. En las tablas 3.6 se indican las mediciones realizadas para esta prueba.

Tabla 3.6: Comparación de las señales mostradas en el osciloscopio Agilent DSO-X 2014A y las señales mostradas en la aplicación Android

VPK (V)	FRECUENCIA (Hz)	VRMS (V)	VPK (V)	ERROR FREC (%)	ERROR VPK (%)
10,70	4,16	7,98	11,28	4	5,42
17,10	10,49	12,86	18,19	9,02	6,37
22,50	15,24	15,91	22,50	1,04	0
1,60	18,4	1,12	1,58	3,77	1,25
3,20	53,2	2,32	3,28	0,75	2,5
8,60	208,2	6,51	9,20	2,07	6,98
10,30	100,67	7,61	10,76	1,30	4,45
13,10	78,5	9,83	13,90	3,09	6,12
16,10	59,53	12,02	17,00	0,78	5,59
18,50	21,57	13,86	19,60	1,75	5,95
	Error promedio			±2,76	±4,46

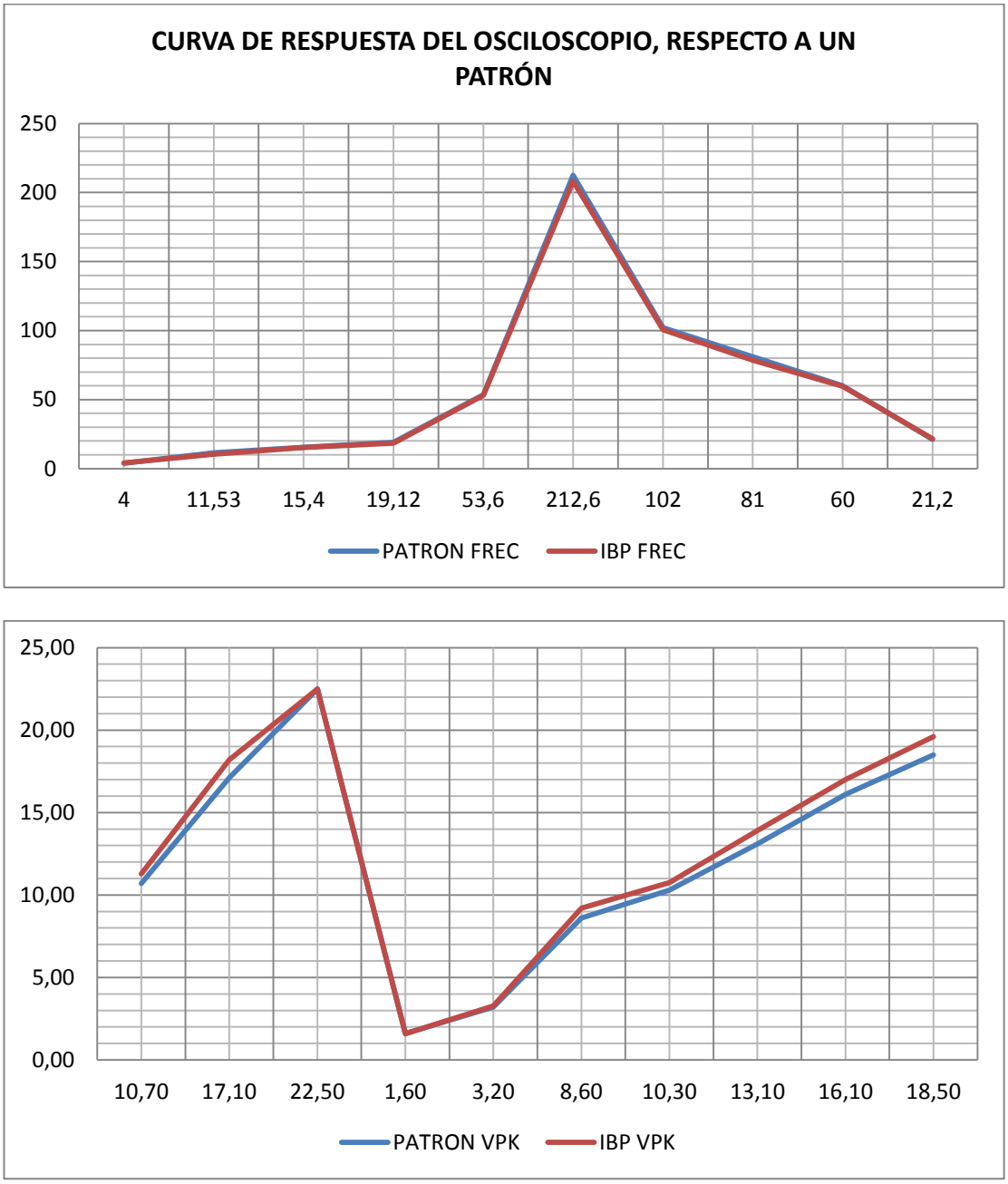


Figura 3.7: Curva de respuesta de un osciloscopio diseñado respecto a un patrón

CAPÍTULO 4

CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- Se cumplió con el objetivo principal del proyecto, el cual consiste en el diseño e implementación de un sistema de adquisición de señales eléctricas, transmisión vía Bluetooth y una interfaz gráfica en un dispositivo móvil bajo la plataforma Android.

- Para la implementación de este proyecto se utilizó la tarjeta Arduino Due que posee un microcontrolador de 32 bits, 8 conversores análogos digitales con una resolución de 12 bits.

- El multímetro implementado permite la medición de corriente y voltaje dentro de un rango aceptable de error. Además presentó una alta velocidad de actualización de los datos.

- Se cumplió con el objetivo de desarrollar un dispositivo compacto que integre las funciones de un multímetro y un osciloscopio básicos en una sola aplicación para dispositivos móviles bajo la plataforma Android.

- Se desarrolló una aplicación móvil sencilla, que satisface las necesidades tanto del multímetro como de un osciloscopio, que permita una fácil conexión entre el dispositivo de adquisición de datos y el dispositivo móvil con sistema Android, a través del protocolo de comunicación inalámbrica bluetooth.

- Los valores medidos difieren en un pequeño porcentaje de los valores calculados, debido principalmente a las tolerancias de los elementos pasivos utilizados.

- El osciloscopio implementado permite medir señales con una frecuencia de hasta 300Hz, esto se debe a las limitaciones que presentan tanto la comunicación serial como la transmisión de datos por bluetooth.

- Para la implementación de este proyecto se utilizaron dos tarjetas Arduino Due y dos módulos bluetooth, puesto que al tratar de integrar las funciones del osciloscopio dentro de la tarjeta que gestiona el multímetro se presentaron conflictos entre las tramas de las dos aplicaciones.

- La medición de la frecuencia de la señal que se presenta en la aplicación del osciloscopio se la realizó mediante la tarjeta que gestiona las funciones del multímetro, debido a que al medir con la misma tarjeta que gestiona el osciloscopio la velocidad de muestreo de la señal se reduce notablemente lo que hace que la señal mostrada en el dispositivo móvil con sistema operativo Android difiera de la señal real.

- Las señales de las variables eléctricas a medir deben ser acondicionadas a un rango de 0 a 3.3 voltios, que es el máximo voltaje que se permite a la entrada de la tarjeta Arduino Due.

- Para la programación del firmware del dispositivo se utilizó el IDE de Arduino, en su versión 1.5.2, puesto que sus versiones anteriores no poseen las librerías necesarias para la tarjeta Arduino Due.

- El desarrollo de la aplicación Android se la realizó en la plataforma de programación JAVA, con la ayuda del software de programación Eclipse, por la gran cantidad de tutoriales que se encuentran disponibles en la web.

- El desarrollo del osciloscopio se lo hizo con la ayuda de la librería androidplot que permite representar gráficamente un vector numérico de n posiciones, de una forma sencilla, además existen tutoriales y algunos ejemplos básicos de la utilización de esta librería en la web.

- La comunicación entre el módulo de adquisición de datos y el dispositivo móvil con sistema Android fue satisfactoria dentro de un rango de 10m, luego del cual la comunicación se ve interrumpida.

4.2 RECOMENDACIONES

- La programación del Arduino Due se la debe hacer usando la versión 1.5.2 o superior del IDE de Arduino.

- Para poder ingresar a la aplicación del multímetro virtual, debe estar encendido el dispositivo de adquisición de datos y desconectada la entrada del osciloscopio.

- En caso de que la aplicación del multímetro virtual no se conecte automáticamente al módulo de adquisición de datos, se debe salir de la aplicación y emparejar el bluetooth del módulo de adquisición de datos con el del dispositivo Android.

- Para emparejar los módulos bluetooth de la tarjeta de adquisición de datos, con el del dispositivo Android por primera vez, se debe ingresar una contraseña que es común (“1234”), tanto para el bluetooth llamado “arduino” y el “osciloscopio”.

- Antes de iniciar la aplicación en el dispositivo Android se debe encender el bluetooth del mismo.

BIBLIOGRAFÍA

- ALCALDE, P. (2003). Principios Fundamentales de Electrónica. Madrid: Thomson Paraninfo, S.A.
- BOYLESTAD, R. NASHELSKY, L. (1997). Fundamentos de Electrónica. Naucalpan de Juárez, Edo. de México: Prentice Hall Hispanoamericana, S.A.
- EDMINISTER, J. (1987). Circuitos eléctricos compendios schaum. Madrid: McGraw-Hill.
- NILSSON, J. RIEDEL, S. (2005). Circuitos eléctricos. Madrid: Pearson Prentice Hall.
- LOVEDAY, G. (1995). Diseño de Hardware Electrónico. Madrid: Paraninfo.
- MULLER, N. (2002). Tecnología Bluetooth. Madrid: McGraw-Hill.

LINKGRAFÍA

- ARDUINO. (2010). *Arduino*. Obtenido de Arduino: http://www.arduino.cc/es/ [citado el 05 de enero del 2014]
- BARRIGA, H. (21 de ENERO de 2013). *WIRING*. Obtenido de WIRING: http://www.wiring.org.co/ [citado el 15 de enero del 2014]
- BEN FRY, C. R. (2004). *PROCESSING2*. Obtenido de PROCESSING2: http://www.processing.org/ [citado el 25 de febrero del 2014]
- ECURED. (2010). CONOCIMIENTO CON TODOS Y PARA TODOS. Obtenido de *ECURED*: http://www.ecured.cu/index.php/Potencia_el%C3%A9ctrica. [citado el 27 de febrero del 2014]
- ELECTRÓNICA, C. C. (2005). *ESTRUCTURA Y HERRAMIENTAS DESARROLLO*. Obtenido de uCs ARM: http://www.redeweb.com/_txt/646/66.pdf [citado el 15 de marzo del 2014]
- Electronics, D. (2014). *Dynamo Electronics*. Obtenido de Dynamo Electronics: http://www.dynamoelectronics.com/dynamo-tienda-virtual.html?page=shop.product_details&flypage=dynamo.tpl&product_id=1044&category_id=82 [citado el 19 de marzo del 2014]
- MANUEL BÁEZ, Á. B. (s.f.). *INTRODUCCIÓN A ANDROID*. Obtenido de G-TeC: http://pendientedemigracion.ucm.es/info/tecnomovil/documentos/android.pdf [citado el 22 de abril del 2014]
- MINERVA, M. (31 de ENERO de 2013). *SISTEMAS OPERATIVOS ANDROID*. Obtenido de BLOGSPOT:

http://androidsistemasopearativos.blogspot.com/ [citado el 25 de abril del 2014]

- MOLINA, M. (28 de junio de 2012). *electromec*. Obtenido de electromec: http://www.profesormolina.com.ar/electromec/concep_fundamen.htm [citado el 28 de abril del 2014]

- VELASQUEZ, I. (2010). *INELE*. Obtenido de Departamento de Ingeniería Eléctrica de la Frontera: http://www.inele.ufro.cl/apuntes/Instrumentacion_y_Control/Ivan_Velazquez/Catedra/Capitulo%207.%20Instrumentacion%20Virtual.pdf [citado el 30 de mayo del 2014]

ANEXOS

ANEXO A

GLOSARIO DE TÉRMINOS

A

A2DP, 34, 36

(Advanced Audio Distribution Profile), distribución de audio avanzada. Define cómo se puede propagar un stream de audio (mono o estéreo) entre dispositivos a través de una conexión Bluetooth.

Acid3, 33

Es un sitio hecho por WaSP (Web Standards Project / grupo de desarrolladores web profesionales) que pone a prueba los navegadores con los estándares web, especialmente los de DOM (Document Object Model / Modelo en Objetos para la Representación de Documentos) y ECMAScript.

ADQUISICIÓN DE DATOS, 1, 9, 41, 42

Consiste en la toma de muestras del mundo real (sistema analógico) para generar datos que puedan ser manipulados por un ordenador u otro sistema electrónico (sistema digital).

AMPERÍMETRO, 1, 41

Es un instrumento que se utiliza para medir la intensidad de corriente que está circulando por un circuito eléctrico, presentando directamente sobre su escala calibrada las unidades empleadas para ello denominadas amperios o bien fracciones de amperios, la medida deseada.

ANDROID, 29, 30, 31, 32, 33, 34, 35, 36, 39, 41, 92, 127, 142, 152, 157, 158, 175, 176

Es un sistema operativo basado en el kernel de Linux diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o tabletas.

Android Inc, 29

Fue una compañía de software ubicada en Palo Alto, California. Desarrolló durante sus inicios Android, un sistema operativo basado en Linux y orientado a dispositivos móviles, como teléfonos inteligentes y tabletas. La compañía fue

adquirida por la multinacional Google en julio de 2005. El desarrollo de Android continuó de la mano de Google y la Open Handset Alliance, un conglomerado de fabricantes y desarrolladores de hardware, software y operadoras de servicio celular.

ARDUINO, 17, 18, 19, 20, 41, 44, 47, 94, 95, 127, 142, 143, 176
Es una plataforma electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Se creó para artistas, diseñadores, aficionados y cualquiera interesado en crear entornos u objetos interactivos.

ARQUITECTURA ABIERTA, 1
Es un tipo de arquitectura de ordenadores o arquitectura de software que permite añadir, modernizar y cambiar sus componentes.

ARQUITECTURA RISC, 14
(Inglés Reduced Instruction Set Computer / Computador con Conjunto de Instrucciones Reducidas) es un tipo de diseño de CPU generalmente utilizado en microprocesadores o microcontroladores con características fundamentales como: instrucciones de tamaño fijo y presentadas en un reducido número de formatos; sólo las instrucciones de carga y almacenamiento acceden a la memoria de datos; además estos procesadores suelen disponer de muchos registros de propósito general.

AVRCP, 34, 36
(Audio-Video Remote Control Profile), control remoto de audio-vídeo, diseñado para ofrecer una interfaz estándar para el control de televisores y aparatos de música entre otros, de forma que un mando único pueda agrupar todo el control. Puede usarse junto con A2DP o VDP.

B

BLUETOOTH, 3, 21, 22, 23, 24, 25, 26, 27, 33, 34, 36, 37, 41, 131, 132, 133, 137, 146, 155, 176

Es una especificación industrial para Redes Inalámbricas de Área Personal (WPAN) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia.

BUGS, 36
Es un error de software, un error o fallo en un programa de computador o sistema de software que desencadena un resultado indeseado.

BYTECODE JAVA, 33
Es el tipo de instrucciones que la máquina virtual Java ejecuta, usualmente es el resultado de utilizar un compilador del lenguaje de programación Java (como javac), pero puede ser generado desde otros lenguajes.

C

CAMCORDER, 36
(Videocámara), es un dispositivo portátil para registrar la imagen y el sonido en el mismo soporte. Por lo tanto, combina las funciones de una cámara de televisión con las de un vídeo, de ahí su nombre, video-cámara de estos dos términos. Es una evolución tecnológica de la generación anterior, que involucraba dos unidades separadas.

CORRIENTE, 20, 41, 167
Es el flujo de carga eléctrica por unidad de tiempo que recorre un material. Se debe al movimiento de las cargas (normalmente electrones) en el interior del material.

D

DALVIK, 29, 33
Es una máquina virtual especializada, diseñada específicamente para Android y optimizada para dipositivos móviles que funcionan con batería y que tienen memoria y procesador limitados.

DIFERENCIA DE POTENCIAL (d.d.p), 4
Es la diferencia de potencial eléctrico entre dos cuerpos.

F

FRECUENCIA, 20, 22
Es una magnitud que mide el número de repeticiones por unidad de tiempo de cualquier fenómeno o suceso periódico.

H

HARDWARE, 8, 9, 13, 17, 24, 29, 32, 40
Se refiere a todas las partes tangibles de un sistema informático; sus componentes son: eléctricos, electrónicos, electromecánicos y mecánicos. Son cables, gabinetes o cajas, periféricos de todo tipo y cualquier otro elemento físico involucrado.

K

KERNEL, 30, 31, 32
Núcleo del sistema operativo

L

LINUX, 10, 15, 16, 29, 30, 31, 32, 36, 37, 39
Es un núcleo libre de sistema operativo basado en Unix. Es uno de los principales ejemplos de software libre y de código abierto.

LMP, 25
Es el protoclo que se encarga de la conectividad entre dos dispositivos bluetooth, temas de calidad del servico, etc.

M

MICROPROCESADOR, 1, 41
Es el circuito integrado central y más complejo de un sistema informático. Es un circuito integrado conformado por millones de componentes electrónicos.

Constituye la unidad central de procesamiento (CPU) de un PC catalogado como microcomputador.

N

NEAR FIELD COMMUNICATION (NFC), 22

Comunicación de Campo Cercano, es una tecnología de comunicación inalámbrica, de corto alcance y alta frecuencia que permite el intercambio de datos entre dispositivos.

O

OBJECT PUSH PROFILE (OPP), 34

El perfil de envío de objeto, contenida en muchos de los teléfonos móviles, incluyendo los construidos en el sistema operativo Android, no es una aplicación, sino que es un componente de la tecnología Bluetooth interno que permite llevar a cabo una tarea inalámbrica. El OPP no es sino uno de los muchos perfiles de Bluetooth, o modos, que se utilizan en dispositivos de computación, tales como teléfonos móviles y ordenadores personales.

ÓHMETRO, 1, 41

Los óhmetros son instrumentos que miden la resistencia de los circuitos y de los componentes eléctricos. La unidad de medida de la resistencia es el ohmio (Ω). Básicamente, la resistencia es un indicador de la dificultad presentada a una corriente eléctrica circulando en un circuito.

OPEN HANDSET ALLIANCE, 29

Es una alianza comercial de 84 compañías que se dedica a desarrollar estándares abiertos para dispositivos móviles. Algunos de sus miembros son Google, HTC, Dell, Intel, Motorola, Qualcomm, Texas Instruments, Samsung, LG, T-Mobile, Nvidia y Wind River Systems.

OpenGL ES, 33
(OpenGL for Embedded Systems), es una variante simplificada de la API gráfica OpenGL, define y promueve el Grupo Khronos, un consorcio de empresas dedicadas a hardware y software gráfico interesadas en APIs gráficas y multimedia.

P

PICONET, 23
Red informática cuyos nodos se conectan utilizando Bluetooth.

POTENCIAL ELÉCTRICO, 5
Se dice que un cuerpo cargado posee una energía o potencial.

PWM, 18, 20
(Pulse-width modulation), modulación por ancho de pulsos de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

S

SCATTERNET, 23
Es una colección de varias picoredes en operación que se traslapan en tiempo y en espacio.

SISTEMA OPERATIVO, 3, 10, 15, 29, 31, 32, 34, 36
Es un programa o conjunto de programas que en un sistema informático gestiona los recursos de hardware y provee servicios a los programas de aplicación, ejecutándose en modo privilegiado respecto de los restantes y anteriores próximos y viceversa.

SMARTPHONE'S, 29, 39
(Teléfono inteligente), es un teléfono móvil construido sobre una plataforma informática móvil, con una mayor capacidad de almacenar datos y realizar

actividades semejantes a una minicomputadora y conectividad que un teléfono móvil convencional. Hace referencia a la capacidad de usarse como un ordenador de bolsillo, llegando incluso a reemplazar a un ordenador personal en algunos casos.

SOFTWARE, 8, 9, 10, 13, 17, 25, 26, 29, 31, 32, 33, 34, 95, 127, 142, 143, 153, 157, 158, 175, 176, 177

Es el equipamiento lógico o soporte lógico de un sistema informático, que comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas.

SQLite, 33

Es una base de datos liviana, que es usada para propósitos de almacenamiento de datos, es un proyecto de dominio público creado por D. Richard Hipp.

STREAMING, 33

(También denominado lectura en continuo, difusión en flujo, lectura en tránsito, difusión en continuo, descarga continua o mediaflujo) es la distribución de multimedia a través de una red de computadoras de manera que el usuario consume el producto, generalmente archivo de video o audio, en paralelo mientras se descarga. La palabra streaming se refiere a: una corriente continua (que fluye sin interrupción).

T

TABLET'S, 28

Es una computadora portátil de mayor tamaño que un teléfono inteligente o una PDA, integrada en una pantalla táctil (sencilla o multitáctil) con la que se interactúa primariamente con los dedos o un estilete (pasivo o activo), sin necesidad de teclado físico ni ratón.

TETHERING, 37

También denominado “anclaje a red” al proceso por el cual un dispositivo móvil con conexión a Internet actúa como pasarela para ofrecer acceso inalámbrico a la

red a otros dispositivos, cualesquiera que estos sean, asumiendo dicho dispositivo móvil un papel similar al de un módem o enrutador inalámbrico. Esto se puede realizar mediante una conexión LAN inalámbrica (Wi-fi), Bluetooth o físicamente mediante un cable, como el USB. Este método puede ser utilizado para permitir el acceso a internet a un dispositivo, como para ahorrar dinero mediante el anclaje a una conexión de tarifa plana.

Turn-By-Turn, 36
Es una característica de algunos dispositivos de navegación GPS, donde las instrucciones de una ruta seleccionada continuamente presentan al usuario en forma de instrucciones habladas y visuales. El sistema lleva al usuario hasta a la fecha de la mejor ruta al destino, y se actualizan a menudo en función de factores tales como las condiciones cambiantes de tráfico y carreteras.

V

VOLTAJE, 20, 41, 166, 181
Es una magnitud física que cuantifica la diferencia de potencial eléctrico entre dos puntos.

VOLTÍMETRO, 1, 41
Es un instrumento que sirve para medir la diferencia de potencial entre dos puntos de un circuito eléctrico.

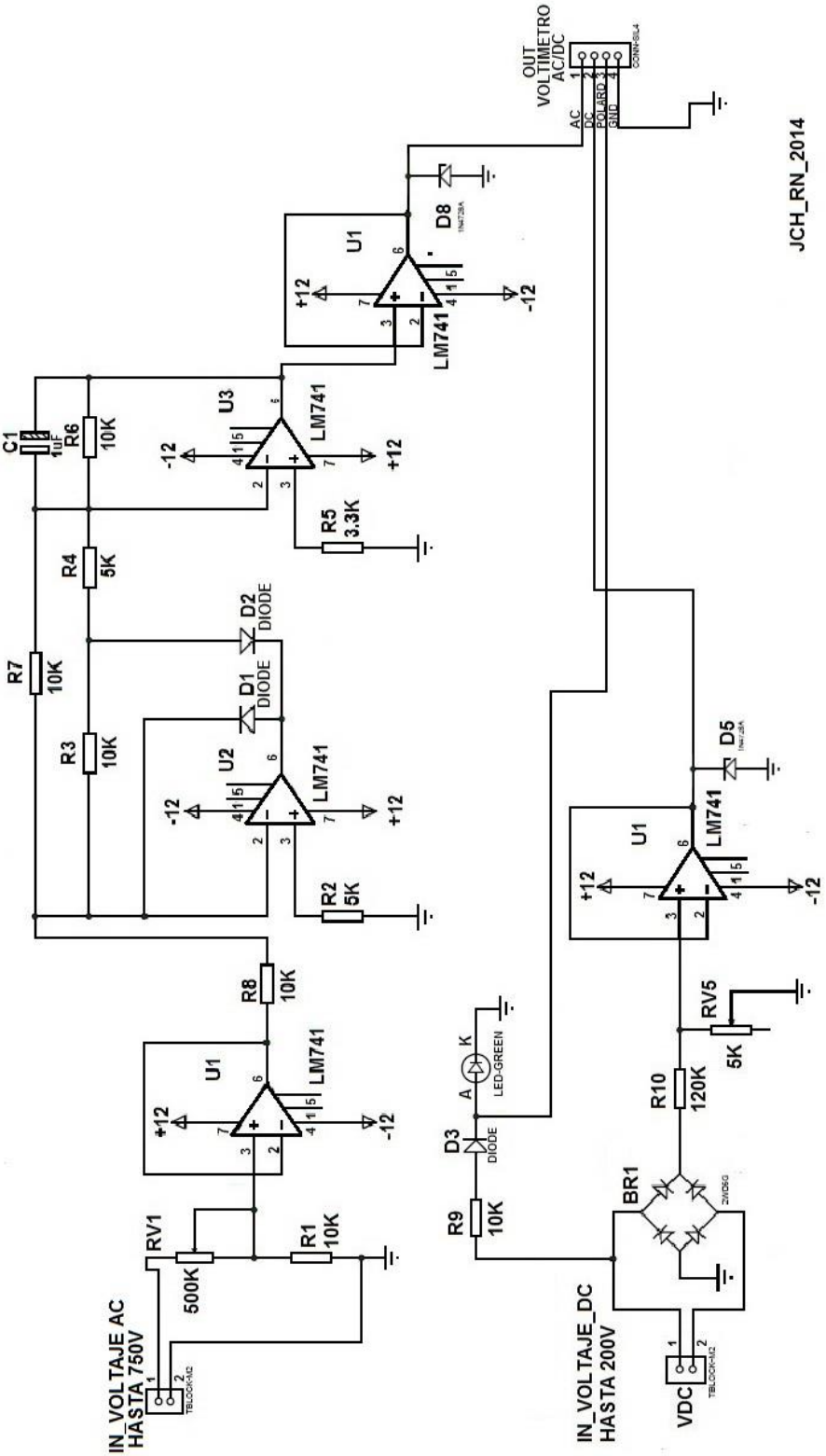
W

WebKit, 32, 33
Es una plataforma para aplicaciones que funciona como base para el navegador web.

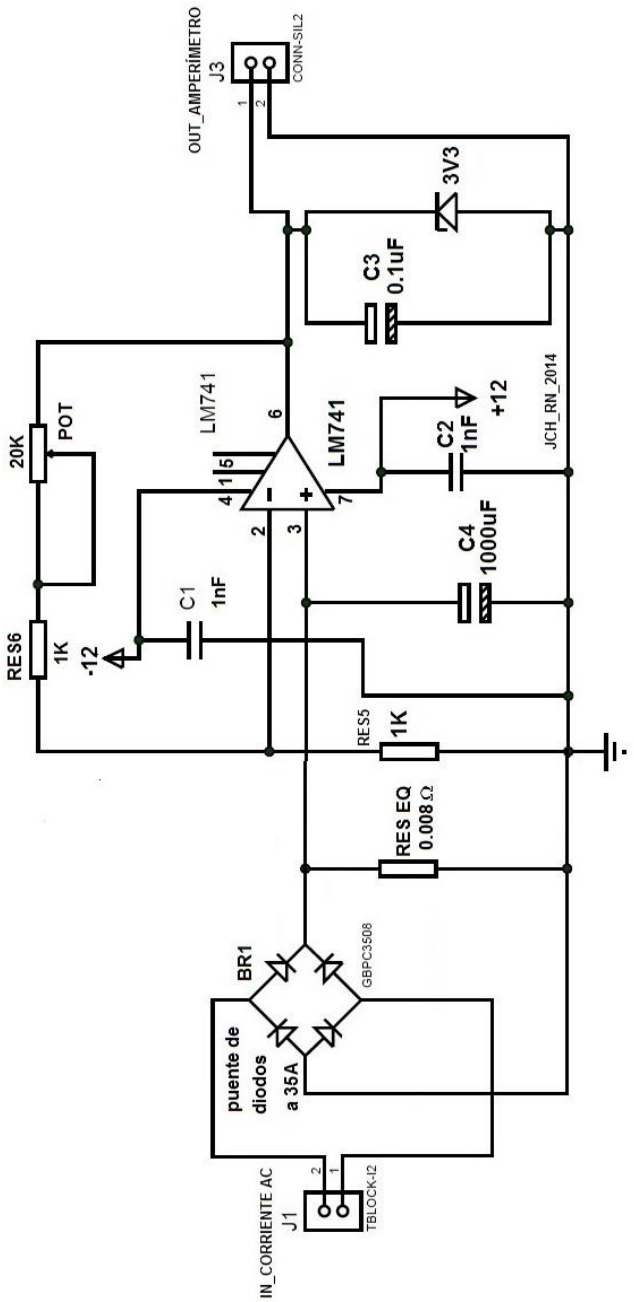
ANEXO B

CIRCUITOS ESQUEMÁTICOS

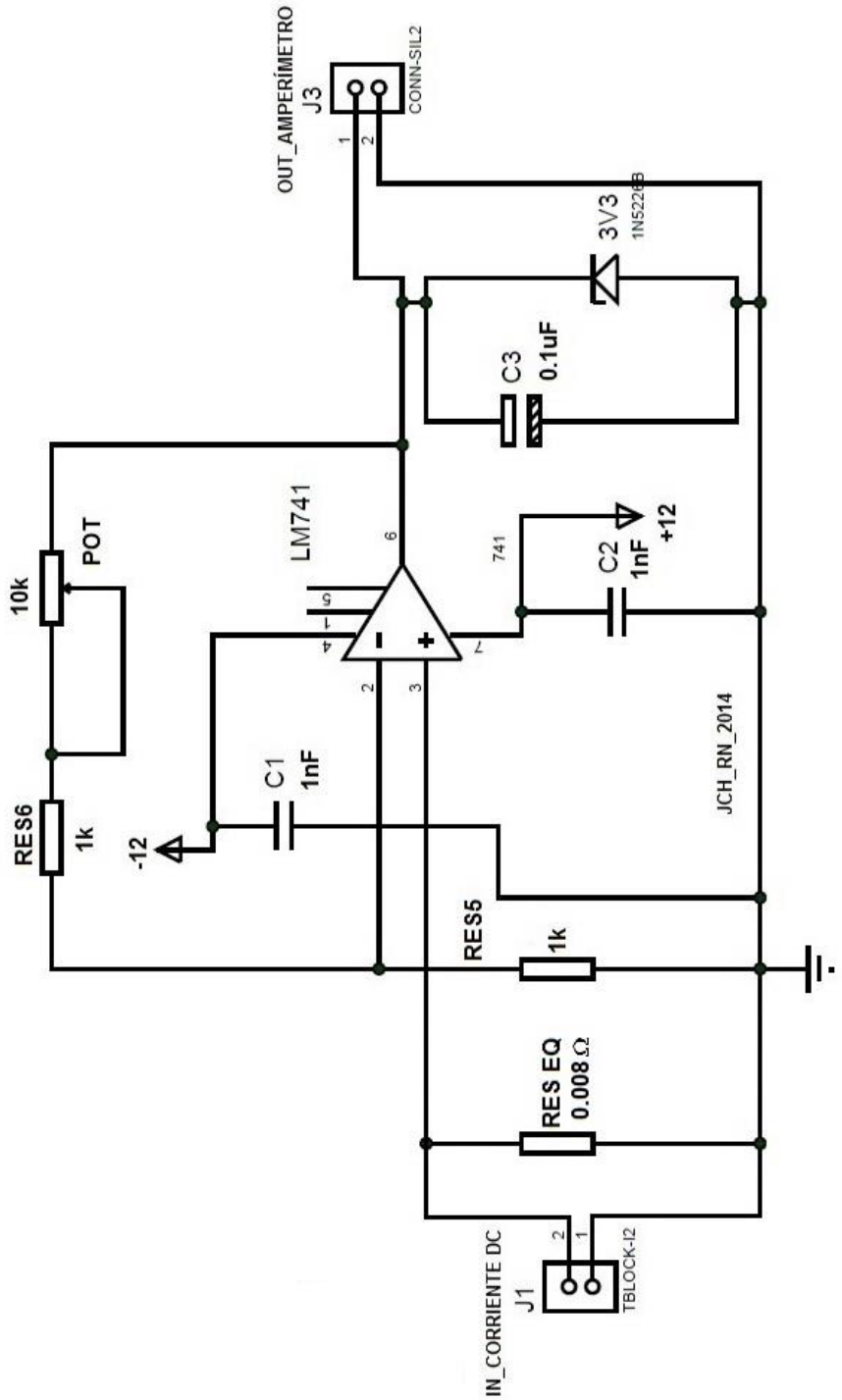
VOLTÍMETRO AC/DC



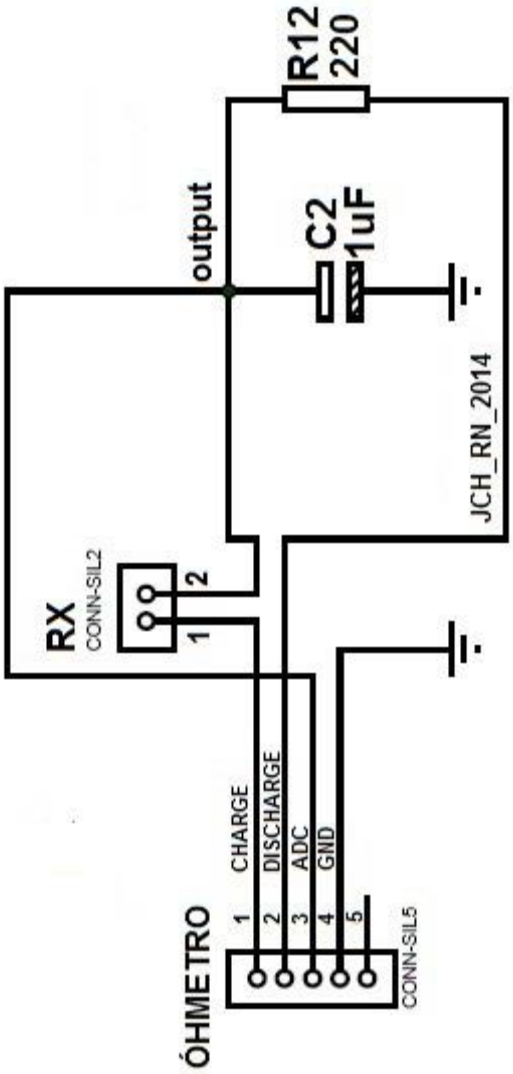
AMPERÍMETRO AC



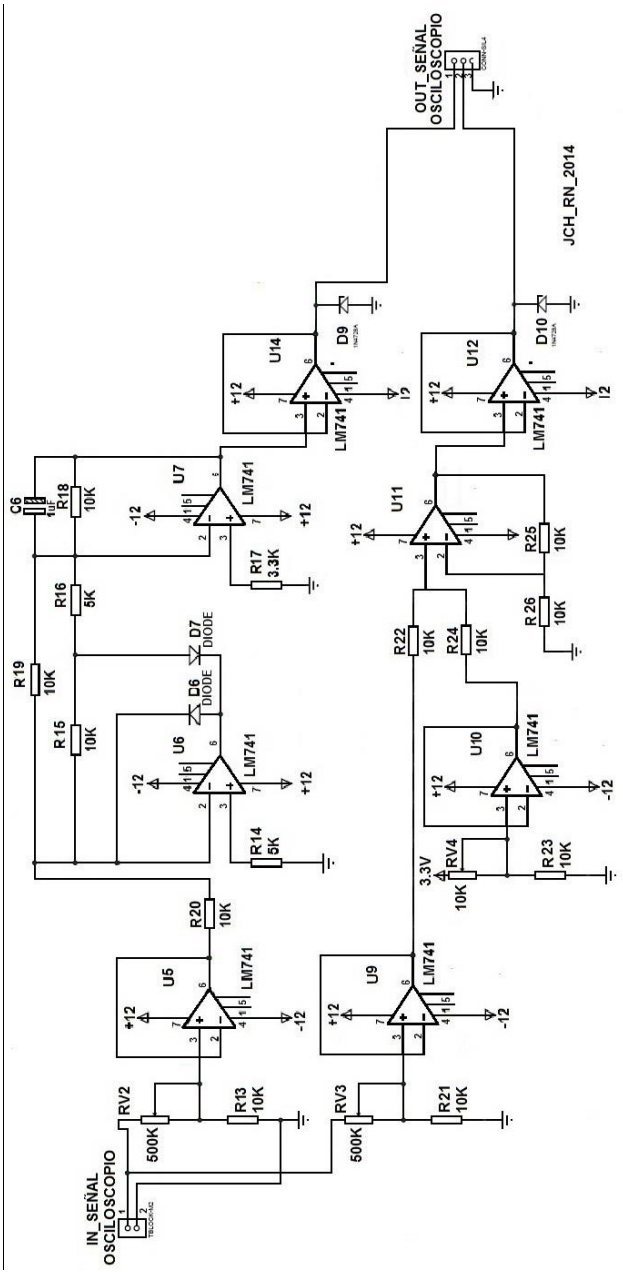
AMPERÍMETRO DC



ÓHMETRO


R12
220
output
C2
1uF
JCH_RN_2014
RX
CONN-SIL2
1
2
ÓHMETRO
1
CHARGE
2
DISCHARGE
3
ADC
4
GND
5
CONN-SIL5

OSCILOSCOPIO



TARJETA ARDUINO DUE

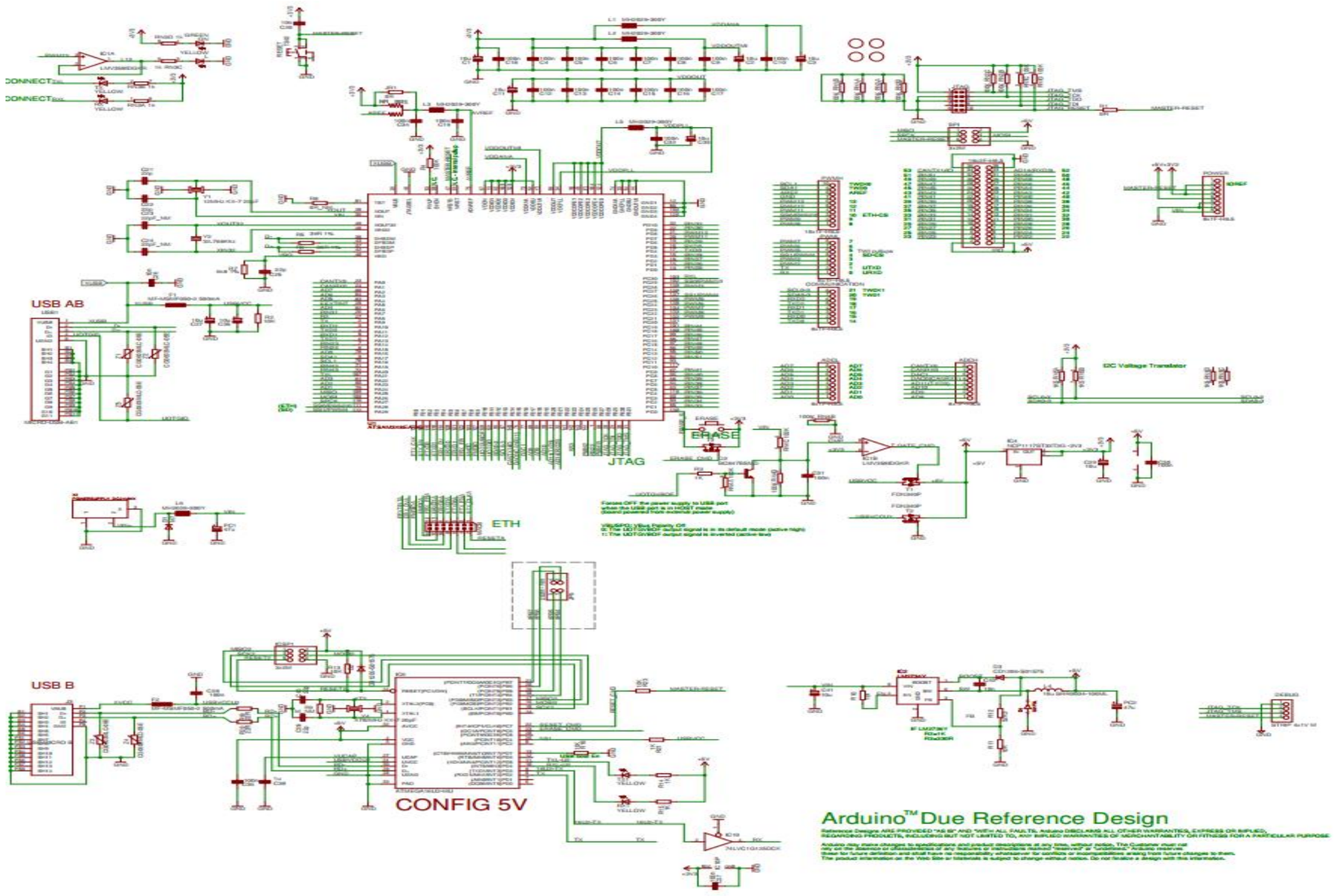
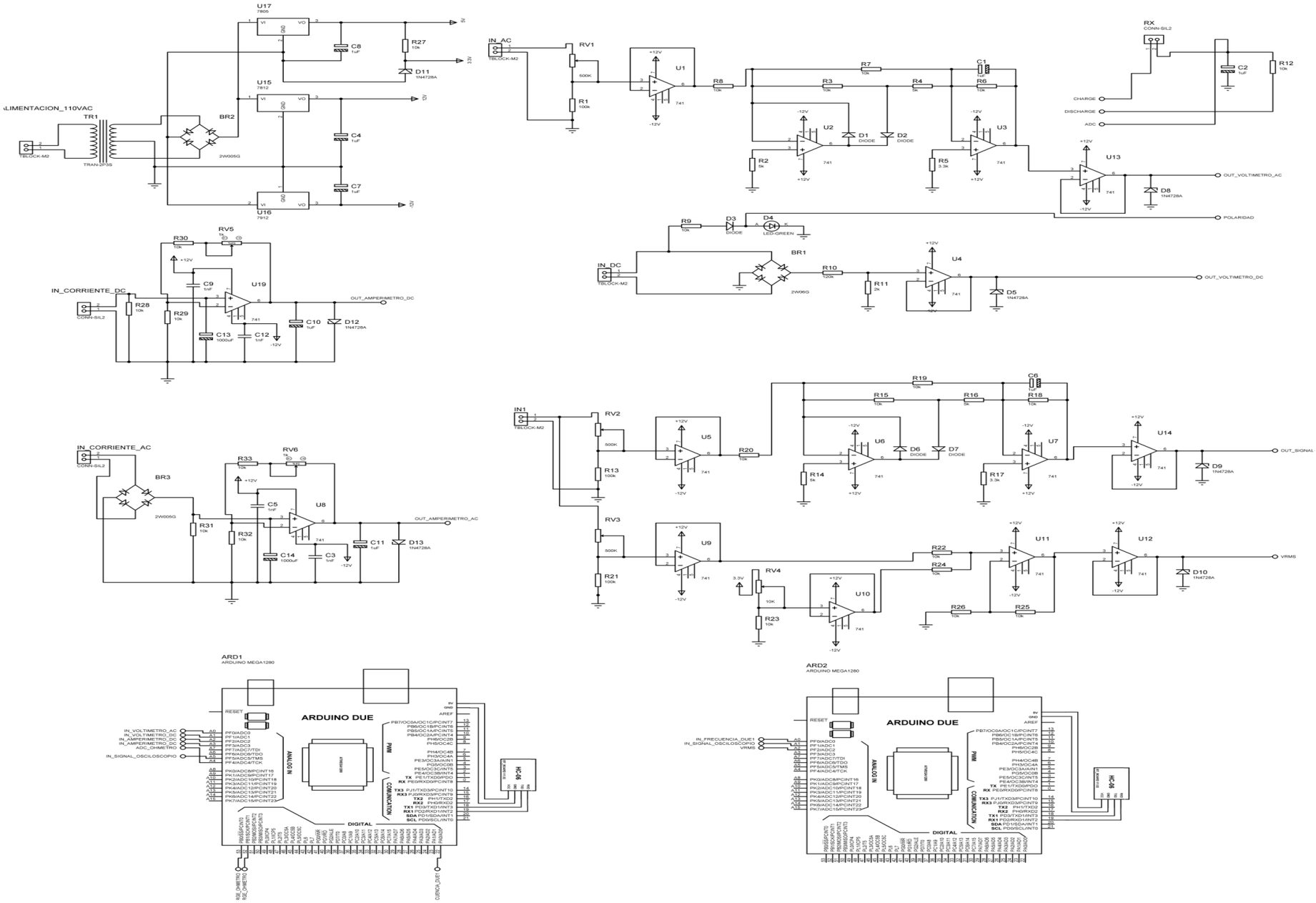


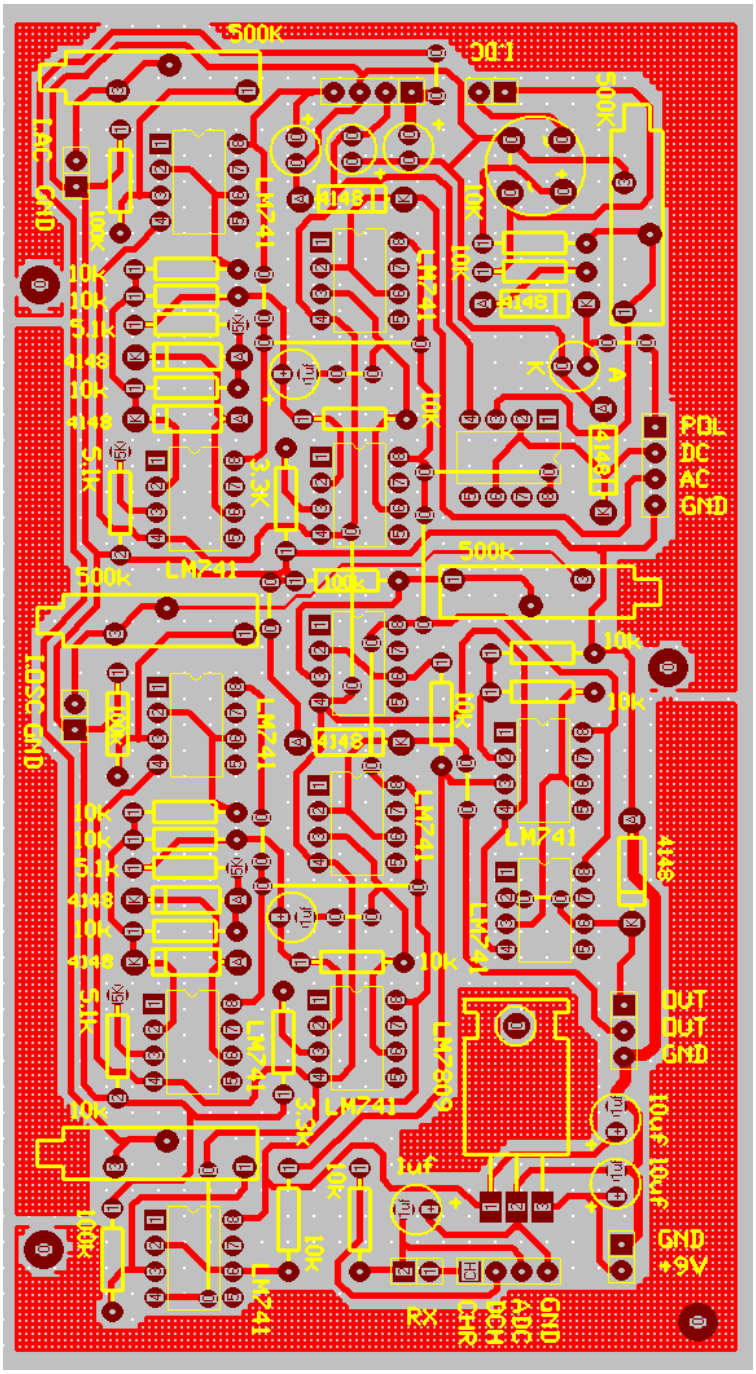
DIAGRAMA ESQUEMÁTICO GENERAL



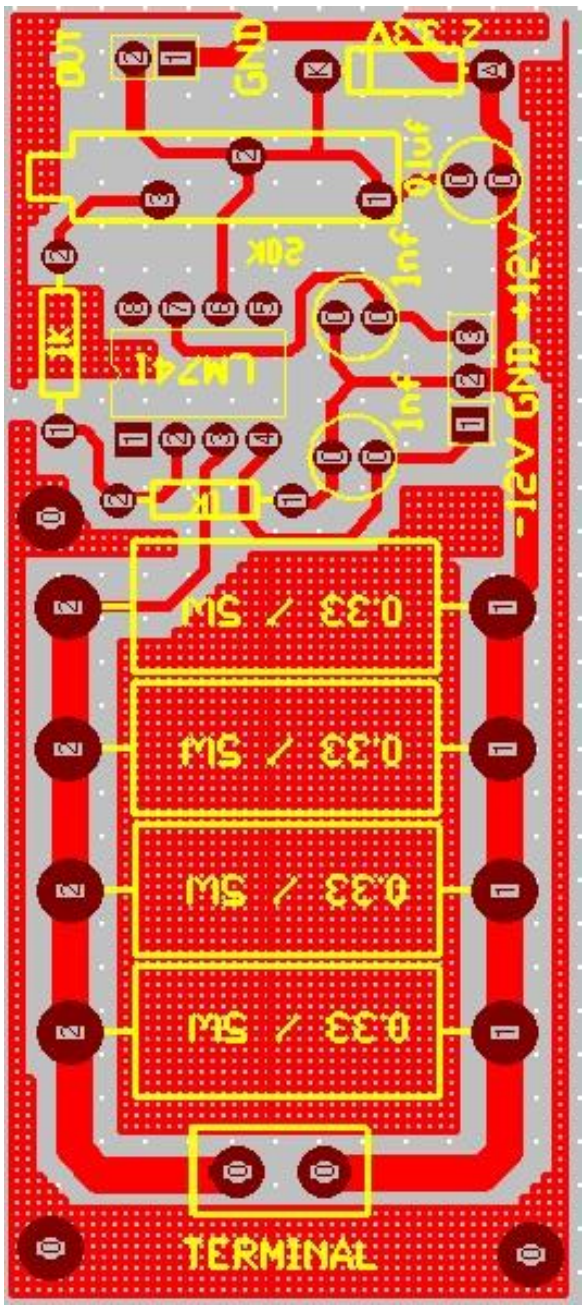
ANEXO C

CIRCUITOS IMPRESOS

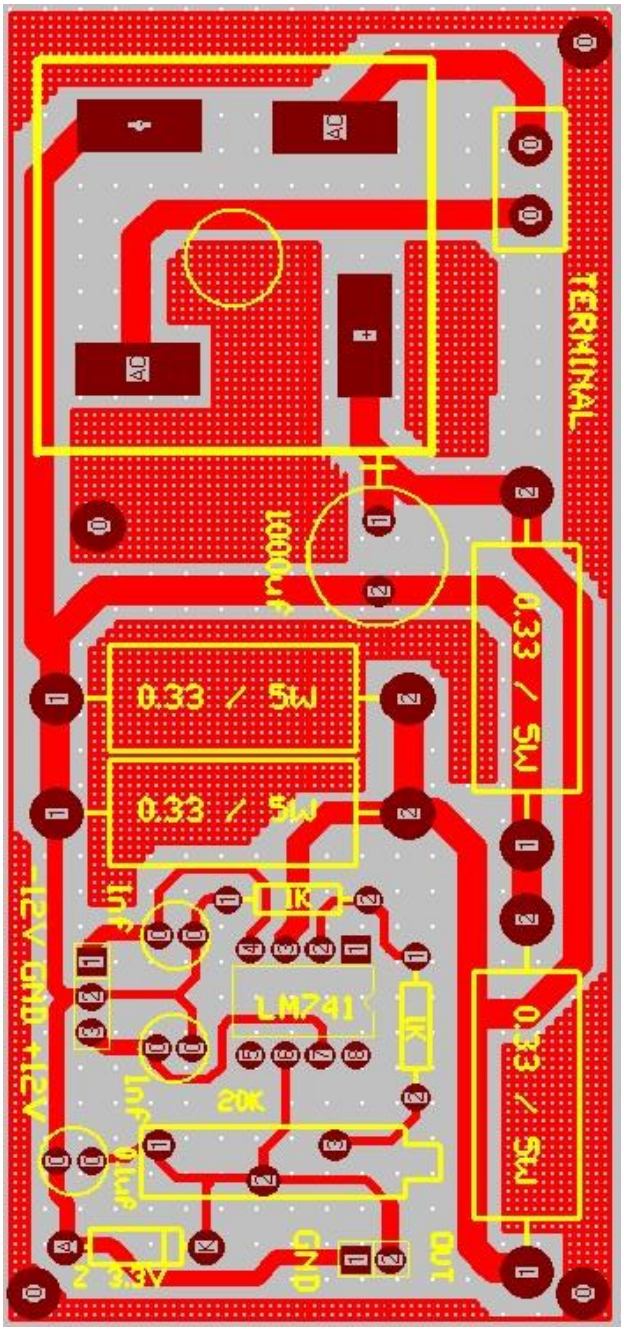
VOLTÍMETRO AC/DC - ÓHMETRO
OSCILOSCOPIO



AMPERÍMETRO_AC



AMPERÍMETRO_DC



ANEXO D

PROGRAMACIÓN ANDROID

PROGRAMACIÓN ANDROID

CLASE INICIO

```
package tesis.android;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Set;
import java.util.UUID;

import android.os.Bundle;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ImageButton;
import android.widget.Toast;

public class Inicio extends Activity implements OnClickListener{

	ImageButton ayuda, multimetro, osciloscopio;
	private Multimetro enviar = new Multimetro();
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.inicio);

		ayuda=(ImageButton)findViewById(R.id.ayuda);
		multimetro=(ImageButton)findViewById(R.id.multimetro);
		osciloscopio=(ImageButton)findViewById(R.id.osciloscopio);

		 ayuda.setOnClickListener(this);
		 multimetro.setOnClickListener(this);
		 osciloscopio.setOnClickListener(this);



	}


	@Override
	public void onClick(View boton) {
		// TODO Auto-generated method stub
```

```
                switch(boton.getId())
                {
                case R.id.ayuda:

                        Intent ayuda_activity = new Intent(this, ayuda.class);
                startActivity(ayuda_activity);
                        break;
                case R.id.multimetro:
                        Intent multimetro_activity = new Intent(this, Multimetro.class);
                startActivity(multimetro_activity);
                        break;
                case R.id.osciloscopio:
                        Intent osciloscopio_activity = new Intent(this,
Osciloscopio.class);
                startActivity(osciloscopio_activity);
                        break;

                }
        }

}
```

CLASE AYUDA

```
package tesis.android;
import android.app.Activity;
import android.os.Bundle;
public class ayuda extends Activity{
        @Override
        protected void onCreate(Bundle savedInstanceState) {
                // TODO Auto-generated method stub
                super.onCreate(savedInstanceState);
                setContentView(R.layout.help);
        }
}
```

CLASE MULTÍMETRO

```
package tesis.android;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Arrays;
import java.util.Set;
import java.util.UUID;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
```

```
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.PowerManager;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;
import java.text.DecimalFormat;

public class Multimetro extends Activity implements OnClickListener{

	private static final int rx = 0;
	ToggleButton Vcc, Vca, Icc, Ica, Res;
	TextView pantalla1, pantalla2;
	BluetoothAdapter mBluetoothAdapter;
	BluetoothSocket mmSocket;
	BluetoothDevice mmDevice;
	OutputStream mmOutputStream;
	InputStream mmInputStream;
	Thread workerThread;
	private Handler mHandler;
	volatile boolean stopWorker;
	String opcion=null;
	byte[] readBuffer;
	int readBufferPosition;
	 protected PowerManager.WakeLock mWakeLock;
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		setContentView(R.layout.multimetro);

		Vcc=(ToggleButton)findViewById(R.id.Vcc);
		Vca=(ToggleButton)findViewById(R.id.Vca);
		Icc=(ToggleButton)findViewById(R.id.Icc);
		Ica=(ToggleButton)findViewById(R.id.Ica);
		Res=(ToggleButton)findViewById(R.id.Resis);
		pantalla1=(TextView)findViewById(R.id.pantalla1);
		pantalla2=(TextView)findViewById(R.id.pantalla2);
		 Vca.setOnClickListener(this);
		 Vcc.setOnClickListener(this);
		 Icc.setOnClickListener(this);
		 Ica.setOnClickListener(this);
		 Res.setOnClickListener(this);
		//abrir bluetooth

		 try {
```

```
                findBT();
            } catch (IOException e) {
                // TODO Auto-generated catch block

            }
             try {
                    sendData("multimetro");
                } catch (IOException ex) {

                }
             PowerManager pm = (PowerManager)
getSystemService(Context.POWER_SERVICE);
        this.mWakeLock =
pm.newWakeLock(PowerManager.SCREEN_BRIGHT_WAKE_LOCK, "My Tag");
        //this.mWakeLock.acquire();
    }

    @Override
    public void onClick(View boton) {

        switch(boton.getId())
        {
        case R.id.Vcc:
            if(Vcc.isChecked())
            {
                Vca.setChecked(false);
                Icc.setChecked(false);
                Ica.setChecked(false);
                Res.setChecked(false);
                opcion="Vcc";
            }
            else
            {
                opcion=("offall");
            }


            break;
        case R.id.Vca:
            if(Vca.isChecked())
            {
                Vcc.setChecked(false);
                Icc.setChecked(false);
                Ica.setChecked(false);
                Res.setChecked(false);
                opcion="Vca";
            }
            else
            {
                opcion=("offall");
```

```
		}

		break;
case R.id.Icc:
		if(Icc.isChecked())
		{
			Vca.setChecked(false);
			Vcc.setChecked(false);
			Ica.setChecked(false);
			Res.setChecked(false);
			opcion="Icc";
		}
		else
		{
			opcion=("offall");
		}
		break;
case R.id.Ica:
		if(Ica.isChecked())
		{
			Vca.setChecked(false);
			Icc.setChecked(false);
			Vcc.setChecked(false);
			Res.setChecked(false);
			opcion=("Ica");
		}
		else
		{
			opcion=("offall");
		}
		break;
case R.id.Resis:
		if(Res.isChecked())
		{
			Vca.setChecked(false);
			Icc.setChecked(false);
			Ica.setChecked(false);
			Vcc.setChecked(false);
			opcion="Res";
		}
		else
		{
			opcion=("offall");
		}
		break;

}

		try {
			sendData(opcion);
```

```
                        opcion=null;
                    } catch (IOException e) {

                    }

        }
        @Override
        public void onDestroy(){
                 if (mWakeLock.isHeld()) {
                mWakeLock.release();
            }
                 super.onDestroy();
                 try {

                        sendData("x");
                        closeBT();
                 } catch (IOException ex) {
                 }
        }
        void openBT() throws IOException {
                 UUID uuid = UUID.fromString("00001101-0000-1000-8000-
00805F9B34FB"); // Standard

                                                                                //
SerialPortService

                                                                                // ID
                 mmSocket = mmDevice.createRfcommSocketToServiceRecord(uuid);
                 mmSocket.connect();
                 mmOutputStream = mmSocket.getOutputStream();
                 mmInputStream = mmSocket.getInputStream();

                 beginListenForData();


        }
        void closeBT() throws IOException {
                 stopWorker = true;
                 mmOutputStream.close();
                 mmInputStream.close();
                 mmSocket.close();

        }
        void sendData(String msg) throws IOException {
                 msg += "\n";
                 mmOutputStream.write(msg.getBytes());
                 msg=null;
        }
        void findBT() throws IOException {
```

```
		Toast.makeText(getApplicationContext(), "Conectando..." ,
Toast.LENGTH_LONG).show();
		mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
		if (mBluetoothAdapter == null) {

		}

		if (!mBluetoothAdapter.isEnabled()) {
			Intent enableBluetooth = new Intent(

	BluetoothAdapter.ACTION_REQUEST_ENABLE);
			startActivityForResult(enableBluetooth, 0);
		}

		Set<BluetoothDevice> pairedDevices = mBluetoothAdapter
				.getBondedDevices();
		if (pairedDevices.size() > 0) {
			for (BluetoothDevice device : pairedDevices) {
				System.out.println(device.getName());
				if (device.getName().equals("ARDUINO")) {
					mmDevice = device;
					openBT();
					break;
				}
			}
		}

	}

	void beginListenForData() {
		final Handler handler = new Handler();
		final byte delimiter = '\n'; // This is the ASCII code for a newline //
										// character
		final int[] datapandding = new int[100];
		Arrays.fill(datapandding, 0);
		stopWorker = false;
		int i = 0;
		readBufferPosition = 0;
		readBuffer = new byte[1024];
		final int aux;
		workerThread = new Thread(new Runnable() {
			public void run() {
				while (!Thread.currentThread().isInterrupted() &&
!stopWorker) {
					try {
						int bytesAvailable =
mmInputStream.available();
						if (bytesAvailable > 0) {
							byte[] packetBytes = new
byte[bytesAvailable];
```

```
            mmInputStream.read(packetBytes);
                                                        for (int i = 0; i < bytesAvailable;
i++) {
                                                                byte b = packetBytes[i];
                                                                if (b == delimiter) {
                                                                        byte[]
encodedBytes = new byte[readBufferPosition];
            System.arraycopy(readBuffer, 0,
            encodedBytes, 0,
            encodedBytes.length);
                                                                        final String data
= new String(
            encodedBytes, "US-ASCII").trim();
                                                                        // Log.i("datos",
data);
            readBufferPosition = 0;
            handler.post(new Runnable() {
                                                                                public
void run() {
            mostrar_datos(data);
                                                                                        }
                                                                        });
                                                                } else {
            readBuffer[readBufferPosition++] = b;
                                                                }
                                                        }
                                                }
                                        } catch (IOException ex) {
                                                stopWorker = true;
                                        }
                                }
                        }
                });
                workerThread.start();
        }
        public void mostrar_datos(String dato)
        {
```

```
                float vpk;


                if(Vca.isChecked())
                {
                        if (!dato.equals("OFF"))
                        {
                                pantalla1.setText(dato+ " VRMS");
                                try
                                {
                                vpk=Float.parseFloat(dato);
                                vpk=(float) (vpk*1.4142);
                                DecimalFormat df = new DecimalFormat("0.00");
                                pantalla2.setText(df.format(vpk)+ " VPk");
                                } catch(Exception e){}
                        }
                        else
                        {
                                pantalla1.setText(dato);
                                pantalla2.setText("");
                        }
                }
                else
                {
                pantalla1.setText(dato);
                pantalla2.setText("");
                }
        }
}
```

CLASE OSCILOSCOPIO

```
* Copyright (C) 2009 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 *
 * Modified by Joe Smallman April 2012 to make appropriate for serial
 * communication with Arduino device
```

```
 *
 */

package tesis.android;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.OutputStream;
import java.text.DecimalFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.LinkedList;
import java.util.Random;

import com.android.*;
import com.androidplot.series.XYSeries;
import com.androidplot.xy.*;

import android.annotation.SuppressLint;
import android.app.ActionBar;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Context;
import android.content.DialogInterface;
import android.content.DialogInterface.OnClickListener;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.Point;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.os.Message;
import android.os.PowerManager;
import android.util.Log;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.inputmethod.EditorInfo;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.CompoundButton;
```

```
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.RadioButton;
import android.widget.TextView;
import android.widget.TextView.OnEditorActionListener;
import android.widget.Toast;
import android.widget.ToggleButton;

public class Osciloscopio extends Activity implements Button.OnClickListener{

        // Debugging
    private static final String TAG = "OsciloscopioVirtual";
    private static final boolean D = true;
    protected PowerManager.WakeLock mWakeLock;
    // Intent request codes
    private static final int REQUEST_CONNECT_DEVICE = 1;
    private static final int REQUEST_ENABLE_BT_AND_CONNECT_DEVICE = 2;
    private static final int REQUEST_ENABLE_BT = 3;
    private static final int REQUEST_FILE = 5;

    private BTcom btCom;
    private BluetoothAdapter mBluetoothAdapter;

    private DataBuffer datBuf;

    // Layout Views
    private XYPlot mySimpleXYPlot;
    private XYPlot dynamicPlot;
    //private ListView mConversationView;
    private EditText mOutEditText;
    private ActionBar mActionBar;

    static String[] timebase = {"5us", "10us", "20us", "50us", "100us", "200us", "500us",
"1ms", "2ms", "5ms", "10ms", "20ms", "50ms" };
        //static String[] ampscale = {"10mV/div", "20mV/div", "50mV/div",
"100mV/div", "200mV/div", "500mV/div", "1V/div", "2V/div", "GND"};
        static byte timebase_index = 5;
        static byte ch1_index = 4;
        static byte ch1_pos = 0; // 0 to 40
    String enviar;
    //botones del osciloscopio

    ToggleButton btn_run;

    Button btn_tmas, btn_tmenos, btn_escalamas, btn_escalamenos;
    ImageButton btn_posmas, btn_posmenos;
    int scroll=20;
    //tetxt Views del osciloscopio
```

```
TextView txt_tiempo, txt_escala1, txt_v1rms;
TextView txt_v1pp, txt_f1;
TextView txt_fu1;
// Plot variables
private XYPlot mDataPlot;
private int mCount = 0;
private int mPlotMemory = 200; // Default size
private ArrayList<Number> mPlotBuffer = new ArrayList<Number>();
private SimpleXYSeries mDataSeries = new SimpleXYSeries("SIGNAL");
private ArrayList<Number> tierra = new ArrayList<Number>();
private SimpleXYSeries mtierra = new SimpleXYSeries("GND");

Double vr1=0.0,  vpp1,  frec1;
int grafCh1=0;
Double voltpp;
        Double voltrms;
        Double menor=120.0;
        Double mayor=-120.0, pp;

int c1=0,c2=0, contador=0;
double escala1=1, pos=0;
boolean accion=false;
int max_samples=100, dominio=100;
// Name of the connected device
private String mConnectedDeviceName = null;

        /** Called when the activity is first created. */
@SuppressWarnings("deprecation")
        @Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    if(D) Log.e(TAG, "+++ ON CREATE +++");

    // view stuff
    setContentView(R.layout.osciloscopio1);
    btn_run=(ToggleButton)findViewById(R.id.btn_run);
    btn_tmas=(Button)findViewById(R.id.bn_tmas);
    btn_tmenos=(Button)findViewById(R.id.btn_tmenos);
    btn_escalamas=(Button)findViewById(R.id.btn_escalamas);
    btn_escalamenos=(Button)findViewById(R.id.btn_escalamenos);
    btn_posmas=(ImageButton)findViewById(R.id.btn_posmas);
    btn_posmenos=(ImageButton)findViewById(R.id.btn_posmenos);


    btn_run.setOnClickListener(this);
    btn_tmas.setOnClickListener(this);
    btn_tmenos.setOnClickListener(this);
    btn_escalamas.setOnClickListener(this);
    btn_escalamenos.setOnClickListener(this);
    btn_posmas.setOnClickListener(this);
```

```
        btn_posmenos.setOnClickListener(this);


        txt_tiempo=(TextView)findViewById(R.id.txt_tiempo);
        txt_escala1=(TextView)findViewById(R.id.txt_escala1);

        txt_v1rms=(TextView)findViewById(R.id.txt_v1rms);
        txt_v1pp=(TextView)findViewById(R.id.txt_v1pp);

        txt_f1=(TextView)findViewById(R.id.txt_f1);

        txt_fu1=(TextView)findViewById(R.id.txt_fu1);

        mDataPlot = (XYPlot)findViewById(R.id.plot_log);

        mDataPlot.addSeries(mDataSeries, new LineAndPointFormatter(Color.RED, null,
null));
        //mDataPlot.addSeries(mtierra, new LineAndPointFormatter(Color.GREEN, null,
null));
        mDataPlot.centerOnRangeOrigin(0);
        //mDataPlot.centerOnDomainOrigin((dominio/2));
        mDataPlot.setBackgroundColor(Color.WHITE);
        mDataPlot.setDomainLabel("Sample");
        mDataPlot.getDomainLabelWidget().pack();
        mDataPlot.setDomainValueFormat(new DecimalFormat("#"));
        mDataPlot.setRangeLabel("Amplitud");
        mDataPlot.setRangeValueFormat(new DecimalFormat("#"));
        mDataPlot.getRangeLabelWidget().pack();
        mDataPlot.setRangeBoundaries(-120, 120, BoundaryMode.FIXED);
        mDataPlot.setDomainBoundaries(0, max_samples, BoundaryMode.FIXED);
        mDataPlot.setDomainStepValue(10);

        mDataPlot.setGridPadding(5, 5, 5, 5);
        mDataPlot.disableAllMarkup();
        mDataPlot.setDomainStepMode(XYStepMode.SUBDIVIDE);

                mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
                if(mBluetoothAdapter == null) {
                        // Device does not support Bluetooth
                        Toast.makeText(this, R.string.bt_not_available,
Toast.LENGTH_LONG).show();
                        finish();
                        return;
                }

                if (!mBluetoothAdapter.isEnabled()) {
        Intent                  enableIntent                =               new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(enableIntent,
REQUEST_ENABLE_BT_AND_CONNECT_DEVICE);
```

```
        } else {
            //Otherwise just request device connection straight off.
            //Launch the DeviceListActivity to see the paired devices,
                        //scan for devices and connect to chosen device
                        Intent serverIntent = new Intent(this, DeviceListActivity.class);
                        startActivityForResult(serverIntent, REQUEST_CONNECT_DEVICE);
        }



                        PowerManager pm = (PowerManager)
getSystemService(Context.POWER_SERVICE);
        this.mWakeLock =
pm.newWakeLock(PowerManager.SCREEN_BRIGHT_WAKE_LOCK, "My Tag");
        this.mWakeLock.acquire();
                        datBuf = new DataBuffer(this, mHandler);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.main_activity, menu);
        return true;
    }

    @Override
    public void onStart() {

        super.onStart();
        if(D) Log.e(TAG, "++ ON START ++");
    }

    @Override
    public synchronized void onResume() {
        super.onResume();
        /*try {
                                Thread.sleep(100);
                        } catch (InterruptedException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }*/
        if(D) Log.e(TAG, "+ ON RESUME +");

        // Performing this check in onResume() covers the case in which BT was
        // not enabled during onStart(), so we were paused to enable it...
        // onResume() will be called when ACTION_REQUEST_ENABLE activity returns.
        if (btCom != null) {
            // Only if the state is STATE_NONE, do we know that we haven't started already
            if (btCom.getState() == BTcom.STATE_NONE) {
              // Start the Bluetooth com services
```

```
        btCom.start();
       }
   }
}

@Override
public synchronized void onPause() {
   super.onPause();
   if(D) Log.e(TAG, "- ON PAUSE -");
}

@Override
public void onStop() {
   super.onStop();
   if(D) Log.e(TAG, "-- ON STOP --");
}

@Override
public void onDestroy() {
       if (mWakeLock.isHeld()) {
      mWakeLock.release();
   }
   super.onDestroy();
   enviar="inicio\n";
                btCom.write(enviar);
   btCom.stop();
   // Stop the Bluetooth chat services
   if (btCom != null) btCom.stop();
   if(D) Log.e(TAG, "--- ON DESTROY ---");
}

       public void setupCom(){
                btCom = new BTcom(this, mHandler);
                return;
       }


@SuppressLint("NewApi")
       private final Handler mHandler = new Handler() {
   @Override
   public void handleMessage(Message msg) {
      switch (msg.what) {

      case BTcom.MESSAGE_STATE_CHANGE:
         if(D) Log.i(TAG, "MESSAGE_STATE_CHANGE: " + msg.arg1);
         switch (msg.arg1) {
         case BTcom.STATE_CONNECTED:
           //  setStatus(R.string.connected);

            break;
```

```
            case BTcom.STATE_CONNECTING:
                    Toast.makeText(getApplicationContext(), "Conectando... ",
Toast.LENGTH_SHORT).show();
                break;
            case BTcom.STATE_LISTEN:
            case BTcom.STATE_NONE:
              //  setStatus(R.string.not_connected);
                break;
            }
            break;

          case BTcom.MESSAGE_WRITE:
            break;

          case BTcom.MESSAGE_READ:
            //datBuf.add((String) msg.obj, true);
            //txt_v1rms.setText((String)msg.obj);
           //Toast.makeText(getApplicationContext(), "Recibido: "
              //      + (String)msg.obj, Toast.LENGTH_SHORT).show();
            break;

          case BTcom.VRMS1:
           Double v1r = Double.parseDouble((String)msg.obj);
           if(grafCh1==1)
           {
                    txt_v1rms.setText(v1r.toString());
           //txt_v1rms.setText(vr1.toString());
           }
           break;

          case BTcom.VPP1:
           try{
           Double v1p = Double.parseDouble((String)msg.obj);
           /*voltpp=v1p;
           if(contador<100) {

                    if(voltpp > mayor){
                    mayor = voltpp;
                    }
                    if(voltpp < menor){
                    menor = voltpp;
                    }
                    contador++;
                    }
           else
           {

                    pp=mayor-menor;
                    voltrms=pp/1.4142;
                    contador=0;
```

```
                mayor=-120.0;
                menor=120.0;
            }

            if(grafCh1==1)
            {
                DecimalFormat voltaje = new DecimalFormat("0.00");
                txt_v1pp.setText(voltaje.format(pp));
                txt_v1rms.setText(voltaje.format(voltrms));
                 datBuf.add((String) msg.obj,true);
            }*/
            datBuf.add((String) msg.obj,true);
            }
            catch(Exception e){}
            break;

           case BTcom.f1:
            try{
            Double f_1 = Double.parseDouble((String)msg.obj);
            DecimalFormat df = new DecimalFormat("0.00");
            f_1=(f_1*(158.201))-93.92;
            if(grafCh1==1)
            {
            txt_f1.setText(df.format(f_1));
            }
            }catch(Exception e){}
            break;

           case BTcom.MESSAGE_DEVICE_NAME:
              // save the connected device's name
              mConnectedDeviceName                                                    =
msg.getData().getString(BTcom.DEVICE_NAME);
              Toast.makeText(getApplicationContext(), "Conectado a: "
                        + mConnectedDeviceName, Toast.LENGTH_SHORT).show();
                        break;

           case BTcom.MESSAGE_TOAST:
              Toast.makeText(getApplicationContext(),
msg.getData().getString(BTcom.TOAST),
                        Toast.LENGTH_SHORT).show();
              break;

           case DataBuffer.MESSAGE_NEW_VALUE:
            vr1 = (Double)msg.obj;
            UpdatePlot(vr1);
            break;

           }
       }
    };
```

```
   private void UpdatePlot(Double vr1) {
        //mDataPlot.centerOnDomainOrigin((max_samples/2));
        accion=true;
        if(mCount>=max_samples)
        {
                mCount=0;

                mPlotBuffer.clear();
                //tierra.clear();
        }
        if(accion==true)
        {
                mPlotBuffer.add(mCount);
                mPlotBuffer.add((escala1*vr1)+pos);
                //tierra.add(mCount);
                //tierra.add(0);
        }
        else
        {
                mPlotBuffer.add(mCount);
                mPlotBuffer.add(vr1);
                //tierra.add(mCount);
                //tierra.add(0);

        }

      if (mPlotBuffer.size() > 2 * mPlotMemory)
      {
          mPlotBuffer.remove(0);
          mPlotBuffer.remove(0);
          //tierra.remove(0);

      }

      if(grafCh1==1)
          {
           mDataPlot.addSeries(mDataSeries,  new  LineAndPointFormatter(Color.RED,
null, null));
         //  mDataPlot.addSeries(mtierra,  new  LineAndPointFormatter(Color.YELLOW,
null,null));
           mDataSeries.setModel(mPlotBuffer,
SimpleXYSeries.ArrayFormat.XY_VALS_INTERLEAVED);
              //mtierra.setModel(tierra,
SimpleXYSeries.ArrayFormat.XY_VALS_INTERLEAVED);
          }


                mDataPlot.redraw();
```

```
                mCount++;

        }

        protected void onActivityResult(int requestCode, int resultCode, Intent data) {
                switch(requestCode){
                case REQUEST_ENABLE_BT_AND_CONNECT_DEVICE:
                        if (resultCode == RESULT_OK) {
                                Toast.makeText(this, R.string.bt_enabled,
Toast.LENGTH_SHORT).show();
                                Intent serverIntent = new Intent(this,
DeviceListActivity.class);
                                startActivityForResult(serverIntent,
REQUEST_CONNECT_DEVICE);
                                break;
                        }
                        else {
                                // User did not enable Bluetooth or an error occurred
          Log.d(TAG, "BT not enabled");
          Toast.makeText(this, R.string.bt_not_enabled_leaving,
Toast.LENGTH_SHORT).show();
          finish();
          break;
                        }
                case REQUEST_CONNECT_DEVICE:
                        if (resultCode == RESULT_OK){
                                if (btCom == null) setupCom();
                                connectDevice(data, true);
                        }
                        break;
                case REQUEST_FILE:
                        if (resultCode == RESULT_OK){
                                String path =
data.getExtras().getString(FileBrowserActivity.EXTRA_FILE_PATH);
                                //Toast.makeText(this, "File Clicked: " + path,
Toast.LENGTH_SHORT).show();
                                plotSavedData(path);
                        }
                        break;
                }
                return;
        }

        private void plotSavedData(String path){

                try {
                        File f = new File(path);
                        BufferedReader reader = new BufferedReader(new
FileReader(f));
```

```
                        String currentLine;

                        datBuf.clear();
                        while ((currentLine = reader.readLine()) != null) {
                                datBuf.add(currentLine, false);
                        }
                }catch (IOException e) {
                        e.printStackTrace();
                }

                mDataSeries.setModel(datBuf.getBuffer(),
SimpleXYSeries.ArrayFormat.Y_VALS_ONLY);
                mDataPlot.redraw();

        }

        private void connectDevice(Intent data, boolean secure) {
                // MAC address
                String address = data.getExtras()
                        .getString(DeviceListActivity.EXTRA_DEVICE_ADDRESS);
                // BluetoothDevice object
                BluetoothDevice                device                =
mBluetoothAdapter.getRemoteDevice(address);
                // Attempt to connect to the device
      btCom.connect(device, secure);
      return;
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
                Intent serverIntent = null;
           // Handle item selection
           switch (item.getItemId()) {
                case R.id.scan_connect:
                        if (btCom != null){
                                btCom.stop();
                        }
                        datBuf.clear();
                        mPlotBuffer.clear();
                        // If BT is not on, request that it be enabled.
                    // A request to connect to device will then be called during
onActivityResult
                        if (!mBluetoothAdapter.isEnabled()) {
                          Intent enableIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                          startActivityForResult(enableIntent,
REQUEST_ENABLE_BT_AND_CONNECT_DEVICE);
                } else {
                          //Otherwise just request device connection straight off.
                          //Launch the DeviceListActivity to see the paired devices,
```

```
                                        //scan for devices and connect to chosen device
                                        serverIntent = new Intent(this, DeviceListActivity.class);
                                        startActivityForResult(serverIntent,
REQUEST_CONNECT_DEVICE);
                        }
                                return true;
                        case R.id.disconnect:
                                if (btCom != null){
                                        btCom.stop();
                                }
                                return true;
                        case R.id.data_save:
                                saveToFile(datBuf.getBuffer(), datBuf.getBufferch2());
                                return true;
                        case R.id.data_clear:
                                datBuf.clear();
                                return true;
                        case R.id.plot_load:
                                //Launch the FileBrowserActivity
                                serverIntent = new Intent(this, FileBrowserActivity.class);
                                startActivityForResult(serverIntent, REQUEST_FILE);
                                return true;
                        case R.id.plot_settings:
                                //TODO
                                return true;
                    default:
                        return super.onOptionsItemSelected(item);
                }
        }

        private void saveToFile(ArrayList<Number> data, ArrayList<Number> dato){
                try {
                        //date format for filename
                        SimpleDateFormat formatter = new
SimpleDateFormat("ddMMMyy");
                        //get date
                        Date now = new Date();
                        //get/make file directory
                        File dir = new File("/sdcard/Osciloscopio/");
                        dir.mkdirs();
                        //check to see if file exists
                        //if doesnt exist create file,
                        //if does exist increment filename couter
                        //filename in format ddMMMyyii.txt
                        File file = new File("default.txt");
                        for (int i = 0; i < 100; i++){
                                String count = String.format("%02d", i);
                                String filename = formatter.format(now) + count +
".txt";
                                file = new File(dir, filename);
```

```
                        if (!file.exists()) break;
                    }
                    BufferedWriter writer = new BufferedWriter(new
FileWriter(file));
                    for (Number n : data) {
                        writer.write(n.toString()+"\r\n");
                    }
                    for(Number n:dato)
                    {
                        writer.write(n.toString()+"\r\n");
                    }
                    writer.flush();
                    writer.close();
                    Toast.makeText(this, R.string.file_saved,
Toast.LENGTH_SHORT).show();
                }catch (IOException e) {
                    e.printStackTrace();
                    Toast.makeText(this, R.string.file_save_error,
Toast.LENGTH_SHORT).show();
                }
        }

        private void sendMessage(){
                // Check that we're actually connected before trying anything
     if (btCom.getState() != BTcom.STATE_CONNECTED) {
        Toast.makeText(this, R.string.not_connected, Toast.LENGTH_SHORT).show();
        return;
     }

                if(btn_run.isChecked())
                {
                enviar="osciloscopio\n";
                btCom.write(enviar);

                }
                else{
                    enviar="offall\n";
                    btCom.write(enviar);
                }
                return;
        }


        class DoneOnEditorActionListener implements OnEditorActionListener {
           @Override
           public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
              if (actionId == EditorInfo.IME_ACTION_DONE) {
                 InputMethodManager imm =
(InputMethodManager)v.getContext().getSystemService(Context.INPUT_METHOD_SE
RVICE);
```

```
                imm.hideSoftInputFromWindow(v.getWindowToken(), 0);
                sendMessage();
                return true;
            }
            return false;
        }
    }


    //@Override
    public void onClick(View v) {
        switch(v.getId()){
        case R.id.btn_run:
            if(btn_run.isChecked())
            {
                grafCh1=1;
            }
            else
            {
                grafCh1=0;
            }
            if (btCom != null){

                sendMessage();
        }

            else{
                Toast.makeText(getApplicationContext(), "Conectese al
dispositivo bluetooth", Toast.LENGTH_SHORT).show();

            }
             break;
        case R.id.bn_tmas:
            if(timebase_index<(timebase.length-1)){
            txt_tiempo.setText(timebase[++timebase_index]);
            max_samples+=10;
            mDataPlot.setDomainBoundaries(0, max_samples,
BoundaryMode.FIXED);

        }
            break;
        case R.id.btn_tmenos:
            if(timebase_index>0){
            txt_tiempo.setText(timebase[--timebase_index]);
            max_samples-=10;
            mDataPlot.setDomainBoundaries(0, max_samples,
BoundaryMode.FIXED);
            }
            break;
        case R.id.btn_escalamas:
```

```
                        //txt_escala1.setText(ampscale[--ch1_index]);
                        //mDataPlot.setRangeBoundaries(-escala, escala,
BoundaryMode.FIXED);
                        escala1+=0.1;

                        //mDataPlot.setRangeBoundaries(-escala, escala,
BoundaryMode.FIXED);
                        break;
                case R.id.btn_escalamenos:

                                escala1-=0.1;
                                //txt_escala1.setText(ampscale[ch1_index]);

                        break;
                case R.id.btn_posmas:
                        pos+=1;
                        break;
                case R.id.btn_posmenos:

                        pos-=1;

                        break;


                }
        }

}
```

CLASE BTcom

```
/*
 * Copyright (C) 2009 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 *
 * Modified by Joe Smallman April 2012 to make appropriate for serial
 * communication with Arduino device
 * Copyright (C) 2009 The Android Open Source Project
```

```
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 *
 * Modified by Joe Smallman April 2012 to make appropriate for serial
 * communication with Arduino device
 *

package tesis.android;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.nio.charset.Charset;
import java.util.UUID;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Context;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;


public class BTcom {

        // Debugging
    private static final String TAG = "BTcom";
    private static final boolean D = true;

    // Name for the SDP record when creating server socket
    //private static final String NAME = "BTcom";

    // Unique UUID for this application
    private static final UUID MY_UUID = UUID.fromString("00001101-0000-1000-
8000-00805f9b34fb"); //Standard SerialPortService ID

    private final BluetoothAdapter mAdapter;
```

```
        private Context mContext;
        private Handler mHandler;
        private ConnectThread mConnectThread;
    private ConnectedThread mConnectedThread;

        // Message types sent to the main activity
    public static final int MESSAGE_STATE_CHANGE = 1;
    public static final int MESSAGE_READ = 2;
    public static final int VRMS1=6;
    public static final int VPP1=8;
    public static final int f1=10;
    public static final int f_u1=12;
    public static final int MESSAGE_WRITE = 3;
    public static final int MESSAGE_DEVICE_NAME = 4;
    public static final int MESSAGE_TOAST = 5;

    // Key names sent to the main activity
    public static final String DEVICE_NAME = "device_name";
    public static final String TOAST = "toast";

        // Constants that indicate the current connection state
    public static final int STATE_NONE = 0;       // we're doing nothing
    public static final int STATE_LISTEN = 1;     // now listening for incoming
connections
    public static final int STATE_CONNECTING = 2; // now initiating an outgoing
connection
    public static final int STATE_CONNECTED = 3;  // now connected to a remote device
    public static final int STATE_RECONNECT = 4;  // reconnecting to remote device
    private int mState;

        public BTcom(Context context, Handler handler){
                mAdapter = BluetoothAdapter.getDefaultAdapter();
                mContext = context;
                mHandler = handler;
                mState = STATE_NONE;
        }

        private synchronized void setState(int state) {
      if (D) Log.d(TAG, "setState() " + mState + " -> " + state);
      mState = state;

      // Give the new state to the Handler so the UI Activity can update
      mHandler.obtainMessage(MESSAGE_STATE_CHANGE,  state,  -
1).sendToTarget();
    }
        public synchronized int getState() {
      return mState;
    }
        /**
     * Start the chat service. Specifically start AcceptThread to begin a
```

```
     * session in listening (server) mode. Called by the Activity onResume() */
    public synchronized void start() {
        if (D) Log.d(TAG, "start");

     // Cancel any thread attempting to make a connection
        if (mConnectThread != null) {mConnectThread.cancel(); mConnectThread = null;}

        // Cancel any thread currently running a connection
        if (mConnectedThread != null) {mConnectedThread.cancel(); mConnectedThread =
null;}

        setState(STATE_LISTEN);

    }
    /**
     * Stop all threads
     */
    public synchronized void stop() {
        if (D) Log.d(TAG, "stop");

        if (mConnectThread != null) {
            mConnectThread.cancel();
            mConnectThread = null;
        }

        if (mConnectedThread != null) {
            mConnectedThread.cancel();
            mConnectedThread = null;
        }

        setState(STATE_NONE);
    }
    /**
     * Start the ConnectThread to initiate a connection to a remote device.
     * @param device  The BluetoothDevice to connect
     * @param secure Socket Security type - Secure (true) , Insecure (false)
     */
    public synchronized void connect(BluetoothDevice device, boolean secure) {
        if (D) Log.d(TAG, "connect to: " + device);

        // Cancel any thread attempting to make a connection
        if (mState == STATE_CONNECTING) {
            if (mConnectThread != null) {mConnectThread.cancel(); mConnectThread =
null;}
        }

        // Cancel any thread currently running a connection
        if (mConnectedThread != null) {mConnectedThread.cancel(); mConnectedThread =
null;}
```

```
        // Start the thread to connect with the given device
        mConnectThread = new ConnectThread(device, secure);
        mConnectThread.start();
        setState(STATE_CONNECTING);
    }
    /**
     * Start the ConnectedThread to begin managing a Bluetooth connection
     * @param socket  The BluetoothSocket on which the connection was made
     * @param device  The BluetoothDevice that has been connected
     */
    public synchronized void connected(BluetoothSocket socket, BluetoothDevice
            device, final String socketType) {
        if (D) Log.d(TAG, "connected, Socket Type:" + socketType);

        // Cancel the thread that completed the connection
        if (mConnectThread != null) {mConnectThread.cancel(); mConnectThread = null;}

        // Cancel any thread currently running a connection
        if (mConnectedThread != null) {mConnectedThread.cancel(); mConnectedThread =
null;}

        // Start the thread to manage the connection and perform transmissions
        mConnectedThread = new ConnectedThread(socket, socketType);
        mConnectedThread.start();

        // Send the name of the connected device back to the UI Activity
        Message msg = mHandler.obtainMessage(MESSAGE_DEVICE_NAME);
        Bundle bundle = new Bundle();
        bundle.putString(DEVICE_NAME, device.getName());
        msg.setData(bundle);
        mHandler.sendMessage(msg);

        setState(STATE_CONNECTED);
    }
            /**
     * Indicate that the connection attempt failed and notify the UI Activity.
     */
    private void connectionFailed() {
        // Send a failure message back to the Activity
        Message msg = mHandler.obtainMessage(MESSAGE_TOAST);
        Bundle bundle = new Bundle();
        bundle.putString(TOAST, "Unable to connect device");
        msg.setData(bundle);
        mHandler.sendMessage(msg);

        // Start the service over to restart listening mode
        BTcom.this.start();
    }

    /**
```

```
 * Indicate that the connection was lost and notify the UI Activity.
 */
private void connectionLost() {
    // Send a failure message back to the Activity
    Message msg = mHandler.obtainMessage(MESSAGE_TOAST);
    Bundle bundle = new Bundle();
    bundle.putString(TOAST, "Conexion perdida");
    msg.setData(bundle);
    mHandler.sendMessage(msg);

    // Start the service over to restart listening mode
    BTcom.this.start();
}
/**
 * Write to the ConnectedThread in an unsynchronized manner
 * @param out The string to write
 * @see ConnectedThread#write(byte[])
 */
public void write(String out) {
    // Create temporary object
    ConnectedThread r;
    // Synchronize a copy of the ConnectedThread
    synchronized (this) {
        if (mState != STATE_CONNECTED) return;
        r = mConnectedThread;
    }
    // Perform the write unsynchronized
    r.write(out);
}

        /**
 * This thread runs while attempting to make an outgoing connection
 * with a device. It runs straight through; the connection either
 * succeeds or fails.
 */
private class ConnectThread extends Thread {
    private final BluetoothSocket mmSocket;
    private final BluetoothDevice mmDevice;
    private String mSocketType;

    public ConnectThread(BluetoothDevice device, boolean secure) {
        mmDevice = device;
        BluetoothSocket tmp = null;
        mSocketType = secure ? "Secure" : "Insecure";

        // Get a BluetoothSocket for a connection with the
        // given BluetoothDevice
        try {
          tmp = device.createRfcommSocketToServiceRecord(
                    MY_UUID);
```

```
        } catch (IOException e) {
            Log.e(TAG, "Socket Type: " + mSocketType + "create() failed", e);
        }
        mmSocket = tmp;
    }

    public void run() {
        Log.i(TAG, "BEGIN mConnectThread SocketType:" + mSocketType);
        setName("ConnectThread" + mSocketType);

        // Always cancel discovery because it will slow down a connection
        mAdapter.cancelDiscovery();
        // Make a connection to the BluetoothSocket
        try {
            // This is a blocking call and will only return on a
            // successful connection or an exception
            mmSocket.connect();
        } catch (IOException e) {
            // Close the socket
            try {
                mmSocket.close();
            } catch (IOException e2) {
                Log.e(TAG, "unable to close() " + mSocketType +
                        " socket during connection failure", e2);
            }
            connectionFailed();
            return;
        }

        // Reset the ConnectThread because we're done
        synchronized (BTcom.this) {
            mConnectThread = null;
        }

        // Start the connected thread
        connected(mmSocket, mmDevice, mSocketType);
    }

    public void cancel() {
        try {
            mmSocket.close();
        } catch (IOException e) {
            Log.e(TAG, "close() of connect " + mSocketType + " socket failed", e);
        }
    }
}


     /**
 * This thread runs during a connection with a remote device.
```

```
 * It handles all incoming and outgoing transmissions.
 */
private class ConnectedThread extends Thread {

        private final BluetoothSocket mmSocket;
    private final InputStream mmInStream;
    private final OutputStream mmOutStream;

        public ConnectedThread(BluetoothSocket socket, String socketType) {
       Log.d(TAG, "create ConnectedThread: " + socketType);
       mmSocket = socket;
       InputStream tmpIn = null;
       OutputStream tmpOut = null;

       // Get the BluetoothSocket input and output streams
       try {
          tmpIn = socket.getInputStream();
          tmpOut = socket.getOutputStream();
       } catch (IOException e) {
          Log.e(TAG, "temp sockets not created", e);
       }

       mmInStream = tmpIn;
       mmOutStream = tmpOut;
    }

        public void run() {
       Log.i(TAG, "BEGIN mConnectedThread");

       char delimeter = '\n';
       byte[] readBuffer = new byte[1024];
       byte[] vrm1=new byte[1024];
       byte[] vrm2=new byte[1024];
       byte[] vp1=new byte[1024];
       byte[] vp2=new byte[1024];
       byte[] f_1=new byte[1024];
       byte[] f_2=new byte[1024];
       byte[] fu1=new byte[1024];
       byte[] fu2=new byte[1024];
       int y=0;
       int readBufferPosition = 0;

       // Keep listening to the InputStream while connected
       while (true) {
          try {
             // Read from the InputStream
                  int bytesAvailable = mmInStream.available();
                  if(bytesAvailable > 0)
                  {
                           byte[] packetBytes = new byte[bytesAvailable];
```

```
                    mmInStream.read(packetBytes);

                    for (int i = 0; i < bytesAvailable; i++)
                    {
                            byte b = packetBytes[i];
                            if (b == delimeter)
                            {



                                    // vrms1
                                    if(readBuffer[0]=='v' && readBuffer[1]=='r'&&
readBuffer[2]=='1')
                                    {
                                            for(int x=3; x < bytesAvailable; x++)
                                            {
                                                    y=x-3;
                                                    if(readBuffer[x]=='&')
                                                    {
                                                            String vr1 = new
String(vrm1, Charset.forName("ASCII")).trim();
                                            mHandler.obtainMessage(VRMS1,
vr1).sendToTarget();
                                            readBufferPosition = 0;
                                                    }
                                                    else
                                                    {
                                                    vrm1[y]=readBuffer[x];
                                                    }

                                            }
                                            readBufferPosition = 0;
                                    }

                                    //vpp1
                                    if(readBuffer[0]=='v' &&
readBuffer[1]=='p'&& readBuffer[2]=='1')
                                    {
                                            for(int x=3; x < bytesAvailable; x++)
                                            {
                                                    y=x-3;
                                                    if(readBuffer[x]=='&')
                                                    {
                                                            String p1 = new
String(vp1, Charset.forName("ASCII")).trim();
                                            mHandler.obtainMessage(VPP1,
p1).sendToTarget();
                                            readBufferPosition = 0;
                                                    }
                                                    else{
```

```
                    vp1[y]=readBuffer[x];
                    }

                }
                readBufferPosition = 0;
            }


           //f1
            if(readBuffer[0]=='f' && readBuffer[1]=='1')
            {
                for(int x=2; x < bytesAvailable; x++)
                {
                    y=x-2;
                    if(readBuffer[x]=='&')
                    {
                        String frec1 = new
String(f_1, Charset.forName("ASCII")).trim();
                mHandler.obtainMessage(f1,
frec1).sendToTarget();
                readBufferPosition = 0;
                    }
                    else
                    {
                    f_1[y]=readBuffer[x];
                    }

                }

                readBufferPosition = 0;
            }


            if(readBuffer[0]!='v'|| readBuffer[0]!='f')
            {
                String s = new String(readBuffer,
Charset.forName("ASCII")).trim();
            mHandler.obtainMessage(MESSAGE_READ,
s).sendToTarget();
                readBufferPosition = 0;
                y=0;
            }

            else
            {

                readBufferPosition = 0;
                y=0;
            }
```

```
                                    }
                                    else
                                    {
                                            readBuffer[readBufferPosition++] = b;
                                    }
                            }
                    }


            } catch (IOException e) {
               Log.e(TAG, "disconnected", e);
               connectionLost();
               // Start the service over to restart listening mode
               BTcom.this.start();
               break;
            }
         }
       }

          /**
       * Write to the connected OutStream.
       * @param buffer  The bytes to write
       */
      public void write(String msg) {
         try {
            mmOutStream.write(msg.getBytes());

            // Share the sent message back to the UI Activity
            mHandler.obtainMessage(MESSAGE_WRITE, -1, -1, msg)
                  .sendToTarget();
         } catch (IOException e) {
            Log.e(TAG, "Exception during write", e);
         }
      }

          public void cancel() {
         try {
            mmSocket.close();
         } catch (IOException e) {
            Log.e(TAG, "close() of connect socket failed", e);
         }
          }
   }
}
```

CLASE DeviceListActivity

```
/*
 * Copyright (C) 2009 The Android Open Source Project
```

```
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package tesis.android;

import java.util.Set;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
//import android.view.Window;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.AdapterView.OnItemClickListener;

/**
 * This Activity appears as a dialog. It lists any paired devices and
 * devices detected in the area after discovery. When a device is chosen
 * by the user, the MAC address of the device is sent back to the parent
 * Activity in the result Intent.
 */
public class DeviceListActivity extends Activity {
    // Debugging
    private static final String TAG = "DeviceListActivity";
    private static final boolean D = true;

    // Return Intent extra
```

```
    public static String EXTRA_DEVICE_ADDRESS = "device_address";

    // Member fields
    private BluetoothAdapter mBtAdapter;
    private ArrayAdapter<String> mPairedDevicesArrayAdapter;
    private ArrayAdapter<String> mNewDevicesArrayAdapter;

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

        // Setup the window
        //requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);
        setContentView(R.layout.device_list);

        // Set result CANCELED in case the user backs out
        setResult(Activity.RESULT_CANCELED);

        // Initialize the button to perform device discovery
        Button scanButton = (Button) findViewById(R.id.button_scan);
        scanButton.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                doDiscovery();
                v.setVisibility(View.GONE);
            }
        });

        // Initialize array adapters. One for already paired devices and
        // one for newly discovered devices
        mPairedDevicesArrayAdapter = new ArrayAdapter<String>(this,
R.layout.device_name);
        mNewDevicesArrayAdapter = new ArrayAdapter<String>(this,
R.layout.device_name);

        // Find and set up the ListView for paired devices
        ListView pairedListView = (ListView) findViewById(R.id.paired_devices);
        pairedListView.setAdapter(mPairedDevicesArrayAdapter);
        pairedListView.setOnItemClickListener(mDeviceClickListener);

        // Find and set up the ListView for newly discovered devices
        ListView newDevicesListView = (ListView) findViewById(R.id.new_devices);
        newDevicesListView.setAdapter(mNewDevicesArrayAdapter);
        newDevicesListView.setOnItemClickListener(mDeviceClickListener);

        // Register for broadcasts when a device is discovered
        IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
        this.registerReceiver(mReceiver, filter);

        // Register for broadcasts when discovery has finished
        filter = new IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
```

```
        this.registerReceiver(mReceiver, filter);

        // Get the local Bluetooth adapter
        mBtAdapter = BluetoothAdapter.getDefaultAdapter();

        // Get a set of currently paired devices
        Set<BluetoothDevice> pairedDevices = mBtAdapter.getBondedDevices();

        // If there are paired devices, add each one to the ArrayAdapter
        if (pairedDevices.size() > 0) {
            findViewById(R.id.title_paired_devices).setVisibility(View.VISIBLE);
            for (BluetoothDevice device : pairedDevices) {
                mPairedDevicesArrayAdapter.add(device.getName() + "\n" +
device.getAddress());
            }
        } else {
            String noDevices = getResources().getText(R.string.none_paired).toString();
            mPairedDevicesArrayAdapter.add(noDevices);
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();

        // Make sure we're not doing discovery anymore
        if (mBtAdapter != null) {
            mBtAdapter.cancelDiscovery();
        }

        // Unregister broadcast listeners
        this.unregisterReceiver(mReceiver);
    }
     * Start device discover with the BluetoothAdapter
     */
    private void doDiscovery() {
        if (D) Log.d(TAG, "doDiscovery()");

        // Indicate scanning in the title
        setProgressBarIndeterminateVisibility(true);
        setTitle(R.string.scanning);

        // Turn on sub-title for new devices
        findViewById(R.id.title_new_devices).setVisibility(View.VISIBLE);

        // If we're already discovering, stop it
        if (mBtAdapter.isDiscovering()) {
            mBtAdapter.cancelDiscovery();
        }
```

```
        // Request discover from BluetoothAdapter
        mBtAdapter.startDiscovery();
    }

    // The on-click listener for all devices in the ListViews
    private OnItemClickListener mDeviceClickListener = new OnItemClickListener() {
        public void onItemClick(AdapterView<?> av, View v, int arg2, long arg3) {
            // Cancel discovery because it's costly and we're about to connect
            mBtAdapter.cancelDiscovery();

            // Get the device MAC address, which is the last 17 chars in the View
            String info = ((TextView) v).getText().toString();
            String address = info.substring(info.length() - 17);

            // Create the result Intent and include the MAC address
            Intent intent = new Intent();
            intent.putExtra(EXTRA_DEVICE_ADDRESS, address);

        }
    };
    // The BroadcastReceiver that listens for discovered devices and
    // changes the title when discovery is finished
    private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            String action = intent.getAction();
            // When discovery finds a device
            if (BluetoothDevice.ACTION_FOUND.equals(action)) {
                // Get the BluetoothDevice object from the Intent
                BluetoothDevice device =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
                // If it's already paired, skip it, because it's been listed already
                if (device.getBondState() != BluetoothDevice.BOND_BONDED) {
                    mNewDevicesArrayAdapter.add(device.getName() + "\n" +
device.getAddress());
                }
            // When discovery is finished, change the Activity title
            } else if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(action))
{
                setProgressBarIndeterminateVisibility(false);
                setTitle(R.string.select_device);
                if (mNewDevicesArrayAdapter.getCount() == 0) {
                    String noDevices = getResources().getText(R.string.none_found).toString();
                    mNewDevicesArrayAdapter.add(noDevices);
                }
            }
        }
    };
```

PROGRAMACIÓN LAYOUT'S

Device_list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Copyright (C) 2009 The Android Open Source Project

     Licensed under the Apache License, Version 2.0 (the "License");
     you may not use this file except in compliance with the License.
     You may obtain a copy of the License at

          http://www.apache.org/licenses/LICENSE-2.0

     Unless required by applicable law or agreed to in writing, software
     distributed under the License is distributed on an "AS IS" BASIS,
     WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
     See the License for the specific language governing permissions and
     limitations under the License.
-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    >
    <TextView android:id="@+id/title_paired_devices"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/title_paired_devices"
        android:visibility="gone"
        android:background="#666"
        android:textColor="#fff"
        android:paddingLeft="5dp"
    />
    <ListView android:id="@+id/paired_devices"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:stackFromBottom="true"
        android:layout_weight="1"
    />
    <TextView android:id="@+id/title_new_devices"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/title_other_devices"
        android:visibility="gone"
        android:background="#666"
        android:textColor="#fff"
        android:paddingLeft="5dp"
    />
    <ListView android:id="@+id/new_devices"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:stackFromBottom="true"
        android:layout_weight="2"
    />
    <Button android:id="@+id/button_scan"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/button_scan"
    />
</LinearLayout>
```

Device_name.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Copyright (C) 2009 The Android Open Source Project

     Licensed under the Apache License, Version 2.0 (the "License");
     you may not use this file except in compliance with the License.
     You may obtain a copy of the License at

          http://www.apache.org/licenses/LICENSE-2.0

     Unless required by applicable law or agreed to in writing, software
     distributed under the License is distributed on an "AS IS" BASIS,
     WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
     See the License for the specific language governing permissions and
     limitations under the License.
-->
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="18sp"
    android:padding="5dp"
                                    />
```

Help.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:background="#000"
    android:layout_height="match_parent" >
     <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:orientation="vertical" >

        <ImageView
            android:id="@+id/imageView3"
            android:layout_width="400dp"
```

```
            android:layout_height="300dp"
            android:layout_gravity="center"
            android:src="@drawable/electronica" />

        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="PROYECTO DE GRADO PARA LA OBTENCIÓN DEL TITULO
DE"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:textColor="#ffff"
            android:textSize="10pt"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/textView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="INGENIERO EN ELECTRÓNICA E INSTRUMENTACIÓN"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:textColor="#ffff"
             android:textStyle="bold"
            android:textSize="10pt" />

        <TextView
            android:id="@+id/textView4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text=""
            android:textAppearance="?android:attr/textAppearanceLarge"
/>

        <TextView
            android:id="@+id/textView5"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="TEMA:"
            android:textColor="#ffff"
            android:textSize="18pt"
            android:textStyle="bold"
            android:textAppearance="?android:attr/textAppearanceLarge"
/>

        <TextView
            android:id="@+id/textView3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="DISEÑO E IMPLEMENTACIÓN DE INSTRUMENTOS"
            android:textAppearance="?android:attr/textAppearanceLarge"
```

```
            android:textColor="#ffff"
            android:textSize="10pt"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/textView6"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="VIRTUALES UTILIZADOS EN LA MEDICIÓN DE
VARIABLES"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:textColor="#ffff"
            android:textSize="10pt"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/textView7"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="ELECTRICAS CON  DISPOSITIVOS MÓVILES BAJO LA"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:textColor="#ffff"
            android:textSize="10pt"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/textView8"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="PLATAFORMA ANDROID, PARA EL LABORATORIO DE"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:textColor="#ffff"
            android:textSize="10pt"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/textView9"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="CIRCUITOS ELECTRÓNICOS"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:textColor="#ffff"
            android:textSize="10pt"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/textView10"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text=""
```

```
            android:textAppearance="?android:attr/textAppearanceLarge"
/>

        <TextView
            android:id="@+id/textView11"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="AUTORES:"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:textColor="#ffff"
            android:textSize="18pt"
            android:textStyle="bold" />

        <TableLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent" >

            <TableRow
                android:id="@+id/tableRow1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" >

                <ImageView
                    android:id="@+id/imageView1"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:src="@drawable/jchicaisa" />

                <LinearLayout
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:orientation="vertical" >

                    <TextView
                        android:id="@+id/textView12"
                        android:layout_width="wrap_content"
                        android:layout_height="match_parent"
                        android:text="Jorge Chicaisa:"

android:textAppearance="?android:attr/textAppearanceLarge"
                        android:textColor="#ffff"
                        android:textSize="10pt"
                        android:textStyle="bold" />

                    <TextView
                        android:id="@+id/textView13"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:text="Nació el 16 de julio de 1989"
                        android:textColor="#ffff"
                        android:textSize="10pt"
                        android:textStyle="bold"
```

```
android:textAppearance="?android:attr/textAppearanceLarge" />

                    <TextView
                        android:id="@+id/textView14"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:text="en la ciudad de Latacunga"
                        android:textColor="#ffff"
                        android:textSize="10pt"
                        android:textStyle="bold"

android:textAppearance="?android:attr/textAppearanceLarge" />

                </LinearLayout>

            </TableRow>

            <TableRow
                android:id="@+id/tableRow3"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" >

                <TextView
                    android:id="@+id/textView18"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text=""

android:textAppearance="?android:attr/textAppearanceLarge" />

            </TableRow>

            <TableRow
                android:id="@+id/tableRow2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" >

                <ImageView
                    android:id="@+id/imageView2"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:src="@drawable/rnieto" />

                <LinearLayout
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:orientation="vertical" >

                    <TextView
                        android:id="@+id/textView15"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:text="Raúl Nieto:"
```

```
                        android:textColor="#ffff"
                        android:textSize="10pt"
                        android:textStyle="bold"

android:textAppearance="?android:attr/textAppearanceLarge" />

                    <TextView
                        android:id="@+id/textView16"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:text="Nació el 10 de diciembre de 1989"
                        android:textColor="#ffff"
                        android:textSize="10pt"
                        android:textStyle="bold"

android:textAppearance="?android:attr/textAppearanceLarge" />

                    <TextView
                        android:id="@+id/textView17"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:text="en la ciudad de Ambato"
                        android:textColor="#ffff"
                        android:textSize="10pt"
                        android:textStyle="bold"

android:textAppearance="?android:attr/textAppearanceLarge" />

                </LinearLayout>

            </TableRow>

        </TableLayout>

    </LinearLayout>

                            </RelativeLayout>
```

Inicio.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:background="@drawable/fondo"
    android:layout_height="match_parent" >

    <RelativeLayout
        android:id="@+id/RelativeLayout1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
```

```
        android:layout_alignParentTop="true"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/titulo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true"
            android:layout_centerHorizontal="true"
            android:layout_gravity="center"
            android:text="ELIJA UNA OPCIÓN"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:textColor="#ffff"
            android:textSize="18pt"
            android:textStyle="bold" />

        <ImageButton
            android:id="@+id/multimetro"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/titulo"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="49dp"
            android:src="@drawable/multimetro2" />

        <ImageButton
            android:id="@+id/ayuda"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignLeft="@+id/multimetro"
            android:layout_below="@+id/osciloscopio"
            android:layout_marginTop="52dp"
            android:src="@drawable/help1" />

        <ImageButton
            android:id="@+id/osciloscopio"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/multimetro"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="37dp"
            android:src="@drawable/osciloscopio1" />

    </RelativeLayout>

                        </RelativeLayout>
```

Multímetro.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:background="@drawable/logo_mult"
```

```
android:layout_height="match_parent" >
<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:text="Seleccione una opcion"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:textColor="#ffff"
    android:textSize="55dp"
    android:textStyle="bold" />

<EditText
    android:id="@+id/pantalla1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentRight="true"
    android:layout_below="@+id/textView2"
    android:clickable="false"
    android:editable="false"
    android:ems="10"
    android:enabled="false"
    android:freezesText="false"
    android:gravity="right"
    android:background="#fff"
    android:textStyle="bold"
    android:textColor="#000"
    android:linksClickable="false"
    android:textSize="90dp" >

    <requestFocus />
</EditText>

<EditText
    android:id="@+id/pantalla2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentRight="true"
    android:layout_below="@+id/pantalla1"
    android:clickable="false"
    android:editable="false"
    android:ems="10"
    android:enabled="false"
    android:freezesText="false"
    android:gravity="right"
    android:background="#fff"
    android:textColor="#000"
    android:linksClickable="false"
    android:textStyle="bold"
    android:textSize="90dp" />
```

```
<ProgressBar
    android:id="@+id/analogico"
    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentRight="true"
    android:layout_below="@+id/pantalla2"
    android:max="750" />

<TableLayout
    android:id="@+id/tableLayout1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentRight="true"
    android:layout_below="@+id/analogico" >

    <TableRow
        android:id="@+id/tableRow2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >

        <ToggleButton
            android:id="@+id/Vca"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"

            android:text="Voltaje Alterno"
            android:textOff="Ac [V]"
            android:textOn="Ac [V]"
            android:textSize="40dp"
            android:textStyle="bold"
            android:width="200dp" />

        <ToggleButton
            android:id="@+id/Vcc"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Voltaje Continuo"

            android:textOff="Dc [V]"
            android:textOn="Dc [V]"
            android:textSize="40dp"
            android:textStyle="bold"
            android:width="200dp" />
    </TableRow>

    <TableRow
        android:id="@+id/tableRow3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >
```

```
            <ToggleButton
                android:id="@+id/Icc"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="Corriente Continua"
                android:textOff="Dc [A]"
                android:textOn="Dc [A]"

                android:textSize="40dp"
                android:textStyle="bold"
                android:width="200dp" />

            <ToggleButton
                android:id="@+id/Ica"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="Corriente Alterna"
                android:textOff="Ac [A]"
                android:textOn="Ac [A]"

                android:textSize="40dp"
                android:textStyle="bold"
                android:width="200dp" />
        </TableRow>

        <TableRow
            android:id="@+id/tableRow4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" >

            <ToggleButton
                android:id="@+id/Resis"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_marginBottom="66dp"
                android:layout_weight="1"
                android:text="Resistencia"
                android:textOff="R"
                android:textOn="R"

                android:textSize="40dp"
                android:textStyle="bold"
                android:width="200dp" />
        </TableRow>
    </TableLayout>

                        </RelativeLayout>
```

Osciloscopio.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/RelativeLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="right"
    android:orientation="vertical" >

    <com.androidplot.xy.XYPlot
        android:id="@+id/plot_log"
        android:layout_width="848dp"
        android:layout_height="wrap_content"
        android:layout_above="@+id/textView8"
        android:layout_below="@+id/textView1"
        android:layout_toLeftOf="@+id/tableLayout1"
        android:layout_toRightOf="@+id/textView7"
        androidPlot.title=" "
        androidPlot.graphWidget.marginTop="20dp"
          androidPlot.graphWidget.marginLeft="15dp"
            androidPlot.graphWidget.marginBottom="25dp"
            androidPlot.graphWidget.marginRight="10dp"

androidPlot.legendWidget.iconSizeMetrics.heightMetric.value="15dp"

androidPlot.legendWidget.iconSizeMetrics.widthMetric.value="15dp"
            androidPlot.legendWidget.heightMetric.value="25dp"
        title=" " />

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:text="OSCILOSCOPIO VIRTUAL"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textSize="40dp" />

    <TextView
        android:id="@+id/textView7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="      " />

    <TableLayout
        android:id="@+id/tableLayout1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true" >

        <TableRow
            android:id="@+id/tableRow6"
            android:layout_width="10px"
            android:layout_height="10px" >
```

```
            <TextView
                android:id="@+id/textView11"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginLeft="56dp"
                android:text="ESTADO"

android:textAppearance="?android:attr/textAppearanceLarge" />

            <ToggleButton
                android:id="@+id/btn_run"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="ToggleButton"
                android:textOff="RUN"
                android:textOn="STOP"
                android:textSize="30dp" />
        </TableRow>

        <TableRow
            android:id="@+id/tableRow1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" >

            <TextView
                android:id="@+id/textView2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:text="T/div:"

android:textAppearance="?android:attr/textAppearanceLarge"
                android:textSize="15dp"
                android:textStyle="bold" />

            <TextView
                android:id="@+id/txt_tiempo"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="100 us"

android:textAppearance="?android:attr/textAppearanceLarge"
                android:textSize="15dp"
                android:textStyle="bold" />
        </TableRow>

        <TableRow
            android:id="@+id/tableRow2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" >

            <Button
```

```
                android:id="@+id/btn_tmenos"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="&lt;"
                android:textSize="50dp"
                android:width="30dp" />

            <Button
                android:id="@+id/bn_tmas"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_gravity="fill"
                android:layout_weight="1"
                android:text=">"
                android:textSize="50dp" />
        </TableRow>

        <TableRow
            android:id="@+id/tableRow3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" >

            <TextView
                android:id="@+id/textView4"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:text="Escala"

android:textAppearance="?android:attr/textAppearanceLarge"
                android:textSize="15dp"
                android:textStyle="bold" />

            <TextView
                android:id="@+id/textView3"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Posicion   "

android:textAppearance="?android:attr/textAppearanceLarge"
                android:textSize="15dp"
                android:textStyle="bold" />
        </TableRow>

        <TableRow
            android:id="@+id/tableRow4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" >

            <Button
                android:id="@+id/btn_escalamas"
                android:layout_width="0dp"
                android:layout_height="match_parent"
```

```
            android:layout_weight="1"
            android:text="+"
            android:textSize="50dp"
            android:width="10dp" />

        <ImageButton
            android:id="@+id/btn_posmas"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:src="@drawable/pos_mas"
            android:text="+"
            android:textSize="50dp" />
    </TableRow>

    <TableRow
        android:id="@+id/tableRow5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >

        <Button
            android:id="@+id/btn_escalamenos"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:text="-"
            android:textSize="50dp"
            android:width="30dp" />

        <ImageButton
            android:id="@+id/btn_posmenos"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:src="@drawable/pos_menos"
            android:text="-"
            android:textSize="50dp" />
    </TableRow>

    <TableRow
        android:id="@+id/tableRow16"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >
    </TableRow>

    <TableRow
        android:id="@+id/tableRow15"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >
    </TableRow>

    <TableRow
        android:id="@+id/tableRow7"
        android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content" >

            <TextView
                android:id="@+id/textView5"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:gravity="right"
                android:text="Volt:"

android:textAppearance="?android:attr/textAppearanceMedium"
                android:textSize="15dp" />

            <TextView
                android:id="@+id/textView12"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:gravity="right"
                android:text=""

android:textAppearance="?android:attr/textAppearanceMedium"
                android:textSize="15dp" />
        </TableRow>

        <TableRow
            android:id="@+id/tableRow8"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" >

            <TextView
                android:id="@+id/textView13"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginLeft="151dp"
                android:gravity="right"

android:textAppearance="?android:attr/textAppearanceMedium"
                android:textSize="15dp" />

            <TextView
                android:id="@+id/txt_v1rms"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:gravity="center"

android:textAppearance="?android:attr/textAppearanceMedium"
                android:textSize="15dp" />

            <TextView
                android:id="@+id/textView6"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="VRMS"

android:textAppearance="?android:attr/textAppearanceMedium"
```

```
                android:textSize="15dp" />
        </TableRow>

        <TableRow
            android:id="@+id/tableRow9"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" >

            <TextView
                android:id="@+id/textView14"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginLeft="28dp"

android:textAppearance="?android:attr/textAppearanceMedium"
                android:textSize="15dp" />

            <TextView
                android:id="@+id/txt_v1pp"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_vertical"
                android:gravity="center"

android:textAppearance="?android:attr/textAppearanceMedium"
                android:textSize="15dp" />

            <TextView
                android:id="@+id/TextView04"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Vp-p"

android:textAppearance="?android:attr/textAppearanceMedium"
                android:textSize="15dp" />
        </TableRow>

        <TableRow
            android:id="@+id/tableRow13"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" >

            <TextView
                android:id="@+id/textView9"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:gravity="right"
                android:text="Frec:  "

android:textAppearance="?android:attr/textAppearanceMedium"
                android:textSize="15dp" />

            <TextView
                android:id="@+id/txt_f1"
```

```
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:gravity="center"

android:textAppearance="?android:attr/textAppearanceMedium"
                android:textSize="15dp" />

            <TextView
                android:id="@+id/txt_fu1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Hz"

android:textAppearance="?android:attr/textAppearanceMedium"
                android:textSize="15dp" />
        </TableRow>

        <TableRow
            android:id="@+id/tableRow14"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" >
        </TableRow>
    </TableLayout>

    <TextView
        android:id="@+id/txt_escala1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/plot_log"
        android:layout_toLeftOf="@+id/tableLayout1"
        android:text=""
        android:textColor="#fff"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textSize="30dp" />

    <TextView
        android:id="@+id/textView8"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_toLeftOf="@+id/textView1"
        android:text=" "
        android:textAppearance="?android:attr/textAppearanceLarge" />

</RelativeLayout>
```

MENU (main_activity.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright (C) 2012 Joe Smallman

     Licensed under the Apache License, Version 2.0 (the "License");
     you may not use this file except in compliance with the License.
     You may obtain a copy of the License at
```

```
          http://www.apache.org/licenses/LICENSE-2.0

     Unless required by applicable law or agreed to in writing, software
     distributed under the License is distributed on an "AS IS" BASIS,
     WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
     See the License for the specific language governing permissions and
     limitations under the License.
-->
<menu xmlns:android="http://schemas.android.com/apk/res/android">
         <item android:id="@+id/connection"
        android:icon="@drawable/bt"
        android:title="@string/connection"
        android:showAsAction="ifRoom|withText" >
       <menu>
             <item android:id="@+id/scan_connect"
                 android:icon="@drawable/bt"
                     android:title="@string/scan_connect" />
             <item android:id="@+id/disconnect"
                  android:icon="@drawable/bt"
                     android:title="@string/disconnect" />
       </menu>
    </item>
    <item android:id="@+id/data"
        android:icon="@android:drawable/ic_menu_agenda"
          android:title="@string/data">
        <menu >
            <item android:id="@+id/data_save"
                android:icon="@android:drawable/ic_menu_save"
                  android:title="@string/data_save"/>

            <item android:id="@+id/data_clear"
                android:icon="@android:drawable/ic_menu_delete"
                  android:title="@string/data_clear"/>
        </menu>
    </item>
    <item android:id="@+id/plot"
        android:icon="@android:drawable/ic_menu_edit"
          android:title="@string/plot">
        <menu >
            <item android:id="@+id/plot_load"
                android:icon="@android:drawable/ic_menu_search"
                  android:title="@string/plot_load"/>
            <item android:id="@+id/plot_settings"
                  android:title="@string/plot_settings"/>
        </menu>
    </item>

                                    </menu>
```

Strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Copyright (C) 2009 The Android Open Source Project
```

```
     Licensed under the Apache License, Version 2.0 (the "License");
     you may not use this file except in compliance with the License.
     You may obtain a copy of the License at

          http://www.apache.org/licenses/LICENSE-2.0

     Unless required by applicable law or agreed to in writing, software
     distributed under the License is distributed on an "AS IS" BASIS,
     WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
     See the License for the specific language governing permissions and
     limitations under the License.

     Modified by Joe Smallman April 2012 to make appropriate for serial
        communication with Arduino device
-->
<resources>

   <!--  Main activity -->
    <string name="hello">Hello World, ArduinoBTComActivity!</string>
    <string name="edittext_hint">Enter text to send to Arduino</string>
    <string name="button_send">Send</string>
    <string name="bt_not_available">Bluetooth no disponible</string>
    <string name="bt_enabled">Bluetooth activado</string>
    <string name="bt_not_enabled_leaving">Bluetooth desactivado</string>
      <string name="connect">Conectar</string>
      <string name="connecting">Conectando...</string>
      <string name="connected">Conectado</string>
      <string name="not_connected">Desconectado</string>
    <string name="ayuda">Acerca de.</string>
    <!--  Options Menu -->
    <string name="connection">Bluetooth</string>
    <string name="scan_connect">Conectar</string>
    <string name="disconnect">Desconectar</string>

    <!--  DeviceListActivity -->
    <string name="scanning">Buscar Dispositivo&amp;#8230;</string>
    <string name="select_device">Seleccione un dispositivo</string>
    <string name="none_paired">No hay dispositivos
sincronizados</string>
    <string name="none_found">No se encontro dispositivos
bluetooth</string>
    <string name="title_paired_devices">Dispositivos
sincronizados</string>
    <string name="title_other_devices">Otros dispositivos
bluetooth</string>
    <string name="button_scan">Buscar dispositivos Bluetooth</string>
    <string name="app_name">Android Measurment</string>
    <string name="data">Dato</string>
    <string name="data_save">Guardar</string>
    <string name="data_clear">Borrar</string>
    <string name="plot">Grafica</string>
    <string name="plot_load">Cargar</string>
```

```
    <string name="plot_settings">Configurar</string>
    <string name="select_file">Abrir</string>
    <string name="file_save_error">Error al guardar datos</string>
    <string name="file_saved">Datos guardados exitosamente</string>


</resources>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Copyright (C) 2012 Joe Smallman

     Licensed under the Apache License, Version 2.0 (the "License");
     you may not use this file except in compliance with the License.
     You may obtain a copy of the License at

          http://www.apache.org/licenses/LICENSE-2.0

     Unless required by applicable law or agreed to in writing, software
     distributed under the License is distributed on an "AS IS" BASIS,
     WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
     See the License for the specific language governing permissions and
     limitations under the License.
-->
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="tesis.android"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="9"
              android:targetSdkVersion="9"/>
      <uses-permission android:name="android.permission.BLUETOOTH" />
      <uses-permission
android:name="android.permission.BLUETOOTH_ADMIN" />
      <uses-permission android:name="android.permission.WAKE_LOCK"/>
      <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".Inicio"
            android:screenOrientation="portrait"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
```

```
        </activity>
        <activity android:name="tesis.android.DeviceListActivity"
                  android:label="@string/select_device"
                  android:theme="@android:style/Theme.Dialog"
                  android:configChanges="orientation|keyboardHidden" />
        <activity android:name="tesis.android.FileBrowserActivity"
                  android:label="@string/select_file"
                  android:theme="@android:style/Theme.Dialog"
                  android:configChanges="orientation|keyboardHidden" />
   <!-- ventana Ayuda-->
         <activity
             android:name="tesis.android.ayuda"
                android:label="@string/ayuda"
                android:screenOrientation="portrait"
                android:theme="@style/AppTheme"/>

     <!-- ventana multimetro-->
<activity
             android:name="tesis.android.Multimetro"
                android:label="@string/app_name"
                android:screenOrientation="portrait"
                android:theme="@style/AppTheme"/>

     <!-- ventana osciloscopio-->
<activity
             android:name="tesis.android.Osciloscopio"
                android:label="@string/app_name"
                android:screenOrientation="landscape"
                android:theme="@style/AppTheme"/>
    </application>

                              </manifest>
```

ANEXO E

PROGRAMACIÓN ARDUINO

PROGRAMACIÓN ARDUINO

CÓDIGO PARA EL MULTÍMETRO

```
float corriente, resistencia, v1, vr, ir, vca, vcc, ica, current;
float canal1, tabla[500], frec;
String rx, rxdata, rxosc, rxdataosc, unidad;
char inrx;
int comp, cruce1, aux=0;
boolean recivido=false;

long rango=2; // este es el rongo por el que se disparará la salida 2 y pasa a estado lógico 1
long ultimamedicion; // contiene el valor de la última medición que disparó a lógico 1, la salida 2
int ciclo=0; // 1=alto 0=bajo
int cambiodeciclo=0;
int picodetension;
int valledetension=1023;
long contadorvisualizacion;
long contadorciclo , vf;
void setup() {
  Serial.begin(38400);
  Serial1.begin(38400);
  pinMode(DAC0,OUTPUT);
  analogWriteResolution(10);

}

void loop() {
 Serial.flush();
  if(Serial1.available()){serialEvent();};
  //Serial.println("lazo loop");
  frec=analogRead(A4);
  frec=analogRead(A4);
  if (frec!=0)
   {

     frecuencia();
   }
  if (recivido)
 {
   if (rx == "multimetro")
   {
```

```
      rx="";
      rxdata="offall";
      recivido=false;
      multimetro();
     }


    rx="";
    recivido=false;

  }
  loop();
}

  // put your main code here, to run repeatedly:
  /*analogReadResolution(12);
  v1=analogRead(A11);
  v1=(v1*3.3)/4095;
  ir=v1/11360000;
  vr=3.3-v1;
  resistencia=vr/ir;
  Serial.print("Voltaje en la resistencia conocida= ");
  Serial.println(v1,6);
   Serial.print("corriente en la resistencia conocida= ");
  Serial.println(ir,4);
  Serial.print("resistencia medida= ");
  Serial.println(resistencia,4);
  delay(500);*/

void multimetro()
{
 //Serial.println("lazo multimetro");
  if(Serial1.available()){serialEvent();};
  analogReadResolution(12);

   if (recivido)
  {
  rx.trim();
  if(rx=="Vca" | rx=="Vcc" | rx=="Icc" | rx=="Ica" | rx=="Res" | rx=="offall" |
rx== "x")
  {
    rxdata=rx;
  }
  if(rxdata=="x")
 {
  rx="";
```

```
 Serial.println(rxdata);
 rxdata="";
 recivido=false;
    loop();
}
   rx="";
   recivido=false;
}

if (rxdata == "Vca")
 {
   vca=analogRead(A0);
   vca=(vca*750)/4095;
   //Serial.println(vca);
   Serial1.println(vca);
    delay(100);
  }
if (rxdata == "Vcc")
 {
   vcc=analogRead(A1);
   vcc=(vcc*200)/4095;
  // Serial.println(vcc);
   Serial1.print(vcc);
   Serial1.println("  V");
    delay(100);
  }
  if (rxdata == "Icc" | rxdata == "Ica")
 {
   current=analogRead(A2);
   current=(current*30)/4095;
   if (current < 1)
   {
     current= current*1000;
     unidad= " mA";
   }
   else
   {
     unidad = " A";
   }
  // Serial.println(current);
   Serial1.print(current);
   Serial1.println(unidad);
    delay(100);
  }

  if (rxdata == "Res")
```

```
 {
  resistencia=analogRead(A3);
  resistencia=(resistencia*10000000)/4095;
   if (resistencia < 1000)
  {
    unidad= " Ohm";
  }
  if (resistencia > 1000 & resistencia < 1000000)
  {
    resistencia= resistencia/1000;
    unidad= " Kohm";
  }
  if (resistencia > 1000000)
  {
    resistencia= resistencia/1000000;
    unidad= " Mohm";
  }


  Serial1.print(resistencia);
  Serial1.println(unidad);
  delay(100);
 }
 if (rxdata == "offall" | rxdata == " ")
 {

  Serial1.println("OFF");
  delay(50);
 }
  multimetro();
}

/*********************************************************************/
/*********************codigo del osciloscopio*********************/
/*********************************************************************/
/*********************************************************************/
/*********************************************************************/

/*void osciloscopio(){
 if(Serial1.available()){serialEvent();};
 if(recivido)
 {
 rx.trim();

 if(rx=="run" | rx=="offall" | rx=="inicio")
 {
```

```
  rxdata=rx;

 }
 if(rxdata=="inicio")
 {
  rx="";
  recivido=false;
  rxdata="";
   loop();
 }
  rx="";
  recivido=false;
 }
  if(rxdata=="run")
  {
    canal1 = analogRead(A4);
    canal1=(canal1*120)/1024;
   Serial1.println(canal1);
  /*
    for(contador=0;contador<500;contador++)
   {
    canal1 = analogRead(A4);
    canal1=(canal1*120)/1024;
    tabla[contador]=canal1;
   }

   for(int aux=0;aux<500;aux++)
    {
       Serial1.println(tabla[aux]);
       Serial.println(aux);
       Serial.println(tabla[aux]);
      //delayMicroseconds(1000);
    }

   delayMicroseconds(1000);
  }
        osciloscopio();
}*/
/*****************************************************************/
/***************frecuencimetro para el osciloscopio*************/
/*****************************************************************/
/*****************************************************************/
/*****************************************************************/
void frecuencia(){
```

```
 long sensorValue = analogRead(A4); // Se vuelca a la varibla sensorValue el valor de la tensión medida por el pin 0 analógico

  if (micros()>contadorvisualizacion+1000000) // Muestra por el puerto serie los resultados, transcurrido un segundo

  {

    // AQUÍ SE IMPRIME EN EL PUERTO SERIE LOS RESULTADOS

   vf=contadorciclo*(3.413);
   if(vf>1023)
   {
    vf=1023;
   }
    analogWrite(DAC0,vf);
    Serial.println(vf);
    if(aux==2)
    {
      analogWrite(DAC0,0);
      aux=0;
     loop();
    }
    if(contadorciclo==0)
   {
     aux++;

    }


      rango=(2+((picodetension-valledetension)/5)); // SE CALCULA EL GANGO MÁS ADECUADO PARA LA SEÑAL, CON EL FIN, DE REDUCIR ERRORES EN SEÑALES CON RUIDO DE FONDO
      contadorvisualizacion=micros();//SE ASIGNA A LA VARIABLE CONTADORVISUALIZACIÓN EL TIEMPO EN MICROSENGUNDOS QUE LLEVA EL PROGRAMA EN EJECUCIÓN
      picodetension=sensorValue;//SE ASIGNA A LA VARIABLE PICODETENSIÓN EL VALOR DE LA TENSIÒN LEIDA POR EL PUERTO ANALÓGICO CERO
      valledetension=sensorValue;//SE ASIGNA A LA VALLEDETENSION EL VALOR DE LA TENSIÒN LEIDA POR EL PUERTO ANALÓGICO CERO
      contadorciclo=0; // SE PONE A CERO LOS CICLOS CONTADOS O HERCIOS.
  }
```

```
 if (sensorValue >= ( ultimamedicion+rango) ) // La salida 2 pasa a 1 logico si la tensión medida en la entrada analógica 0 es mayor que la anterior lectura + latensión de RANGO

 {

  ultimamedicion = sensorValue;  // SE ASIGANA A LA VARIABLE ULTIMAMEDICION EL VALOR LEIDO POR LA ENTRADA ANALÓGICA CERO
  ciclo=1;

  if (sensorValue>picodetension) // SI LA TENSIÓN MEDIDA POR LA ENTRADA CERO, ES LA MAYOR DETECTADA, SE ASIGNA A LA VARIABLE PICODETENSIÓN EL VALOR LEYDO POR AL ENTRADA CERO ANALÓGICA

  {
  picodetension=sensorValue; // SE ASIGNA EL VALOR LEYDO POR LA ENTRADA CERO ANALÓGICA A LA VARIABLE PICODETENSIÓN.
  }

 }



 if (sensorValue <= ( ultimamedicion-rango))  // La salida 2 pasa a 1 lógico si la tensión medida en la entrada analógica 0 es menor que la anterior lectura - la tensión de RANGO

 {

  ultimamedicion = sensorValue; // SE ASIGNA A LA VARIABLE ULTIMAMEDICIÓN LA LECTURA MEDIDA POR EL PUERTO ANALÓGICO CERO
  ciclo=0; // EL CICLO SE PONE A CERO, ES DECIR, QUE EL VOLTAJE EMPIEZA A BAJAR DESDE EL PICO DE TENSIÓN MÁS ALTA

  if (sensorValue<valledetension) // SE CUMPLE LA CONDICIÓN SI LA TENSIÓN DETECTADA POR EL PUERTO ANALÓGICO CERO ES MENOR QUE LA CONTENIDA EN LA VARIABLE VALLEDETENSIÓN
  {
  valledetension=sensorValue; //Se asigna a la variable valledetensión el valor medido por la entrada analógica cero
  }
```

```
  }


if (ciclo==1 && cambiodeciclo==0)

    {
      cambiodeciclo=1;
      contadorciclo++;
    }

if (ciclo==0 && cambiodeciclo==1)

    {
      cambiodeciclo=0;
    }
    frecuencia();
}
void serialEvent() {
  while (Serial1.available()) {
  inrx = Serial1.read();
   if (inrx == '\n') {
     recivido = true;
    }
    else
    {
       rx += inrx;
    }
  }

 Serial1.flush();
}
```

CÓDIGO PARA OSCILOSCOPIO

```
int led = 13;
boolean ledon;
float frecuencia;
int analogInputA1 = A1;

const int MAX_INDEX = 400;
int var[MAX_INDEX];
int varIndex;
float voltajemas, voltajemenos, voltaje;
float inicio, fin, frec;
```

```
volatile boolean dataReady;

int FREQ_1KHz = 80000;

void TC3_Handler(){
   TC_GetStatus(TC1, 0);
   //analogReadResolution(8);
     voltajemas=analogRead(A1);  //read ADC input
     voltaje=((voltajemas-512)*118)/1024;
    // voltaje=(voltajemas*118)/254;
    /* if(voltaje==0)
     {
       voltajemenos=analogRead(A2);
       voltaje=(-1)*((voltajemenos*118)/254);
     }
     //voltajemas=(voltajemas*120)/1023;
     //voltajemenos=analogRead(A2);  //read ADC input
     //voltajemenos=(voltajemenos*120)/1023;
      //voltaje=voltajemas-voltajemenos;
  */
    Serial1.print("vp1");
      Serial1.print(voltaje);
      Serial1.println("&");
     analogReadResolution(10);
     frec=analogRead(A0);
 frec=((frec+60)*2.769)/1023;//

      Serial1.print("f1");
      Serial1.print(frec);
      Serial1.println("&");
       //Serial1.flush();


}

void startTimer(Tc *tc, uint32_t channel, IRQn_Type irq, uint32_t frequency){

   //Enable or disable write protect of PMC registers.
   pmc_set_writeprotect(false);
   //Enable the specified peripheral clock.
   pmc_enable_periph_clk((uint32_t)irq);

   TC_Configure(tc,                                                    channel,
TC_CMR_WAVE|TC_CMR_WAVSEL_UP_RC|TC_CMR_TCCLKS_TIMER_
CLOCK4);
   uint32_t rc = VARIANT_MCK/128/frequency;
```

```
    TC_SetRA(tc, channel, rc/2);
    TC_SetRC(tc, channel, rc);
    TC_Start(tc, channel);

    tc->TC_CHANNEL[channel].TC_IER = TC_IER_CPCS;
    tc->TC_CHANNEL[channel].TC_IDR = ~TC_IER_CPCS;
    NVIC_EnableIRQ(irq);
}

void setup() {
  pinMode(led, OUTPUT);

  Serial1.begin(38400);
  //Serial.begin(38400);
  //analogReadResolution(10);  //set ADC resolution to 8 bits
  dataReady = false;
  varIndex = 0;
  startTimer(TC1, 0, TC3_IRQn, FREQ_1KHz);
}

void loop() {

   Serial1.print("vp1");
      Serial1.print(voltaje);
      Serial1.println("&");
      Serial1.print("f1");
      Serial1.print(frec);
      Serial1.println("&");
    Serial1.flush();

}
```

ANEXO F

CERTIFICADOS DE CALIBRACIÓN

Agilent Technologies

Agilent Technologies Microwave Products (M) Sdn. Bhd.
Bayan Lepas Free Industrial Zone
11900 Penang
Malaysia

5962-0476

Agilent Approved Calibration Provider #93107

Certificate Of Calibration

Certificate No: U1253BMY53120140

Manufacturer: Agilent Technologies
Model No: U1253B
Options Installed With Specifications: N/A

Description: U1253B Handheld Digital Multimeter
Serial No: MY53120140

Date of Calibration: 19-Oct-2013
Temperature: (23±5) °C
Procedure: CELQ-056-AT-25

Humidity: (50 to 80)% RH

This certifies that the above product was calibrated by an approved Agilent calibration provider in accordance with procedures approved by Agilent Technologies.

As Received: Factory tested - No incoming data available

As Shipped Conditions: At the completion of the calibration, measured values were IN SPECIFICATION at the points tested

These calibration procedures and test points are those recommended in a procedure developed by Agilent.

Remarks or special requirements: For inquiries regarding this calibration please contact Agilent Technologies at the above address.

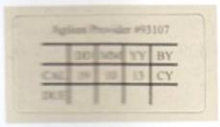
Traceability Information: Traceability is to the International System of Units (SI), consensus standards or ratio type measurements through national standards realiazed and maintained by the NIST U.S., NRC Canada, NMIJ Japan, KRISS Korea, Euramet members (NPL, PTB, etc.), NML-SIRIM in Malaysia or other National Measurement Institutes signatories to the CIPM MRA. Supporting documentation relative to traceability is available for review by appointment. This report shall not be reproduced, except in full, without prior written approval of the calibration facility.

Calibration Equipment Used: ***Date Used: Date equipment used in this Calibration.***

Model Number	Model Description	Trace Number	Date Used	Cal Due Date
Fluke 5520A	Multifunction Calibrator	S/N 2262901	19-Oct-2013	14-Dec-2013

Print Date: 19-Oct-2013

Low Eng Thiam
Quality Manager
Agilent Calibration Provider #93107



Agilent Technologies

Agilent Technologies Microwave Products (M) Sdn. Bhd.
Bayan Lepas Free Industrial Zone
11900 Penang
Malaysia

5965-6007

UK6 Test Report

Certificate No: U1253BMY53120140
Model: U1253B
Serial Number: MY53120140

Test Date: 19-Oct-2013

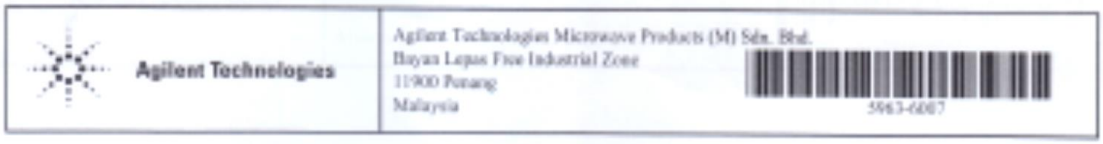
TEST DESCRIPTION	READING	ERROR	1 YEAR SPEC
ACV 5V 1kHz on 5V Range	+5.0000	+0.0002%	+/-0.4500%
ACV 5V 10kHz on 5V Range	+4.9861	-0.2778%	+/-1.5800%
ACV 4.5V 20kHz on 5V Range	+4.5011	+0.0247%	+/-3.7667%
ACV 4.5V 30kHz on 5V Range	+4.5058	+0.1287%	+/-3.7667%
ACV 4.5V 100kHz on 5V Range	+4.5924	+2.0533%	+/-3.7667%
ACV 50V 1kHz on 50V Range	+50.004	+0.0084%	+/-0.4500%
ACV 50V 10kHz on 50V Range	+49.924	-0.1514%	+/-1.5800%
ACV 45V 20kHz on 50V Range	+44.999	-0.0013%	+/-3.7667%
ACV 45V 30kHz on 50V Range	+44.918	-0.1831%	+/-3.7667%
ACV 45V 100kHz on 50V Range	+45.699	+1.5536%	+/-3.7667%
ACV 500V 1kHz on 500V Range	+500.09	+0.0184%	+/-0.4500%
ACV 1000V 1kHz on 1000V Range	+1000.5	+0.0535%	+/-0.8000%
ACV-FREQ 1kHz 0.48V on 9.9999kHz Range	+1.0000	-0.0010%	+/-0.0500%
AC-DUTY 50V 0.05kHz on 5V Range	+050.00	+0.0012%	+/-0.6300%
DCV 5V 0Hz on 5V Range	+5.0001	+0.0016%	+/-0.0350%
DCV 50V 0Hz on 50V Range	+50.000	-0.0002%	+/-0.0350%
DCV 500V 0Hz on 500V Range	+500.02	+0.0036%	+/-0.0400%
DCV 1000V 0Hz on 1000V Range	+1000.0	-0.0049%	+/-0.0800%
(DC/AC)ACV 5V 1kHz on 5V Range	+5.0006	+0.0126%	+/-0.4500%
(DC/AC)ACV 5V 10kHz on 5V Range	+4.9867	-0.2656%	+/-1.5800%
(DC/AC)ACV 4.5V 20kHz on 5V Range	+4.5015	+0.0324%	+/-3.7667%
(DC/AC)ACV 4.5V 100kHz on 5V Range	+4.5928	+2.0616%	+/-3.7667%
(DC/AC)ACV 50V 1kHz on 50V Range	+50.005	+0.0102%	+/-0.4500%
(DC/AC)ACV 50V 10kHz on 50V Range	+49.926	-0.1474%	+/-1.5800%
(DC/AC)ACV 45V 20kHz on 50V Range	+45.002	+0.0053%	+/-3.7667%
(DC/AC)ACV 45V 100kHz on 50V Range	+45.703	+1.5620%	+/-3.7667%
(DC/AC)ACV 500V 1kHz on 500V Range	+500.11	+0.0214%	+/-0.4500%
(DC/AC)ACV 1000V 1kHz on 1000V Range	+1000.5	+0.0535%	+/-0.8000%
DCmV 50mV 0Hz on 50mV Range	+49.998	-0.0046%	+/-0.1500%
DCmV 500mV 0Hz on 500mV Range	+500.02	+0.0048%	+/-0.0350%
DCmV -500mV 0Hz on 500mV Range	-500.02	+0.0048%	+/-0.0350%
DCmV 1000mV 0Hz on 1000mV Range	+1000.0	-0.0001%	+/-0.0750%
DCmV -1000mV 0Hz on 1000mV Range	-1000.0	+0.0010%	+/-0.0750%
ACmV 50mV 1kHz on 50mV Range	+50.010	+0.0208%	+/-0.4800%
ACmV 50mV 10kHz on 50mV Range	+50.025	+0.0490%	+/-0.8300%
ACmV 50mV 20kHz on 50mV Range	+50.014	+0.0282%	+/-3.7400%
ACmV 45mV 30kHz on 50mV Range	+44.980	-0.0447%	+/-3.7667%
ACmV 45mV 100kHz on 50mV Range	+44.375	-1.3891%	+/-3.7667%
ACmV 500mV 45Hz on 500mV Range	+499.16	-0.1672%	+/-0.4500%
ACmV 500mV 1kHz on 500mV Range	+500.07	+0.0136%	+/-0.4500%
ACmV 500mV 10kHz on 500mV Range	+500.57	+0.1144%	+/-0.8300%

UK6 Test Report

Certificate No: U1253BMY53120140
Model: U1253B
Serial Number: MY53120140

Test Date: 19-Oct-2013

TEST DESCRIPTION	READING	ERROR	1 YEAR SPEC
ACmV 500mV 20kHz on 500mV Range	+500.14	+0.0282%	+/-3.7400%
ACmV 450mV 30kHz on 500mV Range	+449.27	-0.1631%	+/-3.7667%
ACmV 450mV 100kHz on 500mV Range	+443.62	-1.4184%	+/-3.7667%
ACmV 1000mV 1kHz on 1000mV Range	+1000.0	-0.0033%	+/-0.6500%
ACmV 1000mV 10kHz on 1000mV Range	+1000.3	+0.0250%	+/-1.1500%
ACmV 1000mV 20kHz on 1000mV Range	+0999.6	-0.0376%	+/-4.7000%
ACmV 1000mV 30kHz on 1000mV Range	+1001.8	+0.1830%	+/-4.7000%
ACmV 1000mV 100kHz on 1000mV Range	+1011.1	+1.1050%	+/-4.7000%
OHM 500ohm on 500ohm Range	+500.01	+0.0010%	+/-0.0700%
OHM 5kohm on 5kohm Range	+5.0007	+0.0132%	+/-0.0600%
OHM 50kohm on 50kohm Range	+50.003	+0.0060%	+/-0.0600%
OHM 500kohm on 500kohm Range	+500.01	+0.0026%	+/-0.0600%
OHM 5Mohm on 5Mohm Range	+4.9996	-0.0084%	+/-0.1600%
OHM 50Mohm on 50Mohm Range	+49.997	-0.0054%	+/-1.0100%
OHM 450Mohm on 500Mohm Range	+449.86	-0.0322%	+/-8.0222%
OHM 50nS on 50nS Range	+050.00	+0.0024%	+/-1.2000%
DIODE 1V 0Hz on 2V Range	+1.0001	+0.0100%	+/-0.1000%
Hz 100kHz 200mV on 999.99kHz Range	+100.00	-0.0013%	+/-0.0520%
Hz 10MHz 600mV on 99.999MHz Range	+10.000	-0.0012%	+/-0.0520%
Cx 10nF on 10nF Range	+10.001	+0.0100%	+/-1.0800%
Cx 100nF on 100nF Range	+099.92	-0.0764%	+/-1.0500%
Cx 1000nF on 1000nF Range	+0999.8	-0.0229%	+/-1.0500%
Cx 10uF on 10uF Range	+10.010	+0.1030%	+/-1.0500%
Cx 100uF on 100uF Range	+100.02	+0.0150%	+/-1.0500%
Cx 1000uF on 1000uF Range	+1000.0	-0.0046%	+/-1.0500%
Cx 10mF on 10mF Range	+10.002	+0.0220%	+/-1.0500%
Cx 100mF on 100mF Range	+099.48	-0.5223%	+/-3.1000%
DCuA 500uA 0Hz on 500uA Range	+500.02	+0.0034%	+/-0.0600%
DCuA 5000uA 0Hz on 5000uA Range	+4999.8	-0.0044%	+/-0.0600%
ACuA 500uA 1kHz on 500uA Range	+500.05	+0.0098%	+/-0.7400%
ACuA 500uA 20kHz on 500uA Range	+500.10	+0.0198%	+/-0.7900%
ACuA 5000uA 1kHz on 5000uA Range	+4999.4	-0.0128%	+/-0.7400%
ACuA 5000uA 20kHz on 5000uA Range	+4999.5	-0.0092%	+/-0.7900%
DCmA 50mA 0Hz on 50mA Range	+49.997	-0.0068%	+/-0.1600%
DCmA 400mA 0Hz on 440mA Range	+399.98	-0.0042%	+/-0.1625%
ACmA 50mA 1kHz on 50mA Range	+50.016	+0.0314%	+/-0.7400%
ACmA 50mA 20kHz on 50mA Range	+50.104	+0.2088%	+/-0.7900%
ACmA 400mA 45Hz on 440mA Range	+399.41	-0.1467%	+/-0.7500%
ACmA 400mA 1kHz on 440mA Range	+400.02	+0.0053%	+/-0.7500%
DCA 5A 0Hz on 5A Range	+4.9997	-0.0066%	+/-0.3200%
DCA 10A 0Hz on 10A Range	+09.996	-0.0428%	+/-0.3500%



UK6 Test Report

Certificate No: U1253BMY53120140 Test Date: 19-Oct-2013
Model: U1253B
Serial Number: MY53120140

TEST DESCRIPTION	READING	ERROR	LINEAR SPEC
ACA 5A 1kHz on 5A Range	+5.0035	+0.0708%	+/-0.7400%
ACA 3A 5kHz on 5A Range	+3.0048	+0.1593%	+/-3.2000%
ACA 10A 1kHz on 10A Range	+09.998	-0.0239%	+/-0.9000%
OUT-Square 120Hz on 2.8V Range	+120.00	+0.0017%	+/-0.0217%
OUT-Square 4800Hz on 2.8V Range	+4800.08	+0.0017%	+/-0.0054%
OUT-DUTY 25V on 2.8V Range	+025.01	+0.0232%	+/-1.5936%
OUT-DUTY 50V on 2.8V Range	+050.01	+0.0124%	+/-0.7968%
OUT-DUTY 75V on 2.8V Range	+075.01	+0.0136%	+/-0.5312%
OUT-Amplitude 2.8V on 2.8V Range	+02.765	-1.2443%	+/-7.1429%
TEMP 100deg C on 1372deg C Range	+0099.9	-0.1157%	+/-2.0000%
TEMP 0CEL on 1372deg C Range	0000.0	+0deg C	+/-0001.0deg C

Agilent Technologies

Agilent Technologies Microwave Products (M) Sdn. Bhd.
Bayan Lepas Free Industrial Zone
11900 Penang
Malaysia

5971-0077

Recommended Due Date for Adjustment/Calibration
(One-Year Storage Limit)

Model No: **U1251A / U1252A / U1253A / U1241A / U1242A**
U1251B / U1252B / U1253B / U1241B / U1242B

We have determined that the factory calibration of this instrument is not significantly affected by storage of up to one year before the first-time use. This instrument is warranted to meet the published specifications at the first adjustment/calibration due date, assuming that the date is determined using the following instructions.

1. Determine the DATE TESTED from the Certificate of Calibration which is shipped with the instrument. This is the date when the instrument was calibrated before it was shipped from the factory.

2. Select the desired adjustment/calibration interval. Refer to the instrument's manuals for calibration interval recommendations or to the requirements of your organization's quality system.
Please note that the Cal Interval may be other than 1 year, e.g.:2 year

3. Determine the date when the next adjustment/calibration is due (refer to the examples shown below).

Example 1: *First-time use of the instrument is* ***less than*** *one year after DATE TESTED.*
Date Tested at Factory: May 1, 2007
Selected Calibration Interval: 1 Year
First-Time Use of Instrument: August 5, 2007

First-Time Use + Selected Interval = Date for Next Adjustment/Calibration
---
August 5, 2007 + 1 Year = August 5, 2008

Example 2: *First-time use of the instrument is* ***more than*** *one year after DATE TESTED.*
Date Tested at Factory: April 3, 2007
Selected Calibration Interval: 1 Year
First-Time Use of Instrument: May 22, 2008
Date Tested at Factory + 365 Days + Selected Interval = Date for Next Adjustment/Calibration
---
April 3, 2007 + 365 Days + 1 Year = April 3, 2009

Tay Eng Su
Quality Manager

Rev.C

ANEXO G

MANUAL DE USUARIO

MANUAL DE USUARIO

INSTRUMENTO VIRTUAL UTILIZADO EN LA MEDICIÓN DE VARIABLES ELÉCTRICAS CON DISPOSITIVOS MÓVILES BAJO LA PLATAFORMA ANDROID

ÍNDICE DE CONTENIDOS

CARACTERÍSTICAS TÉCINAS DEL MÓDULO DEL INSTRUMENTO VIRTUAL 2

PERSPECTIVA DEL MÓDULO DEL INSTRUMENTO VIRTUAL 2

FUNCIONAMIENTO DEL MÓDULO 4

FUNCIÓN MULTIMETRO 4

MEDICION DE VOLTAJE ALTERNO 5

MEDICION DE VOLTAJE CONTINUO 7

MEDICION DE CORRIENTE ALTERNA 8

MEDICION DE CORRIENTE CONTINUA 9

MEDICION DE RESISTENCIA 11

FUNCIONAMIENTO DEL OSCILOSCOPIO 12

PREGUNTAS FRECUENTES Y POSIBLES FALLOS 14

CARACTERÍSTICAS TÉCINAS DEL MÓDULO DEL INSTRUMENTO VIRTUAL

- Dimensiones: Largo: 30 cm
  Ancho: 26 cm
  Alto: 20 cm
- Voltaje de Alimentación: 120 V_{CA} – 50 / 60 H_Z
- Entrada de Voltaje AC: Max. 750V_{AC}
- Entrada de Voltaje DC: Max. 200V_{DC}
- Entrada de Corriente AC: Max. 20A_{AC}
- Entrada de Corriente DC: Max. 20A_{DC}
- Entrada de Resistencia: 100Ω– 2MΩ
- Entrada de la señal al osciloscopio: Max 120 V_{RMS}

PERSPECTIVA DEL MÓDULO DEL INSTRUMENTO VIRTUAL

El Instrumento Virtual consta de varias conexiones que permiten al usuario medir las distintas variables eléctrivas, voltajes AC/DC, corrientes AC/DC, resistencia, y hace la función de un osciloscopio básico.

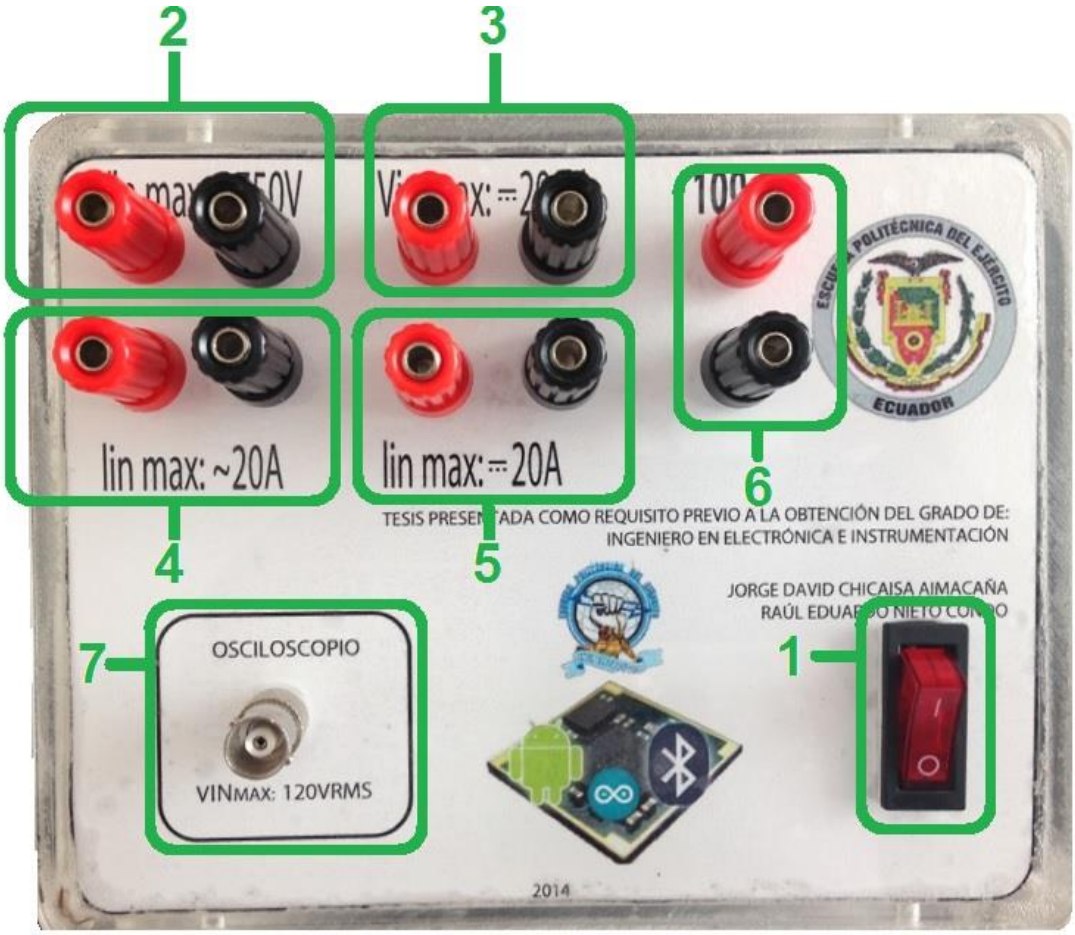


Figura 1. Parte Frontal del Módulo

1. Interruptor con Luz Piloto de Encendido / Apagado
2. Entrada Voltímetro AC
3. Entrada Voltímetro DC
4. Entrada Amperímetro AC
5. Entrada Amperímetro DC
6. Entrada Óhmetro
7. Entrada Osciloscopio

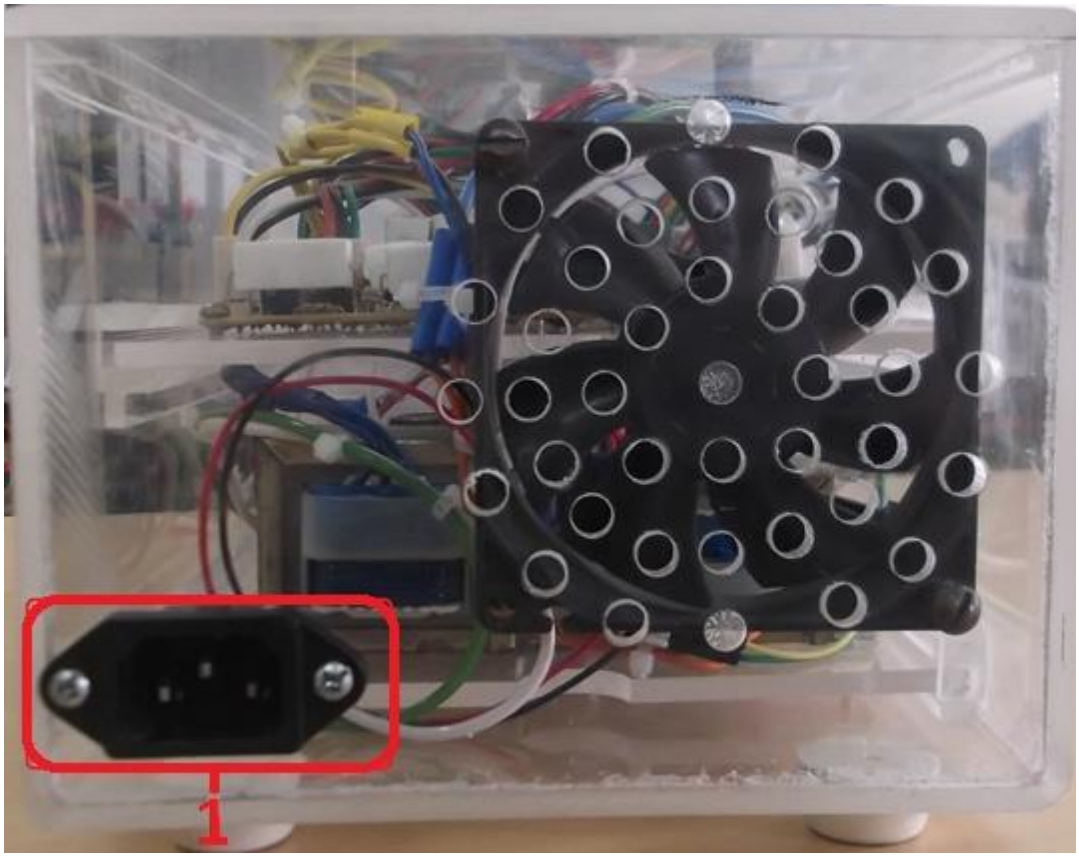


Figura 2. Parte trasera del Módulo

1. Alimentación del módulo 110V_{AC}

FUNCIONAMIENTO DEL MÓDULO

Para tener un correcto funcionamiento del dispositivo hay que tener en cuenta las siguientes recomendaciones:

FUNCIÓN MULTIMETRO

1. Encienda el módulo de adquisición de señales

Figura 3. Encendido del módulo de adquisición de señales

2. Empareje el bluetooth llamado arduino y osciloscopio al dispositivo android, insertando el código “1234” para los dos casos.

MEDICION DE VOLTAJE ALTERNO

1. Conecte los terminales del voltimetro AC, al voltaje que se desea medir, teniendo en cuenta que el voltaje máximo de entrada es de $750V_{AC}$

Figura 4. Conexión voltímetro AC

3. Ejecutar la aplicación en el dispositivo Android, seleccionar la opción multímetro y seleccionar medición de voltaje AC

Figura 4. Layout de la aplicación Android, modo voltímetro AC

MEDICION DE VOLTAJE CONTINUO

1. Conecte los terminales del voltimetro DC, al voltaje que se desea medir, teniendo en cuenta que el voltaje máximo de entrada es de 200 V_{CD}

Figura 5. Conexión voltímetro DC

2. Ejecutar la aplicación en el dispositivo Android, seleccionar la opción multímetro y seleccionar medición de voltaje DC

Figura 6. Layout de la aplicación Android, modo voltímetro DC

MEDICION DE CORRIENTE ALTERNA

1. Conecte los terminales del amperimetro AC en serie al circuito en el cual se desea medir la corriente, teniendo en cuenta que la corriente máxima de entrada es de 20 Amperios.

Figura 7. Conexión amperímetro AC

2. Ejecutar la aplicación en el dispositivo Android, seleccionar la opción multímetro y seleccionar medición de amperímetro AC

Figura 8. Layout de la aplicación Android, modo amperímetro AC

MEDICION DE CORRIENTE CONTINUA

1. Conecte los terminales del amperimetro DC en serie al circuito en el cual se desea medir la corriente, teniendo en cuenta que la corriente máxima de entrada es de 20 Amperios.

Figura 9. Conexión amperímetro DC

2. Ejecutar la aplicación en el dispositivo Android, seleccionar la opción multímetro y seleccionar medición de amperímetro DC

Figura 10. Layout de la aplicación Android, modo amperímetro DC

MEDICION DE RESISTENCIA

1. Conecte los terminales del óhmetro a la carga que se desa medir, teniendo en cuenta que el rango de medición están entre 100 – 2MΩ.

Figura 11. Conexión óhmetro

2. Ejecutar la aplicación en el dispositivo Android, seleccionar la opción multímetro y seleccionar medición de resistencia.

Figura 12. Layout de la aplicación Android, modo óhmetro

FUNCIONAMIENTO DEL OSCILOSCOPIO

1. Conecte la punta del osciloscopio en el conector BNC de entrada.

Figura 13. Conexión osciloscopio

2. Ejecutar la aplicaciónen el dispositivo Android, seleccionar la opción osciloscopio.

Figura 14. Layout de la aplicación Android, modo osciloscopio

3. Presione el botón run.

PREGUNTAS FRECUENTES Y POSIBLES FALLOS

1. La aplicación multímetro no se inicia.

   Sol: Revise que el módulo este encendido y la entrada del osciloscopio desconectada.

2. La señal mostrada en el osciloscopio no corresponde a la señal que ingresa al mismo.

   Sol: Revise que la punta esté conectada correctamente, y que el selector de escala de la misma se encuentre en la posición x10.

3. No se muestra el valor de la corriente DC en la aplicación.

   Sol: Revise que la entrada de corriente este correctamente conectada, asegúrese que el terminal positivo de la corriente que se desea medir ingrese al borne positivo del amperímetro (color rojo), y el borne negativo (color negro) cierre el circuito.

4. El módulo no se enciende.

   Sol.: Cerciorarse que los fusibles del módulo se encuentren en buen estado, para ello desconecte el cable de alimentación, retirar el conector de entrada ($110V_{AC}$) y retirar la tapa frontal del módulo, de tal forma que se puedan extraer las placas que contienen los fusibles.

Latacunga, Mayo del 2014

AUDITORÍA

ELABORADO POR:

Jorge David Chicaisa Aimacaña	Raúl Eduardo Nieto Condo
C.I: 0503251548	C.I: 1804249959

APROBADO POR:

Ing. José Bucheli

DIRECTOR DE LA CARRERA DE INGENIERÍA EN ELECTRÓNUICA E INSTRUMENTACIÓN

CERTIFICADO POR:

Dr. Rodrigo Vaca C.

SECRETARIO ACADÉMICO

UNIDAD DE ADMISIÓN Y REGISTRO