



UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO  
CENTRO UNIVERSITARIO UAEM ECATEPEC



## Multímetro para dispositivos móviles

T E S I S

Para obtener el título de:

**Ingeniero en computación**

P R E S E N T A:

**Jorge Luis Castillo López**

**Víctor Hugo Omaña Vera**

ASESOR:

**Dr. En C. Rodolfo Zolá García Lozano**

Ecatepec de Morelos. Estado de México.



**CARTA DE CESIÓN DE DERECHOS DE AUTOR**

El (la) que suscribe **C.C. JORGE LUIS CASTILLO LOPEZ Y VICTOR HUGO OMAÑA VERA** Autor del trabajo escrito de evaluación profesional en la opción de **TESIS** con el título **“MULTIMETRO PARA DISPOSITIVOS MOVILES”** por medio de la presente con fundamento en lo dispuesto en los artículos 5, 18, 24, 25, 27, 30, 32 y 148 de la Ley Federal de Derechos de Autor, así como los artículos 35 y 36 fracción II de la Ley de la Universidad Autónoma del Estado de México; manifiesto mi autoría y originalidad de la obra mencionada que se presentó en el Centro Universitario UAEM Ecatepec para ser evaluada con el fin de obtener el Título Profesional de **INGENIERIA EN COMPUTACION**

Así mismo expreso mi conformidad de ceder los derechos de reproducción, difusión y circulación de esta obra, en forma NO EXCLUSIVA, a la Universidad Autónoma del Estado de México; se podrá realizar a nivel nacional e internacional, de manera parcial o total a través de cualquier medio de información que sea susceptible para ello, en una o varias ocasiones, así como en cualquier soporte documental, todo ello siempre y cuando sus fines sean académicos, humanísticos, tecnológicos, históricos, artísticos, sociales, científicos u otra manifestación de la cultura.

Entendiendo que dicha cesión no genera obligación alguna para la Universidad Autónoma del Estado de México y que podrá o no ejercer los derechos cedidos.

Por lo que el autor da su consentimiento para la publicación de su trabajo escrito de evaluación profesional.

- a) Texto completo
- b) Por capítulo
- c) Solamente portada y tabla de contenido


Se firma presente en la ciudad de Ecatepec de Morelos, Estado de México, a los 2 días del mes de Junio de 2017.

  
 \_\_\_\_\_  
**JORGE LUIS CASTILLO LOPEZ**

  
 \_\_\_\_\_  
**VICTOR HUGO OMAÑA VERA**



**UAEM** | Universidad Autónoma  
del Estado de México

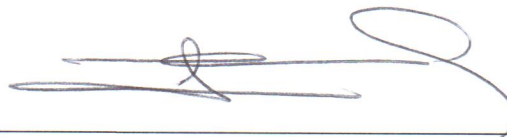
Ecatepec de Morelos, Edo. De Méx., a 8 de Mayo de 2017  
**ASUNTO: VOTO APROBATORIO DE ASESOR**

**LIA. ADRIANA MORALES LICONA**  
**JEFA DEL DEPARTAMENTO DE TITULACION DEL**  
**CENTRO UNIVERSITARIO U.A.E.M ECATEPEC**  
**P R E S E N T E**

Por éste conducto me permito informarle que el (la) pasante **C.C. JORGE LUIS CASTILLO LOPEZ Y VICTOR HUGO OMAÑA VERA** con el número de cuenta **1028663 y 1025486**, de la **INGENIERIA EN COMPUTACION**, ha concluido el desarrollo de su **TESIS**, con el título:

**“MULTIMETRO PARA DISPOSITIVOS MOVILES”**

Manifiesto que el borrador del trabajo escrito reúne las características necesarias para ser revisado por la Comisión especial nombrada para tal efecto.



---

**DR. EN C. RODOLFO ZOLA GARCIA LOZANO**  
**NO. DE CÉDULA PROFESIONAL: 2706467**

**PATRIA, CIENCIA Y TRABAJO**

**“2017, Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos”**



[www.uaemex.mx](http://www.uaemex.mx)





**UAEM** | Universidad Autónoma  
del Estado de México

Ecatepec de Morelos, Edo. De Méx., 26 de Mayo de 2017  
**ASUNTO: VOTO APROBATORIO DE REVISORES**

**LIA. ADRIANA MORALES LICONA**  
**JEFA DEL DEPARTAMENTO DE TITULACION DEL**  
**CENTRO UNIVERSITARIO UAEM ECATEPEC**  
**P R E S E N T E**

Nos es grato comunicarle que el trabajo de **TESIS** titulado:

**"MULTIMETRO PARA DISPOSITIVOS MOVILES"**

Que para obtener el título de: **INGENIERIA EN COMPUTACION**

Presentan los pasantes: **JORGE LUIS CASTILLO LOPEZ Y VICTOR HUGO OMAÑA VERA**  
Con números de cuenta: **1028663 y 1025486**

Cumplen con los requisitos teóricos-metodológicos suficientes para ser aprobada,  
pudiendo continuar con los trámites correspondientes para su impresión.

**REVISORES**

M. en I.S.C. **CUAUHTEMOC HIDALGO CORTES**  
CÉDULA PROFESIONAL: 4797107

M. en I.S.C. **ALEJANDRA MORALES RAMIREZ**  
CÉDULA PROFESIONAL: 5782891

**PATRIA, CIENCIA Y TRABAJO**

**"2017, Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"**



[www.uaemex.mx](http://www.uaemex.mx)



Universidad Autónoma del Estado de México  
Centro universitario UAEM Ecatepec

Ecatepec de Morelos, Estado de México a 2 de Junio de 2017  
**ASUNTO: IMPRESIÓN DE TRABAJO ESCRITO**

**C.C. JORGE LUIS CASTILLO LOPEZ Y VICTOR HUGO OMAÑA VERA**  
**PASANTE DE LA INGENIERIA EN COMPUTACION**  
**DEL CENTRO UNIVERSITARIO UAEM ECATEPEC**  
**P R E S E N T E**

Por este medio le comunico a usted que al haber cubierto los trámites correspondientes al desarrollo del trabajo escrito bajo la modalidad **TESIS** con el fin de obtener el Título Profesional, se le aprueba la **IMPRESIÓN DE SU TRABAJO** con el título:

**“MULTIMETRO PARA DISPOSITIVOS MOVILES”**

Con el objetivo de establecer la fecha de Evaluación Profesional, le recuerdo que la presentación final del trabajo escrito es de su completa responsabilidad.

Sin más por el momento, reciba un cordial saludo.

PATRIA, CIENCIA Y TRABAJO

*“2017, Año del Centenario de la Promulgación de la Constitución de los Estados Unidos Mexicanos”*

**LIA. ADRIANA MORALES LICONA**  
**JEFA DEL DEPARTAMENTO DE TITULACION**  
**DEL CENTRO UNIVERSITARIO UAEM ECATEPEC**



CENTRO UNIVERSITARIO U.A.E.M.  
ECATEPEC  
TITULACION



## **Agradecimientos Jorge Castillo.**

### ***A mi asesor Dr. Zolá.***

*Por preocuparse en formar a profesionales llenos de valores, motivados a siempre buscar más y tener vocación como usted. Gracias por su apoyo para la culminación de nuestros estudios profesionales y para la elaboración de esta tesis.*

### ***A mi madre Leticia y mi padre Luis.***

*Por ser el pilar central en todo lo que soy, por los ejemplos de perseverancia y constancia que los caracterizan y que me han infundado siempre, por sus consejos, sus valores y motivación que me han convertido en una persona de bien, pero principalmente, por su amor.*

### ***A mis familiares.***

*A mi hermana Alejandra por ser el ejemplo de una hermana mayor y de la cual aprendí aciertos de momentos difíciles, gracias por tu apoyo incondicional, te quiero mucho; a mi tío Gabriel, a mi tía Eva, a mi tía Liliana y a mi tío Arturo, por haberme acompañado y guiado, por que siempre han estado al pendiente de mi desarrollo profesional y personal, siempre preocupándose por mi.*

### ***A Ericka Liliana.***

*Por creer en mí, por que te convertiste en la inspiración y felicidad de mi vida, por que siempre me has apoyado en cada momento difícil, gracias por acompañarme y ser parte importante de cada objetivo y logro obtenido.*

### ***A mis Amigos.***

*Víctor, Carlos, Ricardo, Ulises, David y Astorga, creamos un grupo de amigos incondicionales, que siempre busca más y eso nos llevó a lograr cosas importantes en nuestra carrera, nos dimos cuenta que somos capaces de lograr cualquier objetivo.*

*A todos aquellos que participaron directa o indirectamente en la elaboración de esta tesis y fomentar en mi vocación.*

*¡Gracias a ustedes!*

*“La vocación debería ser aquello que nos llena, que da a cada célula de nuestro cuerpo una sensación insuperable, una actividad que mientras la realizamos nos haga sentir que no necesitamos de nada más.”*

*Arturo Maldonado M.*

## AGRADECIMIENTOS VICTOR OMAÑA

### **A mi padre.**

La principal persona que me motivó a llegar a este punto es mi papá, quien me ha enseñado que todo en la vida es posible si se trabaja con dedicación, dando nuestro mayor esfuerzo y haciendo lo que más nos gusta, es por eso que le agradezco que siempre me haya apoyado y aún lo haga.

### **A mi madre.**

Quiero agradecer a mi madre, quien siempre se ha preocupado por mi bienestar y mi futuro.

### **A mis hermanos.**

Con quienes comparto logros tanto personales como en familia, y es importante como estudiante saber que cuenta con su familia.

### **A mi asesor el Doctor Rodolfo.**

Quien se ha preocupado por el nivel de conocimiento que deja en cada estudiante, quien pensó en nosotros para poder desarrollar un excelente proyecto. Agradecerle por todo el conocimiento compartido a lo largo de 7 años y que nos ayudan a ser mejores personas.

### **A mis amigos (los OMEGAVOLTS).**

Quienes fueron y siguen siendo parte importante de quien soy ahora como persona y como estudiante, me alegra pertenecer a un grupo de personas tan capaces y tan dedicadas en lo que hacen. Gracias Jorge, Carlos, Ulises, Ricardo, David y Astorga por ser mis amigos incondicionales por más de 5 años.

## **Resumen**

En este documento se presenta el desarrollo de un Multímetro para dispositivos móviles con sistema operativo Android. El proyecto fue desarrollado haciendo uso de una aplicación programada en el IDE Android Studio y el diseño e implementación de un hardware controlado por dicha aplicación mediante una tarjeta de adquisición de datos de open-source ARDUINO. El proyecto está enfocado al área de ciencia y tecnología, donde se busca poner al alcance de la sociedad elementos tecnológicos innovadores para resolver problemáticas o hacer más eficiente el realizar actividades cotidianas.



## ÍNDICE

1. Planteamiento del problema. ....	1
1.1. Introducción.....	1
1.2. Objetivos.....	2
2. Marco teórico.....	3
2.1. Corriente eléctrica.....	3
2.2. Voltaje eléctrico o diferencia de potencial eléctrico.....	5
2.3. Resistencia. ....	7
2.4. Ley de ohm. ....	9
2.5. Temperatura.....	9
2.6. Circuitos de medición. ....	10
2.6.1.Medidores analógicos de corriente.....	11
2.6.2.Medidor analógico de voltaje.....	13
2.6.3.Convertidores de analógico / digital. ....	15
2.6.4.Voltímetro digital.....	16
2.7. Programación en Android.....	17
2.7.1.Aplicación 1. Hola mundo. ....	19
2.7.2.Aplicación 2. Login.....	27
2.7.3.Aplicación 3. Agregando funcionalidades a la aplicación Login.....	36
2.8. Comunicación bluetooth. ....	39
2.9. Arduino. ....	40
2.9.1.Convertidor analógico digital (CAD) Arduino. ....	42
2.9.2.Comunicación en Arduino. ....	42
2.9.2.1. Serial Arduino. ....	43
2.9.2.2. Bluetooth en Arduino.....	44
2.9.2.2.1. Módulo bluetooth HC-05.....	44
3. Metodología.....	46
3.1. Modelo iterativo.....	46
3.2. Análisis general y requerimientos del proyecto. ....	48
3.3. Diseño general del proyecto. ....	48
4. Desarrollo de la aplicación “multímetro digital”.....	49
4.1. Análisis y requerimientos de aplicación móvil-iteración 1.....	49
4.2. Diseño de aplicación movil-iteración 1. ....	50

4.2.1. Análisis y requerimientos “pantalla 1” - iteración 1 .....	52
4.2.2. Diseño “Pantalla 1”- iteración 1.....	52
4.2.3. Codificación o ensamble “pantalla 1”- iteración 1. ....	53
4.2.3.1. Encendido / apagado.....	56
4.2.3.2. Búsqueda de dispositivos.....	58
4.2.3.3. Conexión con un dispositivo.....	60
4.2.3.4. Transferencia de datos entre dispositivos. ....	63
4.2.4. Prueba / integración “pantalla 1” - iteración 1.....	65
4.2.5. Análisis y requerimientos “ViewPager para pantalla 2 y 3” - iteración 1.....	68
4.2.6. Diseño “ViewPager para pantalla 2 y 3”- iteración 1. ....	69
4.2.6.1. Viewpager.....	69
4.2.6.2. Fragment. ....	70
4.2.7. Codificación o ensamble “ViewPager para pantalla 2 y 3”- iteración 1.....	70
4.2.7.1. Análisis y requerimientos “pantalla 2 - Fragment 1” - iteración 1. ....	71
4.2.7.2. Diseño “pantalla 2 - Fragment 1”- iteración 1. ....	72
4.2.7.3. Codificación o ensamble “pantalla 2 - Fragment 1”- iteración 1.....	73
4.2.7.3.1. Fragment 1: Multimetrofragment. ....	75
4.2.7.4. Prueba / integración “pantalla 2 - Fragment 1”- iteración 1. ....	81
4.2.7.5. Análisis y requerimientos “pantalla 3 - Fragment 2” - iteración 1. ....	82
4.2.7.6. Diseño “pantalla 3 - Fragment 2”- iteración 1. ....	83
4.2.7.7. Codificación o ensamble “pantalla 3 - Fragment 2”- iteración 1.....	83
4.2.7.8. Prueba / integración “pantalla 3 - Fragment 2”- iteración 1. ....	85
4.2.8. Análisis y requerimientos “pantalla 4” - iteración 1. ....	85
4.2.9. Diseño “pantalla 4”- iteración 1.....	86
4.2.10. Codificación o ensamble “pantalla 4”- iteración 1.....	87
4.2.11. Prueba / integración “pantalla 4”- iteración 1.....	90
4.2.12. Análisis y requerimientos “pantalla 5” - iteración 1.....	91
4.2.13. Diseño “pantalla 5” - iteración 1. ....	92
4.2.14. Codificación o ensamble “pantalla 5” - iteración 1. ....	92
4.2.15. Prueba / integración “pantalla 5” - iteración 1.....	96
5. Desarrollo de “circuito medidor”.....	97
5.1. Análisis y requerimientos “circuito medidor” - iteración 1.....	97
5.2. Diseño “circuito medidor”- iteración 1.....	97
5.3. Codificación o ensamble “pantalla 4” - iteración 1. ....	100

5.4. Prueba / integración “pantalla 4” - iteración 1.....	100
6. Desarrollo de programa en “tarjeta Arduino”.....	100
6.1. Análisis y requerimientos de programa “tarjeta Arduino” - iteración 1. ....	101
6.2. Diseño de programa “tarjeta Arduino” - iteración 1.....	101
6.3. Codificación o ensamble de programa “tarjeta Arduino” - iteración 1.....	103
7. Modelo CANVAS .....	104
7.1. Segmento de mercado.....	104
7.2. Propuesta de valor.....	104
7.3. Canal de distribución y comunicación.....	105
7.3.1.Etapa de distribución.....	105
7.3.2.Etapa de evaluación. ....	105
7.3.3.Etapa de compra.....	105
7.3.4.Etapa de entrega.....	105
7.3.5.Etapa de posventa.....	106
7.4. Relación con el cliente.....	106
7.4.1.Estrategias de promoción. ....	106
7.4.1.1. La muestra. ....	106
7.4.2.Paquetes de precio global.....	106
7.5. Recursos clave. ....	106
7.5.1.Recursos físicos.....	106
7.5.2.Recursos intelectuales. ....	107
7.5.3.Recursos humanos.....	108
7.5.4.Recursos financieros. ....	109
7.6. Actividades clave.....	110
7.7. Aliados clave.....	113
7.7.1.Alianza estratégica.....	114
7.7.2.Relación proveedor comprador.....	114
7.8. Fuentes de Ingresos.....	114
7.8.1.Costo unitario.....	114
7.8.2.Precio de venta unitario.....	115
7.8.3.Flujo de ingresos. ....	115
7.9. Estructura de costos. ....	117
7.9.1.Análisis de la propuesta de valor. ....	117
7.9.2.Análisis de la fuente de ingresos. ....	117

7.9.3.Estructura de costos. ....	118
7.9.4.Costos fijos.....	118
7.9.5.Costos variables. ....	119
7.9.6.Infraestructura. ....	119
7.9.7.Financiamiento inicial.....	119
7.9.8.Punto de equilibrio.....	120
8. Conclusiones.....	121
9. Trabajos futuros .....	122

## ÍNDICE DE FIGURAS.

Figura 1. Gráfica de una señal de voltaje en corriente directa. ....	5
Figura 2. Gráfica de una señal de voltaje en corriente alterna. ....	5
Figura 3. Escalas de temperatura de grados Kelvin, Fahrenheit y Celsius.....	10
Figura 4. Movimiento D´Arsonval.....	11
Figura 5. Movimiento D´Arsonval con resistencia shun (divisor de corriente) .....	11
Figura 6. Imagen de Amperímetro de múltiple rango.....	13
Figura 7. Movimiento D´Arsonval conectado a resistencia multiplicadora en serie.....	13
Figura 8. Imagen de un voltímetro de varios rangos.....	14
Figura 9. Conversión de señal analógica a digital con 3 bits. ....	15
Figura 10. Diagrama de bloques voltímetro digital.....	16
Figura 11. Pantalla de configuración de un nuevo proyecto. ....	20
Figura 12. Pantalla de selección de SDK mínimo y otros dispositivos.....	21
Figura 13. Pantalla de porcentaje de uso de versiones Android.....	21
Figura 14. Pantalla para elección de tipo de actividad. ....	22
Figura 15. Pantalla de configuración de nombre de actividad y Layout .....	23
Figura 16. Pantalla con paleta de componentes Android Studio.....	24
Figura 17. Pantalla de elección de dispositivo físico o AVD.....	27
Figura 18. Screenshot de pantalla del dispositivo con la aplicación desplegada .....	27

Figura 19. Vista de la aplicación final del ejemplo .....	28
Figura 20. Ubicación carpeta drawable .....	30
Figura 21. Representación de FrameLayout mediante cuadro naranja. ....	31
Figura 22. Árbol de componentes de Layout.xml .....	31
Figura 23. LinearLayout vertical, LinearLayout horizontal .....	32
Figura 24. LinearLayout vertical con diferentes pesos .....	33
Figura 25. Layouts vertical y horizontal combinados .....	33
Figura 26. Esquema Jerárquico de elementos .....	34
Figura 27. Árbol mostrando Frames vacíos .....	35
Figura 28. Dos actividades en ejecución.....	39
Figura 29. Interfaz de la plataforma Arduino.....	41
Figura 30. Shields y Arduino .....	41
Figura 31. Monitor Serial Arduino.....	43
Figura 32. Selección de NL y CR .....	44
Figura 33. Modelo Iterativo .....	46
Figura 34. Diagrama de bloques del diseño general del proyecto.....	49
Figura 35. Diseño y navegación de pantallas de multímetro digital. ....	51
Figura 36. Diseño pantalla 1 .....	53
Figura 37. Petición de Contraseña para conexión bluetooth .....	61
Figura 38. Ciclo de vida de los dos tipos de servicio .....	62
Figura 39. Mensaje para encender bluetooth.....	65
Figura 40. Petición de encendido de bluetooth. ....	66
Figura 41. Encendido del bluetooth .....	66
Figura 42. Búsqueda de dispositivos visibles.....	67
Figura 43. Lista con dispositivos visibles .....	67
Figura 44. PIN de vinculación bluetooth. ....	68
Figura 45. Diseño ViewPager .....	69
Figura 46. Diseño pantalla 2 .....	72
Figura 47. Fragment 1 interfaz multímetro .....	76



Figura 48. Ejes estipulados por el S.O., Android.....	77
Figura 49 a y Figura 49 b. Eje origen perilla.....	78
Figura 50. Ejemplo de conjunto de pares ordenados para el cálculo del ángulo.....	79
Figura 51. Movimiento de la perilla.....	82
Figura 52. Diseño pantalla 2-Fragment 1.....	83
Figura 53. Pantalla 3, selección de tipo de medición.....	85
Figura 54. Diseño de Pantalla 4.....	86
Figura 55. Prueba pantalla 3.....	91
Figura 56. Diseño de Pantalla 5.....	92
Figura 57. Agregado de librería Android Chart.....	92
Figura 58. Prueba gráfica pantalla 5.....	96
Figura 59. Convertidor analógico / digital de 10 Bits.....	98
Figura 60. Divisor de voltaje.....	98
Figura 61. Circuito medidor de voltaje positivo y negativo (esquema eléctrico).....	99
Figura 62. Circuito medidor de voltaje positivo y negativo (esquema de componentes).....	99
Figura 63. Datasheet de sensor de temperatura LM35.....	100
Figura 64. Circuito multímetro digital (esquema de componentes).....	100
Figura 65. Diagrama de Flujo programa en Arduino.....	102

## ÍNDICE DE TABLAS.

Tabla 1. Tabla de recursos gráficos aplicación Login.....	29
Tabla 2. Tabla comandos AT.....	45
Tabla 3. Requerimientos generales del proyecto.....	48
Tabla 4. Requerimientos para aplicación móvil-Iteración 1.....	50
Tabla 5. Requerimientos para “pantalla 1”- iteración 1.....	52
Tabla 6. Requerimientos para aplicación móvil ViewPager para pantalla 2 y 3 iteración 1..	69
Tabla 7. Requerimientos “pantalla 2 – Fragment 1” - iteración 1.....	71
Tabla 8. Requerimientos “pantalla 3 - Fragment 2” - iteración 1.....	82

Tabla 9. Requerimientos para “pantalla 4” - iteración 1 .....	86
Tabla 10. Requerimientos “pantalla 5” - iteración 1 .....	91
Tabla 11. Requerimientos para desarrollo de “circuito medidor” - iteración 1.....	97
Tabla 12. Requerimientos para programa en “tarjeta Arduino” - iteración 1.....	101
Tabla 13. Recursos físicos.....	107
Tabla 14. "Flujo de efectivo mensual (conservador)" .....	115
Tabla 15 "Flujo efectivo mensual (optimista)" .....	116
Tabla 16. "Flujo de efectivo mensual (Pesimista)" .....	117
Tabla 17. Análisis de la fuente de ingresos .....	118
Tabla 18. Costos fijos.....	118
Tabla 19. Costos variables .....	119
Tabla 20. Infraestructura .....	119
Tabla 21. Financiamiento inicial.....	120
Tabla 22. Punto de equilibrio .....	120

## **ÍNDICE DE CÓDIGOS.**

Código 1. Código XML pantalla 1. ....	55
Código 2. Permisos para adaptador bluetooth.....	56
Código 3. Encendido y apagado de bluetooth. ....	57
Código 1. Método para encender adaptador.....	58
Código 2. Método para recibir respuesta de activación bluetooth .....	58
Código 3. Método para iniciar la búsqueda de dispositivos .....	59
Código 4. BroadcastReceiver espera para encontrar nuevos dispositivos .....	59
Código 5. Evento onClick para seleccionar dispositivo bluetooth a conectar .....	60
Código 6. Se inicia servicio en el Manifest .....	62
Código 7. Se crea interfaz para servicio .....	62
Código 8. Método para obtener el dispositivo elegido e iniciar ConnectThread .....	63
Código 9. La clase ConnectThread crea un socket para enviar y recibir datos .....	63

Código 10. El controlador de hilos obtiene el socket y datos recibidos .....	64
Código 11. Clase ConnectedThread para iniciar hilo de emisión de datos e hilo de recepción de datos.....	64
Código 12. Se declara soporte para fragment en XML .....	70
Código 13. Switch para selección de Fragment .....	71
Código 14. Código XML para pantalla de simulación de Multímetro digital.....	75
Código 15. Selección de texto para ser enviado por bluetooth utilizando el método setEsctibe () .....	81
Código 16. Código XML para creación de la pantalla de parámetros .....	84
Código 17. Intent para llamar otra actividad .....	85
Código 18. Código XML para para selección de intervalos .....	89
Código 19. Intent que lanza nueva actividad .....	90
Código 20. XML para gráfica de mediciones .....	93
Código 21. Configuración básica gráfica .....	94
Código 22. Obtención de valores recibidos por bluetooth .....	95
Código 23. Envío de valores a pantalla para graficarlos .....	95
Código 24. Código para iniciar y detener la gráfica .....	96
Código 25. Código Arduino para envió y recepción de datos a través de bluetooth .....	103

## **1. Planteamiento del problema.**

### **1.1 Introducción.**

En México el crecimiento de usuarios con celulares inteligentes para el 2015 se calcula será de 74.4 millones a comparación de 2013 con 69.6 millones. De este universo, el 30% de los usuarios son jóvenes de entre 15 y 24 años, los cuales utilizan sus dispositivos de manera frecuente y los consideran una herramienta indispensable. (INEGI, 2014). A pesar de la amplia difusión de esta tecnología, la mayoría de las personas no explotan al máximo los recursos y utilidades de sus dispositivos. Peor aún, en México el uso de celulares inteligentes dentro de las instituciones educativas prácticamente no ha sido explotado. Es por eso que se deben encontrar maneras de fomentar su uso en beneficio de la educación. Estas acciones están encaminadas, por un lado, a mejorar las posibilidades de aprendizaje y evaluación dentro de los sistemas de educación y de manera paralela, llaman la atención de los estudiantes por el simple hecho de utilizar la tecnología para realizar prácticas, investigaciones o tareas, elevando el aprovechamiento y la calidad de estudio.

En el área de ingeniería, el uso de instrumentos de mediciones eléctricas es común tanto en el ambiente laboral como en las instituciones de educación superior donde se imparten licenciaturas relacionadas con algún área de la electricidad, electrónica o computación. Un elemento vital para su desarrollo de actividades académicas o de diseño e implementación de proyectos eléctricos y electrónicos, uno de los elementos de vital importancia es el multímetro. Por ejemplo, en las prácticas de laboratorio el multímetro permite corroborar el funcionamiento de los circuitos eléctricos con base en el comportamiento de las variables de voltaje o corriente.

Analizando la cantidad de usuarios de teléfonos inteligentes existentes en la actualidad, así como la necesidad que tienen de explotar al máximo las capacidades de estos dispositivos, en el caso de los estudiantes de las carreras que incluyan unidades de aprendizaje afines a la electrónica y considerando el interés que genera el uso de este tipo de dispositivos se decide desarrollar un producto que resuelva estas necesidades y que permita aprovechar las características del sector de la sociedad interesado. De esta forma el presente proyecto

busca implementar un instrumento de mediciones eléctricas básicas a través de la utilización de los teléfonos inteligentes, por lo que los objetivos del proyecto son:

## **1.2 Objetivos**

### **Objetivo general**

Desarrollar el software y hardware necesario para implementar un multímetro en dispositivos móviles (teléfonos inteligentes y tablets) con Sistema Operativo Android.

### **Objetivos particulares:**

1. Desarrollar una aplicación que sirva de interfaz entre dispositivos móviles con sistema operativo Android y el usuario.
2. La navegación en la aplicación desarrollada debe ser intuitiva, que sea lo más parecida a la de multímetros comerciales, pero con la ventaja de poder almacenar, procesar y desplegar de información que ofrecen los dispositivos móviles actuales como teléfonos inteligentes y tablets.
3. Realizar la implementación de un prototipo capaz de realizar las diferentes mediciones y la transmisión de los datos a través de una tarjeta ARDUINO.
4. Implementar la aplicación en múltiples versiones del Sistema Operativo Android.
5. Realizar la comunicación entre la tarjeta ARDUINO y el dispositivo móvil mediante bluetooth.
6. Permitir realizar las siguientes mediciones:
  - Temperatura de 0°C a 100°C.
  - Resistencia de 0Ω a 1MΩ.
  - Voltaje DC de 40mV a 50V



## 2. Marco teórico.

### 2.1 Corriente eléctrica.

Con el objetivo de construir una definición propia y congruente del concepto de corriente eléctrica a continuación se listan cinco definiciones de diferentes autores:

Según Charles K. y Matthew N, definen *la corriente eléctrica como la tasa de cambio de la carga en el tiempo, y la unidad de medición son los amperes (A)*.

*Matemáticamente, la relación entre la corriente  $i$ , la carga  $q$  y el tiempo  $t$  es:* (Charles K., Matthew N, 2001).

$$i = \frac{dq}{dt} \quad (1)$$

Para Stanley Wolf y Richard Smith, *la corriente eléctrica se define como el número de cargas que se mueven más allá de un punto dado en un circuito en un segundo*.

$$i = \frac{q}{t} \quad (2)$$

*Siendo  $i$  la corriente y  $q$  la carga neta que pasa por el punto en  $t$  segundos.* (Stanley Wolf y Richard Smith, 1992).

Por otro lado, Paul Hewitt hace la siguiente comparación del flujo de corriente eléctrica respecto al flujo de corriente de agua:

*Así como una corriente de agua es el flujo de moléculas de  $H_2O$ , la corriente eléctrica no es más que el flujo de cargas eléctricas. Esto porque uno o más electrones tienen libertad de movimiento por toda la red de átomos.* (Paul Hewitt, 2004).

Joseph A. Ediminister expone: *Un concepto más intuitivo es considerar la corriente eléctrica como cargas en movimiento. Utilizando magnitudes variables con el tiempo se tiene:* (Joseph A. Ediminister, 1997).

$$i(t) = \frac{dq}{dt} \quad (C/s) \quad (3)$$

Por otro lado, Salvador Mosquera *define la intensidad de corriente como el número de cargas que en un segundo pasa por la sección de un conductor. Así, en t segundos pasan Q unidades de carga por un conductor, la intensidad de corriente tiene el valor:* (Salvador Mosquera, 1981).

$$I = \frac{Q}{t} \quad (4)$$

De acuerdo a lo anterior, una definición particular de la corriente eléctrica ( $I$ ) podría ser:

La corriente eléctrica es la cantidad de carga que atraviesa una sección transversal en un segundo. Matemáticamente el cambio de la carga eléctrica que atraviesa la sección transversal respecto al cambio en el tiempo se representa con el concepto de la derivada. Por esta razón la corriente eléctrica se expresaría como:

$$I(t) = \frac{dq}{dt} \quad (C/s) \quad (5)$$

En circuitos eléctricos, el movimiento de la carga es debido fundamentalmente al movimiento de electrones. El movimiento de los electrones se produce cuando existe una diferencia de potencial entre dos puntos de un material que permite el flujo de electrones.

Una corriente eléctrica será de 1 A, se presenta cuando un diferencial de potencial logra que  $6.242 \times 10^{18}$  electrones (1 Coulomb, C) se desplacen a través de una sección transversal en un segundo. En donde la carga del electrón es  $1.6 \times 10^{-19}$  C

En función del cambio de la corriente respecto al tiempo, en la práctica ésta se clasifica en Corriente alterna (CA) y Corriente directa (CD). Por convención el símbolo  $I$  (mayúscula) se utiliza para identificar la corriente directa mientras que  $i$  (minúscula) se utiliza para identificar la corriente alterna.

Los valores instantáneos de las señales eléctricas pueden graficarse cuando varían con el tiempo. Si el valor de una onda permanece constante con el tiempo, a la señal se le llama corriente directa (CD) [Figura 1].

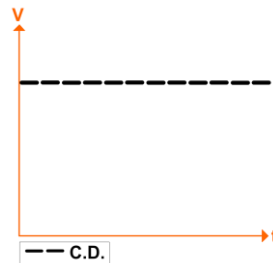


Figura 1. Gráfica de una señal de voltaje en corriente directa

Si una señal varía con el tiempo y tiene valores instantáneos positivos y negativos, a la onda se le llama onda de corriente alterna (CA) [Figura 2].

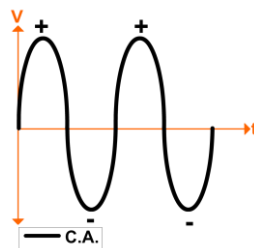


Figura 2. Gráfica de una señal de voltaje en corriente alterna.

## 2.2 Voltaje eléctrico o diferencia de potencial eléctrico.

De manera similar a la sección anterior, para tener una mejor comprensión del concepto de voltaje y realizar una definición propia se hizo una búsqueda y análisis de cinco definiciones de diferentes autores que se citan a continuación:

Charles K. y Matthew N. definen el voltaje de la siguiente manera: *La tensión  $V_{ab}$  entre dos puntos  $a$  y  $b$  en un circuito electrónico es la energía (o trabajo) necesaria para mover una unidad de carga desde  $a$  hasta  $b$*  (Charles K. y Matthew N, 2001).

Por otro lado, para Robert Boylestad el voltaje se define como: *una “presión” externa derivada de la energía que una masa tiene a causa de su posición: energía potencial.* (Boylestad, 2006).

Según Stanley Wolf y Richard Smith, *el concepto de voltaje se relaciona con los conceptos de energía potencial y de trabajo, e indica cuanta energía adquiere o pierde (por unidad de carga) una partícula al moverse dentro del campo eléctrico* (Stanley Wolf y Richard Smith, 1992).

Por otro lado, Salvador Mosquera define el voltaje de la siguiente manera:

*Se dice que entre dos puntos A y B hay una diferencia de potencial cuando para llevar la unidad de carga (C) del punto A al punto B hay que hacer un trabajo (J) de una unidad (v). De modo que:* (Salvador Mosquera, 1981).

$$\text{volt} = \frac{\text{joule}}{\text{coulomb}}, \text{ o sea } V = \frac{J}{C} \quad (6)$$

Otro concepto de *voltaje, tensión o diferencia de potencial es la presión que ejerce una fuente de suministro de energía eléctrica o fuerza electromotriz (FEM) sobre las cargas eléctricas o electrones en un circuito eléctrico cerrado, para que se establezca el flujo de una corriente eléctrica* (Staff Editorial, 2011).

De acuerdo a las citas anteriores para definir el voltaje, se hará una descripción de básica del concepto de energía potencial. Para comenzar la energía es la capacidad de realizar un trabajo. Por consiguiente si una masa ( $m$ ) se eleva a cierta altura ( $h$ ) sobre un plano de referencia, se tiene una energía potencial. Según la ecuación (7), a mayor altura el objeto ganará una mayor energía potencial.

$$\text{Energía potencial (EP)} = mgh \quad (7)$$

Siguiendo esto, el voltaje es una magnitud física que mide la diferencia de potencial entre dos puntos. Es decir, la energía potencial que tiene una carga en función de su posición dentro de un campo eléctrico.

Por ello, se puede decir que el voltaje es la energía necesaria (J) para producir el movimiento de electrones (C) desde un punto A hasta un punto B de un segmento cualquiera sobre de un material conectado a una fuente de energía eléctrica o fuerza electromotriz (fem).

$$V = \frac{J}{C} \quad (8)$$

### 2.3 Resistencia

Siguiendo la misma metodología de las secciones anteriores, a continuación, se citan cuatro diferentes definiciones del concepto de resistencia, que permitirán tener una mayor comprensión y a partir de ello crear una definición propia.

De acuerdo con los escritos de la Nueva enciclopedia autodidactica tomo VII: *La Resistencia eléctrica representa la oposición que ejerce la red de átomos de un conductor a la circulación de los electrones que pasan por ella. Esta resistencia aumenta con la longitud del conductor y disminuye al incrementarse su sección. También está íntimamente relacionada con la naturaleza del conductor (resistividad)* (Nueva enciclopedia autodidactica tomo VII, 2002).

Por otro lado, Charles K. y Matthew N, 2001 definen la resistencia de forma siguiente: *La resistencia de cualquier material con un área de selección transversal uniforme A, depende de ésta misma y de su longitud l.*

$$R = \rho \frac{l}{A} \quad (9)$$

Donde  $\rho$  se conoce como la resistividad del material en ohm-metros. Ohm definió la constante de proporcionalidad para un resistor como la resistencia,  $R$

$$v = i R \quad (10)$$

La resistencia  $R$  indica la capacidad para resistir el flujo de la corriente eléctrica (Charles K. y Matthew N, 2001).



Para Staff Editorial la resistencia se define como: *toda oposición que encuentra la corriente a su paso por un circuito eléctrico cerrado, atenuando o frenando el libre flujo de circulación de las cargas eléctricas o electrones* (Staff Editorial, 2011).

Según William D. Stevenson, *la resistencia de los conductores es la causa de la pérdida de la energía en las líneas de transporte. La resistencia efectiva de un conductor es*

$$R = \frac{\text{pérdida de potencia del conductor}}{I^2} \Omega \quad (11)$$

*Donde la potencia está en vatios e I es la corriente eficaz del conductor, en amperios* (William D. Stevenson, 1991).

Tomando en cuenta las definiciones anteriores el concepto de resistencia se define como la oposición que presenta un material al movimiento de los electrones en un punto cualquiera. La resistencia depende principalmente del tipo de material, largo, grosor y temperatura. De esta forma, entre más largo sea un cable de un material conductor se necesita mayor energía para mover los electrones de un extremo a otro, por lo tanto, entre más largo más resistencia se tiene. Por el contrario, si es más grueso la resistencia es menor.

Cuando el movimiento de electrones provoca colisiones en un material, se convierte la energía cinética de los electrones en calor. Las colisiones que sufren los electrones entorpecen/afectan el flujo de corriente. Por esta razón, si la temperatura aumenta la vibración de la red atómica aumentará, aumentando el número de colisiones de los electrones con la red por lo que la resistividad aumentará, aumentando la resistencia del conductor.

$$R_0 = \frac{\rho l}{A} \quad (12)$$

Dónde  $\rho = \text{resistividad del material}$

$l = \text{longitud}$

$A = \text{área desección transversal}$

Los materiales que tienen una resistividad tan grande como para evitar el paso de la corriente eléctrica se les considera materiales aislantes como lo son el plástico o la madera. Los materiales que presentan poca resistencia al paso de los electrones son considerados conductores, como lo son el cobre, el oro o incluso el agua.

## **2.4 Ley de ohm**

El físico y matemático Alemán George Simon Ohm, demostró que: *La intensidad de la corriente eléctrica que pasa por un conductor en un circuito, es directamente proporcional a la diferencia de potencial aplicado a sus extremos e inversamente proporcional a la resistencia del conductor.*

De acuerdo con lo anterior es posible expresar la relación que existe entre la diferencia potencial eléctrico, corriente eléctrica y resistencia en un circuito eléctrico con la siguiente función matemática, llamada en su honor Ley de Ohm:

$$V = \frac{R}{I} \quad (13)$$

## **2.5 Temperatura.**

Debido a que en el proyecto se desarrollará un módulo para medición de la temperatura, de manera similar a los capítulos anteriores, se hizo una investigación previa para entender mejor el concepto de temperatura.

En el campo de la física, la temperatura es relacionada con el movimiento de átomos y moléculas. Un mayor movimiento de las moléculas significa que la red tiene una mayor temperatura y viceversa. La temperatura afecta a todos los cuerpos y ambientes. Este parámetro cuenta con diferentes escalas de medida, usando los Grados Kelvin como escala absoluta. Mediante esta escala se representa el cero absoluto como el estado en reposo de los átomos y partículas, el cual se alcanza a 0 K.

Otras escalas existentes son los grados Celsius (o centígrados) y los grados Fahrenheit. En los grados Celsius toman como referencia los valores de fusión y ebullición del agua, considerando que 0°C como la temperatura de congelación (fusión) y 100°C como la

temperatura de ebullición del agua. De forma similar, Fahrenheit toma como una de sus referencias la temperatura de una mezcla de hielo, agua y cloruro de amonio. Este es un tipo de mezcla frigorífica, que estabiliza a una temperatura definida, la cual fue establecida como 0 °F. Al igual que en los grados Celsius se utilizaron como referencia los puntos de fusión y ebullición del agua a los cuales se les asignó los valores de 32° y 212° respectivamente [Figura 3].

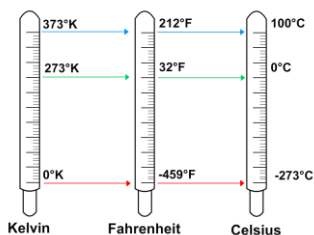


Figura 3. Escalas de temperatura de grados Kelvin, Fahrenheit y Celsius.

## 2.6 Circuitos de medición

En los siguientes subtemas se describen algunos tipos de circuitos de medición analógica y digital utilizados en amperímetros y voltímetros.

El movimiento electromecánico de D´Arsonval o mecanismo de imán permanente y bobina móvil (PMMC), es el mecanismo sensor más común en los amperímetros y multímetro, así como en algunos óhmetros.

Este dispositivo se compone por un imán permanente en forma de herradura con piezas polares magnéticas unidas a él. Entre las piezas polares se encuentra una bobina de alambre y un núcleo cilíndrico de hierro suave. El cilindro y la bobina pueden girar en el espacio entre el núcleo de hierro y las piezas polares, tal y como se muestra en la [Figura 4]. Al giro se oponen un par de resortes, que calibrados dan un ángulo que está en función de la corriente que circula por la bobina. De esta manera se convierte la corriente eléctrica aplicada a la bobina en un movimiento giratorio y se utiliza un puntero unido a la bobina que muestra la rotación en una escala conocida.

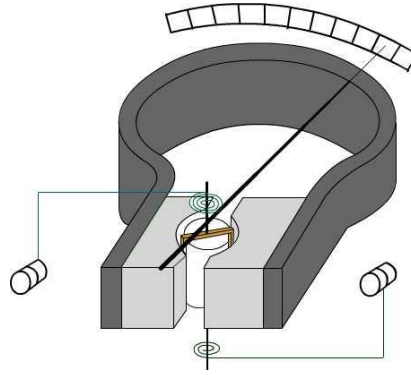


Figura 4. Movimiento D'Arsonval

Cabe mencionar que este mecanismo es ampliamente utilizado por su gran sensibilidad y extrema exactitud ya que puede medir corrientes pequeñas de hasta  $1.0 \times 10^{-13} \text{ A}$ .

### 2.6.1 Medidores analógicos de corriente.

Dado que el movimiento de D'Arsonval utiliza corriente para mover la bobina interna del dispositivo, es considerado como un Amperímetro para corrientes pequeñas. Para lograr medir corrientes mayores es necesario que se desvíe la mayor parte de la corriente, de forma que solo una pequeña porción de la corriente a medir pase por la bobina. Para esto se utiliza una resistencia conectada en paralelo (resistencia shunt) que formará un divisor de corriente [Figura 5].

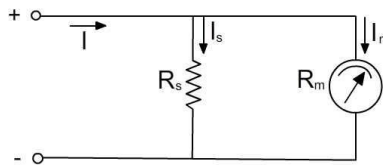


Figura 5. Movimiento D'Arsonval con resistencia shun (divisor de corriente).

La resistencia shunt ( $R_s$ ), se calcula haciendo un análisis de circuito. Por ejemplo, considere que se cuenta con un movimiento D'Arsonval que soporta una corriente ( $I_m$ ) de

1mA y con una resistencia interna ( $R_m$ ) de  $100\Omega$ . Calcular la resistencia shunt para implementar un amperímetro con un rango de medición de corriente ( $I_s$ ) de entre 0-100mA.

Tomando en cuenta que la conexión de las resistencias es en paralelo, el voltaje en la resistencia shunt y en el movimiento D'Arsonval debe ser el mismo, por lo tanto:

$$V_m = V_s \quad (14)$$

Siguiendo la ley de ohm, donde:

$$V = R * I \quad (15)$$

Se obtiene:

$$I_s * R_s = I_m * R_m \quad (16)$$

$$R_s = \frac{I_m R_m}{I_s} \quad (17)$$

Entonces la resistencia shunt indicada para implementar un amperímetro con un rango de medición de 0-100mA es:

$$R_s = \frac{I_m R_m}{I_s} = \frac{1mA * 100ohms}{99mA} = 1.01\Omega \quad (18)$$

Algunos amperímetros usan varias terminales externas para cambiar el rango. Si se emplea un interruptor [Figura 6], el interruptor debe estar siempre en contacto con las resistencias vecinas antes ponerse en contacto con la resistencia que va a ser utilizada, esto para que el circuito esté protegido y no interrumpa divisor de corriente.

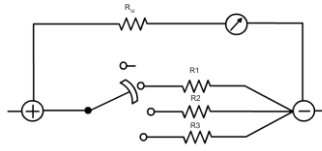


Figura 6. Imagen de Amperímetro de múltiple rango

### 2.6.2 Medidor analógico de voltaje.

El movimiento de D'Arsonval puede ser considerado un voltímetro. Para esto es necesario convertir la escala de Amperes a Volts. Para hacer la conversión a volts se debe tomar en cuenta la resistencia interna y la corriente máxima del movimiento [Figura 7]. Entonces, sí se tiene una corriente máxima ( $I_{max}$ ) de 1mA y una resistencia interna ( $R_m$ ) de  $20\Omega$ , por ley de ohm se obtiene el voltaje máximo que tendrá la escala:

$$V_{max} = I_{max} * R_m \quad (19)$$

El voltaje máximo en la escala del movimiento D'Arsonval queda definido por:

$$V_{max} = 1mA * 20\Omega = 20mV \quad (20)$$

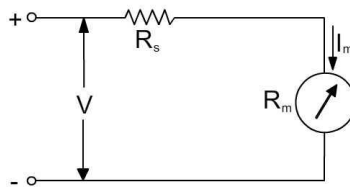


Figura 7. Movimiento D'Arsonval conectado a resistencia multiplicadora en serie.

Si se le conecta una resistencia en serie a la del movimiento, la resistencia adicional (multiplicador) limita la corriente que circula por el movimiento D'Arsonval, haciendo que

la escala de medición cambie. Para determinar el límite del voltaje se busca la resistencia adecuada mediante ley de ohm.

Para:

$$V = \frac{I_m(R_s + R_m)}{I_m} \quad (21)$$

Por lo tanto, para:

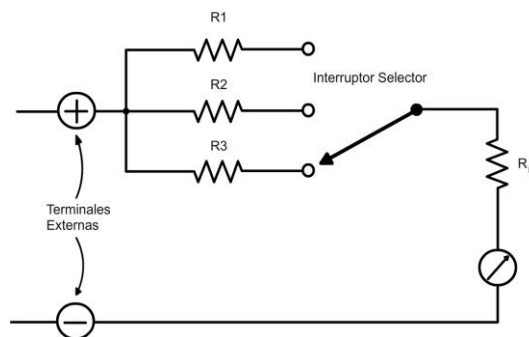
$$R_s = \frac{V - I_m R_m}{I_m} = \frac{V}{I_m} - R_m \quad (22)$$

De acuerdo a lo anterior la resistencia multiplicadora adecuada para extender el rango de voltaje de 0-10 Volts, con una resistencia interna de  $100\Omega$  y una corriente de  $1\text{mA}$  se calcula de la siguiente manera:

$$R_s = \frac{10V}{1\text{mA}} - 100\Omega = 9,900\Omega \quad (23)$$

$$R_s = 9,900\Omega$$

Para construir un voltímetro de múltiple rango, se puede emplear un interruptor selector que conecte resistencias de diferentes magnitudes en serie [Figura 8].



**Figura 8. Imagen de un voltímetro de varios rangos.**

### 2.6.3 Convertidores de analógico / digital.

En el caso de los medidores digitales, utilizan un dispositivo que toma la señal analógica de entrada y lo convierte a bits, lo que ayuda a que las mediciones sean más precisas, este dispositivo es llamado convertidor analógico / digital (CAD).

Un CAD es un circuito integrado cuya salida es resultado en forma binaria de la conversión de la señal analógica de entrada. La conversión se realiza en dos fases: cuantificación y codificación.

En la cuantificación, a cada valor analógico obtenido en la señal de entrada se le un valor o estado, que depende del número de bits del CAD. En la cuantificación se codifica el valor de salida del CAD en una palabra digital [Figura 9].

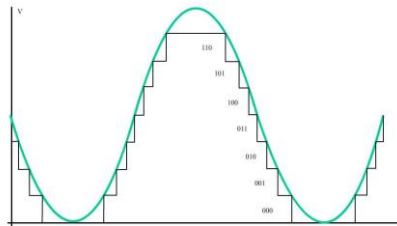


Figura 9. Conversión de señal analógica a digital con 3 bits.

Existen distintos métodos de conversión, entre los que se encuentran:

1. Aproximaciones sucesivas
2. Rampa de escalera
3. Doble rampa
4. Voltaje a frecuencia
5. Paralelo (o instantáneo)

De los anteriores el método de aproximaciones sucesivas es el más ampliamente utilizado. *Se utilizan ampliamente debido a su combinación de alta resolución y velocidad (es decir, pueden efectuar conversiones entre 1 y 50 $\mu$ s en lugar de los milisegundos necesarios por los convertidores de rampa de escalera, doble rampa y voltaje a frecuencias (Cooper,1985).*



## 2.6.4 Voltímetro digital

En los voltímetros digitales emplean convertidores A/D y contadores BCD para convertir las señales de voltaje de entrada a palabras digitales codificadas en binario que emplean para activar los dispositivos digitales de despliegue. Los voltímetros digitales más sencillos y menos costosos tienen la menor resolución (expresada como el número de dígitos en la pantalla) y emplean convertidores integradores de voltaje a frecuencia para efectuar sus conversiones digitales.

Como se mencionó en los temas anteriores, el CAD más utilizado es el de aproximación sucesiva. A continuación, se muestra un diagrama de bloques donde se muestra el proceso que sigue un voltímetro digital utilizando un CAD de aproximación sucesiva [Figura 10].

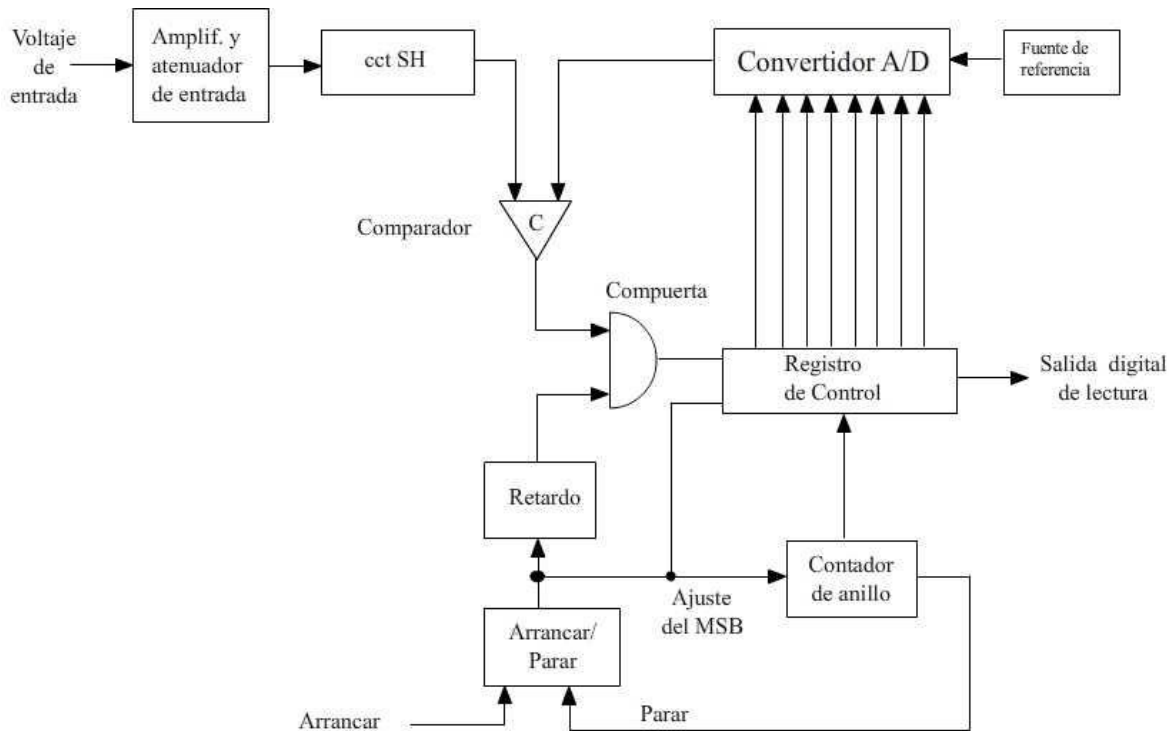


Figura 20. Diagrama de bloques voltímetro digital.

## 2.7 Programación en Android

### Herramientas

Para realizar los temas de programación en Android, es necesario el conocimiento de herramientas de desarrollo y conceptos fundamentales del entorno de aplicaciones para Android. Android Studio, SDK Android y JDK de java son esenciales en el desarrollo de aplicaciones móviles.

### Android Studio

Android Studio es el ambiente de desarrollo integrado (Integrated Development Environment, IDE) oficial para el desarrollo de aplicaciones para Android. Pertenece a la compañía Google quien ofrece este entorno de desarrollo completamente gratuito y puede ser descargado de la página oficial de Android Developer (Google, 2016).

Una vez en el sitio web seleccionar “Desarrollar” y posteriormente “Herramientas” para descargar.

Google ofrece un paquete de descarga que incluye el IDE (Integrated Development Environment) de desarrollo (Android Studio), el SDK de Android (Software Development Kit) que es un conjunto de herramientas de desarrollo que incluyen, por ejemplo, un emulador para probar las aplicaciones desarrolladas. Por último, incluye la última API de Android. Una API es un conjunto de instrucciones o procedimientos que nos ofrece una librería que el programador utiliza para desarrollar en este caso una APP, además se toma como referencia para conocer la versión de Android para la que se está desarrollando. Por ejemplo, la última API actual es la API 24 y el nombre comercial de esta versión es “Nougat”.

En un principio Android Studio fue enfocado solo para el desarrollo de aplicaciones para los dispositivos móviles que presenten la funcionalidad *touch*. Sin embargo, en estos tiempos existen dispositivos con sistema operativo Android como televisores, relojes inteligentes, tablets y una extensa variedad de smartphones.

## **Desarrollo de aplicaciones**

En este trabajo se presentan ejemplos de programas básicos en Android con el objetivo de dar a conocer los elementos que comprenden un proyecto en Android Studio, así como de facilitar la explicación en el desarrollo del multímetro digital.

### **Ejecutar aplicaciones: Emulador o dispositivo físico.**

Antes de desarrollar las primeras aplicaciones para Android es necesario conocer y elegir una de las dos maneras posibles para ejecutar y testear las aplicaciones desarrolladas en Android Studio. En las siguientes secciones se describen ambas formas de desarrollo.

- Android Virtual Device (AVD)

El SDK de Android incluye un emulador de dispositivos móviles - un dispositivo móvil virtual que se ejecuta en su computadora (Google, 2016).

El emulador requiere de un Dispositivo Virtual de Android (AVD) y mediante la herramienta *AVD Manager* se crean los dispositivos virtuales.

Es posible crear un AVD, basta con elegir un modelo de *Smartphone* de la línea Nexus. Esta selección se hace debido a que esta serie de dispositivos móviles es desarrollada por Google en colaboración de otras empresas que fabrican el hardware. Estos modelos ya tienen definidas las especificaciones dentro del AVD Manager. Sin embargo, la herramienta también permite crear dispositivos y estipular las características de manera específica como lo es el tamaño de pantalla, resolución, API, recursos de memoria virtual, recursos designados para su ejecución, etc.

Cabe resaltar que el desempeño del emulador depende de los recursos designados para su ejecución y estos son recursos que brinda directamente la computadora. También se debe saber que el emulador no presenta elementos que un dispositivo físico tiene como, por ejemplo, el acelerómetro o el Bluetooth.

- Dispositivo Físico

Antes de distribuir cualquier aplicación o en caso de no contar con una computadora que permita ejecutar el emulador con un alto rendimiento, es posible y se recomienda probar las aplicaciones directamente en un dispositivo físico.

Para esta modalidad se requiere de un Smartphone que tenga habilitadas las opciones de desarrollador. Estas opciones se habilitan en *Ajustes/Opciones de Desarrollador*. Para los dispositivos móviles con Android 4.2.2 o superior las opciones están ocultas y se habilitan desde *Ajustes/Acerca del teléfono* y tocar *Número de compilación* siete veces.

Las opciones de desarrollador permiten habilitar la depuración por USB, presentar los informes de fallos que llegara a presentar la aplicación, mostrar el rendimiento del dispositivo móvil en tiempo real, donde se puede observar la memoria, CPU, etc.

Para el desarrollo tanto de los ejemplos presentados en este documento como para el multímetro digital se optó probar las aplicaciones en el dispositivo físico con el fin de observar el comportamiento directamente en el smartphone.

En las siguientes secciones se analizarán los puntos básicos necesarios para realizar la programación de dispositivos móviles, esto con el objetivo de que el lector tenga un panorama más preciso de la plataforma.

### **2.7.1 Aplicación 1. Hola mundo**

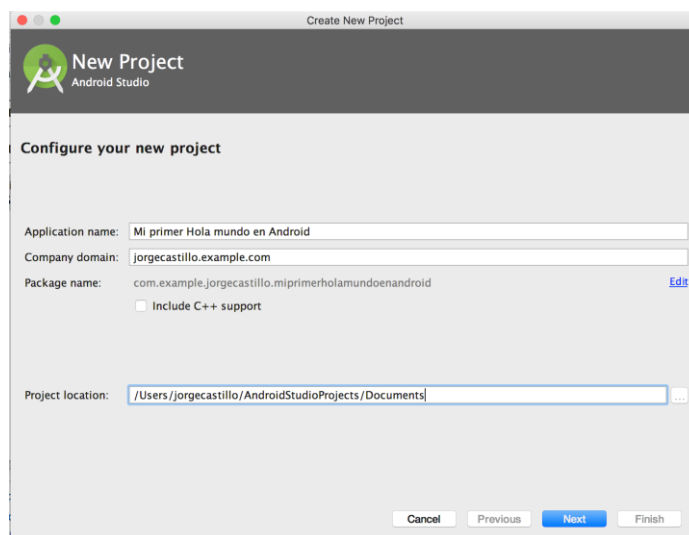
Para crear un nuevo proyecto en Android Studio, el primer paso es dar clic en *File>New Project*, inmediatamente se abre una ventana en la que se configura el nuevo proyecto [Figura 11]. Esta ventana permite nombrar el proyecto y modificar el dominio de la empresa. Este campo es de vital importancia, ya que cuando se planea compartir la aplicación en la Play Store, se debe agregar un calificativo o nombre a la aplicación con el fin de hacer única dicha APP, aun cuando existan diversas APP con el mismo nombre. Sin embargo, para este ejemplo, se utiliza el dominio de la empresa que tiene por defecto.

Otro campo que puede ser modificado, es el nombre del paquete, donde se muestra el nombre completo del proyecto (se siguen las mismas reglas que las de nombramiento de los paquetes en el lenguaje de programación Java) [Figura 11]. El nombre del paquete debe ser único en todos los paquetes instalados en el sistema Android (Google, 2016).

Por último, se puede establecer una ubicación, la cual debe ser utilizada para guardar los proyectos, esta ubicación debe ser diferente a la que Android Studio crea por defecto al momento de su instalación.

Los parámetros para este ejemplo son los siguientes:

- Nombre de la aplicación: Mi primer Hola mundo en Android
- Dominio de la empresa: Dejar el que tiene por default
- Nombre del Paquete: Dejar el que tiene por default
- Ubicación del Proyecto: Dejar el que tiene por default



**Figura 11. Pantalla de configuración de un nuevo proyecto.**

Al dar clic en *siguiente* se abre una nueva ventana [Figura 12], la cual permite seleccionar el tipo de dispositivo (Smartphone, tablet, tv, etc.) y el SDK mínimo en el que se va a desarrollar la aplicación. Este último es importante, debido a que, de acuerdo al SDK seleccionado, es el alcance que tiene la APP en el mercado. Dicho de otra manera, la

aplicación puede ser ejecutada en cualquier dispositivo que tenga una versión de SDK igual o superior a la elegida.

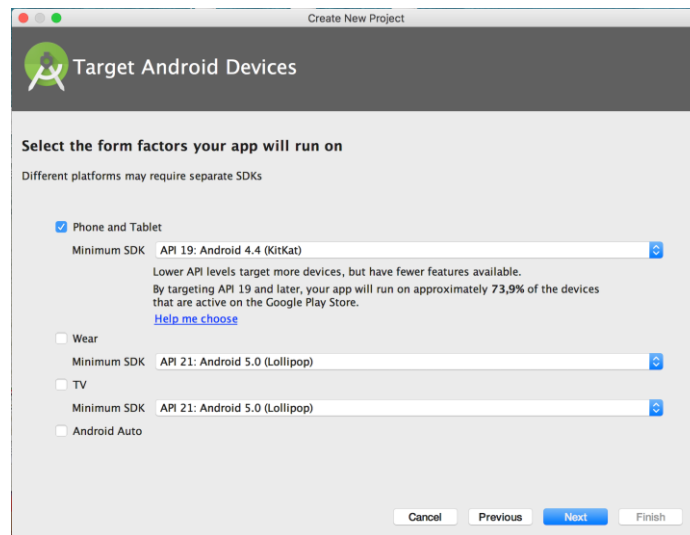


Figura 12. Pantalla de selección de SDK mínimo y otros dispositivos

Como ayuda para saber que SDK seleccionar, Android Studio ofrece un gráfico [Figura 13] que al presionar *Help me choose* ubicado bajo el campo *Mínimo SDK* y permite al desarrollador hacer una mejor elección de SDK a utilizar para su aplicación. El gráfico representa con porcentajes la distribución que tienen los SDK en el mercado.

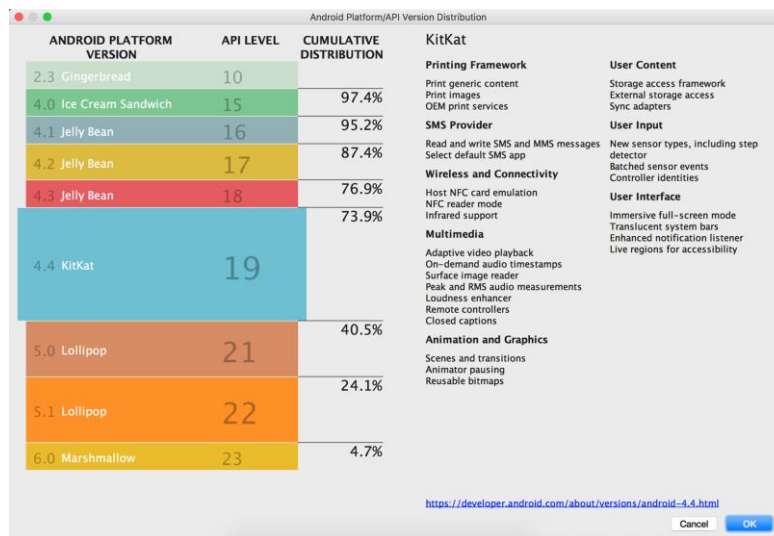


Figura 13. Pantalla de porcentaje de uso de versiones Android

Para cada proyecto de Android Studio presentados en este documento, se estipula como mínimo SDK la API 19: Android 4.4 (KitKat), debido al alcance que presenta desarrollar en esta API cubriendo con aproximadamente el 73.9% de dispositivos activos en *Play Store*.

El siguiente paso es elegir un tipo de actividad (*Activity*). Una actividad es algo que el usuario puede hacer. Casi todas las actividades interactúan con el usuario, por lo que el tipo de actividad se encarga de crear una ventana para el usuario (Google, 2016).

Una actividad se compone de una clase JAVA y un *Layout XML*. Dentro de la clase JAVA se programa el comportamiento que la clase debe de tener mediante estructuras de control, métodos, objetos, etc. Esto para darle sentido y un cometido a esta actividad.

El *Layaout XML*, como su nombre lo dice, contiene todos los elementos que conforman el diseño de la actividad, en otras palabras, es con lo que el usuario final va a interactuar. Incluye campos de texto, botones, imágenes, etc.

Con el objetivo de facilitar y optimizar el tiempo de desarrollo Android Studio ofrece diversos tipos de Actividades [Figura 14] que presentan diversas funcionalidades con estructuras definidas.

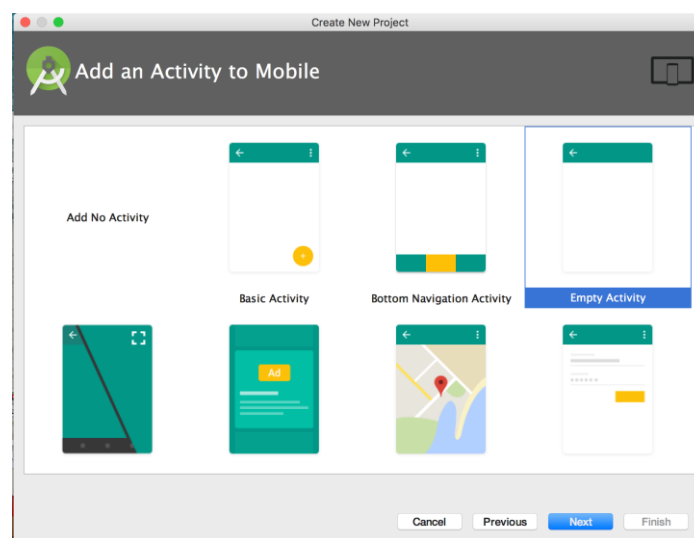


Figura 14. Pantalla para elección de tipo de actividad

Para cada proyecto que se presenta en este documento, se ha elegido la actividad *Empty Activity*, debido a que la actividad no presenta estructura alguna, está completamente vacía.

Al presionar *siguiente* aparece la ventana de configuración [Figura 15]. En esta ventana se debe nombrar la actividad, de manera automática se modifica el nombre del *Layout XML*. El nombre estipulado para la actividad es *ActividadHolaMundo* [Figura 15] se presiona *finalizar*.

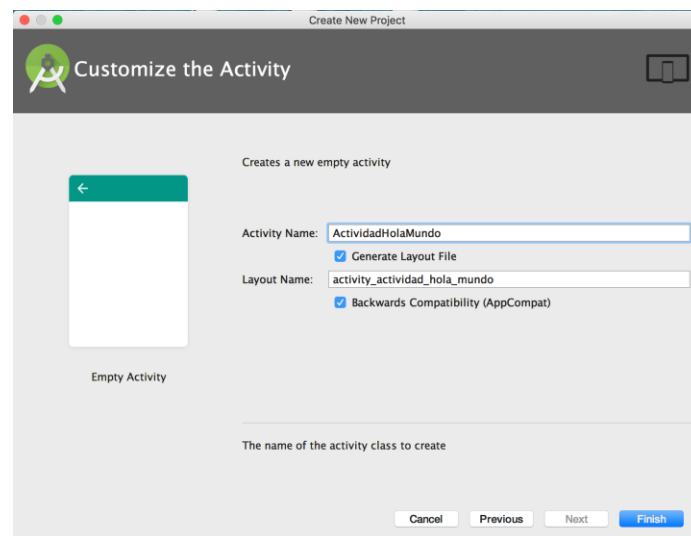


Figura 15. Pantalla de configuración de nombre de actividad y layout

Un proyecto en Android Studio se compone por una amplia variedad de elementos, clases, archivos XML, recursos de tipo *mipmap* utilizados para los iconos que lanzan la aplicación, recursos *drawable* para cualquier recurso gráfico, librerías externas que complementan la aplicación, *Manifest*, etc. Estos elementos se irán explicando acorde a la evolución de los proyectos.

## Diseño de Layout – Mi primer hola mundo en Android

El *Layout* de la actividad se encuentra en el directorio:

*Project>MiprimerHolaMundoenAndroid>app>src>main>res>layout.*



Hay dos maneras diferentes de visualizar el contenido del *Layout*. La primera es mediante el *Diseño* que permite pre visualizar en un dispositivo móvil virtual los elementos que se integran a la interfaz gráfica (no confundir con emulador). La otra manera es mediante *Texto*. Este permite ver el contenido del *Layout* mediante código XML. Nos apoyaremos de ambas perspectivas con la intención de explotar al máximo el entorno del IDE.

Al estar en *Diseño* se puede observar de lado izquierdo un panel llamado *Paleta* con cada elemento que se puede integrar a la interfaz. Se arrastra el elemento dentro del dispositivo en la previsualización, o arrastrar hasta el panel superior derecho llamado *Árbol de componentes*.

Dentro de la Paleta de componentes ubicar la categoría *Widgets*. Posteriormente arrastrar el elemento llamado *Plain TextView* dentro del dispositivo en la pre-visualización [Figura 16].

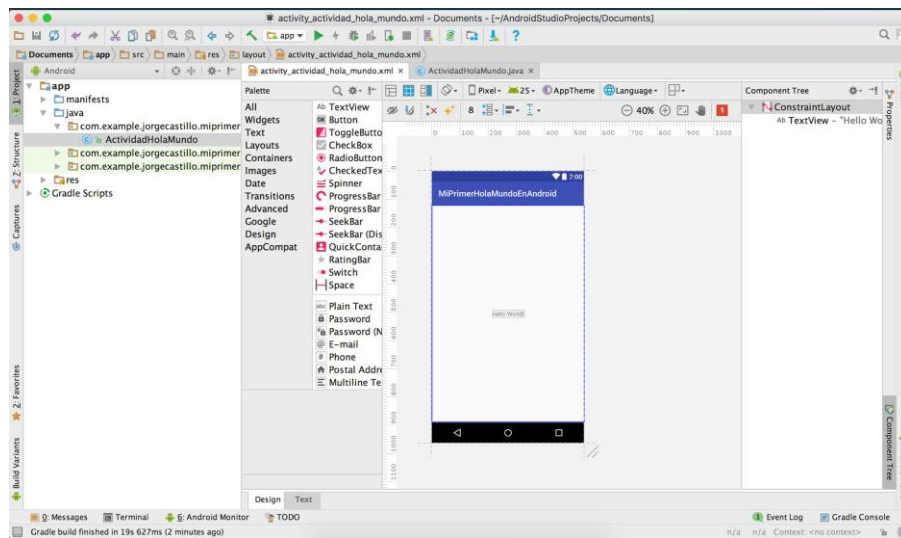


Figura 16. Pantalla con paleta de componentes Android Studio

Nótese que de manera automática en el panel *Árbol de componentes* se ha incluido el elemento que se acaba de integrar a la interfaz.

Al hacer doble clic sobre el elemento se abre una pequeña ventana que permite modificar el texto que el *Plain TextView* deberá mostrar, así como definir el *id* que es un identificador único para cada elemento.

Los parámetros para estos campos son:

**Text:** Hola mundo, esto es Android.

**Id:** txt\_EtiquetaSaludo

Debido a que en esta aplicación se modifican pocos elementos, no es necesario modificar nada en la clase JAVA de la actividad.

### **Probar aplicación – Hola mundo**

Por último, resta probar la aplicación directamente en el dispositivo físico.

Una vez conectado mediante USB el dispositivo móvil es necesario hacer clic en el botón Run que se encuentra en la barra de herramientas de Android Studio. Al ser la primera vez que se ejecuta una aplicación en un dispositivo móvil, este pedirá permisos de desarrollador.

Las opciones de desarrollador en cualquier dispositivo móvil se encuentran ocultas debido a que, para el usuario común, esta funcionalidad le será inservible. Sin embargo, para los desarrolladores de aplicaciones móviles esta opción es imprescindible. Las opciones de desarrollador son una serie de herramientas que ayudan a los desarrolladores a desplegar de manera rápida y eficaz las aplicaciones, a continuación, se describen las más importantes:

**Depuración por USB:** Permite a las aplicaciones que se están desplegando acceder a todos los recursos del sistema.

**Ubicaciones y SMS simulados:** Esta herramienta es necesaria si de ubicación se trata debido a que el emulador no cuenta con GPS, por lo tanto, las pruebas se pueden hacer únicamente en un dispositivo físico.

**Mostrar pulsaciones / Ubicación del puntero:** Muestra cada pulsación en la pantalla del dispositivo, representando cada pulsación con un círculo de color azul.

Informar un error: Devuelve la descripción del error sucedido durante la ejecución de la aplicación.

La manera de activar las opciones de desarrollador depende de la versión de S.O Android con la que cuente el dispositivo. En este documento se muestra la configuración para un equipo con la versión Android 6.0.1 Marshmallow.

1. Abrir Ajustes del dispositivo.
2. Dar clic en info. software
3. Tocar 7 veces la opción Número de Compilación.

Una vez hecho este proceso las opciones de desarrollador se encuentran dentro del menú de Ajustes. La manera de configurar el dispositivo para poder desplegar las aplicaciones, se habilitar la Depuración por USB y la opción *Informar un Error*. Lo que resta, es conectar el dispositivo móvil al ordenador y resetear nuestra aplicación.

Una vez en el IDE y al presionar el botón *Run* se abrirá una ventana [Figura 17] que permite seleccionar un dispositivo físico conectado a la computadora, o lanzar el emulador con un AVD elegido.

A continuación, se muestra un dispositivo conectado a la computadora, se debe seleccionar y por último presionar el botón *Ok* [Figura 17].

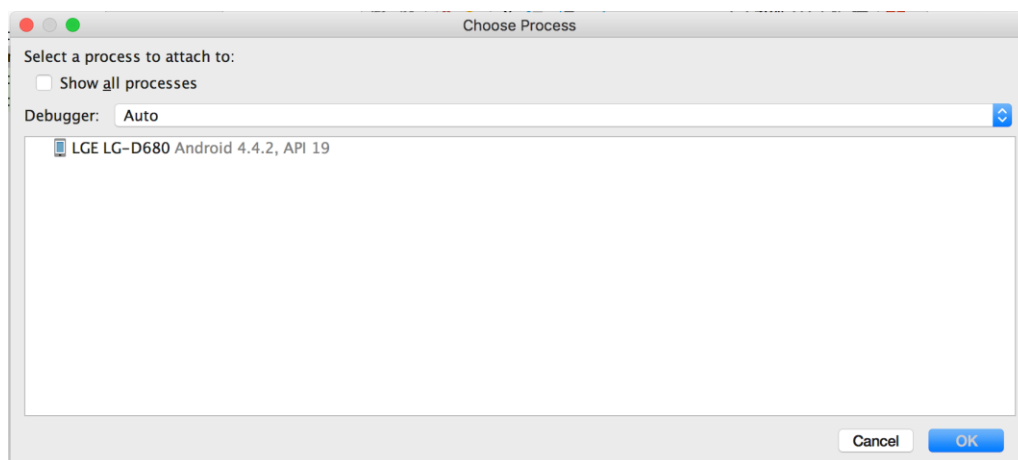


Figura 17. Pantalla de elección de dispositivo físico o AVD

La aplicación se descarga en el dispositivo móvil, es instalada automáticamente y al finalizar la instalación se ejecuta y muestra el resultado del proyecto. A continuación, se muestra un *Screenshot* (captura de pantalla) tomado directamente del dispositivo móvil, se observa la aplicación ya ejecutada [Figura 18].



Figura 18. Screenshot de pantalla del dispositivo con la aplicación desplegada.

### 2.7.2 Aplicación 2. Login.

El objetivo de esta aplicación es conocer, utilizar y diseñar interfaces vistosas e intuitivas que hagan uso de diversos componentes e incluso hacer uso de elementos como lo es una imagen y asignarle el funcionamiento de botón.

A continuación, se representa el resultado final de esta aplicación [Figura 19]. La cual ocupa imágenes para el fondo y para representar a los usuarios, un par de *EditText* para ingresar un usuario y una contraseña, además de una imagen configurada como botón.






**Figura 19. Vista de la aplicación final del ejemplo.**

Para este ejercicio es necesario crear un nuevo proyecto retomando los primeros pasos en la aplicación anterior Hola mundo. Los parámetros para esta nueva aplicación son:

- Nombre de la aplicación: Login
- Dominio de la empresa: Dejar el que tiene por default
- Nombre del Paquete: Dejar el que tiene por default
- Ubicación del Proyecto: Dejar el que tiene por default
- API Mínima: API 19, Android 4.4 (KitKat)
- Tipo de Actividad: Empty Activity
- Nombre de la Actividad: MainLogin

Los recursos gráficos necesarios para la elaboración de esta aplicación se presentan en la Tabla 1.

Imagen	Nombre	Medidas en pixeles (Ancho x Alto)
--------	--------	--------------------------------------

	Imgfondo.png	1364 X 1364
	Imgusuario.png	420 X 694
	Imgboton.png	200 X 200

**Tabla 1. Tabla de recursos gráficos aplicación Login**

Android Studio utiliza el directorio *Drawable* para almacenar cualquier recurso gráfico como las imágenes utilizadas dentro de la interfaz. La manera de dar soporte a los diversos tamaños de pantalla es mediante la clasificación de recursos dentro de las 6 variantes estandarizadas por Android Studio con base en a la densidad de pixeles (dpi) (Google, 2016):

- drawable\_ldpi (low) 120dpi,
- drawable\_mdpi (medium) 160dpi,
- drawable\_hdpi (high) 240dpi,
- drawable\_xhdpi (extra-high) 320dpi,
- drawable\_xxhdpi (extra-extra-high) 480dpi,
- drawable\_xxxhdpi (extra-extra-extra-high) 640dpi.

A pesar de esta clasificación existe un directorio genérico el cual busca la mejor representación del recurso dentro de la interfaz, dicho directorio es llamado *drawable*. Para la aplicación Login y la del Multímetro Digital se utilizó ésta opción.

Para almacenar los gráficos de la Tabla 1 en el directorio *Drawable* se arrastran los elementos directamente al directorio ubicado en la pestaña *Project/Login/app/src/main/res/drawable* [Figura 20].

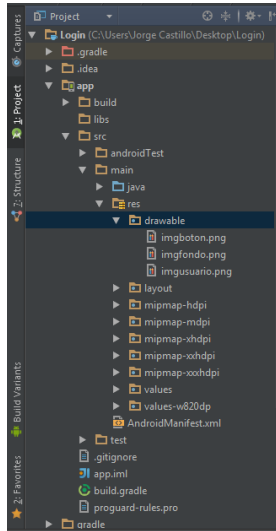


Figura 20. Ubicación carpeta drawable

El siguiente paso es abrir el *Layout* `activity_main_login.xml` para hacer la vista que el usuario tendrá.

Para esta aplicación y para el multímetro digital se utilizó como base de distribución de los elementos en la pantalla dos tipos de *layout*, *LinearLayout* y *FrameLayout*.

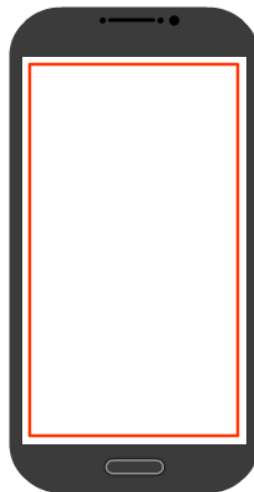


Figura 21. Representación de `FrameLayout` mediante cuadro naranja.

El *FrameLayout* [Figura 21] puede ser definido como un contenedor al que se le pueden agregar diversos elementos, como botones, *TextView*, *EditText*, etc. Por defecto son empalmados uno sobre otro, sin embargo, pueden ser distribuidos en toda la pantalla y

estipular medidas. Para este ejercicio es recomendable que el *FrameLayout* únicamente contenga *Layouts* y aprovechar las características que brinda, por ejemplo, a continuación, se muestra en el árbol de componentes de un *Layout.xml* [Figura 22].

Si se utiliza un *FrameLayout* padre, y a su vez este contiene un par de *FrameLayout* hijos, estos últimos se comportan como capas/layers con fondo transparente, que visualmente se verían empalmados uno sobre otro, ya que no cuentan con un margen el cual permite una distribución adecuada y evita que un layout cubra a otros.

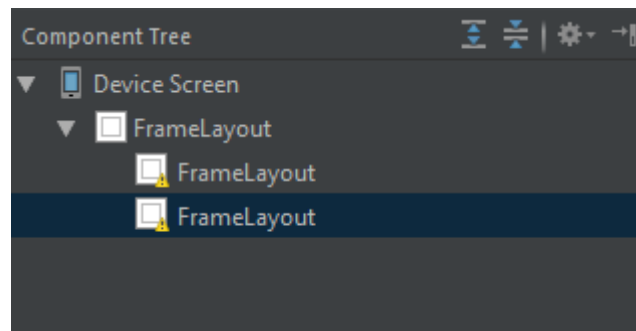


Figura 22. Árbol de componentes de *Layout.xml*

Los componentes *android: layout\_width* y *android: layout\_height* incluidos en un *FrameLayout* pueden modificarse tomando las siguientes propiedades:

- *Fill\_parent*: El control padre toma la dimensión de su layout contenido
- *match\_parent*: El control hijo toma la dimensión de su layout contenedor
- *wrap\_content*: El control hijo toma la dimensión de su contenido.

Dicho de otra manera, si el *FrameLayout* padre [Figura 22] tiene la propiedad *fill\_parent* tanto en *android: layout\_width* y *android: layout\_height* se adaptará al ancho y alto del componente padre que lo contiene, en este caso el Smartphone, y por lo tanto será del tamaño de la pantalla del dispositivo móvil (Gómez, 2014).

Por otro lado, el layout llamado *LinearLayout* a diferencia del *FrameLayout* coloca cada elemento contenido en el uno después de otro sin ser empalmados.



Las propiedades más importantes de este layout son *android:orientation* y *android:weight*, la primera de estas toma los valores *horizontal* o *vertical*. A continuación, se muestra la diferencia entre el tipo de orientación [Figura 23].

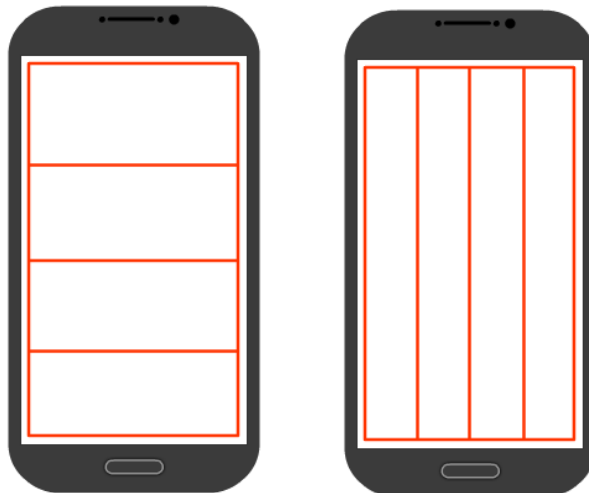


Figura 23. LinearLayout vertical, LinearLayout horizontal.

La propiedad *android:weight* permite designar las dimensiones o pesos que el elemento contenido va a tener [Figura 24].

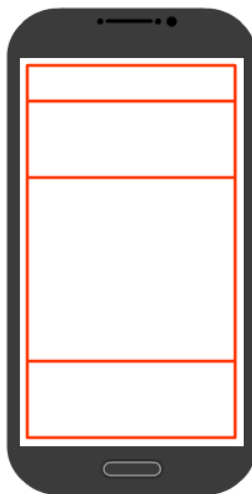
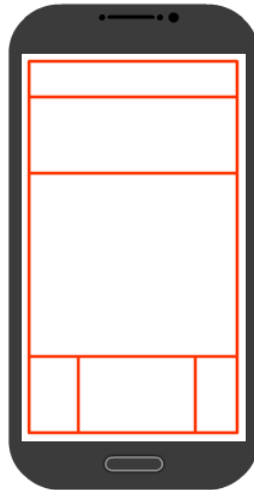


Figura 24. LinearLayout vertical con diferentes pesos

Con el objetivo de cumplir con un diseño estipulado y representarlo de la misma manera para cualquier tamaño de pantalla, es posible combinar diversos tipos de layouts. Véase la

[Figura 25] que tiene como layout principal un `FrameLayout`. Este contiene un `LinearLayout` con orientación vertical.

Cabe destacar que este layout contiene 4 elementos, los primeros 3 a pesar de verse vacíos están presentes, siendo `FrameLayouts` necesarios para la distribución deseada, el cuarto elemento es un `LinearLayout` con orientación horizontal que contiene 3 elementos más.



**Figura 25. Layouts vertical y horizontal combinados**

Para el diseño de la interfaz representada en la [Figura 19], se hace uso de ésta técnica (combinación de diversos tipos de *Layouts*). En la siguiente figura se esquematizan de manera jerárquica los elementos necesarios para el desarrollo de la interfaz [Figura 26], no sin antes mencionar que se utilizan *FrameLayouts* vacíos considerados como espacios de distribución. Estos elementos son importantes y hacen uso de la propiedad *android:weight*, en los *FrameLayouts* que no son considerados vacíos serán colocados componentes de tipo *ImageView* y *EditText*, además de los elementos de la Tabla 1.

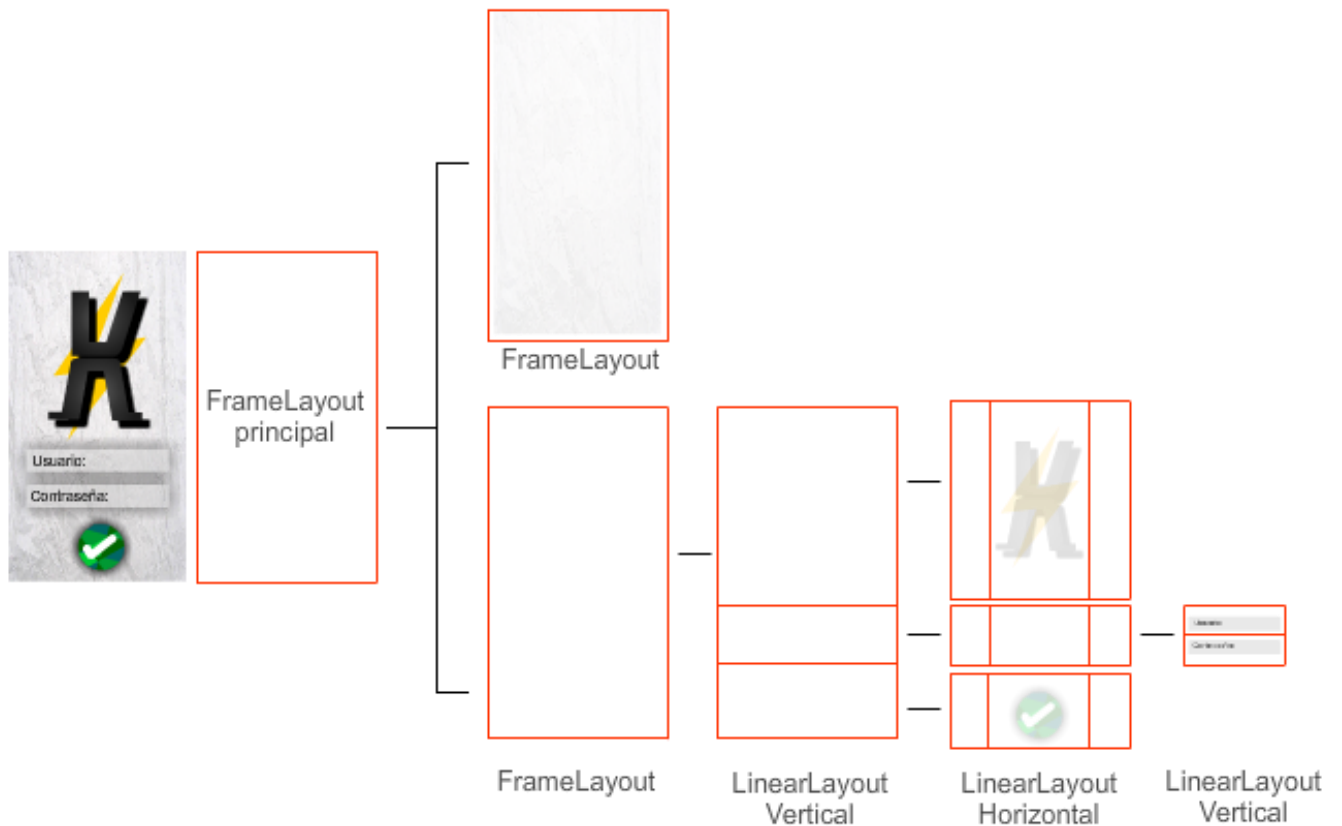


Figura 26. Esquema Jerárquico de elementos

En un principio, y por defecto el *Layout* que presenta Android Studio en el *Layout* del *activity\_main\_login.xml*, es de tipo *RelativeLayout* con un componente de tipo *TextView*, los cuales son modificados o eliminados.

Complementando la [Figura 22], A continuación, se visualizan los *FrameLayouts* utilizados como espacios vacíos y sus respectivos pesos, así como los elementos de la Tabla 1 [Figura 27].

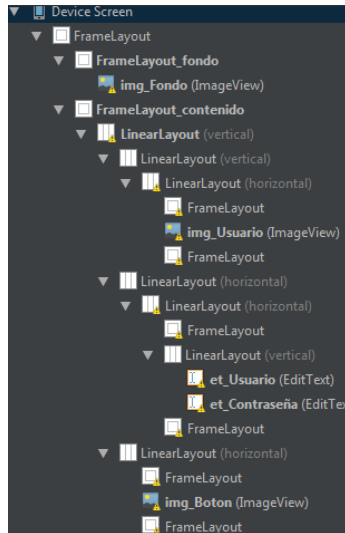


Figura 27. Árbol mostrando Frames vacíos

Con el fin de asignar los recursos del directorio *Drawable* a los elementos insertados en el archivo XML, el siguiente paso es modificar el archivo .java de la actividad llamada MainLogin.java donde se escribe el código necesario para hacer el enlace entre el directorio Drawable y el archivo XML. El archivo MainLogin.java queda de la siguiente manera:

```
package com.example.jorgecastillo.login;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ImageView;

public class MainLogin extends Activity {

    //Declarar variables
    ImageView img_Fondo,img_Usuario, img_Boton;

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

```

super.onCreate(savedInstanceState);
setContentViewById(R.layout.activity_main_login);

//Asignar recursos del activity_main_login.xml a las variables creadas
img_Fondo = (ImageView)findViewById(R.id.img_Fondo);
img_Usuario = (ImageView)findViewById(R.id.img_Usuario);
img_Boton = (ImageView)findViewById(R.id.img_Boton)

//Asignar imagen a los recursos
img_Fondo.setImageResource(R.drawable.imgfondo);
img_Usuario.setImageResource(R.drawable.imgusuario);
img_Boton.setImageResource(R.drawable.imgboton);

//Escalar las imágenes para su representación
img_Fondo.setScaleType(ImageView.ScaleType.CENTER_CROP);
img_Usuario.setScaleType(ImageView.ScaleType.CENTER_CROP);
img_Boton.setScaleType(ImageView.ScaleType.CENTER_INSIDE);

}

}

```

Por último, se ejecuta la aplicación siguiendo los mismos pasos realizados en la aplicación Hola mundo. Como resultado se espera obtener la vista de la actividad, sin embargo, no realiza ninguna acción o evento hasta este punto.

### **2.7.3 Aplicación 3. Agregando funciones a la aplicación Login.**

El siguiente paso de la aplicación Login, es designar y gestionar un evento tipo OnClickListener a una imagen para que funcione como botón, donde el resultado de ese evento sea lanzar una nueva actividad.

El primer paso es agregar una segunda actividad al proyecto, esto se logra dando clic sobre *File/New/Activity/Empty Activity*, inmediatamente se abre una ventana que pide el nombre de la actividad, para esta actividad se utilizan los siguientes parámetros:

- **Nombre de la Aactividad:** ActividadDos
- **Nombre del Layout:** activity\_actividad\_dos.xml

Esta actividad contendrá el mismo fondo utilizado en la primera actividad, además de tener un *TextView* con los siguientes parámetros:

- **Text:** Bienvenido
- **Id:** txt\_EtiquetaBienvenido

El siguiente paso es asignarle un evento a la imagen *Imgboton.png* para que actúe como botón. Existen diversos eventos que pueden ser utilizados para gestionar eventos touch, por ejemplo, *OnClickListener*, *OnTouchListener*, entre otros más. Sin embargo, para este ejercicio se usa *OnClickListener*, el cual se utiliza para ejecutar una parte de código cuando se toca un botón u otra parte de la interfaz de usuario (clic) (Google, 2016).

Al presionar la imagen *imgBoton* se compara el contenido de los *editText* contra datos estipulados en el código para saber si permite el *logea* del usuario, los datos son:

- Usuario: omegavolts
- Contraseña: multimetrodigital

En caso de no coincidir el contenido de los *EditText* aparecerá un mensaje en pantalla diciendo que se intente de nuevo y se limpiaran los campos. Por el contrario, si los datos coinciden se lanza la segunda actividad. La manera de lanzar una nueva actividad es mediante un *Intent*.

*El Intent es una descripción abstracta de una operación a realizar. Su uso más significativo está en la puesta en marcha de las actividades, puede ser pensado como el pegamento entre las actividades. Se trata básicamente de una estructura de datos pasiva que sostiene una descripción abstracta de una acción a realizar* (Google, 2016).

El código que se muestra a continuación debe ser ubicado después de escalar las imágenes en la primera actividad.

*//Estipular OnClickListener a la imagen*

```
img_Boton.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {

        //Comparar contenido de los EditText
        if((et_Usuario.getText().toString().equals("omegavolts"))&&
            et_Contraseña.getText().toString().equals("multimetro")){

            //Lanzar Actividad
            Intent intent = new Intent(getApplicationContext(), ActividadDos.class );
            startActivity(intent);

        }else{

            //Lanza mensaje

            Toast.makeText(getApplicationContext(),"Intente de nuevo",
            Toast.LENGTH_SHORT).show();

            et_Usuario.setText("");
            et_Contraseña.setText("");

        }

    }

});
```

A continuación, se muestra las capturas de pantalla desde el dispositivo móvil de las dos actividades en ejecución [Figura 28].

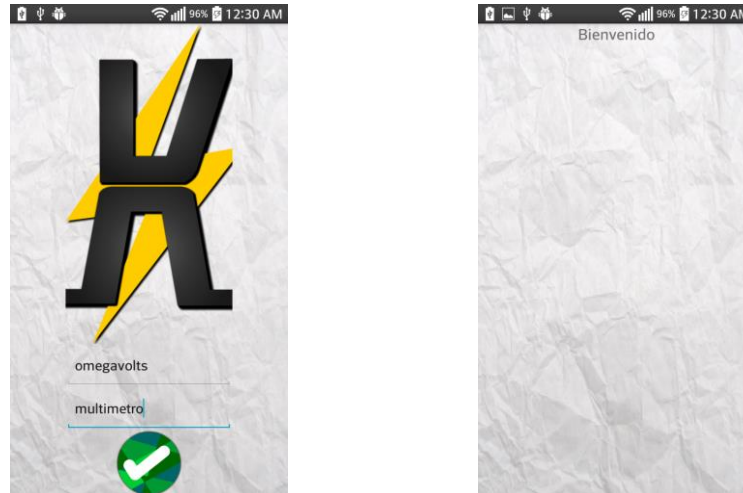


Figura 28. Dos actividades en ejecución.

## 2.8 Comunicación bluetooth

Un aspecto importante en el desarrollo de este proyecto es la comunicación que habrá entre el hardware (Tarjeta Arduino) y la aplicación Android, donde la transmisión de datos es realizada mediante tecnología bluetooth. Para poder entender mejor los alcances de esta tecnología, a continuación, se describen los aspectos más sobresalientes.

La tecnología Bluetooth fue desarrollada a partir de las investigaciones que realizó la compañía Ericsson en 1994. Su objetivo era crear una conexión inalámbrica que permitiera el envío y recepción de datos a través de ondas de radio, así como ser una tecnología de bajo consumo de energía. Sin embargo, en 1998 surgió Special Interest Group (Bluetooth SIG) un grupo conformado por más de 200 compañías quienes en ese momento se dan a la tarea de crear las especificaciones para la primera versión de Bluetooth.

La tecnología bluetooth trabaja a una frecuencia de 2.4 GHz y cuenta con parámetros de seguridad el cual un PIN es exigido para soportar la conexión, sin embargo, la distancia a la que dos dispositivos pueden estar conectados es relativamente corta, de hasta 100m de acuerdo a la potencia de transmisión utilizada. La velocidad de hasta 64 Mbps.



## 2.9 Arduino

Para la conversión de la información analógica obtenida del circuito a forma digital, se hace uso de la tarjeta de adquisición de datos Arduino. A continuación, se mencionan algunos aspectos importantes de dicha tarjeta.

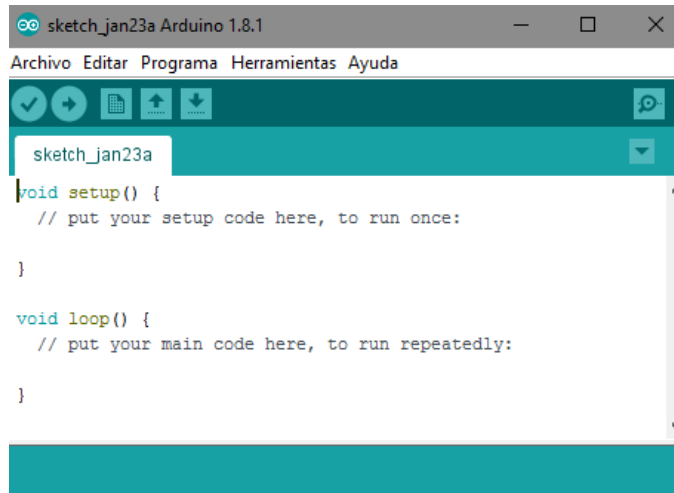
De acuerdo Garrido (s.f), *Arduino es una plataforma de prototipado electrónico de hardware libre y de placa única. Forma parte del concepto de hardware y software libre dado que es una contribución de toda la sociedad. Arduino es creado en Italia y consiste básicamente en una placa microcontrolador.*

Arduino cuenta con un lenguaje de programación basado en C/C++ en un entorno de desarrollo que soporta la entrada y salida de datos. *Fue creado en el año 2005 con el objetivo de servir como base para proyectos de bajo costo y es lo suficientemente simple para ser utilizado por los desarrolladores. Arduino es flexible y no requiere de un profundo conocimiento sobre el campo de la electrónica* (Garrido, s.f),

Como se mencionó antes, Arduino se compone de dos partes principales, el Hardware y el Software. El software se compone de una aplicación basada en las plataformas Processing y Wiring, que permiten que la programación sea un poco más sencilla, teniendo solo que definir las funciones a ejecutar. Además, es una herramienta multiplataforma utilizada en sistemas operativos como Windows, MAC OS y Linux.

La estructura de programación se divide en dos:

- **setup ():** Es el inicio del código, es llamado una sola vez al encender o reiniciar la tarjeta y dentro de éste se inicializan y declaran variables, pines como entradas o salidas y la velocidad de transmisión.
- **void loop ():** Es llamada después del setup (), y se ejecuta hasta que la tarjeta es desconectada de su fuente de energía, dentro del bloque void loop se ejecutan los comandos escritos por el usuario para realizar una tarea en específico [Figura 29].



**Figura 29. Interfaz de la plataforma Arduino**

El hardware es una placa conformada principalmente por un microcontrolador Atmega que permite el manejo de entradas y salidas de información a través de pines a los cuales se conectan otros circuitos y dispositivos discretos como leds o sensores. Los pines están ordenados de manera que otros componentes llamados shields como tarjetas Ethernet, pantallas LCD, joystick, bluetooth, entre otros puedan ser acoplados a la tarjeta sin necesidad de usar [Figura 30].



**Figura 30. Shields y Arduino**

**Crespo, C. (2015). Shields y Arduino**

Las tarjetas Arduino en su mayoría cuentan con un oscilador de 16 MHz, pines de entrada y salida de tipo digital y analógico, así como salidas de voltaje de 5 y 3.3 V y hasta 50mA por cada pin de salida. Existen diferentes tipos de tarjetas Arduino, las cuales se pueden utilizar

dependiendo del tipo de proyecto y cada una se basa en diferentes formas y configuraciones de hardware. El Arduino Uno es el más utilizado pero el Mega Arduino, por ejemplo, tiene más puertos de entrada, posibilitando la creación de dispositivos más grandes y más complejos. El Arduino Nano, como el nombre dice, es una versión reducida de un Arduino común, para la creación de objetos de electrónica más pequeña (Gironés, 2015).

### **2.9.1 Convertidor analógico digital (CAD) Arduino**

Como se menciona en los temas anteriores, Arduino es utilizada para hacer la conversión de los datos análogos leídos a forma digital, para ser enviados por bluetooth al dispositivo móvil con la aplicación Android. Para la conversión se usa el convertidos analógico a digital con el que cuenta Arduino. A continuación, se describe el comportamiento del CAD de Arduino.

Arduino cuenta con muchas características útiles, una de las más importantes es un convertidor analógico digital. Arduino es capaz de pasar una señal en forma de ondas que puede tomas múltiples valores con respecto al tiempo a bits (0's y 1's) siendo más fácil el manejo de la información.

Las tarjetas Arduino con chips Atmega cuentan con un CAD de 10 bits para datos de entrada y con un convertidor digital a analógico (CDA) de 8 bits para datos de salida, lo que significa que puede tomar valores entre 0 y 1023 para la entrada y valores entre 0 y 255 para la salida. Siendo estos voltajes de 0 a  $V_{ref}$ .

Para la lectura y escritura de datos analógicos se utilizan las sentencias `analogRead()` y `analogWrite()` correspondientemente, mientras que para datos digitales se utiliza `digitalRead()` y `digitalWrite()`.

### **2.9.2 Comunicación en Arduino.**

En el desarrollo de este proyecto Arduino debe mantener una comunicación constante con los dispositivos conectados a la placa. Ya sea mediante los pines de entradas y salidas analógicas y digitales, o por una comunicación serial. En este tipo de comunicación se ven

involucrados dispositivos ajenos a la tarjeta como un shield o la comunicación con el PC para la visualización y procesamiento de datos mediante el puerto usb.

### 2.9.2.1 Serial Arduino.

En este apartado se mencionan las propiedades con las que cuenta Arduino para la comunicación serial. Android cuenta con pines que permiten comunicarse con otro dispositivo, para poder comunicarse se necesita un pin que permita la transmisión (TX) y otro pin para la recepción de bytes (RX), así como una frecuencia en baudios que por default es de 9600 para la comunicación Arduino-PC.

Algunas de las tarjetas solo cuentan con un par de pines (TX y RX) para la conexión, por lo que necesitan de una librería especial para ocupar otros puertos de manera serial, mientras que tarjetas como la Arduino Mega 2560 cuentan con más de un par de pines designados para transmisión de datos (TX1 y RX1,..., TXn y RXn), solo necesitan ser conectados y declara el número de pines se estén manejando para esa función, para TX y RX se utilizara “begin.Serial()” y para TX1 y RX1 “begin.Serial1()”, esto de acuerdo al número de pines utilizados y con el que cuente cada una de las diferentes tarjetas.

La plataforma Arduino cuenta con un monitor serial que permite mostrar si se desea la información que fluye a través de estos pines [Figura 31].

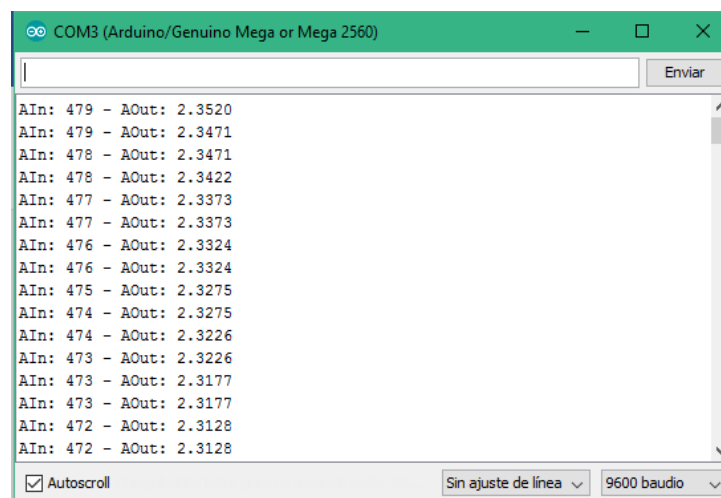


Figura 31. Monitor Serial Arduino.

### 2.9.2.2 Bluetooth en Arduino.

Arduino cuenta con shields especiales para la comunicación bluetooth con otros dispositivos, como los módulos HC-05 y HC-06. La conexión con éste dispositivo bluetooth se hace de manera serial con arduino, por lo que se necesita una frecuencia de envío de bytes la cual debe ser la misma en la arduino como del bluetooth. Esta frecuencia se establece en baudios. Arduino siempre manda al final de cada conjunto de datos enviados un salto de líneas (NL) y un retorno de carro (CR) las cuales pueden mostrarse u ocultarse en el monitor serial [Figura 32].

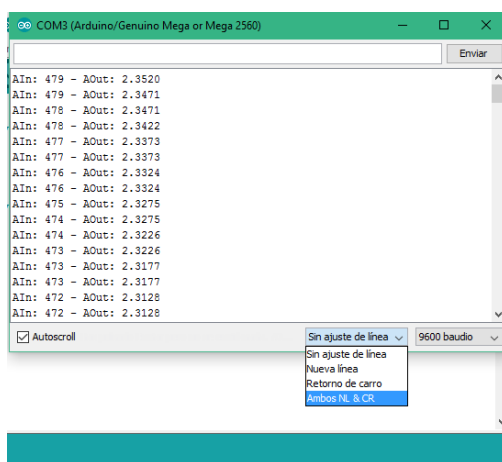


Figura 32. Selección de NL y CR.

#### 2.9.2.2.1. Módulo bluetooth HC-05

El módulo Bluetooth HC-05 es una herramienta Bluetooth SPP (Serial Port Profile) diseñada para la conexión en serie inalámbrica, permitiendo así conectar dos dispositivos Bluetooth habilitados

El módulo está configurado como maestro y esclavo, esto permite que además de recibir peticiones de conexión, éste pueda buscar y enviar peticiones de conexión a otros dispositivos. Las especificaciones que presenta el modulo Bluetooth HC-05 son:

- Sensibilidad -80 dBm
- Alcance +4dBm RF

- Potencia de operación 1.8 V, 3.3 a 5 V.
- Control PIO (Entradas y Salidas Programables).
- Interfaz UART (Universal Asynchronous Receiver-Transmitter) con velocidad de transmisión programable.

La manera de conectar este dispositivo con arduino es poner el pin TX del bluetooth en el pin RX de arduino designado para este dispositivo, y el pin RX del bluetooth en el pin TX, el pin de VCC del dispositivo a la salida de 5V de arduino y GND al pin GND de arduino.

Los comandos AT son instrucciones codificadas que logran establecer comunicación entre el hombre y un Terminal. Su nombre es la abreviación de attention comand. Los comandos AT sirven para configurar las diferentes funcionalidades del dispositivo bluetooth. Para el bluetooth HC-05 es necesario primero hacer que el dispositivo esté en modo AT, para ello se debe conectar el pin Key (Pin 34) a voltaje o presionar el botón que algunos modelos traen integrado.

Una vez que el dispositivo esté conectado en modo AT, desde el monitor serial de arduino se ingresa un comando para verificar una conexión exitosa (Escribir en monitor serial “AT”) y después escribir en el monitor serial los comandos según lo que se desee configurar. Estos son algunos comandos AT más utilizados en el bluetooth HC-05.

<b>Comando</b>	<b>Respuesta</b>
AT	OK
AT+PSWD=<4 dígitos>	Cambia la contraseña
AT+PIN<4 dígitos>	Cambia la clave de emparejamiento
AT+BAUD<numero>	Baudios a los que se trabajará
AT+ROLE = 1	Módulo como maestro
AT+ROLE = 0	Módulo como esclavo
AT+VERSION?	Muestra versión firmware
AT+UART = 115200,1,2	Modo puente

**Tabla 2. Tabla comandos AT**

### 3. Metodología

Existen diversas metodologías o modelos de desarrollo, y cada uno cuenta con un enfoque y procesos diferentes. Para este proyecto se opta por el Modelo iterativo, ya que, por la naturaleza académica del proyecto, este se divide en dos bloques principales desde un inicio, en programación y desarrollo de circuitos electrónicos. El modelo iterativo permite separar el proyecto en diferentes actividades o bloques y dividir cada bloque en fases, las cuales muestran versiones previas a la final lo que permite hacer correcciones individuales y precisas unificando las ideas del cliente con la visión del programador.

#### 3.1 Modelo iterativo.

Para entender mejor el modelo iterativo, a continuación, se describen los procesos que integran dicho modelo y como estos procesos se acoplan a este proyecto. Cabe mencionar que el modelo iterativo es también conocido como evolutivo y se deriva del ciclo de vida en cascada, cada ciclo se integra a una iteración que a su vez es una versión del proyecto [Figura 33].

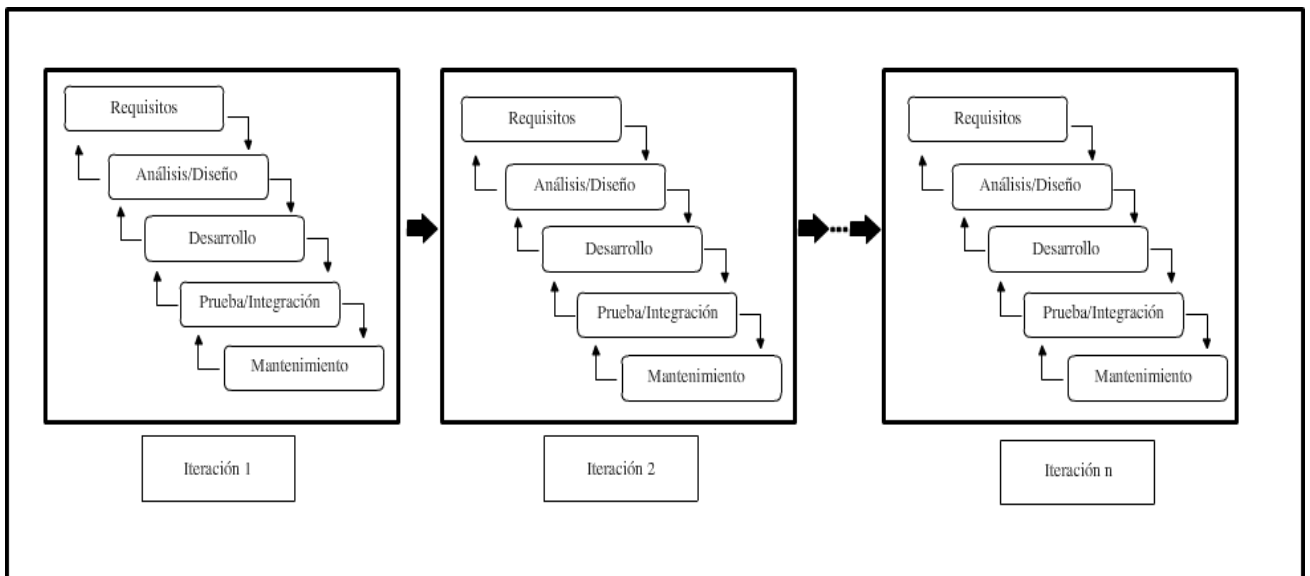


Figura 33. Modelo Iterativo

Al final de cada iteración se le entrega al cliente una versión mejorada o con mayores funcionalidades. El cliente es quien después de cada iteración, evalúa el producto y lo

corrige o propone mejoras. Estas iteraciones se repetirán hasta obtener un producto que satisfaga al cliente.

Se suele utilizar en proyectos en los que los requerimientos no están claros por parte del usuario. Cada iteración cuenta con seis fases de desarrollo en cascada, cabe mencionar que el ciclo de vida en cascada cuenta con variantes en las fases y pueden agregarse u omitir alguna de ellas, sin embargo, para este proyecto se tomaran las siguientes seis fases:

1. Análisis: El trabajo comienza estableciendo los requisitos de todos los elementos del sistema. Esto se hace mediante un diagrama de bloques que muestre cada sistema y los subsistemas que puedan depender de cada bloque.

2. Requisitos: En el proceso de recopilación de los requisitos el ingeniero de debe comprender cada parte del sistema, así como su funcionalidad, el rendimiento y las interfaces requeridas.

3. Diseño: El proceso de diseño traduce los requisitos en una representación con la calidad requerida antes de que comience la codificación o implementación.

4. Codificación o ensamble: el diseño debe traducirse en una forma legible para la maquina o el armado de un circuito. Si el diseño se realiza de una manera detallada, la codificación puede realizarse mecánicamente.

5. Prueba/Integración: Una vez que se ha generado el código comienza al realizar pruebas que aseguren que la codificación o ensamble arroja los resultados que realmente se requieren.

6. Mantenimiento: El software sufrirá cambios después de que se entrega al cliente. Los cambios ocurrirán debidos a que se haya encontrado errores o a que el sistema deba adaptarse a cambios del entorno externo (sistema operativo o dispositivos periféricos).

Una vez comprendido el funcionamiento del modelo iterativo, se comienza por identificar las partes por las que se conforma el proyecto.



### 3.2 Análisis general y requerimientos del proyecto.

A continuación, se describen los objetivos y requisitos generales para el desarrollo del multímetro digital para dispositivos móviles.

Descripción General del producto:	Multímetro Digital para Android que reciba información de valores con circuito medidor de voltaje, resistencia y temperatura, conectados mediante tecnología bluetooth.		
Objetivo:	Diseñar y desarrollar un sistema con interfaz de un multímetro, comunicado inalámbricamente con un circuito medidor de voltaje, resistencia y temperatura.		
Requerimientos Generales del Proyecto			
No.	Descripción	Funcional	No Funcional
1	Uso de un dispositivo (teléfonos inteligentes y tablets) con sistema operativo Android.	✓	
2	Captura y procesamiento de la información obtenida por el circuito medidor de voltaje, resistencia y temperatura a través de una tarjeta de adquisición de datos.	✓	
3	Uso de modulo bluetooth que permita el envío y recepción de información.	✓	

Tabla 3. Requerimientos generales del proyecto

### 3.3 Diseño general del proyecto

Siguiendo los requisitos generales, y de acuerdo a los conocimientos adquiridos en la carrera, en la siguiente figura [Figura 34] se presenta el diagrama a bloques del proyecto. Las variables a medir suministradas por el circuito medidor deben ser convertidas de datos analógicos a datos digitales. Para ello se decide utilizar la tarjeta de adquisición de datos Arduino Mega 2650, ya que además de contar con un convertidor analógico / digital (CAD), se puede integrar un módulo bluetooth que permita el envío y recepción de datos con la aplicación móvil. El módulo bluetooth que se utiliza es el HC-05 siendo un shield conocido para Arduino. La aplicación móvil se desarrolla en el IDE Android Studio, ya que es una plataforma pensada para el sistema Android, además de ser una plataforma Open Source.

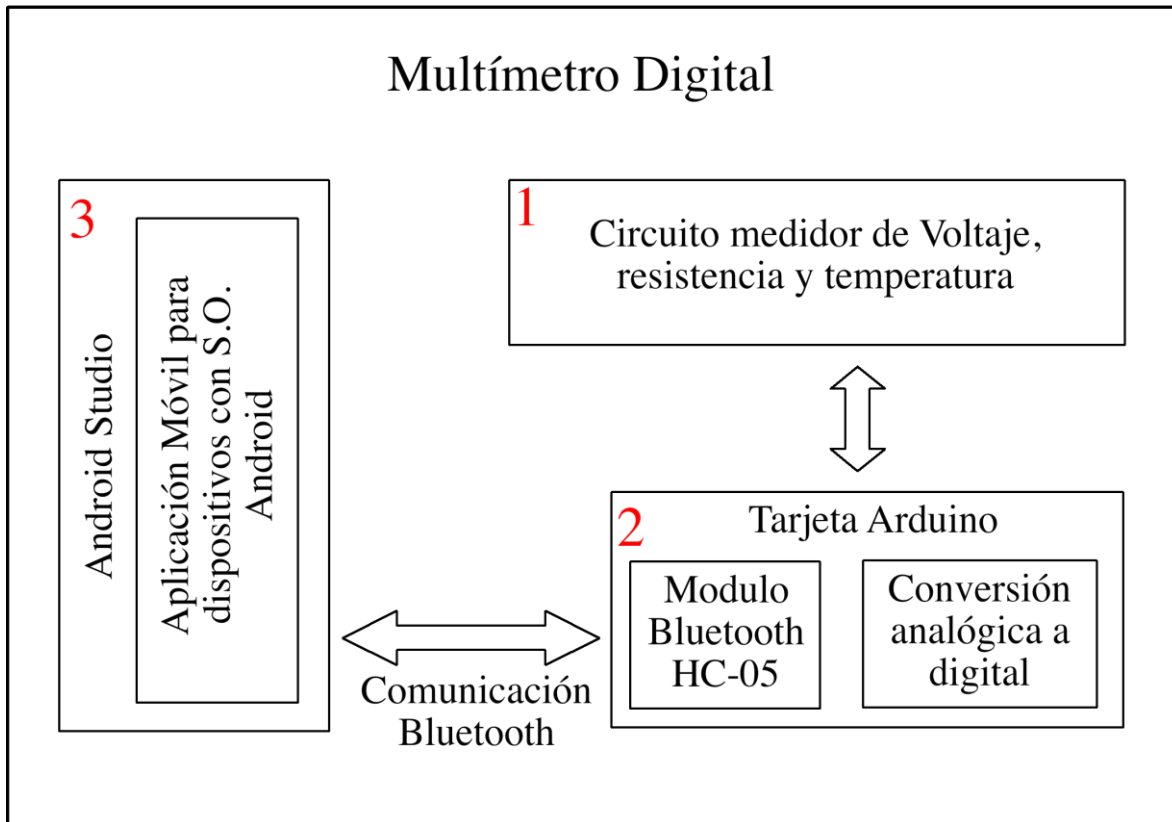


Figura 34. Diagrama de bloques del diseño general del proyecto

Hasta este punto se terminará la primera iteración ya que se analizará cada una de los bloques para hacer iteraciones individuales y tener un mejor control del proyecto.

#### 4. Desarrollo de la aplicación “multímetro digital”.

Siguiendo el diseño general del proyecto, se creará una iteración para el desarrollo del bloque 3 (Aplicación móvil), el cual se puede observar en la [Figura 34]. En este bloque se hacer una iteración para cada requisito, lo que ayuda a tener un mejor control del desarrollo del sistema.

##### 4.1 Análisis y requerimientos de aplicación móvil - iteración 1.

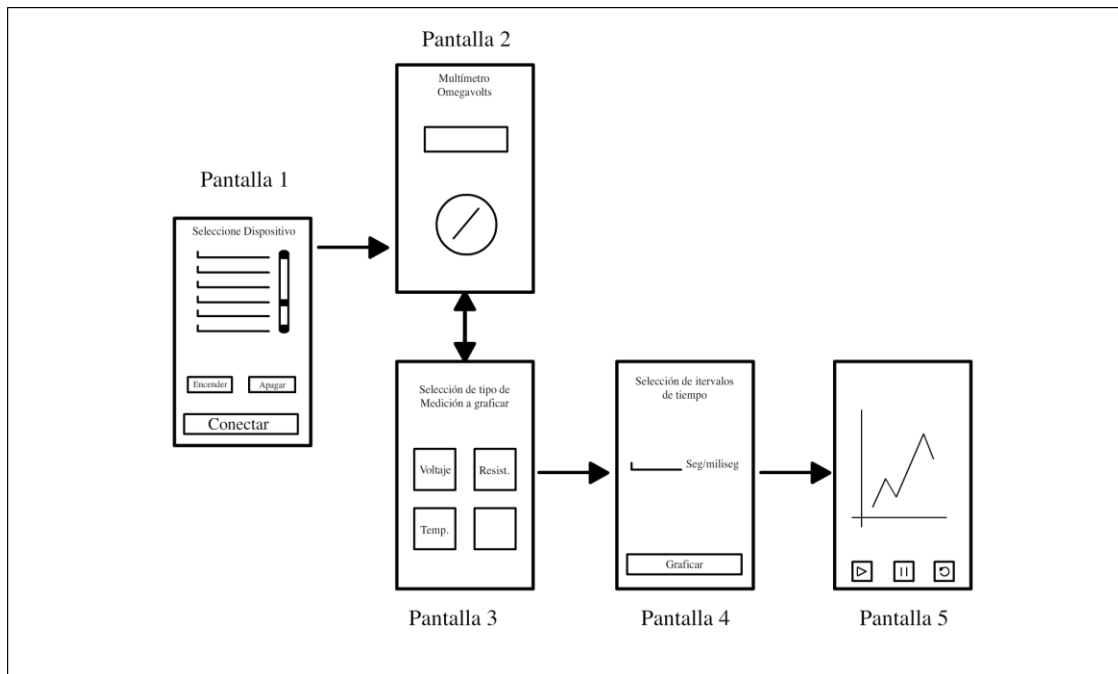
A continuación, se describen los requerimientos de la primera iteración del bloque 3 del diseño general (Aplicación móvil) [Figura 34].

Descripción de la Aplicación Móvil:	<b>Aplicación móvil para dispositivos con sistema operativo Android.</b>		
Objetivo:	Diseñar y desarrollar una aplicación para dispositivos móviles con sistema Android utilizando la plataforma <b>Android Studio</b> .		
Requerimientos para Aplicación móvil-Iteración 1			
No.	Descripción	Funcional	No Funcional
1	La aplicación debe permitir al usuario utilizar el bluetooth y seleccionar el dispositivo adecuado para la medición.	✓	
2	La aplicación debe contener una interfaz parecida a los multímetros reales con una perilla y un display.	✓	
3	La aplicación debe contener una interfaz donde el usuario pueda elegir el tipo de medición que puede graficar.	✓	
4	La aplicación debe permitir al usuario seleccionar el intervalo de tiempo en el que la gráfica mostrará la medición.	✓	
5	La gráfica debe permitir detener la recolección de datos, seguir con la medición después de la detención, reiniciarla y así como hacer zoom.	✓	
6	La aplicación debe ser capaz de enviar y recibir información por bluetooth.	✓	

Tabla 4. Requerimientos para aplicación móvil-Iteración 1.

## 4.2 Diseño de Aplicación móvil - iteración 1

En la siguiente figura [Figura 35], se observa el diseño y navegación de las pantallas para la aplicación “multímetro digital”.



**Figura 35. Diseño y navegación de pantallas de multímetro digital.**

En la figura anterior [Figura 35] se observa la navegabilidad que se tiene entre pantallas, donde la “pantalla 1” contiene un botón de encendido y uno de apagado para el bluetooth del dispositivo móvil. Una vez que el bluetooth del dispositivo esté encendido, se realiza la búsqueda de otros dispositivos visibles. Si hay respuesta de conexión del dispositivo bluetooth, se lanza la “pantalla 2”, la cual tendrá una interfaz similar a un multímetro real, con una perilla para seleccionar el tipo de medición (voltaje, resistencia, etc.) cuyos valores medidos se muestran en la simulación de display del multímetro. La “pantalla 3” será parte de la recolección de datos necesarios para graficar, en esta pantalla se elige el tipo de medición y complementando a la “pantalla 3”, en la “Pantalla 4” se le solicitará al usuario que proporcione el intervalo de tiempo entre cada medición. Completados los pasos anteriores, se manda a la “pantalla 5” donde se muestra la gráfica de la medición en tiempo real.

De acuerdo a la descripción anterior de los requisitos, es más sencillo manejar cada pantalla como una iteración individual. Por lo tanto, aquí termina la iteración 1 del bloque 3 de Aplicación móvil.

#### 4.2.1 Análisis y requerimientos “pantalla 1”-Iteración 1.

En este capítulo se describen los requerimientos para la pantalla 1 de acuerdo al diseño de la [Figura 35].

Descripción de la Aplicación Móvil:	<b>Aplicación móvil para dispositivos con sistema operativo Android pantalla 1.</b>		
Objetivo:	Diseñar y desarrollar la interfaz y funcionamiento de la pantalla 1.		
Requerimientos para “pantalla 1”- iteración 1			
No.	Descripción	Funcional	No Funcional
1	Manejo del adaptador del dispositivo bluetooth (Encendido y apagado).		✓
2	Botones de encendido y apagado de bluetooth.	✓	
3	Búsqueda de dispositivos mediante dirección MAC.		✓
4	Mostrar los dispositivos encontrados en una lista.	✓	
5	Evento para conexión bluetooth mediante socket con otros dispositivos al seleccionar dispositivo de la lista.		✓

Tabla 5. Requerimientos para “pantalla 1”- iteración 1.

#### 4.2.2 Diseño “pantalla 1”-Iteración 1.

A continuación, se muestra el diseño de la interfaz de “pantalla 1”. Donde la interfaz contiene dos botones para el encendido y apagado, así como la lista para mostrar los dispositivos encontrados [Figura 36].

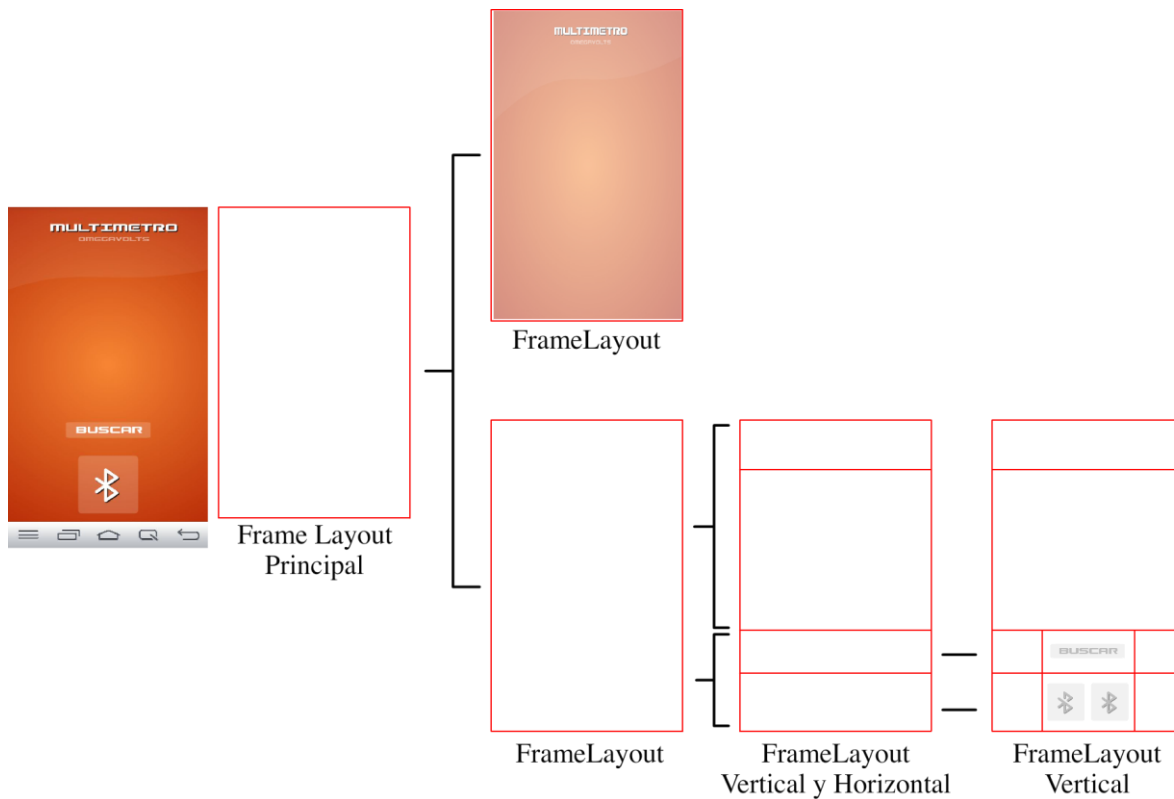


Figura 36. Diseño pantalla 1.

#### 4.2.3 Codificación o ensamble “pantalla 1”- iteración 1.

Para la parte visual se utilizan imágenes diseñadas por los desarrolladores del proyecto, con ayuda de la Suite Adobe Flash CS6.

Como se explicó en el apartado 2.7 el diseño de la aplicación se crea en un archivo XML donde se pueden ver los componentes o funcionalidades que sean agregados. El código XML del diseño de la “pantalla 1” queda de la siguiente manera:

```

1  <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
2  xmlns:tools="http://schemas.android.com/tools"
3  android:layout_width="match_parent"
4  android:layout_height="match_parent"
5  tools:context=".MainActivity">
6
7
8  <FrameLayout
9  android:layout_width="fill_parent"
10 android:layout_height="fill_parent">
11
12     <ImageView
13     android:id="@+id/imgV_FondoBlu"
14     android:layout_width="fill_parent"
15     android:layout_height="fill_parent" />
16 </FrameLayout>
17
18 <FrameLayout
19 android:layout_width="match_parent"
20 android:layout_height="match_parent">
21
22     <LinearLayout
23     android:layout_width="match_parent"
24     android:layout_height="match_parent"
25     android:orientation="vertical">
26
27         <LinearLayout
28         android:layout_width="match_parent"
29         android:layout_height="match_parent"
30         android:layout_weight="1.7"
31         android:orientation="vertical"></LinearLayout>
32
33         <LinearLayout
34         android:layout_width="match_parent"
35         android:layout_height="match_parent"
36         android:layout_weight="1"
37         android:orientation="vertical">
38
39             <ListView
40             android:id="@+id/lv_Dispositivos"
41             android:layout_width="match_parent"
42             android:layout_height="match_parent"
43             android:layout_above="@+id/btn_BuscarBT"
44             android:layout_alignParentLeft="true"
45             android:layout_alignParentStart="true"
46             android:layout_below="@+id/btn_EncenderBT"
47             android:choiceMode="singleChoice"
48             android:visibility="visible" />
49 </LinearLayout>
50

```

```

51     <LinearLayout
52     android:layout_width="match_parent"
53     android:layout_height="match_parent"
54     android:layout_weight="1.7"
55     android:orientation="horizontal">
56
57         <LinearLayout
58         android:layout_width="match_parent"
59         android:layout_height="match_parent"
60         android:layout_weight="2"
61         android:orientation="vertical"></LinearLayout>
62
63         <ImageView
64         android:id="@+id/imgV_Buscar"
65         android:layout_width="wrap_content"
66         android:layout_height="wrap_content"
67         android:layout_gravity="bottom"
68         android:layout_weight="1" />
69
70         <LinearLayout
71         android:layout_width="match_parent"
72         android:layout_height="match_parent"
73         android:layout_weight="2"
74         android:orientation="vertical"></LinearLayout>
75
76 </LinearLayout>
77
78 <LinearLayout
79 android:layout_width="match_parent"
80 android:layout_height="match_parent"
81 android:layout_weight="1.5"
82 android:orientation="horizontal">
83
84     <LinearLayout
85     android:layout_width="match_parent"
86     android:layout_height="match_parent"
87     android:layout_weight="2"
88     android:orientation="vertical"></LinearLayout>
89
90     <ImageView
91     android:id="@+id/imgV_Bluetooth"
92     android:layout_width="wrap_content"
93     android:layout_height="wrap_content"
94     android:layout_gravity="center"
95     android:layout_weight="1" />
96

```

```

97      <LinearLayout
98          android:layout_width="match_parent"
99          android:layout_height="match_parent"
100         android:layout_weight="2"
101         android:orientation="vertical"></LinearLayout>
102     </LinearLayout>
103
104     </LinearLayout>
105 </FrameLayout>
106
107 </FrameLayout>
108

```

**Código 26. Código XML pantalla 1.**

En el Código XML se observa que cada Layout cuenta con sus propios parámetros o características, como ancho y largo (`layout_width`, `layout height`), el peso que tendrá si es que está en el mismo nivel con otros layouts (`layout_weigth`) y la orientación, ya que de está depende la forma en la que se acomodan los elementos que estén dentro del Layout. Al igual que los layouts, las imágenes y botones cuentan con parámetros únicos y los que más se modifican son ancho y largo, estos parámetros se adaptan al Layout que los contiene, por lo que quedaría `layout_width="wrap_content"` y `layout_height="wrap_content"`, otro parámetro utilizado es la visibilidad (`android: visibility="visible"` o `android: visibility="invisible"`).

A continuación, se explicará el desarrollo de la conexión bluetooth. En la interfaz de “pantalla 1” se permite acceder a las propiedades del bluetooth del dispositivo, lo que hace posible poder crear la conexión entre la aplicación android y el bluetooth HC-05 que más adelante se explica su desarrollo.

Al crear aplicaciones es necesario que Android Studio le permita al desarrollador utilizar el adaptador bluetooth del dispositivo, así como sus propiedades para lograr el correcto funcionamiento en las aplicaciones.

Otro aspecto importante es el protocolo de conexión bluetooth. En este caso, Radio Frequency Communication (RFCOMM), el cual permite hasta 60 conexiones simultaneas. Android Studio utiliza el mismo protocolo de conexión, en donde la conexión básica del bluetooth consta de cuatro etapas necesarias:

- La configuración de Bluetooth.
- La búsqueda de dispositivos.



- La conexión con un dispositivo o emparejamiento.
- La transferencia de datos entre dispositivos.

#### 4.2.3.1. Encendido / Apagado.

Para la etapa de configuración del bluetooth del dispositivo android, es necesario que se den de alta los permisos para ocupar funcionalidades de este, es decir, declarar en el “manifest” las siguientes líneas:

```
5 <uses-permission android:name="android.permission.BLUETOOTH" />  
6 <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

##### Código 27. Permisos para adaptador bluetooth.

La línea 5, android.permission.BLUETOOTH es necesaria para poder realizar una conexión a otro dispositivo, aceptar una conexión de otro dispositivo y realizar la transferencia de datos. Por otro lado, android.permission.BLUETOOTH\_ADMIN se utiliza para realizar la búsqueda de dispositivos y manipular la configuración del bluetooth. Si se utiliza android.permission.BLUETOOTH\_ADMIN, android.permission.BLUETOOTH debe agregarse, para que la aplicación funcione adecuadamente.

Una vez que se agregan los permisos necesarios, se trabaja sobre el documento “.java”. Se inicia con la creación y configuración de botones que permitan encender, apagar y la búsqueda de dispositivos usando adaptador bluetooth, así como la inclusión de una lista para realizar la conexión con un dispositivo. Es necesario declarar una variable de tipo BluetoothAdapter, que permite el acceso a las configuraciones del adaptador bluetooth local con el que cuentan la mayoría de los dispositivos.

```

52     lv_Dispositivos = (ListView) findViewById(R.id.lv_Dispositivos);
53     arrayAdaptador = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1);
54     lv_Dispositivos.setAdapter(arrayAdaptador);
55     adaptadorBT = BluetoothAdapter.getDefaultAdapter();
56     al_NombreBT = new ArrayList<String>();
57
58
59     imgV_FondoBlue = (ImageView) findViewById(R.id.imgV_FondoBlu);
60     imgV_FondoBlue.setImageResource(R.drawable.bLuefondo);
61     imgV_FondoBlue.setScaleType(ImageView.ScaleType.CENTER_CROP);
62     imgV_Buscar = (ImageView) findViewById(R.id.imgV_Buscar);
63     imgV_Buscar.setImageResource(R.drawable.bluebuscar);
64     imgV_Buscar.setScaleType(ImageView.ScaleType.CENTER_INSIDE);
65     imgV_Buscar.setVisibility(View.GONE);
66     imgV_Buscar.setOnTouchListener((view, motionEvent) -> {
67         if (motionEvent.getAction() == MotionEvent.ACTION_DOWN) {
68             imgV_Buscar.setImageResource(R.drawable.bluebuscarpulsado);
69         } else if (motionEvent.getAction() == MotionEvent.ACTION_UP) {
70             imgV_Buscar.setImageResource(R.drawable.bluebuscar);
71             Buscar(view);
72         }
73     });
74     return true;
75
76
77
78
79
80     imgV_Bluetooth = (ImageView) findViewById(R.id.imgV_Bluetooth);
81     imgV_Bluetooth.setImageResource(R.drawable.bLueapagado);
82     imgV_Bluetooth.setScaleType(ImageView.ScaleType.CENTER_INSIDE);
83     imgV_Bluetooth.setOnTouchListener((view, motionEvent) -> {
84         if (motionEvent.getAction() == MotionEvent.ACTION_DOWN) {
85             imgV_Bluetooth.setImageResource(R.drawable.bLuepulsado);
86         }
87         else if (motionEvent.getAction() == MotionEvent.ACTION_UP) {
88             imgV_Bluetooth.setImageResource(R.drawable.bLueapagado);
89             if (!adaptadorBT.isEnabled()) {
90                 Encender(view);
91                 estado = false;
92             } else {
93                 Apagar(view);
94                 estado = true;
95             }
96         }
97     });
98     return true;
99
100
101
102

```

**Código 28. Encendido y apagado de bluetooth.**

Para este proyecto se toman tres imágenes y se convierten en botones, una para el encendido del bluetooth, uno para el apagado y uno para la búsqueda de dispositivos.

Al iniciar la aplicación se debe verificar si el bluetooth se encuentra encendido, de no ser así debe solicitar que se encienda, de lo contrario comienza la búsqueda de dispositivos para su conexión y uso de las siguientes interfaces dentro de la aplicación.

Existen métodos estáticos de la clase “BluetoothAdapter” que permiten el manejo del encendido, apagado del de dispositivos como lo son:

- getDefaultAdapter()
- enable() y disable()
- startDiscovery() y cancelDiscovery()

El método “getDefaultAdapter()” que se muestra en la línea 55 del código anterior, devuelve el estado de soporte del bluetooth, si el método devuelve un valor nulo, significa

que el usuario no podrá manipular el bluetooth desde la aplicación y deberá hacerlo de manera manual desde la configuración de sistemas o puede no contar con soporte bluetooth. Los métodos *enable()* y *disable()* permiten controlar el encendido y apagado del bluetooth.

Si el dispositivo cuenta con soporte bluetooth y el adaptador está apagado, al iniciar la aplicación o al presionar el botón encender, se crea un nuevo *Intent* para la solicitud del inicio del servicio bluetooth a través del método *startActivityForResult()*, el cual cuenta con dos parámetros, la acción (*Intent*) y el identificador. Donde el *Intent(adapterBT.ACTION\_REQUEST\_ENABLE)* es la respuesta al encendido del bluetooth y el entero (1) que se envía sirve para lanzar una ventana y saber si el usuario acepta encender o no su bluetooth.

```
228 public void Encender(View view) {
229     if (!adaptadorBT.isEnabled()) { //si el bluetooth está apagado
230         enciende = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE); //manda solicitud a iniciar
231         startActivityForResult(enciende, 1); //envía el estado al ActivityResult
232     } else {
233         imgV_Bluetooth.setImageResource(R.drawable.blueencendido);
234         listaDispNuevos();
235         imgV_Buscar.setVisibility(View.VISIBLE);
236     }
237 }
```

**Código 29. Método para encender adaptador.**

La respuesta a la acción de la ventana lanzada se maneja en el método *onActivityResult*, si el usuario acepta que la aplicación haga uso del bluetooth, se inicia con la búsqueda de dispositivos con el uso del método *listaDispNuevos()*.

```
203 protected void onActivityResult(int requestCode, int resultCode, Intent outputIntent) //obedece al
204 {
205     if (resultCode == RESULT_OK) { //si la respuesta a la conexión es si
206         imgV_Buscar.setVisibility(View.VISIBLE); //pone el boton busqueda activo
207         imgV_Bluetooth.setImageResource(R.drawable.blueencendido);
208         listaDispNuevos(); //hace llamada al metodo
209         Toast.makeText(getApplicationContext(), "Bluetooth Encendido", Toast.LENGTH_SHORT).show();
210     }
211 }
212 }
```

**Código 30. Método para recibir respuesta de activación bluetooth.**

#### 4.2.3.2. Búsqueda de dispositivos.

Para la búsqueda de dispositivos se sigue usando el Adaptador bluetooth, existen dos formas de conectarse a otro dispositivo:

- Mediante los dispositivos vinculados o existentes en la lista bluetooth, sin necesidad de que se muestren visibles.

- Realizando una búsqueda de todos los dispositivos existentes y visibles.

Para tener una interfaz que sea más intuitiva se opta por dejar solo el descubrimiento de todos los dispositivos, ya que el adaptador bluetooth HC-05 actúa permanentemente como un servidor, lo que significa que estará siempre visible para aceptar una nueva conexión.

Una vez que el bluetooth está encendido, se debe realizar una búsqueda de los dispositivos que se encuentran visibles dentro del radio de detección para que se realice una conexión exitosa, para ello se utiliza el método *startDiscovery()* que está dentro del método *listaDispNuevos()*, *startDiscovery()* inicia la búsqueda de nuevos dispositivos.

```

222 public void listaDispNuevos() { //metodo para mostrar los dispositivos encontrados
223     borrar();
224     adaptadorBT.startDiscovery();//inicia la búsqueda
225 }
226

```

**Código 31. Método para iniciar la búsqueda de dispositivos.**

El reconocimiento de los dispositivos que están dentro del radio tiene una duración de 12 segundos, por lo que se implementa el botón de búsqueda si es necesario para encontrar nuevos dispositivos visibles y ayuda a un mejor funcionamiento, una vez que se encuentra el dispositivo al cual se desea hacer una conexión, es recomendable que se utilice el método *cancelDiscovery()* el cual detiene la búsqueda ya que ésta acción consume una gran cantidad de recursos.

En la plataforma de programación para detectar un nuevo dispositivo visible dentro del radio, se hace uso de un *Intent* que será manejado por un *BroadcastReceiver* que monitorea si se encuentran nuevos dispositivos, para después obtener su nombre, su dirección MAC y los guarde en arrays necesarios para su uso posterior en la conexión con el dispositivo.

```

144 final BroadcastReceiver recibir = (context, intent) -> { //recibe el contexto e intent a registrar
147     al_DispositivosBT = new ArrayList<BluetoothDevice>(); //se inicia el metodo para el array
148     String action = intent.getAction(); //se obtiene el tipo de intent encontrado
149
150     switch (action) {
151         case BluetoothDevice.ACTION_FOUND:
152             BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
153             if (!al_DispositivosBT.contains(device)) { //si se tiene dispositivo encontrado
154                 al_DispositivosBT.add(device); // se agrega información de dispositivo al array
155                 arrayAdaptador.add(device.getName() + "\n" + device.getAddress()); // se agrega
156                 al_NombreBT.add(device.getAddress()); //se guarda solo la Mac para poder hacer
157             }
158             break;
159

```

**Código 32. BroadcastReceiver espera para encontrar nuevos dispositivos.**

Otra parte indispensable para la búsqueda de dispositivos es tener donde mostrarlos, la forma más útil es mediante un listado, para ello en android se utiliza un *ListView* el cual es un array que guarda el nombre y la dirección MAC.

#### 4.2.3.3. Conexión con un dispositivo.

Para la conexión cabe aclarar que la aplicación solo funcionará como cliente y se dedica a buscar dispositivos que estén siendo usados como servidores (dedicado solo a aceptar conexiones) o cliente y servidores (dedicado a aceptar y mandar peticiones), por lo que no es necesario que la aplicación esté habilitada para aceptar conexiones.

Para iniciar la conexión se debe seleccionar el dispositivo con el que se quiere hacer la conexión, para esto se utiliza el *ListView*, el cual estará habilitado para la detección de un evento *touch*, al hacer uso de este evento se toma la posición del elemento dentro del *ListView* y ésta posición será utilizada para moverse en los diferentes *Arrays* creados para lograr la conexión.

```
117
118
121
122
123
124
125
126
127
128
129
130
    /*Muestra los dispositivos encontrados en lv_Dispositivos*/
    lv_Dispositivos.setOnItemClickListener((parent, view, position, id) -> {
        mHandler.obtainMessage(0).sendToTarget();
        if (adaptadorBT.isDiscovering())//verifica si aun se busca dispositivos
            adaptadorBT.cancelDiscovery();//si encuentra búsqueda se cancela
        posicion = position;
        BluetoothDevice device = adaptadorBT.getRemoteDevice(al_NombreBT.get(posicion)); //s
        myService.setBt(device);
    });
```

**Código 33.** Evento *onClick* para seleccionar dispositivo bluetooth a conectar.

La primera etapa que deben superar los dispositivos para compartir información es vincularse a través de la petición de una conexión entrante pidiendo una contraseña que normalmente proporciona el servidor al cliente [Figura 37].

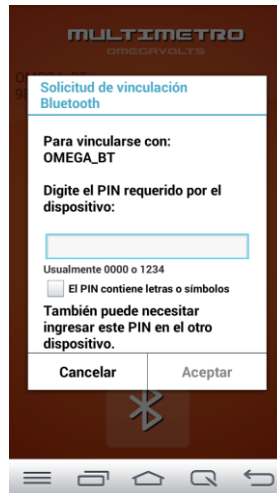
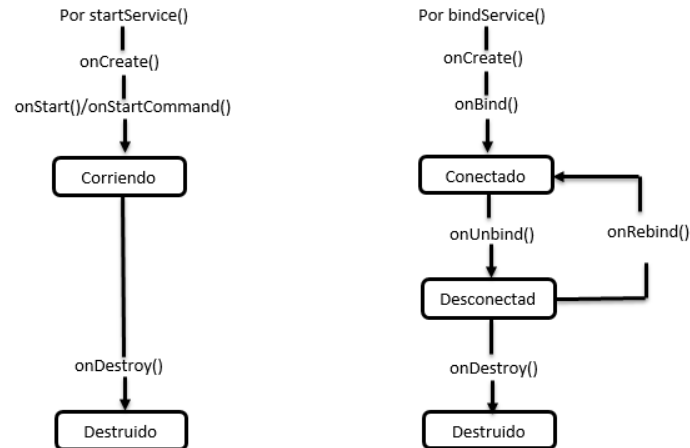


Figura 37. Petición de Contraseña para conexión bluetooth.

Una vez que se selecciona el dispositivo y se pone la contraseña correctamente, se manda su dirección MAC al método *setBt* de la clase *MyService* donde están las sub clases *ConnectThread* y *ConnectedThread* que heredan de la clase *Thread* lo que permite que puedan implementar el método *run*, este método indica que se correrá un nuevo hilo de ejecución.

Para lograr la comunicación entre los diferentes fragments se usa un servicio, el cual contiene las clases *ConnecteThread* y *ConnectedThread*, que permiten la creación de un socket de comunicación y el envío y recepción de datos. Se utiliza un servicio cuando se necesita realizar algún tipo de acción en segundo plano y ésta acción o fragmento de código pueda ser utilizado por distintas actividades.

Existen dos tipos de servicio. El primero, *startService()* o servicio iniciado, necesita que la actividad o algún otro componente le indique cuando comenzar, puede quedar ejecutándose en segundo plano de manera indefinida, pero una vez que termina la acción, devolverá un único resultado y el servicio terminará. Mientras que el segundo tipo de servicio *bindService()* o servicio de enlace, tiene la particularidad de funcionar como cliente y servidor, lo que permite la interacción de actividades o componentes con el servicio hasta que se llama al método *stopService()* o *stopSelf()* y detiene el servicio [Figura 38].



**Figura 38. Ciclo de vida de los dos tipos de servicio.**

Para un mejor funcionamiento se utiliza el servicio de enlace *bindeService()*. Esto debido a que la aplicación cuenta con distintas actividades que envían o reciben datos a través de un socket bluetooth, el cual necesita de un hilo de ejecución independiente que evite la interrupción de la comunicación y por ende el envío y recepción de información.

Para que los servicios puedan ser utilizados por otras aplicaciones o actividades primero deben ser declarados en el manifest dentro de la etiqueta <service>.

```
<service android:name=".MyService" >
</service>
```

**Código 34. Se inicia servicio en el Manifest.**

Una vez que se tienen definido el servicio en el manifest, la clase principal donde se encuentre el Servicio, en éste caso la clase es llamada *MyService* debe extender de *Service*.

```
public class MyService extends Service {
    private BluetoothSocket so;
    private BluetoothDevice bt;
    private String dato,envio,tipo_grafica, escala;
    private int intervalo;
    private float unidades;
    private BluetoothSocket mmSocket;
    private static final UUID MY_UUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
```

**Código 35. Se crea interfaz para servicio.**

Cuando el método *setBt* de la clase *MyService* obtiene el dispositivo con la información necesaria para la conexión (MAC address), inicia la clase *ConnectThread*. Ésta clase se

encargará de crear el socket que terminará por establecer la conexión, el socket es creado basándose en la información del dispositivo servidor y un identificador único universal (UUID) el cual consta de 16 bytes o bien 128 bits.

```

48 public void setBt (BluetoothDevice bt1){
49     this.bt = bt1;
50     if (bt != null) {
51         ConnectThread connectThread = new ConnectThread(bt);
52         connectThread.start();
53     }
54 }

```

**Código 36. Método para obtener el dispositivo elegido e iniciar ConnectThread.**

```

105 public class ConnectThread extends Thread { //nueva clase para conexión mediante un socket
106     private final BluetoothDevice mmDevice;
107     public ConnectThread(BluetoothDevice device) {
108
109         BluetoothSocket tmp = null; //se pone en blanco el socket
110         this.mmDevice = device; // se iguala la variable tipo Dispositivoblurtooth con la macBT
111
112         try {
113             tmp = device.createRfcommSocketToServiceRecord(MY_UUID); //se crea un socket con un
114         } catch (IOException e) { }
115         mmSocket = tmp;
116         so = tmp;
117     }
118
119     public void run() {
120         try {
121             //se hace la conexión
122             mmSocket.connect();
123         } catch (IOException connectException) {
124             // Unable to connect; close the socket and get out
125             try {
126                 mmSocket.close();// si algo sale mal se termina la conexión
127                 mHandler.obtainMessage(2).sendToTarget();
128             } catch (IOException closeException) { }
129             return;
130         }
131         mHandler.obtainMessage(0, mmSocket).sendToTarget(); // se envia al handler el tipo de ev
132     }
133 }

```

**Código 37. La clase ConnectThread crea un socket para enviar y recibir datos.**

Una vez que la conexión se logra se manda un aviso a *Handler* con dos parámetros, el primero es un entero con el que va a ser identificado en un switch con varios identificadores y el segundo el socket recién creado que servirá para inicializar la clase *ConnectedThread* que sirve para el envío y recepción de datos.

#### 4.2.3.4. Transferencia de datos entre dispositivos.

Un *Handler* se encarga de la gestión de procesos en segundo plano o lo que es igual gestión de hilos. En este caso el hilo que es implementado en la clase *ConnectThread* manda la información de que la conexión es exitosa con el dispositivo bluetooth al *Handler*. Al tener un socket abierto es posible enviar y recibir información, para ello se inicia la clase *ConnectedThread* a través del *Handler*.



```

207 public Handler mHandler = handleMessage(msg) -> {
210     super.handleMessage(msg);
211
212     switch (msg.what){
213
214         case 0:
215             ConnectedThread connectThread = new ConnectedThread((BluetoothSocket)msg.obj);
216             Toast.makeText(getApplicationContext(), "Conectado al dispositivo ", Toast.LENGTH_SHORT).show();
217             String s = "Conexion exitosa OMEGAS";
218             estadoDeSocket = 1;
219             connectThread.write(s.getBytes());
220             connectThread.start();
221
222             break;
223
224         case 1:
225
226             byte[] readBuf = (byte[]) msg.obj;
227             String readMessage = new String(readBuf, 0, msg.arg1);
228             envio = readMessage;
229             break;
230         default:
231             break;
232     }
233 };
234
235
236 }

```

**Código 38.** El controlador de hilos obtiene el socket y datos recibidos.

Si la conexión es exitosa se lanza la clase *ConnectedThread*.

```

135 public class ConnectedThread extends Thread { //Nueva clase para el control de entradas y salida
136     private final BluetoothSocket mmSocket;
137     private final InputStream mmInStream;
138     private final OutputStream mmOutStream;
139
140     public ConnectedThread(BluetoothSocket socket) {
141         this.mmSocket = socket; //recibe el socket creado y conectado
142         InputStream tmpIn = null;
143         OutputStream tmpOut = null;
144
145         // Get the input and output streams, using temp objects because
146         // member streams are final
147         try {
148             tmpIn = socket.getInputStream(); //obtiene el hilo de entrada
149             tmpOut = socket.getOutputStream(); //obtiene el hilo de salida
150         } catch (IOException e) {
151             e.getMessage();
152         }
153
154         mmInStream = tmpIn;
155         mmOutStream = tmpOut;
156     }
157
158 }

```

```

159 public void run() {
160     int bytes = 0; // inicializa los bytes en 0 que será donde empezará a leer el array de b
161
162     byte[] buffer = new byte[1024];
163
164     while (estadoDeSocket == 1) { // mientras haya que leer se ejecuta el sig código
165
166         try {
167             buffer[bytes] = (byte) mmInStream.read(); //el array de buffer en la posición act
168         } catch (IOException e) {
169             estadoDeSocket = 3;
170             try {
171                 mmInStream.close();
172             } catch (IOException e1) {
173                 e1.printStackTrace();
174             }
175             Intent mainActivity = new Intent(getApplicationContext(), MainActivity.class);
176             mainActivity.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
177             startActivity(mainActivity);
178             break;
179         }
180         if ((buffer[bytes] == '\n')) //hasta que se detecte un salto de line que se incluye
181         {
182             mHandler.obtainMessage(1, bytes, -1, buffer).sendToTarget(); // se envia al handl
183             bytes=0; //regresa a 0 para leer nuevamente laas entradas
184         }
185         else {
186             if (buffer[bytes] != 0){ //si en el buffer capta algun dato recibido
187                 bytes++; //se incrementara una posición para agregar en el buffer
188             }else{ //si no se mandan datos del arduino
189                 break; //termina la recepción de datos
190             }
191         }
192     }
193 }
194 }

```

**Código 39.** Clase *ConnectedThread* para iniciar hilo de emisión de datos e hilo de recepción de datos.

La clase *ConnectedThread* cuenta con dos métodos que son necesarios para el envío de datos que utiliza el método *write(byte[] buffer)*, en el cual se recibe una cadena convertida a bytes guardados en un buffer que posteriormente será enviado haciendo uso del socket. El segundo método es el que permite recibir datos del dispositivo con el que se tiene conexión, el método utilizado es un hilo o método *run*, que se mantiene en espera para la recepción de bytes que son guardados en un buffer para después utilizarlos de acuerdo a la navegación de la aplicación.

#### 4.3.4 Prueba / integración “pantalla 1” - iteración 1.

En este bloque se presentan imágenes de las pruebas realizadas para la parte de la aplicación llamada “pantalla 1”, en donde como primer paso la aplicación pide al usuario que se encienda el bluetooth [Figura 39].



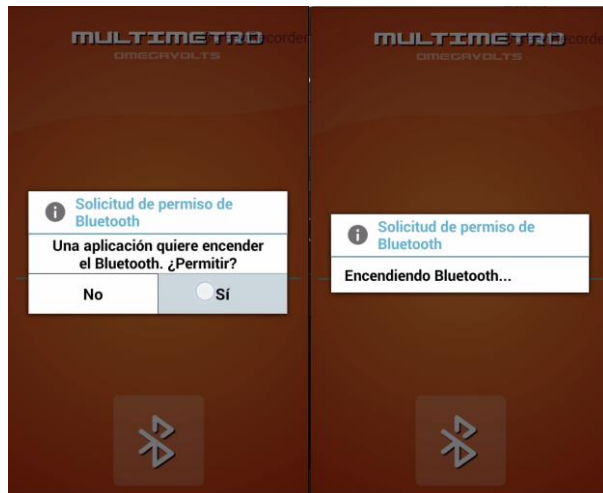
Figura 39. Mensaje para encender bluetooth.

Cabe mencionar que, si el dispositivo bluetooth es encendido por el usuario desde ajustes del dispositivo, este mensaje no aparece, en esta prueba se hace la suposición de que el usuario tiene apagado su bluetooth, a continuación, se observa como la aplicación reacciona e indica al usuario si desea encender el bluetooth [Figura 40].



**Figura 40. Petición de encendido de bluetooth.**

En caso de que el usuario oprima el botón NO, la aplicación no tendrá ningún cambio y seguirá funcionando correctamente. Cuando se presiona el botón SI, la aplicación le indicará al usuario que el bluetooth se está siendo encendido [Figura 41].



**Figura 41. Encendido del bluetooth.**

Cuando el bluetooth es encendido la aplicación avisa al usuario y comienza la búsqueda de dispositivos visibles [Figura 42].



**Figura 42. Búsqueda de dispositivos visibles.**

Si la aplicación no detecta dispositivos se notifica al usuario mediante un mensaje y éste puede presionar el botón buscar para intentar las veces que lo requiera hasta encontrar un dispositivo. Los dispositivos aparecerán en una lista y pueden seleccionarse para una conexión [Figura 43].



**Figura 43. Lista con dispositivos visibles.**

Si es la primera vez que se hace conexión con un dispositivo encontrado, se pedirá un PIN para vincular o emparejar los dispositivos [Figura 44].

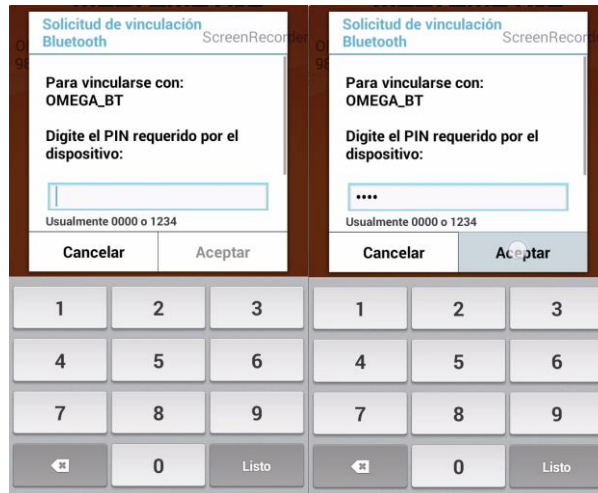


Figura 44. PIN de vinculación bluetooth.

Como se puede observar la navegabilidad para esta pantalla es muy sencilla y funciona de manera adecuada.

#### 4.2.5 Análisis y requerimientos “ViewPager para pantalla 2 y 3” - iteración 1.

En este apartado se describen los requerimientos para la pantalla 2 y 3 de acuerdo a el diseño de la [Figura 35] ya que ambas pantallas son parte de un viewPager, el cual se explica más adelante.

Descripción de la Aplicación Móvil:	<b>Aplicación móvil para dispositivos con sistema operativo Android ViewPager para pantalla 2 y 3.</b>		
Objetivo:	Diseñar y desarrollar la interfaz y funcionamiento ViewPager para pantalla 2 y pantalla 3.		
Requerimientos para aplicación móvil ViewPager para pantalla 2 y 3 - iteración 1			
No.	Descripción	Funcional	No Funcional
1	Estructura View Pager para las pantallas 2 y 3.	✓	
2	Elaboración del Fragment para la pantalla 2		✓
3	Elaboración del Fragment para la pantalla 3		✓
4	Evento de selección entre los diferentes Fragments.		✓

Tabla 6. Requerimientos para aplicación móvil ViewPager para pantalla 2 y 3 - iteración 1

#### 4.2.6 Diseño “ViewPager para pantalla 2 y 3” - iteración 1.

Para las pantallas 2 y 3 se utiliza una estructura compleja llamada View Pager que ofrece características que favorecen la manipulación de la aplicación haciéndola intuitiva y amigable para el usuario [Figura 45].

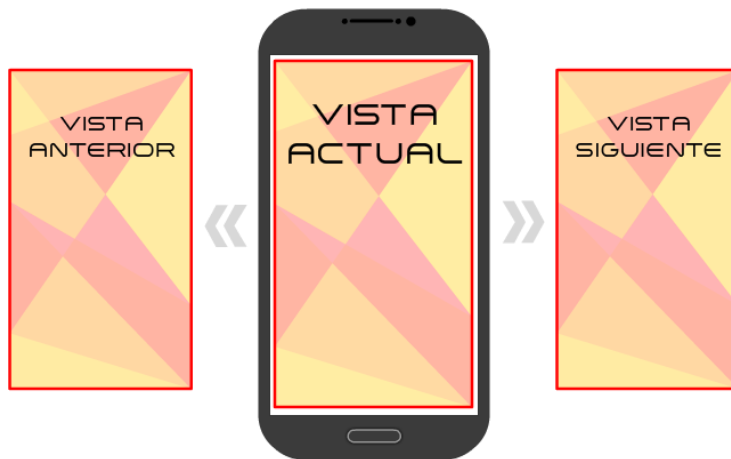


Figura 45. Diseño ViewPager.

##### 4.2.6.1 Viewpager

Con la intención de presentar una navegación intuitiva entre la aplicación de Multímetro Digital es necesario utilizar estructuras más complejas que una simple navegación llana entre actividades mediante botones.

Es por eso que el controlador de distribución ViewPager es la herramienta que brinda una serie de características con una implementación fácil a pesar de ser un tema avanzado en la programación de Android.

El ViewPager permite la navegación de izquierda a derecha de swipe views, o dicho de otra manera permite interactuar al usuario con la aplicación simplemente deslizando el dedo de izquierda a derecha para moverse entre vistas. Para el multímetro digital el ViewPager

trabaja en conjunto con Fragments permitiendo de esta manera la navegación entre Fragments.

#### 4.2.6.2 Fragment

Un Fragment puede ser definido como un módulo dentro de la interfaz de usuario de una actividad. Esta interfaz de usuario puede ser elaborada con diversos módulos o Fragments.

Los Fragments presentan características interesantes e indispensables. Un Fragment siempre debe estar embebido dentro de una actividad, por lo tanto, el ciclo de vida de dicho Fragment está directamente afectado por el ciclo de vida de la actividad que lo contiene. Sin embargo, el Fragment tiene su propio ciclo de vida, es decir, un Fragment puede ser agregado o eliminado mientras la actividad que lo contiene sigue en ejecución, además de poder gestionar sus propios elementos de entrada y tener su propia interfaz.

Las ventajas de utilizar Fragments son las de producir interfaces dinámicas y flexibles. Se refiere a flexibles a la implementación de la aplicación en dispositivos como las tablets que presentan una pantalla de mayor tamaño. Dejando así la opción de crear interfaces especiales para dichos dispositivos sin la necesidad de reestructurar toda la aplicación, sino simplemente darle soporte a dicho tamaño de pantalla.

#### 4.2.7 Codificación o ensamble “ViewPager para pantalla 2 y 3” - iteración 1.

```
1 <android.support.v4.view.ViewPager
2     xmlns:android="http://schemas.android.com/apk/res/android"
3     android:id="@+id/pager"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent">
6
7 </android.support.v4.view.ViewPager>
8
```

**Código 40.** Se declara soporte para fragment en XML.

A continuación, se explicará el desarrollo del controlador de distribución ViewPager. Para el multímetro digital se requiere de un documento “.java” que gestione el funcionamiento de los Fragments que corresponden a la “pantalla 2” y “pantalla 3”. Esta clase debe extender de *FragmentPagerAdapter* la cual sobre escribe el método de tipo *Fragment*

`getItem(int arg0)`; Dicho método es un switch y esta estructura de control devuelve el Fragment seleccionado.

```

17      @Override
18      public Fragment getItem(int arg0) {
19          switch (arg0) {
20              case 0:
21                  return new MultimetroFragment();
22              case 1:
23                  FragmentTwo f2 = new FragmentTwo();
24                  return new FragmentTwo(f2);
25              default:
26                  break;
27          }
28          return null;
29      }
30      @Override
31      public int getCount() {
32          return 2;
33      }

```

**Código 41. Switch para selección de Fragment.**

Como ya se mencionó con anterioridad los Fragment tienen su propia interfaz y pueden gestionar sus propios elementos mediante un documento “.java”, por lo tanto, para que el controlador de distribución reconozca a los Fragments es necesario que los documentos “.java” que corresponden a la “pantalla 2” y “pantalla 3” deban extender de la clase *Fragment*. Una vez teniendo el controlador de distribución *ViewPager*, se crean los Fragments.

#### 4.2.7.1 Análisis y requerimientos “pantalla 2 – Fragment 1” - iteración 1.

En este apartado se describen los requerimientos para el Fragment que representará la pantalla 2 de acuerdo al diseño puesto en la [Figura 35].

Descripción de la Aplicación Móvil:	<b>Aplicación móvil para dispositivos con sistema operativo Android Pantalla 2 – Fragment 1.</b>		
Objetivo:	Diseñar y desarrollar la interfaz y funcionamiento de la pantalla 2.		
Requerimientos “pantalla 2 – Fragment 1” - iteración 1			
No.	Descripción	Funcional	No Funcional
1	Desplegar la información de lo que se está midiendo.	✓	
2	Evento de selección entre las diferentes mediciones.		✓
3	Elemento visual de selección entre las diferentes mediciones.	✓	

**Tabla 7. Requerimientos “pantalla 2 – Fragment 1” - iteración 1**



#### 4.2.7.2 Diseño “pantalla 2 – Fragment 1” - iteración 1.

En la [Figura 46] se muestra el diseño de la interfaz de “pantalla 2”. Donde la interfaz contiene un elemento de tipo .png que es utilizado para la seleccionar la medición que se quiere realizar. Además de una etiqueta en el centro de la pantalla en donde se despliega la información de las mediciones.

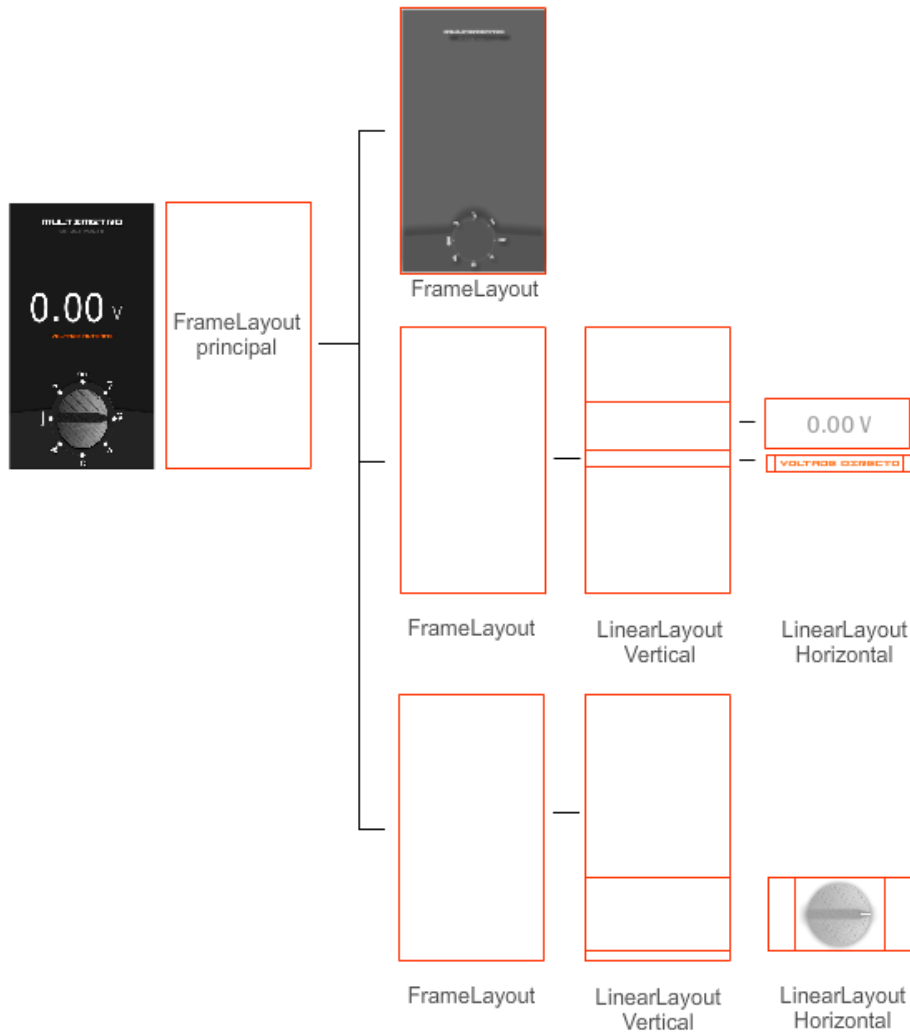


Figura 46. Diseño pantalla 2.

### 4.2.7.3 Codificación o ensamble “pantalla 2 – Fragment 1” - iteración 1.

A continuación, se muestra el código XML de la “pantalla 2” llamada MultimetroFragment, el cual simula un multímetro real.

```
1 <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
2   xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
3   android:layout_height="match_parent"
4   tools:context=".MainActivity">
5
6
7   <ImageView
8     android:layout_width="fill_parent"
9     android:layout_height="fill_parent"
10    android:id="@+id/imgV_Fondo" />
11
12   <FrameLayout
13     android:layout_width="match_parent"
14     android:layout_height="match_parent">
15
16     <LinearLayout
17       android:orientation="vertical"
18       android:layout_width="match_parent"
19       android:layout_height="match_parent"
20       android:layout_gravity="right|top">
21
22       <LinearLayout
23         android:orientation="vertical"
24         android:layout_width="match_parent"
25         android:layout_height="match_parent">
26
27         <LinearLayout
28           android:orientation="vertical"
29           android:layout_width="match_parent"
30           android:layout_height="match_parent"
31           android:layout_weight="2"></LinearLayout>
32
33       <LinearLayout
34
35         android:orientation="vertical"
36         android:layout_width="match_parent"
37         android:layout_height="match_parent"
38         android:layout_weight="1">
39
40         <TextView
41           android:layout_width="wrap_content"
42           android:layout_height="wrap_content"
43           android:textAppearance="?android:attr/textAppearanceLarge"
44           android:id="@+id/txt_Numeros"
45           android:layout_weight="1"
46           android:layout_gravity="center_horizontal"
47           android:textColor="#ffffff"
48           android:textIsSelectable="true"
49           android:textSize="80dp" />
50       </LinearLayout>
51     </LinearLayout>
52   </FrameLayout>
53
54   <FrameLayout
55     android:layout_width="match_parent"
56     android:layout_height="match_parent">
57
58     <LinearLayout
59       android:orientation="vertical"
60       android:layout_width="fill_parent"
61       android:layout_height="fill_parent">
62
63       <LinearLayout
64         android:layout_width="fill_parent"
```

```

65         android:layout_height="fill_parent"
66         android:weightSum="1"
67         android:orientation="vertical"
68         android:layout_weight="1">
69
70     </LinearLayout>
71
72     <LinearLayout
73         android:orientation="vertical"
74         android:layout_width="fill_parent"
75         android:layout_height="fill_parent"
76         android:layout_weight="1">
77
78         <LinearLayout
79             android:orientation="vertical"
80             android:layout_width="match_parent"
81             android:layout_height="match_parent"
82             android:layout_weight="4">
83
84             <LinearLayout
85                 android:orientation="horizontal"
86                 android:layout_width="match_parent"
87                 android:layout_height="match_parent">
88
89                 <LinearLayout
90                     android:orientation="vertical"
91                     android:layout_width="match_parent"
92                     android:layout_height="match_parent"
93                     android:layout_weight="3"></LinearLayout>
94                 </LinearLayout>
95     </LinearLayout>

```

```

95
96     <ImageView
97         android:layout_width="fill_parent"
98         android:layout_height="fill_parent"
99         android:id="@+id/imgV_Etiqueta"
100        android:scaleType="centerCrop"
101        android:layout_weight="1" />
102
103     <LinearLayout
104         android:orientation="vertical"
105         android:layout_width="match_parent"
106         android:layout_height="match_parent"
107         android:layout_weight="3"></LinearLayout>
108
109 </LinearLayout>
110 </LinearLayout>
111
112 <LinearLayout
113     android:orientation="vertical"
114     android:layout_width="match_parent"
115     android:layout_height="match_parent"
116     android:layout_weight="1">
117
118 </LinearLayout>
119
120 </LinearLayout>
121
122 </LinearLayout>
123 </FrameLayout>
124
125 <FrameLayout
126     android:layout_width="match_parent"

```

```

127         android:layout_height="match_parent">
128
129         <LinearLayout
130             android:orientation="vertical"
131             android:layout_width="match_parent"
132             android:layout_height="match_parent">
133
134             <LinearLayout
135                 android:orientation="vertical"
136                 android:layout_width="match_parent"
137                 android:layout_height="match_parent"
138                 android:layout_weight="1.4"></LinearLayout>
139
140             <LinearLayout
141                 android:orientation="horizontal"
142                 android:layout_width="match_parent"
143                 android:layout_height="match_parent"
144                 android:layout_weight="3">
145
146                 <LinearLayout
147                     android:orientation="vertical"
148                     android:layout_width="match_parent"
149                     android:layout_height="match_parent"
150                     android:layout_weight="1.6"></LinearLayout>
151
152                 <ImageView
153                     android:layout_width="wrap_content"
154                     android:layout_height="wrap_content"
155                     android:id="@+id/imgV_Perilla"
156                     android:layout_weight="1"
157
158                     android:background="@drawable/perillaespacio"
159                     android:layout_gravity="center_horizontal|bottom"
160                     android:scaleType="fitCenter" />
161
162                 <LinearLayout
163                     android:orientation="vertical"
164                     android:layout_width="match_parent"
165                     android:layout_height="match_parent"
166                     android:layout_weight="1.6"></LinearLayout>
167
168             </LinearLayout>
169
170             <LinearLayout
171                 android:orientation="vertical"
172                 android:layout_width="match_parent"
173                 android:layout_height="match_parent"
174                 android:layout_weight="4.2"></LinearLayout>
175
176         </LinearLayout>
177     </FrameLayout>
178 </FrameLayout>
179

```

**Código 42.** Código XML para pantalla de simulación de Multímetro digital.

#### 4.4.3.1 Fragment 1: Multimetrofragment

El primer Fragment dentro del ViewPager es la interfaz de un multímetro [Figura 47], tiene un diseño innovador, estético y congruente con los multímetros físicos. Se preservan elementos indispensables como la perilla, la cual permite seleccionar el tipo de medición; adicionalmente se presentan los valores medidos en un área céntrica con un estilo de letra grande facilitando la visualización por el usuario. También hay etiquetas con información que le permite al usuario una mejor comprensión de los eventos al interactuar con la aplicación como el nombre del tipo de medición y las unidades de los valores medidos.

Los colores utilizados en cada una de las interfaces de la aplicación siguen los principios utilizados en herramientas físicas de este tipo.

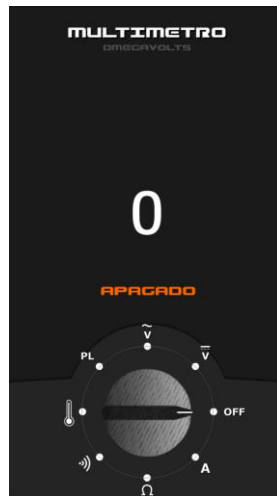


Figura 47. Fragment 1 interfaz multímetro.

## Perilla

Es una imagen propia de tipo .PNG diseñada con la suite de Adobe. Permite seleccionar el tipo de medición y esto sucede mientras se mantiene pulsada. En caso que la perilla fuese posicionada entre dos opciones al soltar el touch del dispositivo se posiciona de manera automática en la opción más cercana.

El evento `onTouchListener()` permite gestionar cada una de las funciones que la perilla realiza y se ejecuta cada que un evento táctil sucede sobre el elemento que lo contiene, en éste caso la perilla misma. Existen diversos movimientos (`MotionEvent`), originados por diferentes elementos que se pueden administrar con este método, sin embargo, los 2 `MotionEvent` que cubren las funciones de este elemento son:

### 1. `MotionEvent.ACTION_MOVE`

Este `MotionEvent` se ejecuta mientras el evento táctil sucede, es decir el dedo pulsa la perilla y sin ser levantado se mueve dentro de la interfaz. Traduce el movimiento en información, en este caso se obtiene la posición del dedo dentro de la interfaz devolviendo coordenadas en los ejes 'x' y 'y', vitales para calcular el ángulo al que apuntará la perilla.

Android implementa un sistema de coordenadas peculiar en los dispositivos móviles estipulando el origen en la esquina superior izquierda [Figura 48].

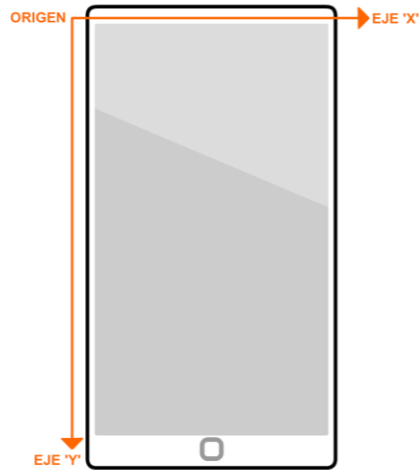


Figura 48. Ejes estipulados por el S.O., Android.

No obstante, es necesario crear un plano cartesiano dentro de la interfaz el cual tenga como origen el centro de la perilla para que ésta gire alrededor de ese punto [Figura 49 a]. Esto se logra gracias a que la perilla es una imagen cuadrada y está ubicada justo en el centro tomando de referencia el eje 'x'.

Se necesita conocer el nuevo par ordenado correspondiente al origen, para esto se obtiene el valor del ancho de la pantalla (X) y dividirla entre dos para obtener la coordenada correspondiente al eje 'x'. Para la coordenada correspondiente al eje 'y' se sabe que la imagen es cuadrada y basta con obtener el valor del largo de esta (L) y dividirla entre dos. En la [Figura 49 b]. El producto de esta operación se debe de restar al valor del largo de la pantalla (Y), tenemos entonces que:

$$\text{Origen} = \left[ \left( \frac{X}{2} \right), \left( Y - \left( \frac{L}{2} \right) \right) \right]$$

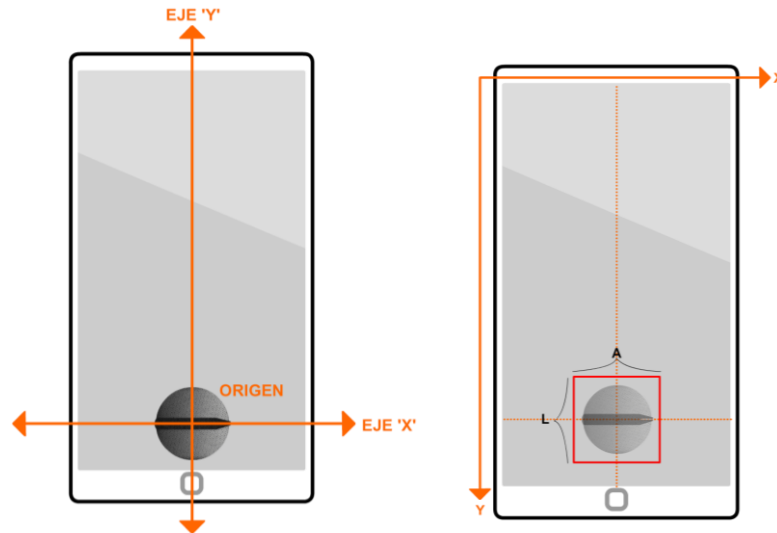


Figura 49 a y Figura 49 b. Eje origen perilla.

Mediante programación se estipulan los límites entre cuadrantes utilizando estructuras de control 'if'. Esto facilita el cálculo del ángulo el cual se obtiene mediante trigonometría, teorema de Pitágoras y leyes de catetos.

## 2. MotionEvent.ACTION\_UP

El fragmento de código que se ejecuta en este tipo de MotionEvent sucede cuando el dedo deja de tocar la superficie táctil, dicho de otra manera, el dedo es alzado.

Esto desencadena una serie de acciones, en primer lugar, se conoce el par ordenado del evento táctil por lo tanto es posible conocer la opción o tipo de medida que se quiere hacer.

Para realizar estas acciones se debe conocer el ángulo al que girará la perilla y apuntar a la opción deseada es por esto que se busca crear un triángulo utilizando las coordenadas obtenidas del evento táctil (T), las coordenadas del origen (O) y un par ordenado calculado (C).

El par calcular (C) se obtiene de la combinación de las coordenadas (T) y (O), quedando de la siguiente manera:

$$\text{Origen} = \left[ \left( \frac{X}{2} \right), \left( Y - \left( \frac{L}{2} \right) \right) \right]$$

$$\text{Evento táctil} = [M, N]$$

$$\text{Calculado} = \left[ M, \left( Y - \left( \frac{L}{2} \right) \right) \right]$$

A continuación, se observa un ejemplo del conjunto de pares ordenados [Figura 50].

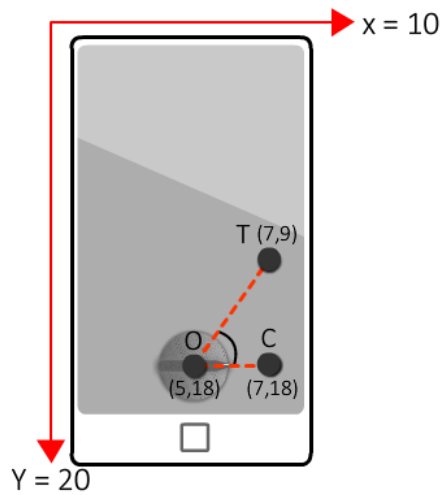


Figura 50. Ejemplo de conjunto de pares ordenados para el cálculo del ángulo.

En la figura anterior [Figura 50], se crea un triángulo al unir cada uno de los pares ordenados, por lo tanto, es posible calcular las longitudes del triángulo.

El Cateto adyacente se representa por la longitud entre el par ordenado O y C. El cateto opuesto estará representado por la longitud entre los pares ordenados T y C.

De tal manera que:

$$\text{Longitud Adyacente (LA)} = M - \left( \frac{X}{2} \right)$$

$$\text{Longitud Opuesta (LO)} = \left( Y - \left( \frac{L}{2} \right) \right) - N$$



Por último, se calcula el ángulo de la siguiente manera:

$$\alpha = \tan^{-1}\left(\frac{LO}{LA}\right)$$

Sin embargo, se debe hacer un ajuste con respecto a los 4 cuadrantes y la manera de calcular la longitud de cada cateto.

- Primer Cuadrante
  - *Longitud Adyacente (LA)* =  $M - \left(\frac{X}{2}\right)$
  - *Longitud Opuesta (LO)* =  $\left(Y - \left(\frac{L}{2}\right)\right) - N$
- Segundo Cuadrante
  - *Longitud Adyacente (LA)* =  $\left(\frac{X}{2}\right) - M$
  - *Longitud Opuesta (LO)* =  $\left(Y - \left(\frac{L}{2}\right)\right) - N$
- Tercer Cuadrante
  - *Longitud Adyacente (LA)* =  $\left(\frac{X}{2}\right) - M$
  - *Longitud Opuesta (LO)* =  $N - \left(Y - \left(\frac{L}{2}\right)\right)$
- Cuarto Cuadrante
  - *Longitud Adyacente (LA)* =  $M - \left(\frac{X}{2}\right)$
  - *Longitud Opuesta (LO)* =  $N - \left(Y - \left(\frac{L}{2}\right)\right)$

De esta manera se calcula el ángulo al que deberá girar la imagen de la perilla y así poder seleccionar la opción que se desea. Para el multímetro digital se ha hecho una implementación para el funcionamiento de la perilla con base en el número de funciones que se pueden realizar (Corriente, Voltaje, etc.). Al conocer la cantidad de funcionalidades también es posible conocer los ángulos exactos en los que se posicionan las opciones alrededor de la perilla. Esto con el objetivo de gestionar los casos en los que la perilla quede a la mitad de dos opciones. Cuando este caso se presente con una simple comparación entre distancias la perilla deberá apuntar a la opción más cercana.

Al conocer la opción elegida por el usuario se despliega en pantalla el tipo de medición que se está haciendo y envía mediante el método `setEscribe(String dato)` a la clase `MyService.java` el tipo de medición:

```
253 public void Servicio(View view, int opcion ){
254     int elemento = opcion;
255     String message;
256     if(elemento ==1){
257         myService.setEscribe("Apagado");
258     }else if(elemento==2){
259         //Corriente
260         myService.setEscribe("Corriente");
261     }else if(elemento==3){
262         //Resistencia
263         myService.setEscribe("Resistencia");
264     }else if(elemento==4){
265         //Continuidad
266         myService.setEscribe("Continuidad");
267     }else if (elemento==5){
268         //Temperatura
269         myService.setEscribe("Temperatura");
270     }else if (elemento==6){
271         //Punta lógica
272         myService.setEscribe("Punta lógica");
273     }else if (elemento==7){
274         //Voltaje Alterno
275         myService.setEscribe("Voltaje Alterno");
276     }else if (elemento ==8){
277         //Voltaje Directo
278         myService.setEscribe("Voltaje Directo");
279     }else{
280
281     }
282
283 }
```

**Código 43.** Selección de texto para ser enviado por bluetooth utilizando el método `setEscribe()`.

La clase `MyService.java` logra la comunicación entre los diferentes fragments y dicha clase contiene las clases `ConnecteThread` y `ConnectedThread`, que permiten la creación de un socket de comunicación y el envío y recepción de datos. Se utiliza un servicio cuando se necesita realizar algún tipo de acción en segundo plano y ésta acción o fragmento de código pueda ser utilizado por distintas actividades. Sin embargo, la importancia de esta clase se explica más a adelante en este documento.

#### 4.2.7.4 Prueba / integración “pantalla 2 – Fragment 1” - iteración 1.

Un proceso complicado que lleva esta interfaz es el de mantener la imagen ubicada en la posición correcta al momento de girar la perilla y que quede posicionada en el lugar correcto, lo cual se realiza de forma correcta [Figura 51].

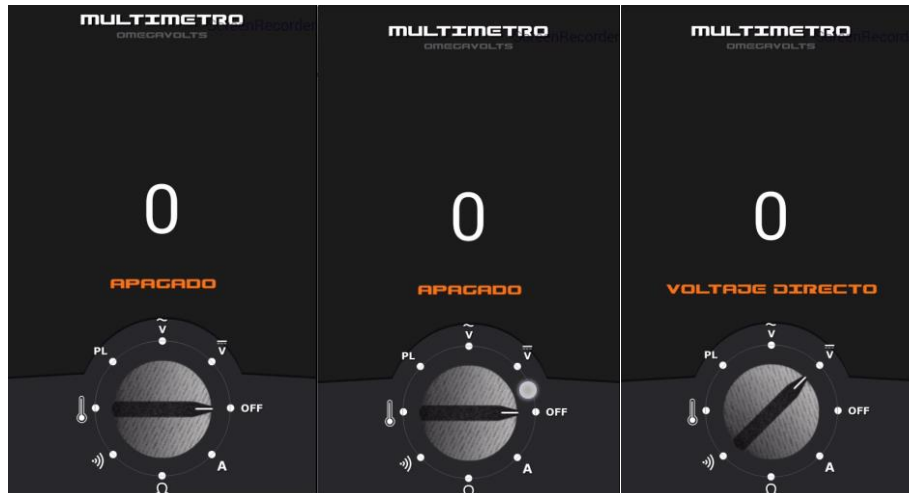


Figura 51. Movimiento de la perilla.

Como se observa en la figura anterior, al cambiar la perilla de posición, la aplicación indica mediante el texto en rojo que pasó de Apagado a Voltaje Directo sin perder la posición o dimensiones de la imagen de la perilla.

#### 4.2.7.5 Análisis y requerimientos “pantalla 3 – Fragment 2” - iteración 1.

En este apartado se describen los requerimientos para el fragment que representará la pantalla 3 de acuerdo a el diseño de las pantallas del multímetro [Figura 35].

Descripción de la Aplicación Móvil:	<b>Aplicación móvil para dispositivos con sistema operativo Android “pantalla 3” - Fragment 2.</b>		
Objetivo:	Diseñar y desarrollar la interfaz y funcionamiento de la pantalla 3.		
Requerimientos “pantalla 3 - Fragment 2” - iteración 1			
No.	Descripción	Funcional	No Funcional
1	Botones para seleccionar el tipo de graficación		✓
2	Eventos para cada uno de los botones	✓	
3	Intent que lanza la Pantalla 4		✓
4	Interacción con clase MyService		✓

Tabla 8. Requerimientos “pantalla 3 - Fragment 2” - iteración 1.

#### 4.2.7.6 Diseño “pantalla 3 – Fragment 2” - iteración 1.

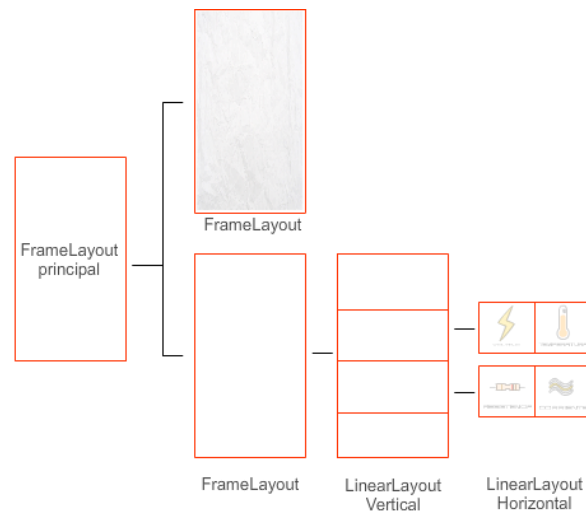


Figura 52. Diseño pantalla 2-Fragment 1.

#### 4.2.7.7 Codificación o ensamble “pantalla 3 - Fragment 2” - iteración 1.

A continuación, se muestra el código XML de la “pantalla 3” llamada Parámetros, el cual presenta una selección de mediciones a graficar.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <FrameLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent">
6
7     <ImageView
8         android:layout_width="fill_parent"
9         android:layout_height="fill_parent"
10        android:id="@+id/imgV_Fondo" />
11
12    <FrameLayout
13        android:layout_width="match_parent"
14        android:layout_height="match_parent">
15
16        <LinearLayout
17            android:orientation="vertical"
18            android:layout_width="match_parent"
19            android:layout_height="match_parent">
20
21            <LinearLayout
22                android:orientation="vertical"
23                android:layout_width="match_parent"
24                android:layout_height="match_parent"
25                android:layout_weight="1"></LinearLayout>
26
27            <LinearLayout
28                android:orientation="horizontal"
29                android:layout_width="match_parent"
30                android:layout_height="match_parent"
31                android:layout_weight="1">
```

```

32
33 <ImageView
34     android:layout_width="wrap_content"
35     android:layout_height="wrap_content"
36     android:id="@+id/imgB_Voltaje"
37     android:layout_weight="1" />
38
39 <ImageView
40     android:layout_width="wrap_content"
41     android:layout_height="wrap_content"
42     android:id="@+id/imgB_Corriente"
43     android:layout_weight="1" />
44 </LinearLayout>
45
46 <LinearLayout
47     android:orientation="horizontal"
48     android:layout_width="match_parent"
49     android:layout_height="match_parent"
50     android:layout_weight="1">
51
52     <ImageView
53         android:layout_width="wrap_content"
54         android:layout_height="wrap_content"
55         android:id="@+id/imgB_Resistencia"
56         android:layout_weight="1" />
57
58     <ImageView
59         android:layout_width="wrap_content"
60         android:layout_height="wrap_content"
61         android:id="@+id/imgB_Temperatura"
62         android:layout_weight="1" />
63 </LinearLayout>

```

```

64
65 <LinearLayout
66     android:orientation="vertical"
67     android:layout_width="match_parent"
68     android:layout_height="match_parent"
69     android:layout_weight="1"></LinearLayout>
70 </LinearLayout>
71 </FrameLayout>
72
73 </FrameLayout>
74

```

**Código 44.** Código XML para creación de la pantalla de parámetros.

La pantalla 3 contiene 4 botones que definen el tipo de gráfica que se desea hacer, y es quizá la pantalla con menos dificultad de toda la aplicación.

Los botones son imágenes de tipo .png a las que se les han asignado el `MotionEvent.ACTION_UP` de esta manera tienen la misma funcionalidad que un botón.

Al ser pulsados estos botones, se estipula el tipo de gráfica en la clase `MyService` mediante el método `myService.setTipoGrafica(String tipoGrafica)`. Así como iniciar la pantalla 4 en la cual se estipula la configuración de la gráfica.

La manera de llamar a la pantalla 4 es mediante un Intent. Un Intent *es una descripción abstracta de una operación a realizar* (Google, 2016).

Dicho de otra manera, es una estructura de datos que en tiempo de ejecución que tiene una instrucción de una acción a realizar y su uso más común es la de lanzar actividades, sin embargo, también se pueden llamar servicios o hasta iniciar otras aplicaciones.

En el multímetro digital es utilizado para llamar otra actividad o mejor dicho ejecutar o mostrar la pantalla 4 de la siguiente manera:

```
111 | | | Intent intent = new Intent(getActivity(), Parametros.class);
```

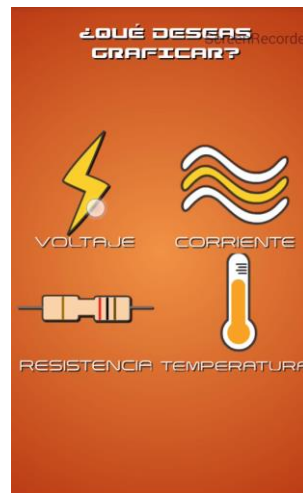
**Código 45. Intent para llamar otra actividad.**

Siendo “Parámetros” el nombre de la pantalla 4.

El objetivo de esta pantalla es conocer el tipo de gráfica que se quiere realizar para que Arduino conozca qué proceso deberá hacer mediante la clase MyService.

#### **4.2.7.5 Prueba / integración “pantalla 3 – Fragment 2” - iteración 1.**

Al igual que la “pantalla 2”, esta pantalla es parte de un Fragment lo que permite que la “pantalla 3” sea visible al deslizar la “pantalla 2” hacia la izquierda. La “pantalla 3” permite seleccionar que tipo de medición se va a graficar [Figura 53].



**Figura 53. Pantalla 3, selección de tipo de medición.**

#### **4.2.8 Análisis y requerimientos “pantalla 4” - iteración 1.**

En este apartado se describen los requerimientos la “pantalla 4” siguiendo el diseño de la [Figura 35].

Descripción de la Aplicación Móvil:	<b>Aplicación móvil para dispositivos con sistema operativo Android “pantalla 4”.</b>		
Objetivo:	Diseñar y desarrollar la interfaz y funcionamiento de la “pantalla 4”.		
Requerimientos para “pantalla 4” - iteración 1			
No.	Descripción	Funcional	No Funcional
1	Parámetro intervalos de tiempo		✓
2	Escala de tiempo (segundos o milisegundos)		✓
3	Botón Graficar		✓
4	Intent que lanza la actividad de la gráfica.	✓	

Tabla 9. Requerimientos para “pantalla 4” - iteración 1.

#### 4.2.9 Diseño “pantalla 4” - iteración 1.

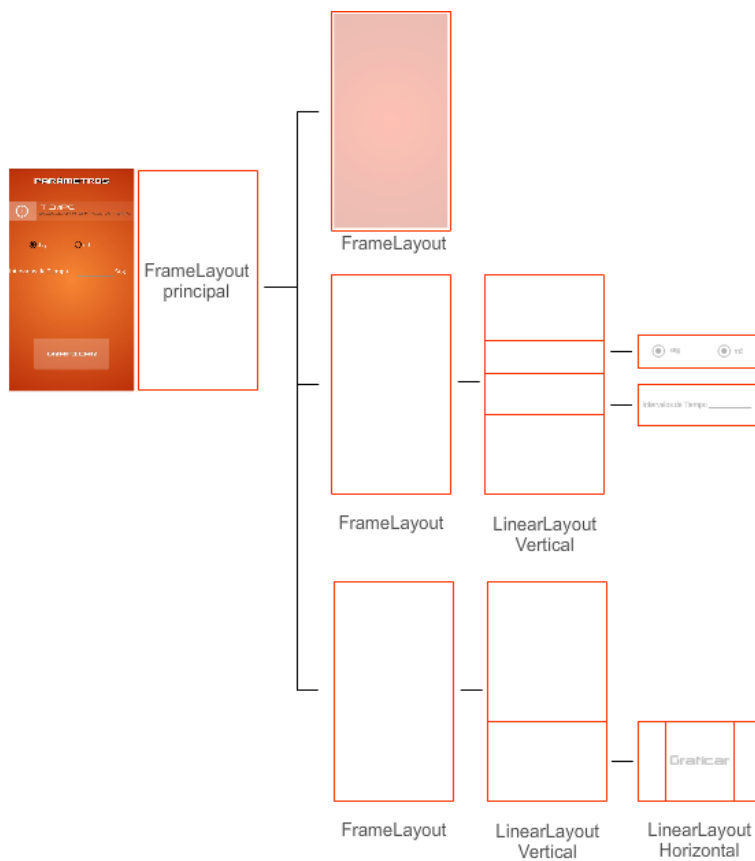


Figura 54. Diseño de Pantalla 4.

## 4.2.10 Codificación o ensamble “pantalla 4” - iteración 1.

A continuación, se muestra el código XML de la “pantalla 4”, en esta pantalla se seleccionan las unidades de tiempo que tendrán los intervalos entre cada medición.

```
1 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2   xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
3   android:layout_height="match_parent"
4   tools:context="com.jorgecastillo.pc.aplicacion_15.Grafica_Resistencia">
5
6   <FrameLayout
7     android:orientation="horizontal"
8     android:layout_width="fill_parent"
9     android:layout_height="fill_parent">
10
11     <ImageView
12       android:layout_width="fill_parent"
13       android:layout_height="fill_parent"
14       android:id="@+id/imgV_Fondo"/>
15
16     <FrameLayout
17       android:layout_width="fill_parent"
18       android:layout_height="fill_parent">
19
20       <LinearLayout
21         android:orientation="vertical"
22         android:layout_width="fill_parent"
23         android:layout_height="fill_parent">
24
25         <LinearLayout
26           android:orientation="vertical"
27           android:layout_width="fill_parent"
28           android:layout_height="wrap_content"
29           android:layout_weight="2"></LinearLayout>
```

```
31
32     <LinearLayout
33       android:orientation="horizontal"
34       android:layout_width="fill_parent"
35       android:layout_height="wrap_content"
36       android:layout_weight="1">
37
38       <LinearLayout
39         android:orientation="horizontal"
40         android:layout_width="match_parent"
41         android:layout_height="match_parent"
42         android:layout_weight="3"></LinearLayout>
43
44     <RadioGroup
45       android:layout_width="match_parent"
46       android:layout_height="match_parent"
47       android:orientation="horizontal"
48       android:layout_weight="1">
49
50       <RadioButton
51         android:layout_width="wrap_content"
52         android:layout_height="wrap_content"
53         android:text="Seg"
54         android:id="@+id/rbtn_SegundosCorriente"
55         android:checked="true"
56         android:layout_weight="1"
57         android:onClick="eventrbtn"
58         android:textColor="#ffffffff"
59         android:layout_gravity="center" />
```



```

60
61
62     <RadioButton
63         android:layout_width="wrap_content"
64         android:layout_height="wrap_content"
65         android:text="mS"
66         android:id="@+id/rbtn_msCorriente"
67         android:checked="false"
68         android:layout_weight="1"
69         android:onClick="eventrbtn"
70         android:textColor="#ffffffff"
71         android:layout_gravity="center" />
72
73     </RadioGroup>
74
75     <LinearLayout
76         android:orientation="vertical"
77         android:layout_width="match_parent"
78         android:layout_height="match_parent"
79         android:layout_weight="3"></LinearLayout>
80
81 </LinearLayout>
82
83 <LinearLayout
84     android:orientation="horizontal"
85     android:layout_width="fill_parent"
86     android:layout_height="wrap_content"
87     android:layout_weight="1">
88
89     <TextView
90         android:layout_width="wrap_content"
91         android:layout_height="wrap_content"

```

```

90         android:textAppearance="?android:attr/textAppearanceMedium"
91         android:text="Intervalos de Tiempo"
92         android:id="@+id/label_IntervalosdeTiempo"
93         android:textColor="#ffffffff"
94         android:layout_weight="2" />
95
96     <EditText
97         android:layout_width="wrap_content"
98         android:layout_height="wrap_content"
99         android:inputType="numberDecimal"
100        android:id="@+id/ET_UnidadTiempoC"
101        android:layout_weight="2"
102        android:textColor="#ffffffff"
103        android:ems="4"
104        android:maxLength="3" />
105    ET_TiempoResistencia
106    <TextView
107        android:layout_width="wrap_content"
108        android:layout_height="wrap_content"
109        android:textAppearance="?android:attr/textAppearanceMedium"
110        android:text="Seg"
111        android:id="@+id/txt_UnidadTiempoCorriente"
112        android:textColor="#ffffffff"
113        android:layout_weight="2" />
114    </LinearLayout>
115
116    <LinearLayout
117        android:orientation="vertical"
118        android:layout_width="fill_parent"
119        android:layout_height="wrap_content"
120        android:layout_weight="3"
121        android:layout_gravity="top"></LinearLayout>

```

```

122     </LinearLayout>
123
124 </FrameLayout>
125
126 <FrameLayout
127     android:layout_width="match_parent"
128     android:layout_height="match_parent">
129
130     <LinearLayout
131         android:orientation="vertical"
132         android:layout_width="fill_parent"
133         android:layout_height="fill_parent"
134         android:layout_weight="2"
135         android:layout_gravity="bottom">
136
137         <LinearLayout
138             android:orientation="vertical"
139             android:layout_width="match_parent"
140             android:layout_height="match_parent"
141             android:layout_weight="1">
142
143             <LinearLayout>
144
145             <LinearLayout
146                 android:orientation="vertical"
147                 android:layout_width="match_parent"
148                 android:layout_height="match_parent"
149                 android:layout_weight="1">
150
151                 <LinearLayout
152                     android:orientation="vertical"
153                     android:layout_width="match_parent"
154
155                     android:layout_height="match_parent"
156                     android:layout_weight="2"></LinearLayout>
157
158                 <LinearLayout
159                     android:orientation="horizontal"
160                     android:layout_width="match_parent"
161                     android:layout_height="match_parent"
162                     android:layout_weight="1">
163
164                     <LinearLayout
165                         android:orientation="horizontal"
166                         android:layout_width="match_parent"
167                         android:layout_height="match_parent"
168                         android:layout_weight="2"></LinearLayout>
169
170                     <ImageView
171                         android:layout_width="fill_parent"
172                         android:layout_height="fill_parent"
173                         android:id="@+id/btn_Graficar"
174                         android:layout_gravity="center"
175                         android:layout_weight="1" />
176
177                 <LinearLayout
178                     android:orientation="horizontal"
179                     android:layout_width="match_parent"
180                     android:layout_height="match_parent"
181                     android:layout_weight="2"></LinearLayout>
182     </LinearLayout>
183 </FrameLayout>
184 </FrameLayout>
185 </FrameLayout>
186 </FrameLayout>
187 </FrameLayout>
188 </FrameLayout>
189 </LinearLayout>
190

```

Código 46. Código XML para para selección de intervalos.

La complejidad de esta pantalla [Figura 54], tiene un nivel bajo, sin embargo, la importancia de la misma impacta sobre una característica con gran importancia dentro de toda la aplicación del multímetro digital, como lo es la representación visual de las mediciones mediante una gráfica.

Como se mencionó con anterioridad la pantalla 4 tiene el objetivo de configurar los parámetros de graficación. Cuenta con un par de radio buttons con los cuales se estipula la escala de los intervalos de tiempo permitiendo elegir entre dos opciones, segundos y mili segundos. También mediante un input de valores numéricos se definen en los intervalos en los que se tomará el valor de la medición que se está haciendo.

Para el multímetro digital se ha establecido como valor mínimo de intervalos de tiempo 5 mili segundos. Esto debido a que la representación de la gráfica es tan potente que, al tomar valores de intervalos de tiempo menores al establecido, provoca una representación veloz de la gráfica en cualquier dispositivo móvil, complicado así, la interpretación del fenómeno o medición que se está realizando.

Por último, en la parte inferior de la pantalla [Figura 54] se encuentra un botón que mediante en un principio estipula la escala y el intervalo de tiempo en la clase MyService mediante el método `myService.setUnidades_escala(Float unidadTiempo, String tipoEscala)`, una vez que los parámetros fueron estipulados, un intent lanza la actividad de la pantalla de graficación como se muestra a continuación:

```
Intent Multimetro_Graficar = new Intent(Parametros.this, MultimetroGrafica.class);  
  
startActivity(Multimetro_Graficar);}
```

**Código 47. Intent que lanza nueva actividad.**

#### **4.2.11 Prueba / integración “pantalla 4” - iteración 1.**

En esta pantalla se permite seleccionar dos medidas de tiempo diferentes, en milisegundos y en segundos, si la aplicación detecta que se están ingresando menos de 5 milisegundos, enviará un mensaje al usuario para que se ingresen valores mayores [Figura 55]. Esto debido a que la obtención y envío de datos no es tan rápida.

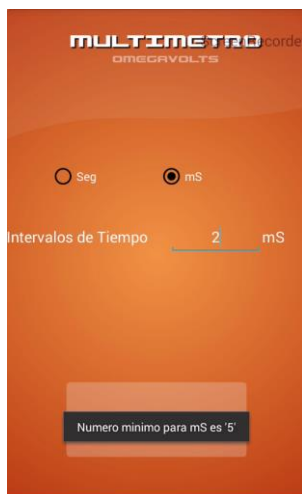


Figura 55. Prueba pantalla 3.

#### 4.2.12 Análisis y requerimientos “pantalla 5” - iteración 1.

En este apartado se describen los requerimientos para la pantalla 5 de acuerdo a el diseño la [Figura 35].

Descripción de la Aplicación Móvil:	<b>Aplicación móvil para dispositivos con sistema operativo Android “pantalla 5”.</b>		
Objetivo:	Diseñar y desarrollar la interfaz y funcionamiento de la “pantalla 5”.		
Requerimientos “pantalla 5” - iteración 1			
No.	Descripción	Funcional	No Funcional
1	Graficar mediciones en tiempo real		✓
2	Botón de play y pause	✓	
3	La gráfica debe poder reiniciar	✓	
4	La gráfica debe permitir hacer zoom	✓	

Tabla 10. Requerimientos “pantalla 5” - iteración 1.

#### 4.2.13 Diseño “pantalla 5” - iteración 1.

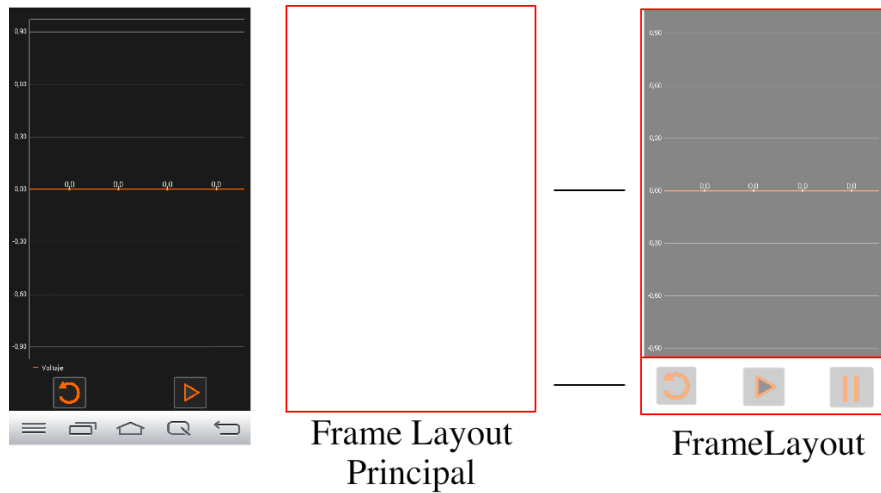


Figura 56. Diseño de Pantalla 5.

#### 4.2.14 Codificación o ensamble “pantalla 5” - iteración 1.

Para el desarrollo de esta pantalla se utiliza *Android Chart*, una librería no incluida en Android Studio que permite desplegar datos en forma de gráficos, para que pueda ser utilizada necesita agregarse el archivo .jar al directorio *libs* [Figura 57].

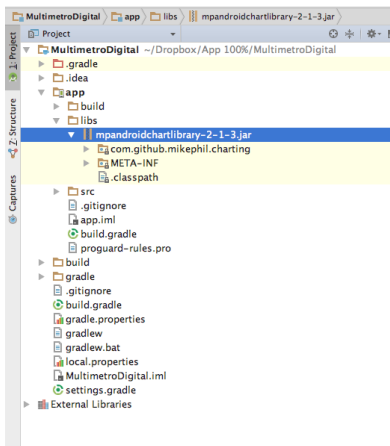


Figura 57. Agregado de librería Android Chart.

Después de agregar la librería, en el archivo .xml de la actividad para utilizar los componentes se utiliza un encabezado distinto que hace referencia a la librería. Los componentes como los Layouts o los botones se siguen manejando de manera normal.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3  android:orientation="vertical" android:layout_width="match_parent"
4  android:layout_height="match_parent">
5
6  <LinearLayout
7  android:orientation="horizontal"
8  android:layout_width="match_parent"
9  android:layout_height="match_parent"
10 android:layout_weight="1">
11
12     <com.github.mikephil.charting.charts.LineChart
13     android:id="@+id/chart1"
14     android:layout_width="match_parent"
15     android:layout_height="match_parent" />
16 </LinearLayout>
17
18 <LinearLayout
19 android:orientation="horizontal"
20 android:layout_width="match_parent"
21 android:layout_height="match_parent"
22 android:layout_weight="10"
23 android:background="#1a1a1a">
24
25     <LinearLayout
26     android:orientation="horizontal"
27     android:layout_width="match_parent"
28     android:layout_height="match_parent">
29
30         <ImageView
31         android:layout_width="wrap_content"
32         android:layout_height="wrap_content"
33         android:id="@+id/imgV_Reset"
34         android:layout_weight="1" />
35
36         <ImageView
37         android:layout_width="wrap_content"
38         android:layout_height="wrap_content"
39         android:id="@+id/imgV_PlayPause"
40         android:layout_weight="1" />
41     </LinearLayout>
42 </LinearLayout>
43
44 </LinearLayout>

```

**Código 48.** XML para grafica de mediciones.

Una vez que se han incluido todos los elementos de la parte visual, la siguiente parte es modificar la funcionalidad de los componentes en el archivo .java. Para quitar características que vienen por default en la librería o habilitar otras es necesario hacer lo al iniciar en el método *Bundle*.

```

43  @Override
44  protected void onCreate(Bundle savedInstanceState) {
45      super.onCreate(savedInstanceState);
46      setContentView(R.layout.activity_multimetro_grafica);
47      getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN, WindowManager.LayoutParams.FLAG_FULLSCREEN);
48
49      Intent intent = new Intent(getBaseContext(), MyService.class);
50      bindService(intent, mConnection, Context.BIND_AUTO_CREATE);
51
52
53      mHandler.obtainMessage(0).sendToTarget();
54
55      grafica = (LineChart) findViewById(R.id.chart1);
56      grafica.setDescription("");
57      grafica.setNoDataTextDescription("No hay datos por el momento");
58      grafica.setDrawGridBackground(false);
59      grafica.setBackgroundColor(Color.rgb(26, 26, 26));
60      grafica.getAxisLeft().setStartAtZero(!grafica.getAxisLeft().isStartAtZeroEnabled());
61      grafica.getAxisRight().setStartAtZero(!grafica.getAxisRight().isStartAtZeroEnabled());
62      grafica.setAutoScaleMinMaxEnabled(true);
63
64      LineData datos = new LineData();
65      grafica.setData(datos);
66
67      Legend l = grafica.getLegend();
68      l.setForm(Legend.LegendForm.LINE);
69      l.setTextColor(Color.WHITE);
70
71      YAxis y1 = grafica.getAxisLeft();
72      y1.setTextColor(Color.WHITE);
73      y1.setDrawLimitLinesBehindData(true);
74
75      XAxis x1 = grafica.getXAxis();
76      x1.setTextColor(Color.WHITE);
77      x1.setDrawGridLines(false);
78
79      YAxis y2 = grafica.getAxisRight();
80      y2.setEnabled(false);
81  }

```

#### Código 49. Configuración básica gráfica.

Se utilizan dos métodos que sirven para el almacenamiento de datos obtenidos y para el lanzamiento de los mismos en pantalla. En esta parte se implementa un hilo dentro Handler, el cual estará conectado a la clase MyService revisando los parámetros como intervalo y si se detiene el envío de los mismos.

El primer método (*addEntery()*), inicia revisando si existen algún tipo de características para gráficos utilizando el método *y* y si no los hay, son agregados con el método *createSet()*. Posteriormente una variable de tipo servicio revisa si hay los datos que se reciben del bluetooth son nulos, y de ser así manda un 0, esto para evitar errores de tipo de dato.

```

84 public void addEntry() {
85     final LineData datos = grafica.getData();
86
87     if (datos != null) {
88         LineDataSet set = datos.getDataSetByIndex(0);
89         if (set == null) {
90             datos.addDataSet(set);
91             set = createSet();
92             datos.addDataSet(set);
93         }
94
95         datos.addXValue("");
96
97         if (myService.mensajeRecibido() == null) {
98             num = "0";
99         } else {
100            num = myService.mensajeRecibido();
101        }
102
103        n = Double.parseDouble((num));
104        datos.addEntry(new Entry((float) (n), set.getEntryCount(), 0));
105
106        grafica.notifyDataSetChanged();
107        //grafica.setVisibleXRange(1, 10);
108        grafica.moveViewToX(datos.getXValCount());
109    }
110    imgV_PlayPause = (ImageView) findViewById(R.id.imgV_PlayPause);
111    imgV_PlayPause.setImageResource(R.drawable.grafpausa);
112    imgV_PlayPause.setScaleType(ImageView.ScaleType.CENTER_INSIDE);
113
114    imgV_PlayPause.setOnClickListener((view) -> {
115        if (stop == false) {
116            imgV_PlayPause.setImageResource(R.drawable.grafplay);
117            stop = true;
118        } else if (stop == true) {
119            imgV_PlayPause.setImageResource(R.drawable.grafpausa);
120            stop = false;
121        }
122    });
123
124
125

```

**Código 50. Obtención de valores recibidos por bluetooth.**

```

149 private LineDataSet createSet() {
150     LineDataSet set = new LineDataSet(null, myService.getTipo_grafica());
151     set.setDrawHighlightIndicators(false);
152     set.setDrawCubic(true);
153     set.setCubicIntensity(0.05f);
154     set.setAxisDependency(YAxis.AxisDependency.LEFT);
155     set.setColor(Color.rgb(255, 102, 0));
156     set.setCircleColor(Color.rgb(255, 102, 0));
157     set.setLineWidth(2f);
158     set.setCircleSize(2f);
159     set.setValueTextColor(Color.WHITE);
160
161     return set;
162 }

```

**Código 51. Envío de valores a pantalla para graficarlos.**

Como se observa en el código anterior los botones hacen uso de una variable llamada *stop*, que está dentro del hilo de ejecución que lanza la gráfica, si se presiona el botón stop la variable detiene el hilo hasta que el botón play o reiniciar sean presionados.



```

179 public Handler mHandler = handleMessage(msg) -> {
182     super.handleMessage(msg);
183
184     switch (msg.what) {
185
186     case 0:
187
188         new Thread((Runnable) () -> {
189             while (hilo) {
190                 runOnUiThread() -> {
191                     if (stop == false) {
192                         if (myService.estadoSocket() == 3) {
193                             finish();
194                         } else {
195                             addEntry();
196                         }
197                     }
198                 }
199             }
200         });
201
202         try {
203             Thread.sleep(myService.intervalos()); // cada cuanto tiempo lee
204         } catch (InterruptedException e) {
205             e.printStackTrace();
206         }
207     }
208 }
209 }.start();
210
211 break;
212
213
214
215 };

```

**Código 52. Código para iniciar y detener la gráfica.**

En el código anterior se observa que dentro del hilo se encuentra la llamada al método `addEntry ()`, esto ya que la variable `stop` es definida por el estado de los botones `pause` y `play` resultando en la detención o inicio del hilo de ejecución. Cabe destacar que este hilo de ejecución es creado de manera separa del hilo principal lo que ayuda a un mejor procesamiento de la información.

#### 4.2.15 Prueba / integración “pantalla 5” - iteración 1.

En la Figura 58 se observa que la gráfica tiene el comportamiento deseado y los valores obtenidos son los deseados.



**Figura 58. Prueba gráfica pantalla 5.**

## 5. Desarrollo de “circuito medidor”

Al igual que en el bloque anterior, se crea una del bloque 1 (Circuito medidor) [Figura 34].

### 5.1 Análisis y requerimientos de “circuito medidor” - iteración 1.

Descripción de la Aplicación Móvil:	<b>Desarrollo de un circuito medidor de voltaje, resistencia y temperatura.</b>		
Objetivo:	Diseñar y ensamblar un circuito que permita hacer mediciones de voltaje, resistencia y temperatura.		
Requerimientos para desarrollo de “ circuito medidor ” - iteración 1			
No.	Descripción	Funcional	No Funcional
1	Diseñar y ensamblar un circuito que permita hacer mediciones de voltaje.		✓
2	Diseñar y ensamblar un circuito que permita hacer mediciones de resistencia.		✓
3	Diseñar y ensamblar un circuito que permita hacer mediciones de temperatura.		✓

Tabla 11. Requerimientos para desarrollo de “circuito medidor” - iteración 1.

### 5.2 Diseño de “circuito medidor” - iteración 1.

Para el diseño del circuito primero se realiza un análisis de las características con las que cuenta Arduino para la lectura de datos analógicos. Existen algunas limitantes que hacen que el diseño del circuito sea dirigido únicamente para Arduino.

Entre las limitantes que tiene Arduino es que su CAD va desde cero Volts hasta 5 Volts, por lo que, para hacer mediciones de mayor voltaje, se use un divisor de voltaje, teniendo en cuenta que el CAD es de 10 bits, y el voltaje soportado por arduino es 5v la resolución quedaría de la siguiente manera [Figura 59].

## Convertidor Analógico/Digital de 10 Bits

10	9	8	7	6	5	4	3	2	1
512	256	128	64	32	16	8	4	2	1

Total de códigos digitales = 1024

Para una entrada de 5v la resolución es:

$$\frac{5v}{1024} = 0.0048828125 v = 4.8 mv$$

Para una entrada de 50v la resolución es:

$$\frac{50v}{1024} = 0.048828125 v = 48 mv$$

Figura 59. Convertidor analógico / digital de 10 Bits.

Complementando lo anterior, para hacer mediciones de voltaje mayores a 5 volts, se establecer un voltaje máximo de entrada, en este caso 50 volts para después calcular para los valores de las resistencias de un divisor de voltaje que permita un voltaje de salida de 0 a 5V. La fórmula para calcular un divisor de voltaje es la siguiente [Figura 60]:

$$V_{Salida} = V_{Entrada} * \frac{R_2}{R_1 + R_2} \quad (24)$$

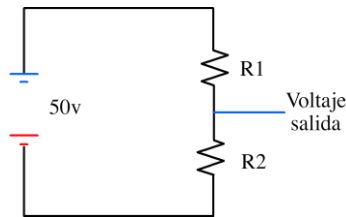


Figura 60. Divisor de voltaje.

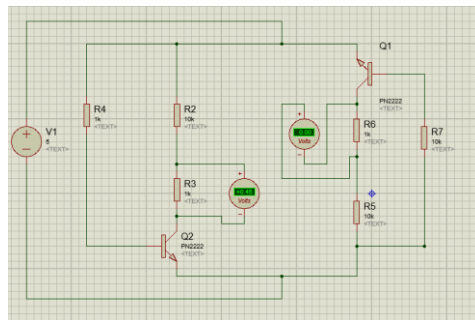
Siguiendo la fórmula anterior y tomando el voltaje de salida como 5 volts, un voltaje de entrada de 50 volts y la resistencia R2 es igual a 1 KΩ, los cálculos para la resistencia R1 quedan de la siguiente manera.

$$5v = 50v * \frac{1 K}{R_1 + 1 K} \quad (25)$$

$$R_1 = \frac{1 K}{\frac{5v}{50v}} - 1 K = 9K\Omega \quad (26)$$

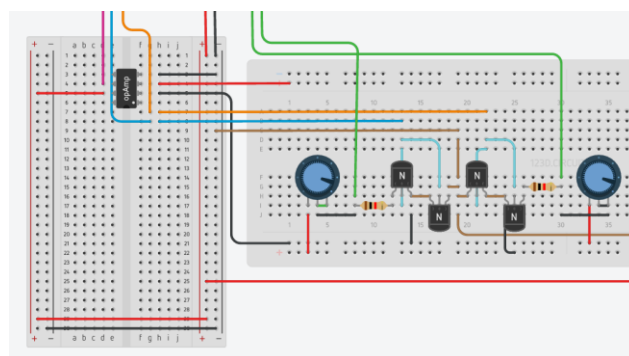
Por lo tanto, para medir un máximo de 50 V y que a la salida se tengan valores de 0-5 V, se utiliza una resistencia de  $1\text{K}\Omega$  y una de  $9\text{K}\Omega$  para el divisor. Hay que tener en cuenta el divisor de voltaje al momento de programar el cálculo del voltaje y multiplicar el valor leído de 0-5 volt la resolución de los 50 volts, que es 48 mV.

Otro aspecto importante que se debe tener en cuenta al momento de diseñar un circuito que va a ser leído por Arduino es que Arduino no lee valores negativos por lo que se propone el siguiente diseño [Figura 61].



**Figura 61. Circuito medidor de voltaje positivo y negativo (esquema eléctrico).**

Para lograr detectar que líneas son positivas y negativas se utiliza un amplificador operacional como comparador. De la misma manera, para medir la resistencia no es necesario utilizar un divisor, ya que de acuerdo al diseño de la [Figura 62], se usan los 5 volts de Arduino para realizar las mediciones correspondientes.



**Figura 62. Circuito medidor de voltaje positivo y negativo (esquema de componentes).**

Por último, para realizar la medición de temperatura se hace uso del sensor LM35 el cual se conecta de la siguiente manera de acuerdo al datasheet [Figura 63].



## 6.1 Análisis y requerimientos de programa en “tarjeta Arduino” - iteración 1.

Descripción de la Aplicación Móvil:	<b>Programa para adquisición y envío de información usando Arduino</b>		
Objetivo:	Diseñar y desarrollar un programa que permita adquirir datos de un circuito medidor y enviarlos a la aplicación Android usando bluetooth.		
Requerimientos para programa en “tarjeta Arduino” - iteración 1			
No.	Descripción	Funcional	No Funcional
1	Crear el código necesario para leer los datos analógicos que mida el circuito medidor.		✓
2	Crear el código necesario que permita el envío y recepción de datos con el módulo bluetooth.		✓
3	Configurar módulo bluetooth HC -05 para el correcto envío y recepción de datos.		✓

Tabla 12. Requerimientos para programa en “tarjeta Arduino” - iteración 1.

## 6.2 Diseño de programa en “tarjeta Arduino” - iteración 1.

La programación para Arduino puede contar con una parte de entrada de datos a través de la interfaz, sin embargo, para este proyecto las entradas son proporcionadas por el circuito medidor. Por esto, solo se realiza el diagrama de flujo para lectura de datos de un puerto serial, dicho puerto pertenece al módulo bluetooth HC-05.

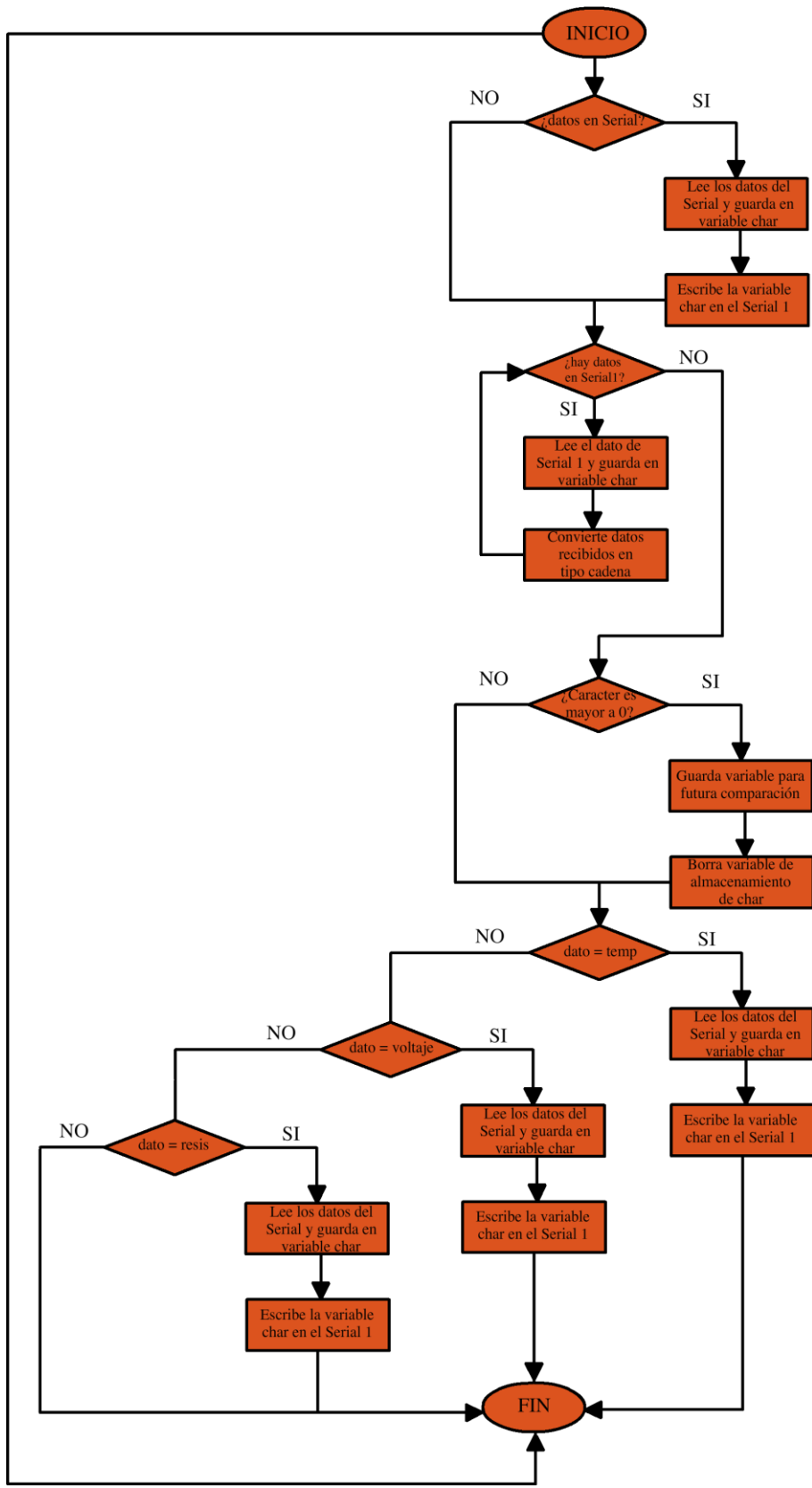


Figura 65. Diagrama de Flujo programa en Arduino.

### 6.3 Codificación o ensamble de programa en “tarjeta Arduino” - iteración 1.

Arduino utiliza su propio IDE de desarrollo en alto nivel, esto ayuda a que la programación sea más sencilla. A continuación, se muestra el código escrito para la lectura y escritura de los datos de medición.

```
15 void loop() {
16
17   if (Serial.available()) { //si hay datos disponibles en el puerto serial
18     delay(3);
19     char serial = Serial.read(); // cada dato se convierte en caracter para ser enviado de manera serial
20     Serial1.write(serial); //se manda o escribe el caracter o conjunto de caracteres
21     Serial.print(serial); // se imprime en el monitor serial pero no es muy necesario
22   }
23
24   while (Serial1.available()) { //mientras haya datos disponibles en en puerto Serial1
25     delay(3);
26     char serial1 = Serial1.read(); //la variable serial1 lee un caracter
27     readString += serial1; // si aún hay disponibles hace una cadena con los caracteres nuevos
28   }
29
30   if (readString.length() > 0) { // una vez que no hay datos disponibles en el puerto Serial1 y readString es mayor a 0
31     Serial.println(readString); // se imprime la cadena recibida en el monitor serial
32     dato_compara = readString; // el dato se guarda en otra variable
33     readString = ""; //se limpia la variable y si hay nuevos datos no se sobre escriba
34   }
35
36   if (dato_compara == "temperatura") { // se puede usar un case y usar enteros en lugar de strings pero se compara el valor que llega del Serial1
37     Serial.println(analogRead(2) * 4.9 / 10); // se sigue la formula para sacar la temperatura
38     Serial1.println((analogRead(2) * 4.9) / 10);
39     delay(200);
40   } else if (dato_compara == "voltaje directo") {
41     if (digitalRead(7) == LOW) {
42       negativo = 1;
43     } else {
44       negativo = 0;
45     }
46
47     if (Contador <= 5) {
48       SumaVoltaje = SumaVoltaje + (analogRead(negativo) * (.048367546));
49       Contador = Contador + 1;
50       delay(150);
51     } else {
52       PromedioVoltaje = ((SumaVoltaje / 5));
53       if (PromedioVoltaje > .2) {
54         PromedioVoltaje = PromedioVoltaje - .2;
55       }
56       if (digitalRead(7) == LOW) {
57         PromedioVoltaje = PromedioVoltaje * (-1);
58       }
59
60       Serial.println(PromedioVoltaje);
61       Serial1.println(PromedioVoltaje);
62       Contador = 0;
63       SumaVoltaje = 0;
64     }
65   } else if (dato_compara == "resistencia") {
66     Serial.println((((1000) / (analogRead(1)*.00488)) * 5) - 1000);
67     Serial1.println((((1000) / (analogRead(1)*.00488)) * 5) - 1000);
68     delay(200);
69   }
70 }
```

Código 53. Código Arduino para envío y recepción de datos a través de bluetooth.

Como se puede ver en el código para verificar datos disponibles en un puerto serial se utiliza el método *available()*, y una vez que este termina continua con las siguientes líneas de código.



## **7 Modelo CANVAS.**

Como parte importante en el desarrollo de este proyecto, se participa en el 14 Concurso del universitario emprendedor, donde se hace el análisis de este proyecto basados en el Modelo Canvas, el cual es descrito en los siguientes bloques.

### **7.1 Segmento de mercado**

OMEGAVOLTS ofrece una contra segmentación en el mercado el cual está integrado por profesionistas, técnicos y estudiantes que utilizan la herramienta de Desarrollo Arduino, además que requieran hacer mediciones eléctricas en las áreas de:

- Electricidad
- Electrónica
- Computación

Por tanto, es un elemento vital en diseño e implementación de proyectos eléctricos o electrónicos.

### **7.2 Propuesta de valor**

OMEGAVOLTS es una nueva iniciativa que ofrecerá al mercado un multímetro digital, el cuál es un instrumento básico para realizar mediciones eléctricas. Implementado en dispositivos móviles con sistema operativo Android acompañado de un shield diseñado para la tarjeta Arduino.

Ofrece las mediciones eléctricas básicas (Voltaje, Corriente, Continuidad y Resistencia) además de mediciones complementarias como temperatura gracias al sensor LM35 y una punta lógica para mediciones digitales. Para cada una de estas mediciones eléctricas es posible visualizar su comportamiento de manera gráfica en tiempo real.

Existen diferentes dispositivos en el mercado similares al “multímetro digital” los cuales comparten la estructura que consta de una aplicación móvil, un dispositivo hardware y la comunicación entre estos mediante tecnología bluetooth. Productos como METRAHIT

ULTRA, Keysight Mobilt Meter y Digitech DMM que no son tecnología mexicana y requieren de un hardware costoso por encima de los \$1,000.00 M.N., además comparados con el “Multímetro digital” estos carecen de una interfaz intuitiva y amigable para el usuario.

Cabe resaltar que sólo 2 de estos permiten realizar gráficas de las mediciones eléctricas como lo hace “Multímetro digital” y ninguno mide temperatura.

Además, que Omegavolts dirige sus servicios a aquellas personas que utilizan tarjetas Arduino.

### **7.3 Canales de distribución y comunicación.**

#### **7.3.1 Etapa de información**

Se mostrarán los beneficios del proyecto en la en la etapa de introducción al mercado, en instituciones educativas, de la Ciudad de México y área Metropolitana, que tengan carreras afines a la electrónica por medio de visitas domiciliadas en las que se ofrecerá información apoyada por medios publicitarios.

#### **7.3.2 Etapa de evaluación**

Se ayudará a los clientes potenciales a evaluar el multímetro como mejor opción, al mostrar las bondades o beneficios que ofrece el mismo, en el compromiso educativo y ejercicio profesional y de investigación.

#### **7.3.3 Etapa de compra**

La manera de conocer los productos que ofrece la empresa OMEGAVOLTS es por medio de una página web que contendrá apartados en los que se encontrará información de productos, cotizaciones, ventas, soporte y contacto. Donde el usuario puede verificar precios, así como el domicilio donde puede adquirir el producto de OMEGAVOLTS.

#### **7.3.4 Etapa de entrega**

La aplicación estará en la Play Store para que cualquier persona con dispositivo Android pueda descargarla, Mientras que el Hardware una vez comprado en el punto de venta del

distribuidor autorizado y pagado en su totalidad, será entregado en un empaque de cartón duro, el cual contendrá una bolsa anti estática con el hardware dentro.

### 7.3.5 Etapa de posventa

OMEGAVOLTS, ofrecerá servicio técnico en línea de un mal funcionamiento del hardware, manuales de uso en PDF descargables en el apartado de soporte de la página web, actualizaciones de la aplicación en la play store. Al entregar el producto se tendrá una garantía y una guía de inicio rápido.

## 7.4 Relación con el cliente

La relación que busca OMEGAVOLTS en sus clientes es de confianza por los servicios adquiridos y la calidad de sus actualizaciones, como medio de promoción ofrecerá:

### 7.4.1 Estrategias de promoción

#### 7.4.1.1 La muestra

Consiste en regalar talleres de exhibición en instituciones del área, con la finalidad de mostrar su aplicación.

#### 7.4.1.2 Paquetes de precio global

En esta estrategia estarán incluidos el hardware con la aplicación totalmente gratis.

## 7.5 Recursos clave

### 7.5.1 Recursos físicos

Partida	Nombre	Características	Marca
Inmobiliario			
1	Oficina	4mx4m, C.U. UAEM Ecatepec Incubadora de Empresas.	UAEM
2	Computadora	Intel Core i3-32717U, 6Gb DDR 3, 100 GB HDD,15.6 HD Multi-touch LCD, Intel HD Graphics 4000, 802.11b/g/n + BT, 4-cell Li-ion battery.	Gateway NV570P02m

3	Escritorios	A acabado en madera y finish foil, 2 cajones, 75.5 cm de alto, 120 cm de ancho y 45 cm de profundidad.	Escritorio Politorno Castaño
<b>Servicios</b>			
4	Telefonía e Internet	100 Llamadas locales, Internet de banda ancha hasta 5 Mbps.	Paquete 333 de Telmex
5	Luz	150 KWH al mes	CFE
6	Multímetro	Dimensiones generales de 1.2m de largo x 0.52 m de ancho y alto 0.75m , collarete de 5 hilos, motor de ½ hp, alimentación trifásica 220 Volts	HOSEKI HSK500B-02
<b>Recursos de producción</b>			
7	Resistencias	Resistencias de 10k, 1M,1k de ¼ de watt	STEREN
8	Capacitor	Capacitor lenteja de 1mf,10mf	STEREN
9	Puntas para multímetro	Puntas para multímetro universal MUL-510	STEREN
10	Placas de Cobre	Placa de cobre de 5x5cm	STEREN
11	Cloruro Férrico	Botella 930 ml	STEREN
12	Placa Arduino	Placa Arduino Mega 2560 16 canales Analógicos 8KB de RAM Voltajes de salida de 3.3 y 5 Volts	ARDUINO

Tabla 13. Recursos físicos.

### 7.5.2 Recursos intelectuales.

Los recursos intelectuales se definen en las siguientes categorías:

- Programación.
  - Programación en Android.
    - Lenguaje JAVA.
      - Comunicación Bluetooth.
      - Servicios Background.
    - XML.
      - Interfaces.
      - Maquetación.
  - Programación en Arduino.

- Lenguaje C++.
    - Comunicación Bluetooth.
    - Entradas Analógicas.
    - Estructuras de control.
- Electrónica
  - Circuitos Electrónicos.
    - Divisores de Voltaje.
    - Sensor LM35.
    - Sensor o circuito medidor de Corriente.
    - Circuito medidor de continuidad.
    - Diseño de PCB.
    - Elaboración circuitos en placas fenólicas.
- Diseño gráfico.
  - Teoría del color.
  - Elementos de Navegación.
  - Sencillez de la interfaz.
  - Tipo de presentación (gráfica, audio, texto).
- Bluetooth hc-06
  - Configuración de comandos AT.
  - Conexión con placa Arduino.

### **7.5.3. Recursos humanos**

El personal operativo idóneo para el buen funcionamiento del negocio.



Función de puestos:

- Administrador general: Encargado de las relaciones con los clientes, proveedores, firma de contratos.
- Asistente: Encargado de la documentación necesaria para declaración de impuestos, derechos de autor del software, diseño de objetivos, metas y procedimientos.
- Ingeniero en computación: Encargado del desarrollo del software necesario para el correcto funcionamiento de la interfaz.
- Ingeniero en electrónica: Encargado del desarrollo y armado de los circuitos necesarios para la medición de las diferentes magnitudes eléctricas.
- Diseñador: Encargado de desarrollo de la imagen de la interfaz de cada pantalla y logos.

#### **7.5.4 Recursos financieros**

Los recursos financieros que OMEGAVOLTS requiere para el desarrollo del producto provienen principalmente de los socios de la empresa.

En una segunda instancia y con la misma importancia la empresa debe ser apoyada con recursos financieros ajenos que al ser estratégicamente elegidas deben fungir en pro del

crecimiento de OMEGAVOLTS. Estos recursos deben provenir desde iniciativas dirigidas a emprendedores como lo son:

- UAEM
  - XIV Concurso Universitario Emprendedor
  - Incubadoras de empresas
- Gobierno del Estado de México
  - Instituto Mexiquense del Emprendedor
- CONACYT
  - Fondo Emprendedores CONACYT-NAFIN
- Convocatorias lanzadas por el INADEM
  - Inscribirse en la convocatoria adecuada
  - Contar con RFC y FIEL y la documentación del proyecto señalada en la convocatoria.
  - Ingresar solicitud de apoyo.
  - Los recursos se repartirán de acuerdo a las calificaciones dadas por los expertos (Entre más alta la clasificación se toma más en cuenta el proyecto y al emprendedor).
- Consejo Nacional de Comunicación
  - Iniciativa Pepe y Toño

Las modalidades de apoyo antes mencionadas, facilitan recursos y permiten acceder a capital con otros inversionistas, para desarrollar y consolidar a OMEGAVOLTS con alto valor agregado. Adicionalmente aportan asesoría tecnológica, financiera y legal para fortalecer la posición competitiva en el largo plazo de OMEGAVOLTS.

## **7.6 Actividades clave**

Las actividades claves y procesos que OMEGAVOLTS estipula para la producción del Shield, se describen a continuación:

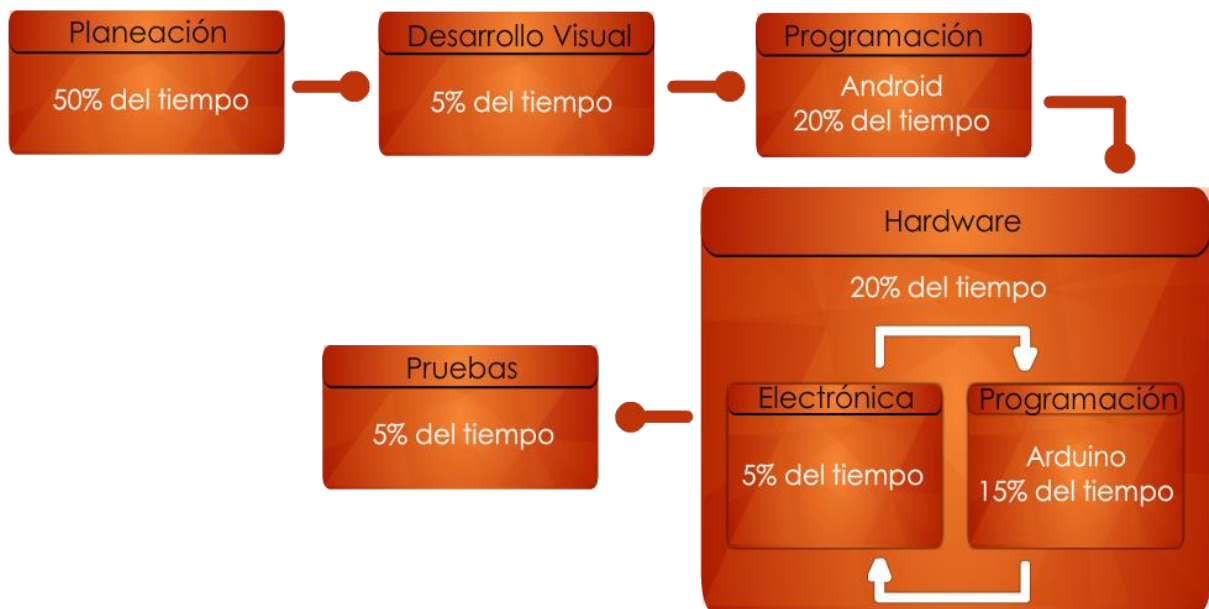
- **Planeación global del proyecto.**
  - Fundamentar idea

- Seleccionar y definir segmento de mercado
- Estipular requerimientos
- Diseño Visual
  - Elegir estilo
  - Colores acordes a la aplicación
  - Boceto
- 1.5.- Diseño aplicación móvil
  - Mapa de navegación
  - Estructuras independientes
  - Comunicación
  - Compatibilidad de objetos con Arduino
- Diseño aplicación en Arduino
  - Maquetado
  - Comunicación
  - Estructuras independientes
  - Compatibilidad de objetos con Android
- Diseño circuito eléctrico
  - Análisis por módulo
  - Cálculos numéricos
    - Definir estructura
    - Definir componentes
  - Simulación
- Elaboración de Shield
  - Diseño de PCB
  - Armado de Shield.
- **Desarrollo visual**
  - Desarrollo de boceto
  - Productos con formato y dimensiones especificadas.
  - Prueba
- **Programación Android**
  - Desarrollo de actividades



- Desarrollo de interfaces
- Prueba
- **Programación Arduino**
  - Desarrollo de estructuras
  - Prueba
- **5.- Elaboración Shield**
  - Impresión y elaboración de PCB
  - Barrenado
  - Soldado de componentes
  - Prueba

**6.- Prueba final global del proyecto.**



**Información:**

Se mostrarán los beneficios del proyecto en instituciones que tengan carreras afines a la electrónica por medio de visitas domiciliadas en las que se ofrecerá información apoyada por medios publicitarios.

**Venta:**

Es por medio de una página web que contendrá apartados en los que se encontrará información de productos, cotizaciones, ventas, soporte y contacto. Donde el usuario puede verificar precios, así como hacer pedidos de los diferentes productos o servicios con los que cuenta OMEGAVOLTS.

**Entrega:**

La aplicación estará en la Play Store para que cualquier persona con dispositivo Android pueda descargarla, Mientras que el Hardware una vez comprado en la página de internet y pagada en su totalidad será entregada personalmente, entregado en un empaque de cartón duro, el cual contendrá una bolsa anti estática con el hardware dentro, envuelta en poli burbuja para evitar daños.

**Posventa:**

OMEGAVOLTS, ofrecerá servicio técnico en línea de un mal funcionamiento del hardware, manuales de uso en PDF descargables en el apartado de soporte de la página web, actualizaciones de la aplicación en la Play store.

Al entregar el producto se tendrá una garantía y una guía de inicio rápido.

**7.7 Aliados clave**

La Universidad Autónoma del Estado de México ofrece servicios de incubadoras de empresas, siendo el lugar indicado para que OMEGAVOLTS se consolide.

La incubadora ofrece un entorno lleno de oportunidades potenciales para el éxito de OMEGAVOLTS. La alianza ofrece únicamente beneficios para la empresa comprendidos

por asesorías legales, financieras, mercadológicas y oportunidades reales en el mercado como lo es el XIV Concurso Universitario Emprendedor.

Los recursos físicos necesarios para la producción y denominados componentes discretos son adquiridos directamente con la comercializadora de productos electrónicos STEREN y esto lo convierte en un proveedor esencial para OMEGAVOLTS.

Se busca crear una alianza que comprende en ceder la exclusividad con la compra de dichos componentes y al mismo tiempo ser el lugar donde OMEGAVOLTS pueda ofrecer el Shield aprovechando el posicionamiento que STEREN tiene en la república, contemplando las políticas necesarias requeridas para convertirse en proveedor de la comercializadora.

#### **7.7.1 Alianza estratégica**

La alianza principal debido al enfoque del producto que OMEGAVOLTS ofrece es con Arduino. Existen diversos Shields en el mercado enfocados para Arduino, pero no hay ninguno como el Shield de multímetro digital de OMEGAVOLTS.

Los alcances de la comunidad de Arduino se extiende en todo el mundo que, traducido en otras palabras, OMEGAVOLTS tiene un mercado extenso ya consolidado. Se debe tener en claro limitar el mercado con el objetivo de cumplir con el cliente debido a que OMEGAVOLTS no cuenta con la infraestructura de cubrir un mercado tan extenso.

#### **7.7.2 Relación proveedor comprador**

Al delimitar el mercado y tener una estructura sólida se es posible cubrir con las demandas debido a que el tiempo de producción es rápido.

### **7.8 Fuentes de ingresos**

#### **7.8.1 Costo unitario**

Se determinaron sus costos fijos mensuales considerando que se producirán 50 voltímetros por mes. Los costos variables incluyen los materiales que se utilizarán, así como los sueldos de técnicos y obreros que intervendrán en su fabricación. Las sumas de ambos proporcionan el costo unitario de \$164.84

## 7.8.2 Precio de venta unitario

Sobre el costo unitario se definió un 20% de utilidad de \$32.97 para determinar el precio de venta unitario, quedando en \$197.81. Se comentó que no existe competencia por lo que se determinó que el precio de venta unitario es competitivo.

## 7.8.3 Flujo de ingresos

En base al precio de venta unitario y con la venta de 27 voltímetros por mes, se elaboró la proyección de los ingresos por un año en tres escenarios: El conservador, el optimista y el pesimista. En los tres se determinaron flujos de efectivo positivos, por lo que el proyecto se observa que será rentable.

FLUJO DE EFECTIVO MENSUAL (CONSERVADOR)													
Proyecto: OMEGA													
Periodo: Enero-Diciembre 2016													
CONCEPTO:	Mes 1	Mes 2	Mes 3	Mes 4	Mes 5	Mes 6	Mes 7	Mes 8	Mes 9	Mes 10	Mes 11	Mes 12	TOTAL
<b>Saldo Inicial</b>	\$ -	\$ 1,758	\$ 3,516	\$ 5,274	\$ 7,032	\$ 8,790	\$ 10,548	\$ 12,306	\$ 14,064	\$ 15,822	\$ 17,580	\$ 19,338	\$ -
<b>Ingresos:</b>													
Ventas Producto 1	\$ 10,000	\$ 10,000	\$ 10,000	\$ 10,000	\$ 10,000	\$ 10,000	\$ 10,000	\$ 10,000	\$ 10,000	\$ 10,000	\$ 10,000	\$ 10,000	\$ 120,000
<b>Total de ingresos</b>	\$ 10,000	\$ 10,000	\$ 10,000	\$ 10,000	\$ 10,000	\$ 10,000	\$ 10,000	\$ 10,000	\$ 10,000	\$ 10,000	\$ 10,000	\$ 10,000	\$ 120,000
<b>Egresos:</b>													
Sueldos	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 4,800
Luz	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 4,668
Teléfono	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 24,000
Renta	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 1,200
Agua	\$ 5,353	\$ 5,353	\$ 5,353	\$ 5,353	\$ 5,353	\$ 5,353	\$ 5,353	\$ 5,353	\$ 5,353	\$ 5,353	\$ 5,353	\$ 5,353	\$ 64,236
Materia Prima													\$ -
Gas													\$ -
Materiales de Limpieza													\$ -
Accesorios y lubricantes													\$ -
<b>Total de egresos</b>	\$ 8,242	\$ 8,242	\$ 8,242	\$ 8,242	\$ 8,242	\$ 8,242	\$ 8,242	\$ 8,242	\$ 8,242	\$ 8,242	\$ 8,242	\$ 8,242	\$ 98,904
<b>Flujo Neto (Efectivo Disp.)</b>	\$ 1,758	\$ 1,758	\$ 1,758	\$ 1,758	\$ 1,758	\$ 1,758	\$ 1,758	\$ 1,758	\$ 1,758	\$ 1,758	\$ 1,758	\$ 1,758	\$ 21,096
<b>Saldo Final</b>	\$ 1,758	\$ 3,516	\$ 5,274	\$ 7,032	\$ 8,790	\$ 10,548	\$ 12,306	\$ 14,064	\$ 15,822	\$ 17,580	\$ 19,338	\$ 21,096	\$ 21,096

Tabla 14. "Flujo de efectivo mensual (conservador)"

FLUJO DE EFECTIVO MENSUAL (OPTIMISTA)													
Proyecto: OMEGA													
Periodo: Enero-Diciembre 2016													
CONCEPTO:	Mes 1	Mes 2	Mes 3	Mes 4	Mes 5	Mes 6	Mes 7	Mes 8	Mes 9	Mes 10	Mes 11	Mes 12	TOTAL
<b>Saldo Inicial</b>	\$ -	\$ 4,026	\$ 8,052	\$ 12,078	\$ 16,104	\$ 20,130	\$ 24,156	\$ 28,182	\$ 32,208	\$ 36,234	\$ 40,260	\$ 44,286	\$ -
<b>Ingresos:</b>													
Ventas Producto 1	\$ 12,000	\$ 12,000	\$ 12,000	\$ 12,000	\$ 12,000	\$ 12,000	\$ 12,000	\$ 12,000	\$ 12,000	\$ 12,000	\$ 12,000	\$ 12,000	\$ 144,000
Total de ingresos	\$ 12,000	\$ 12,000	\$ 12,000	\$ 12,000	\$ 12,000	\$ 12,000	\$ 12,000	\$ 12,000	\$ 12,000	\$ 12,000	\$ 12,000	\$ 12,000	\$ 144,000
<b>Egresos:</b>													
Sueldos													
Luz	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 4,800
Teléfono	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 4,668
Renta	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 24,000
Agua	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 1,200
Mantenimiento Prima	\$ 5,085	\$ 5,085	\$ 5,085	\$ 5,085	\$ 5,085	\$ 5,085	\$ 5,085	\$ 5,085	\$ 5,085	\$ 5,085	\$ 5,085	\$ 5,085	\$ 61,020
Gas													\$ -
Materiales de Limpieza													\$ -
Aceites y lubricantes													\$ -
Total de egresos	\$ 7,974	\$ 7,974	\$ 7,974	\$ 7,974	\$ 7,974	\$ 7,974	\$ 7,974	\$ 7,974	\$ 7,974	\$ 7,974	\$ 7,974	\$ 7,974	\$ 95,688
Flujo Neto (Efectivo Disp.)	\$ 4,026	\$ 4,026	\$ 4,026	\$ 4,026	\$ 4,026	\$ 4,026	\$ 4,026	\$ 4,026	\$ 4,026	\$ 4,026	\$ 4,026	\$ 4,026	\$ 48,312
<b>Saldo Final</b>	\$ 4,026	\$ 8,052	\$ 12,078	\$ 16,104	\$ 20,130	\$ 24,156	\$ 28,182	\$ 32,208	\$ 36,234	\$ 40,260	\$ 44,286	\$ 48,312	\$ 48,312

Tabla 15 "Flujo efectivo mensual (optimista)"

FLUJO DE EFECTIVO MENSUAL (PESIMISTA)													
Proyecto: OMEGA													
Periodo: Enero-Diciembre 2016													
CONCEPTO:	Mes 1	Mes 2	Mes 3	Mes 4	Mes 5	Mes 6	Mes 7	Mes 8	Mes 9	Mes 10	Mes 11	Mes 12	TOTAL
Saldo Inicial	0	223	446	669	892	1115	1338	1561	1784	2007	2230	2453	0
Ingresos:													
Ventas Producto 1	\$ 9,000	\$ 9,000	\$ 9,000	\$ 9,000	\$ 9,000	\$ 9,000	\$ 9,000	\$ 9,000	\$ 9,000	\$ 9,000	\$ 9,000	\$ 9,000	\$ 108,000
Total de ingresos	\$ 9,000	\$ 9,000	\$ 9,000	\$ 9,000	\$ 9,000	\$ 9,000	\$ 9,000	\$ 9,000	\$ 9,000	\$ 9,000	\$ 9,000	\$ 9,000	\$ 108,000
Egresos:													
Sueldos	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 400	\$ 4,800
Luz	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 389	\$ 4,668
Teléfono	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 2,000	\$ 24,000
Renta	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 100	\$ 1,200
Agua	\$ 5,888	\$ 5,888	\$ 5,888	\$ 5,888	\$ 5,888	\$ 5,888	\$ 5,888	\$ 5,888	\$ 5,888	\$ 5,888	\$ 5,888	\$ 5,888	\$ 70,656
Gas													\$ -
Materiales de Limpieza													\$ -
Actores y lubricantes													\$ -
Total de egresos	\$ 8,777	\$ 8,777	\$ 8,777	\$ 8,777	\$ 8,777	\$ 8,777	\$ 8,777	\$ 8,777	\$ 8,777	\$ 8,777	\$ 8,777	\$ 8,777	\$ 105,204
Flujo Neto (Efectivo Disp.)	\$ 223	\$ 223	\$ 223	\$ 223	\$ 223	\$ 223	\$ 223	\$ 223	\$ 223	\$ 223	\$ 223	\$ 223	\$ 2,676
Saldo Final	\$ 223	\$ 446	\$ 669	\$ 892	\$ 1,115	\$ 1,338	\$ 1,561	\$ 1,784	\$ 2,007	\$ 2,230	\$ 2,453	\$ 2,676	\$ 2,676

Tabla 16. "Flujo de efectivo mensual (Pesimista)"

## 7.9 Estructura de costos

### 7.9.1 Análisis de la propuesta de valor.

OMEGAVOLTS es una nueva iniciativa que ofrecerá al mercado un multímetro digital, el cual es un instrumento básico para realizar mediciones eléctricas. Implementado en dispositivos móviles con sistema operativo Android acompañado de un shield diseñado para la tarjeta Arduino. Ofrece las mediciones eléctricas básicas (Voltaje, Corriente, Continuidad y Resistencia) además de mediciones complementarias como temperatura gracias al sensor LM35 y una punta lógica para mediciones digitales. Para cada una de estas mediciones eléctricas es posible visualizar su comportamiento de manera gráfica en tiempo real.

### 7.9.2 Análisis de la fuente de ingresos

La fuente principal de sus ingresos será por medio de la fabricación y venta de este tipo de

producto, para la elaboración del estudio se consideró un total de sesenta y tres como producción y venta a un precio de venta unitario estimado de 197.81, estas unidades son estimación mensual.

De acuerdo a su proyección los importes monetarios que requieren son:

Concepto	Importe Mensual	Importe Anual
Ingresos proyectado	13,363.00	148,357.00

**Tabla 17. Análisis de la fuente de ingresos.**

### 7.9.3 Estructura de costos

Dentro de las bondades que tiene el modelo CANVAS es que no requiere un detalle de los gastos y costos que integran la idea de negocio, para efectos del apoyo que se brindó a los estudiantes del concurso se comparten los gastos que integran su idea de negocio, con la intención de motivarlos para que descubrieran nuevos conceptos.

### 7.9.4 Costos fijos

Concepto	Importe Mensual	Importe Anual
Sueldos (Programador)	5,000.00	60,000.00
Total	5,000.00	60,000.00

**Tabla 18. Costos fijos.**

### 7.9.5 Costos variables

Por la naturaleza de la idea de negocio, su importe es muy pequeño, siendo aproximadamente 64.84 por unidad de producción. Los importes que se comparten, son mensuales.

Concepto	Importe Mensual	Importe Anual
Variables	4,050.00	48,630.00

Tabla 19. Costos variables.

### 7.9.6 Infraestructura

En este apartado, se incluye con lo que actualmente cuenta esta idea de negocio, en lo que se comúnmente como inversión inicial.

Concepto	Importe Inicial
Computadora (Valor de reposición estimado)	5,500.00
Escritorio	1,500.00
Total	7,000.00

Tabla 20. Infraestructura.

### 7.9.7 financiamiento inicial

Una de las ventajas de la estructura del modelo CANVAS, es que permite dimensionar al emprendedor lo que requieren para poder iniciar su idea de negocio. Por lo que a continuación se presenta el primer financiamiento que se necesitaría.



Concepto	Importe Inicial
Para Renovación de Equipo de Computo	11,000.00
Para Adquisición de un escritorio	1,000.00
Para Adquisición de mesa de trabajo	1,000.00
Total	13,000.00

**Tabla 21. Financiamiento inicial.**

### 7.9.8 Punto de equilibrio

En base a los valores económicos de los que se realizó la referencia en apartado anteriores. Se determinó que esta idea de negocios en una primera etapa, deberá obtener los siguientes datos.

Concepto	Importe Mensual	Importe Anual
Ingresos (para punto de Equilibrio)	7,438.00	89,257.00

**Tabla 22. Punto de equilibrio.**

## **8. Conclusiones.**

Como parte de este proyecto de tesis se desarrolló el software y hardware necesario para implementar un multímetro digital que usa como pantalla de despliegue de información dispositivos móviles tales como teléfonos inteligentes y/o tablets que funcionen con sistema operativo Android.

La interfaz gráfica fue desarrollada de forma que emula la apariencia de un multímetro digital convencional. La interfaz cuenta con una perilla de selección que le permite al usuario definir la variable de medición. Las mediciones realizadas son desplegadas en la pantalla. Estas funciones fueron programadas para tener apariencia similar a como se haría o visualizaría en un multímetro digital convencional.

Adicionalmente el sistema cuenta con la modalidad de despliegue gráfico, en tiempo real, de las mediciones realizadas. De esta forma, el usuario tiene la posibilidad de visualizar el comportamiento de la variable medida como una función del tiempo de medición.

El sistema puede realizar la medición de las siguientes magnitudes: Voltaje de CD, Resistencia eléctrica y Temperatura.

El módulo de medición fue implementado utilizando una tarjeta de desarrollo Arduino. La transmisión de los datos entre el circuito de medición y el dispositivo de despliegue se hace a través de Bluetooth.

Un obstáculo encontrado al momento de implementar el módulo de medición con la tarjeta Arduino, fue que el rango de medición de las entradas analógicas es de 0 a 5v, haciendo imposible medir voltajes negativos. Debido a que los multímetros convencionales cuentan con la capacidad de medir las señales de voltaje en polaridad positiva o negativa por lo que fue necesario implementar un circuito medidor que permita cumplir con esta función.

Para el desarrollo de la aplicación móvil, existen diferentes plataformas que facilitan la programación. Por ejemplo, App inventor, que es una plataforma que ayudan a crear aplicaciones móviles e incluso permite configurar el puerto bluetooth del dispositivo. Sin embargo, tiene las desventajas que hacen que la aplicación sea muy pesada, además limita

aspectos de diseño visual y de programación. Para evitar las problemáticas mencionadas para el desarrollo de la aplicación se utilizó Android Studio. Esta es una plataforma enfocada al desarrollo más especializado de aplicaciones móviles para Android. Se hace especial enfoque a que existen pocas aplicaciones desarrolladas en Android Studio para envío y recepción utilizando bluetooth.

A lo largo del desarrollo de este proyecto surgen situaciones que ayudaron a los integrantes a crear estructuras de diseño tanto de programación como de circuitos electrónicos. Estas estructuras pueden ser integradas a proyectos similares, en los que se tengan dos o más áreas involucradas.

Como se observó, los equipos de mediciones en los laboratorios pueden ser adaptados para ser usados a través de aplicaciones móviles.

La navegación de la aplicación es muy buena y cuenta con métodos que detectan la desconexión del bluetooth, regresando a la pantalla de búsqueda, esto ayuda a no tener lecturas no deseadas.

Con este proyecto se participó en el Concurso Emprendedor organizado por la Universidad Autónoma del Estado de México obteniendo el cuarto lugar en la categoría de tecnología básica en el año 2016. El desarrollo del plan de negocios permitió complementar la visión del proyecto desde perspectiva económica. Producto de esta investigación, actualmente se tiene visualizado el mercado potencial de este tipo de aplicaciones.

## **8. Trabajos futuros.**

Como trabajos futuros se consideran los siguientes aspectos:

- Es necesario complementar la funcionalidad del multímetro digital para que sea capaz de realizar mediciones de corriente eléctrica de CD.
- Otra opción que se debe de considerar el desarrollo de un módulo de medición de corriente eléctrica y voltaje de CA.

- Con el objetivo de reducir costos y optimizar las dimensiones del módulo de medición es necesario migrar lo desarrollado a través de la tarjeta Arduino a un circuito diseñado con microcontroladores. Una opción a evaluar puede ser el uso del Pic 16F887, que cuentan con un CAD y puertos RT y RX para el envío y recepción de información serial.
- La ventaja de que el sistema fue programado en JAVA es que las nuevas versiones permiten la utilización conjunta de JAVA y lenguaje C. Esto incrementa las opciones de desarrollo y puede servir para incrementar la velocidad de procesamiento de la información.
- Se pueden incluir funciones de procesamiento matemático de la información tales como derivación, integración, cálculo de la regresión lineal, entre otras. Estas funciones aportarían a la capacidad de medición y procesamiento de la información del dispositivo, utilizando de una manera más eficiente las capacidades del dispositivo móvil de despliegue.
- Otro de los aspectos a desarrollar es la incorporación de funciones que faciliten el diseño y análisis de circuitos digitales. Ejemplo de estas funciones podría ser el desarrollo de un generador de palabras, un probador de circuitos combinatoriales, comprobador de tablas de verdad, punta lógica, entre otros.
- Un aspecto muy importante a considerar como trabajo futuro es la comercialización del multímetro digital.

## REFERENCIAS

Android Studio (versión 7.0) [Software de computación]. Google:E.U

Bluetooth SIG, Inc. (2017). Bluetooth 5 Core. Recuperado de <https://developer.bluetooth.org/TechnologyOverview/Pages/SPP.aspx>

Cooper, W. D. (1985). *Instrumentación Electrónica y Mediciones*. México: Prentice-Hall Hispanoamericana

Franco, O. (Febrero, 2014). México móvil 2014: Consumidores y Publicistas conectan vía Smartphones. *e-Marketer*. Recuperado de: <http://es.slideshare.net/iabmexico/e-marketer-mexicomovil2014consumidoresypublicistasconectanviasma?related=1>

Gironés, J. T. (2015). El gran libro de Android. México: Alfaomega Grupo Editor

Gomez, S. (Agosto, 2014). Interfaz de usuario en Android: Layouts. Recuperado de: <http://www.sgoliver.net/blog/interfaz-de-usuario-en-android-layouts/>

Google. (2016). Android Studio: View Pager. Recuperado de: <http://developer.android.com/reference/android/support/v4/view/ViewPager.html>

Google. (2016). Android Studio: The Emulator. Recuperado de: <http://developer.android.com/intl/es/tools/help/emulator.html>

Google. (2016). Android Studio: Cómo crear un proyecto de Android. Recuperado de: <http://developer.android.com/intl/es/training/basics/firstapp/creating-project.html>

Google. (2016). Android Studio: Activity. Recuperado de: <http://developer.android.com/intl/es/reference/android/app/Activity.html>

Google. (2016). Android Studio: Compatibilidad con diferentes pantallas. Recuperado de: [http://developer.android.com/intl/es/guide/practices/screens\\_support.html](http://developer.android.com/intl/es/guide/practices/screens_support.html)

- Google. (2016). Android Studio: View OnClickListener. Recuperado de: <http://developer.android.com/intl/es/reference/android/view/View.OnClickListener.html>
- Google. (2016). Android Studio: Intent. Recuperado de: <http://developer.android.com/intl/es/reference/android/content/Intent.html>
- Instituto Nacional de Estadística y Geografía [INEGI]. (2014). Estadística sobre disponibilidad y uso de tecnología de información y comunicaciones en los hogares, 2013. Recuperado de [http://internet.contenidos.inegi.org.mx/contenidos/Productos/prod\\_serv/contenidos/espanol/bvinegi/productos/metodologias/MODUTIH/MODUTIH2013/MODUTIH2013.pdf](http://internet.contenidos.inegi.org.mx/contenidos/Productos/prod_serv/contenidos/espanol/bvinegi/productos/metodologias/MODUTIH/MODUTIH2013/MODUTIH2013.pdf)
- Itead Studio. (2010). HC-05 Bluetooth module. Recuperado de [http://www.robotshop.com/media/files/pdf/rb-ite-12-bluetooth\\_hc05.pdf](http://www.robotshop.com/media/files/pdf/rb-ite-12-bluetooth_hc05.pdf)
- Keysight Technologies. (2014). Adaptador de IR a Bluetooth. Recuperado de: <http://literature.cdn.keysight.com/litweb/pdf/5990-9531ESE.pdf>
- Martinez, R. (2014). Kernel/Núcleo. Recuperado de <http://www.linux-es.org/kernel>.
- Mileaf, Harry. (1979). *Electrónica cinco*. México: Limusa
- Rolle, K. C. (2006). *Termodinámica*. México: Pearson Educación
- Stanley, W & Smith, R. (1992). *Guía para mediciones electrónicas y prácticas de laboratorio*. México: Prentice Hall.
- Tipler, P. A. & Mosca G. (2005). *Física para la ciencia y la tecnología*. España: Reverté
- Marketing Directo. (enero, 2014). Uno de cada cuatro mexicanos tendrá un smartphone en 2014. Recuperado de: <http://www.marketingdirecto.com/especiales/latinoamerica-especiales/uno-de-cada-cuatro-mexicanos-tendra-un-smartphone-en-2014/>

Rojas, C. (Junio, 2013). Programación Extrema (XP) - Aprende a Programar. *Codejobs*. Recuperado de: <http://www.codejobs.biz/es/blog/2013/06/05/programacion-extrema-xp#sthash.XSkpaVPI.dpbs>

Virrueta, A. (2010). *Metodologías de desarrollo de software*. Michoacán: Trillas.

Báez, M., Borrego, A., Cordero, J., Cruz, L., González, M., Hernández, F., Palomero, D. & et al. (2014). Introducción a Android. *Universidad Complutense de Madrid*. Recuperado de: <http://www.it-docs.net/ddata/18.pdf>

García, A. (2013). Comunicación serial con Arduino. *Panama Hitek*. Recuperado de: <http://panamahitek.com/comunicacion-serial-con-arduino/>

Fuente: Crespo, C. (2015). Shields y Arduino. Recuperado de: <https://aprendiendoarduino.wordpress.com/2015/03/23/shields-para-arduino/>