SEDE GUAYAQUIL

CARRERA DE INGENIERÍA ELECTRÓNICA

Tesis previa a la obtención del Título de
Ingeniero en Electrónica mención Sistemas Computacionales

TEMA

"DISEÑO E IMPLEMENTACIÓN DE UN MÓDULO DIDÁCTICO Y DESARROLLO DE APLICACIONES ELECTRÓNICAS SOBRE PLATAFORMA ANDROID"

AUTORES

Tec. Jorge Manuel Abudeye Vinueza
Tec. Luis Fabricio Silva Cruz

DIRECTOR:

Ing. Orlando Barcia Ayala, MSC

Guayaquil, Enero 2015

DECLARATORIA DE RESPONSABILIDAD

Todos los conceptos, ideas, afirmaciones y conclusiones de este proyecto de tesis son de exclusiva responsabilidad de los autores, y el patrimonio intelectual le pertenece a la Universidad Politécnica Salesiana.

Guayaquil, 19 de Enero del 2015

Jorge Manuel Abudeye Vinueza
C.I.: 0918800277

Luis Fabricio Silva Cruz
C.I.: 0920515798

DEDICATORIA

A Dios por el amor infinito que ha tenido conmigo y por los regalos que me ha brindado desde que llegue a este mundo, mi familia. A mi Madre por su lucha incansable, por darnos lo mejor de forma física y espiritual. A mi esposa por su amor y paciencia en el desarrollo de mis potencialidades. A Jezrael y Axel por darme la inspiración cada día.

Jorge Manuel Abudeye Vinueza

DEDICATORIA

“Un hombre no es otra cosa que lo que hace de sí mismo” Jean-Paul Sartre

Es Dios por sobre todo, a quien deseo dedicar este logro, pues gracias a él he tenido la vida que se me ha dado y ha puesto en ella las personas y momentos indicados para enseñarme a crecer y creer en que la Fe y la Voluntad son las fuerzas más sorprendentes de este mundo.

Es Dios quien me dio la vida y con ella unos padres ejemplares, Luis y Sonia, y una familia única e irremplazable, en cuyo hogar forme desde pequeño mi carácter y cualidades que me han traído hoy hasta aquí, a lograr esta, una de las metas de mi vida.

Es Dios quien puso frente a mí, en las aulas de clases, a los mejores tutores y maestros que compartieron su conocimiento conmigo.

Eres tú Dios aquel que me permitió compartir con los mejores compañeros y amigos cada momento bueno y malo dentro y fuera de un aula de clases.

Y Dios, como no dedicarte este logro en mi vida, si me regalaste a la mejor mujer como esposa, Blanca Denisse, y con ella el tesoro más grande y hermoso que hasta hoy he podido recibir, Dara Paulina.

Luis Fabricio Silva Cruz

AGRADECIMIENTO

Agradezco a Dios por su amor y misericordia.

A mi madre por su lucha incansable.

A mi esposa y a mis hijos por ser mi fuente de inspiración.

A mis maestros y amigos por ser parte del desarrollo profesional e integral.

Jorge Manuel Abudeye Vinueza

AGRADECIMIENTO

A Dios ante todo, por darme la motivación, la sabiduría y la fe de creer en mí y mis objetivos.

A mis padres, autores aquí en la tierra de cada paso y lucha que emprendo.

A mi esposa y mi hija, que con su comprensión, ternura y apoyo incondicional me inspiran a ser mejor cada día.

A mis amigos, maestros y colegas, que con sus consejos, experiencias y enseñanzas me han ayudado a trazar el camino correcto hacia mi crecimiento profesional y humano.

Luis Fabricio Silva Cruz

ÍNDICE GENERAL

ÍNDICE DE TABLAS

CAPÍTULO II

CAPÍTULO III

CAPITULO IV

ÍNDICE DE FIGURAS

CAPÍTULO I

CAPÍTULO II

CAPÍTULO III

CAPÍTULO IV

CAPÍTULO V

RESUMEN

AÑO	ALUMNOS	DIRECTOR DE TESIS	TEMA TESIS
2014	JORGE MANUEL ABUDEYE VINUEZA LUIS FABRICIO SILVA CRUZ	ING. ORLANDO BARCIA	DISEÑO E IMPLEMENTACIÓN DE UN MÓDULO DIDÁCTICO Y DESARROLLO DE APLICACIONES ELECTRÓNICAS SOBRE PLATAFORMA ANDROID

La presente tesis: **"DISEÑO E IMPLEMENTACIÓN DE UN MÓDULO DIDÁCTICO Y DESARROLLO DE APLICACIONES ELECTRÓNICAS SOBRE PLATAFORMA ANDROID"**, se basa en aplicaciones Android con Arduino. Los casos presentados son: Decompilación de Apks, encendidos de luces, Conexiones Ethernet – USB – bluetooth, Mando Servo motor, Control de velocidad y cambio de giro, Control de temperatura y humedad, Control de video.

El objetivo es brindar a los estudiantes las bases fundamentales entre la comunicación Android y Arduino para el diseño de soluciones electrónicas utilizando la plataforma Android basada en Linux.

El desarrollo del proyecto complementa un análisis desarrollado con experimentos y pruebas, que crean un sistema didáctico de prácticas para los estudiantes de las materias de programación, Sistemas Microprocesados, Club de robótica, seminarios y similares.

PALABRAS CLAVES

Android, Arduino, Comunicación bluetooth.

ABSTRACT

YEAR	STUDENTS	THESIS DIRECTOR	THESIS TOPIC
2014	JORGE MANUEL ABUDEYE VINUEZA LUIS FABRICIO SILVA CRUZ	ING. ORLANDO BARCIA	DESIGNING AND IMPLEMENTING A DIDACTIC MODULE AND DEVELOPMENT OF ELECTRONIC APPLICATIONS ABOUT THE ANDROID PLATFORM.

This thesis: **"DESIGNING AND IMPLEMENTING A DIDACTIC MODULE AND DEVELOPMENT OF ELECTRONIC APPLICATIONS ABOUT THE ANDROID PLATFORM"** is based on Android applications with Arduino. The cases studied are: Decompilation from Apks, switching on of lights, Ethernet –USB-Bluetooth connections, servomotor control, speed and rotation change, temperature and humidity control and video control.

The objective is to give students the principal basis between Android and Arduino communication for the designing of electronic solutions using the Android platform based on Linux.

The development of the Project complements the analysis developed with experiments and tests, which create a didactic system of practice for the students of the subjects of programming, micro processed systems, robotics club and seminars.

KEY WORDS

Android, Arduino, Bluetooth Communication.

INTRODUCCIÓN

El presente proyecto de tesis está orientado a la dotación de elementos y conceptos didácticos - prácticos para el desarrollo de aplicaciones sobre la plataforma Android estará constituido de los siguientes elementos:

- Aplicación desarrollada sobre la plataforma Android, la cual permitirá interactuar con los diferentes elementos y dispositivos que conforman cada una de las prácticas.
- Dispositivos Arduino (USB, Bluetooth, Ethernet), el cual permitirá establecer la comunicación entre la aplicación y los elementos electrónicos.
- Sensores de humedad y temperatura, elementos electrónicos, motor/servomotor DC, cámara de video IP compatibles con el módulo de entrenamiento Arduino.
- Maletín didáctico, el cual se estructurará de tal modo que contenga de manera organizada y didáctica los diferentes elementos que conforman el presente proyecto.
- Dispositivo Tablet con Sistema Operativo Android.

El proyecto se desarrolló de manera didáctica en diferentes prácticas, realizando la introducción y preparación del ambiente de desarrollo de las aplicaciones aquí utilizadas, como lo son los pre-requisitos de software y sub-aplicaciones que permiten y facilitan esta tarea (Plataformas de Desarrollo en JAVA, subrutinas y compiladores propios del sistema operativo Android.

En el capítulo 1 se detallan el problema, planteamiento del problema, metodologías, técnicas, beneficiarios del proyecto.

En el capítulo 2 se define el marco teórico, fundamental para la obtención de los conceptos necesarios para el desarrollo de prácticas sobre módulo de entrenamiento Arduino UNO y creación de aplicaciones sobre la plataforma Android.

En el capítulo 3 se muestra el desarrollo del diseño y construcción del módulo didáctico para el montaje de las diferentes prácticas propuestas en el documento de tesis.

En el capítulo 4 se puntualiza el diseño y construcción del robot móvil explorador para el desarrollo de la práctica 13 que integra las diferentes prácticas para la visualización del alcance de soluciones tecnológicas.

En el capítulo 5 se documentan las pruebas realizadas mediantes las prácticas propuestas mediante la programación de las aplicaciones Android y las conexiones de los elementos con la tarjeta de entrenamiento arduino. Cada práctica está plenamente individualizada tomando como referencia el marco teórico antes descrito.

CAPÍTULO I

PLANTEAMIENTO DEL PROBLEMA

1. El problema

1.1 Planteamiento del Problema.

En la actualidad se ha incrementado el uso de dispositivos móviles que forman parte de las herramientas de uso cotidiano, debido a esto, la electrónica presenta un reto para los estudiantes en la creación y desarrollo de aplicaciones que interactúen entre los dispositivos móviles y el medio físico.

Las herramientas que en este momento se encuentran al alcance de los estudiantes limitan en cierta medida disponer de una referencia o punto de partida que les permita el desarrollo de soluciones tecnológicas basadas en plataforma Android, esto representa una desventaja para los estudiantes de la UPS frente al vertiginoso avance en el desarrollo de aplicaciones basadas en esta plataforma.

1.2 Delimitación del problema.

1.2.1 Temporal

El periodo de realización de este proyecto será desde Julio del 2013 hasta el Enero del 2015, se prevé que este plazo es el tiempo necesario para la recopilación de toda la información requerida que se utilizará como base.

1.2.2 Espacial

El desarrollo de este proyecto será desarrollado en la Universidad Politécnica Salesiana Sede Guayaquil.

Figura 1. 1 Croquis de la Universidad Politécnica Salesiana Sede Guayaquil

Figura: Google Maps, Google Inc. (2015)

1.2.3 Académica

En relación al área académica que interviene en el desarrollo de este proyecto, se han utilizado los conocimientos de Ingeniería Electrónica en las ramas y materias de:

- Programación.
- Sistemas Microprocesados.
- Club de Robótica.
- Seminarios.

Se utiliza como base la plataforma Android en conjunto con tarjetas de entrenamiento Arduino y sensores compatibles.

1.3 Objetivos

1.3.1 Objetivo General

- Diseñar e implementar un módulo didáctico que permita a los estudiantes desarrollar aplicaciones Android para la construcción de soluciones electrónicas.

1.3.2 Objetivos específicos

- Desarrollar aplicaciones Android para comunicación bluetooth con el módulo de entrenamiento Arduino.
- Interactuar con el medio físico a través de sensores compatibles con el módulo de entrenamiento Arduino bluetooth, Ethernet, USB, temperatura y humedad.
- Desarrollar un módulo para el aprendizaje, comprensión y aplicación para funcionamiento de elementos electrónicos con la plataforma Android.
- Implementar y demostrar, de modo práctico-experimental, las aplicaciones que puede ofrecer la plataforma Android, en conjunto con los diferentes elementos que ayudan a la interacción con el exterior.
- Demostrar en aplicaciones reales la interacción de los dispositivos y elementos electrónicos con la plataforma Android, con lo cual se posiciona a este tipo de implementaciones como una herramienta y método de solución a problemas y necesidades de la vida cotidiana.

1.4 Justificación

Contribuyendo con la universidad para contar con los recursos necesarios para adaptarse al uso de nuevas tecnologías específicamente la plataforma Android ya que la mayoría de dispositivos móviles o aplicaciones están soportadas y desarrolladas sobre el sistema operativo de Google. En primer lugar se proporcionará a los estudiantes un conjunto de conceptos fundamentales para el uso de esta tecnología y las aplicaciones bases que pueden desarrollarse. Es conveniente que este aprendizaje se incluya para la elaboración de proyectos que resuelvan problemas cotidianos a través de la implementación de los sistemas desde un punto de vista práctico.

1.5 Hipótesis

Cuanto mayor sea el desarrollo de aplicaciones sobre plataforma Android con Arduino, mayor serán las soluciones electrónicas que pueden ser aplicadas por los estudiantes de la Universidad Politécnica salesiana para la resolución de problemas de la vida cotidiana.

1.6 Variables e Indicadores

Variable independiente: Elaboración de las prácticas electrónicas en tarjetas Arduino con plataforma Android.

Variable dependiente: Diseño y construcción del módulo didáctico para el desarrollo de soluciones tecnológicas con plataforma Android y tarjetas arduino.

1.7 Metodología

1.7.1 Método teórico y sistémico.

Para el desarrollo y cumplimiento del proyecto de tesis se realizó el levantamiento de la información:

- Funcionamiento de la plataforma Android y sus componentes
- Funcionamiento de las tarjetas de entrenamiento arduino.
- Plataformas de desarrollo para aplicaciones Android
- Búsqueda de información en páginas oficiales arduino y google
- Consulta de materiales posibles para la construcción del módulo

Luego de los estudios y análisis de la información recopilada se procedió del siguiente modo:

- Diseño y construcción de módulo de acuerdo a las medidas y con los materiales presupuestados.
- Construcción e instalación de fuentes de poder y elementos del panel principal.
- Ejecución de prácticas implementando funcionalidades de Arduino, Android y los componentes seleccionados para cada práctica.
- Implementación conjunta de prácticas en una práctica general sobre un móvil explorador.
- Correcciones y mejoras sugeridas por el tutor de tesis luego de las reuniones mantenidas.

- Realización del documento tesis, mediante la recopilación de la información obtenida en las prácticas y pruebas ejecutadas.

1.8 Población y Muestra

1.8.1 Población

Para el desarrollo del presente proyecto de Tesis la población seleccionada ha sido aquella conformada por los estudiantes de la **Carrera de Ingeniería Electrónica** de la Universidad Politécnica Salesiana – Sede Guayaquil – contabilizada aproximadamente en 1000, entre los cuales se incluyen aquellos aspirantes que se encuentran con examen de admisión o pre-universitario aprobado.

Esta selección obedece a que las características técnicas definidas para nuestro proyecto, se apegan en gran medida a las ofrecidas por la Universidad e impartida por los docentes y adquiridas por los estudiantes en el transcurso de la carrera; características entre las cuales podemos mencionar la *lógica de programación, el diseño de circuitos electrónicos, desarrollo de aplicaciones e interfaces de comunicación software-hardware, etc.*

1.8.2 Muestra

Como muestra intencionada se tomó como referencia a los estudiantes que forman parte del **Club de Robótica** de la Universidad Politécnica Salesiana – Sede Guayaquil – además de aquellos estudiantes inscritos en las materias de **Sistemas Microprocesados I y II,** y la materia de **Programación II,** ambas asignaturas de la carrera de **Ingeniería Electrónica,** los cuales suman un total de 1000 estudiantes.

Esta muestra específica fue seleccionada al estar compuesta por estudiantes que se relacionan y desarrollan clase a clase durante el semestre en curso aplicaciones y prácticas que mejoran el entendimiento de la implementación conjunta de los elementos que constituyen nuestro proyecto de tesis.

1.9 Descripción de la Propuesta

El presente proyecto de Tesis orientado a la dotación de elementos y conceptos didácticos y prácticos para el desarrollo de aplicaciones sobre la plataforma Android estará constituido de lo siguiente:

- Aplicación desarrollada sobre la plataforma Android, la cual permitirá interactuar con los diferentes elementos y dispositivos que conforman cada una de las prácticas.
- Dispositivos Arduino (USB, Bluetooth, Ethernet), el cual permitirá establecer la comunicación entre la aplicación y los elementos electrónicos.
- Sensores de humedad y temperatura, elementos electrónicos, motor/servomotor DC, cámara de video IP compatibles con el módulo de entrenamiento Arduino.
- Maletín didáctico, que se estructuró de tal modo que contenga de manera organizada y didáctica los diferentes elementos que conforman el presente proyecto.
- Dispositivo Tablet con Sistema Operativo Android.

El proyecto realiza de manera didáctica diferentes prácticas y realiza también la introducción y preparación del ambiente de desarrollo de las aplicaciones aquí utilizadas, como lo son los pre-requisitos de software y sub-aplicaciones que permiten y facilitan esta tarea (Plataformas de Desarrollo en JAVA, subrutinas y compiladores propios del sistema operativo Android.

Se empieza con una explicación Teórico-Práctica de cómo realizar la Decompilación de aplicaciones desarrolladas sobre Android, ya existentes y disponibles en la tienda virtual de google, lo que nos entrega un entendimiento real de los procedimientos y buenas prácticas de desarrollo que permitieron ajustar las aplicaciones aquí implementadas a los estándares manejados por esta plataforma.

En la siguiente práctica de desarrollo e implementación se establecen los parámetros y comandos que hacen posible la interacción de las aplicaciones sobre Android con los elementos Arduino USB, Arduino bluetooth y Arduino Ethernet.

Una vez que se establece, desarrolla e implementa de manera correcta y exitosa la comunicación entre la aplicación y las interfaces que hacen posible su interacción con el exterior, se procede con la ejecución de las prácticas para el control de Leds, lo cual a su vez permite un mayor entendimiento de las características de entrada y salida de datos digitales del dispositivo Arduino y su interacción de éstas con la aplicación.

En la práctica siguiente, al trabajar con Sensores, se realizan e implementan elementos electrónicos que hacen posible el censado de señales análogas y a su vez la respuesta generada desde la aplicación ante este tipo de estímulos exteriores, con lo cual se mejoró el entendimiento de la captura de datos hacia la aplicación.

A continuación, se diseña e implementa la aplicación sobre Android y los parámetros que posibilitan la captura y recepción de imágenes vía Ethernet, utilizando el dispositivo Arduino Ethernet como interfaz de comunicación entre la aplicación y el elemento de hardware, la cámara para este caso.

En la práctica final, se implementa en conjunto cada uno de los anteriores módulos del proyecto en una aplicación y hardware único, el cual consiste en un Robot explorador móvil con control de velocidad, control y regulación de luces, sensores de temperatura y humedad y una cámara IP para observar el entorno; todo esto controlado y monitoreado por la aplicación desarrollada sobre la plataforma Android e instalada en el Dispositivo Tablet.

Como fue referido anteriormente, todos estos elementos y prácticas se presentan en un maletín con características didácticas, lo que hace más amigable el entendimiento y motivación al desarrollo y construcción de aplicaciones electrónicas en las cuales su plataforma de control sea la plataforma Android.

CAPÍTULO II

MARCO TEÓRICO

2.1 Android

Android ha sido calificado como un sistema de desarrollo para aplicaciones móviles de primera categoría, esto ha marcado el enfoque de este sistema operativo al desarrollo de aplicaciones para empresas que utilizan los dispositivos móviles como parte del negocio y del mismo modo para expertos que buscan con sus creaciones resolver problemas de la vida cotidiana.

"*El primer teléfono Android vendido por T-Mobile G1- fue desarrollado por el fabricante de teléfonos HTC ƒ con el servicio proporcionado por T-Mobile. Fue lanzado en octubre de 2008. Muchos otros teléfonos Android fueron programados para 2009 y principios de 2010. La plataforma cobró impulso con relativa rapidez. Para el cuarto trimestre de 2010, Android había llegado a dominar el mercado de teléfonos inteligentes, ganando terreno constantemente contra las plataformas competitivas como RIM BlackBerry, Apple iOS y Windows Mobile".* (Annuzzi, J.Jr., Darcey, L., Conder, S., 2013)

Catalogada por muchos usuarios como una plataforma innovadora y amigable, Android se ha posicionado de manera exitosa para hacer frente al demandante mercado de dispositivos móviles, que no deja de crecer y se está expandiendo por sobre los usuarios de gama baja hacia los de gama media a alta.

Comunidades de desarrolladores de aplicaciones de diversos enfoques, coinciden en que Android es una plataforma completa, abierta y libre para el desarrollo de estas aplicaciones.

"Completo: Los diseñadores tomaron un enfoque integral cuando se desarrolló la plataforma Android. Comenzaron con un sistema operativo seguro y construyeron un marco de software robusto en la parte superior que permite ricas oportunidades de desarrollo de aplicaciones.

Abierto: La plataforma Android se proporciona a través de licencias de código abierto. Los desarrolladores tienen acceso sin precedentes a las características del dispositivo en el desarrollo de aplicaciones.

Libre: aplicaciones Android son libres de desarrollar. No hay derechos de licencia para el desarrollo de la plataforma. No se requiere cuotas de afiliación. No se requiere honorarios de la prueba. No se requiere la firma o certificación de honorarios. Aplicaciones Android pueden ser distribuidos y comercializados en una variedad de maneras." (Annuzzi, J.Jr., Darcey, L., Conder, S., 2013)

Es así que llegamos a una definición formal para Android de acuerdo a las características antes descritas:

Android es un sistema operativo y una plataforma de software sobre la que se desarrollan las aplicaciones. Un conjunto básico de aplicaciones para las tareas cotidianas, como la navegación web y correo electrónico, están incluidos en los dispositivos Android. (Annuzzi, J.Jr., Darcey, L., Conder, S., 2013)

Al combinar de manera eficiente los elementos que conforman el dispositivo móvil, Android entrega a los desarrolladores un abanico extenso de posibilidades para la creación de sus aplicaciones móviles.

"Android proporciona un marco de aplicación rica que le permite construir aplicaciones y juegos innovadores para dispositivos móviles en un entorno de lenguaje Java." (Google Inc, 2015)

Figura 2.1 Versiones de Android

Figura: Versiones existentes para dispositivos móviles (Annuzzi, J.Jr., Darcey, L., Conder, S., 2013)

2.1.1 Entorno de Programación Android

Cada nuevo proyecto sobre eclipse, genera una estructura de elementos que forman parte integral de la aplicación que se desarrollará.

"*Las Aplicaciones de Android están escritas en el lenguaje de programación Java. Las herramientas del SDK de Android compilar su código-junto con todos los datos y archivos-y recursos en un APK: paquete de Android, que es un archivo con un sufijo .apk. Un archivo APK contiene todo el contenido de una aplicación para Android y es el archivo que los dispositivos con Android utilizan para instalar la aplicación*." (Google Inc, 2015)

Figura 2.2 Elementos de proyecto Android

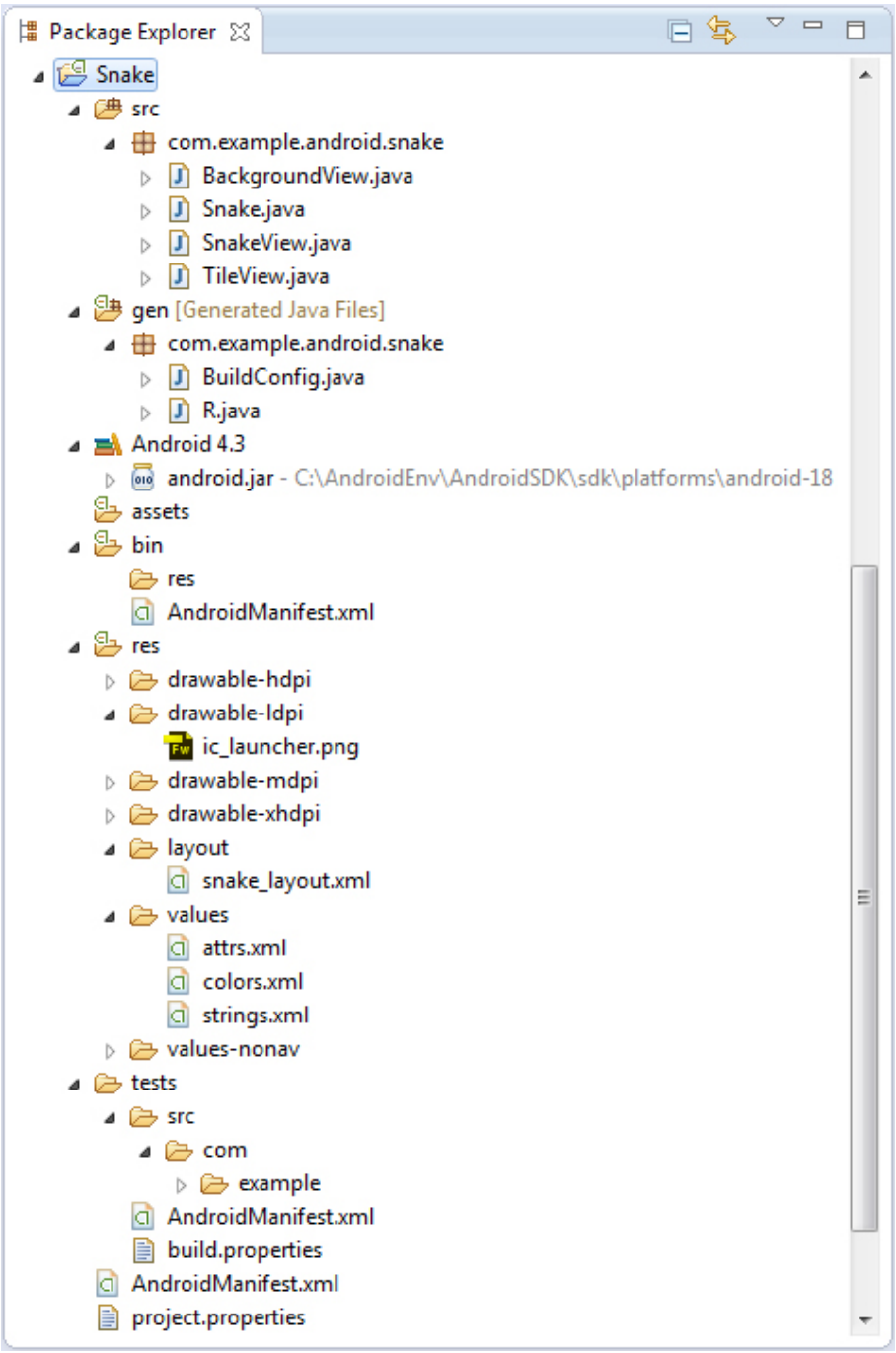


Figura: Estructura de un proyecto Android en Eclipse (Annuzzi, J.Jr., Darcey, L., Conder, S., 2013)

Vamos a continuación a describir la función de cada directorio dentro de nuestro proyecto Android según Annuzzi, J.Jr., et all (2013):

Carpeta /src/ - Esta carpeta contendrá todo el código fuente de la aplicación, código de la interfaz gráfica, clases auxiliares, etc.

Carpeta /res/ - Contiene todos los ficheros y recursos necesarios para el proyecto: imágenes, vídeos, cadenas de texto, etc.

Carpeta /gen/ - Contiene una serie de elementos de código generados automáticamente al compilar el proyecto.

Carpeta /assets/ - Contiene todos los demás ficheros auxiliares necesarios para la aplicación y que se incluirán automáticamente en su propio paquete, como por ejemplo ficheros de configuración, de datos, etc.

Carpeta /bin/ - Contiene los elementos compilados de la aplicación y otros ficheros auxiliares. No será necesario manipular los archivos al interior de la misma.

Carpeta /libs/ - Contendrá las librerías auxiliares, normalmente en formato ".jar" que utilicemos en nuestra aplicación Android.

AndroidManifest.xml - Es el archivo de configuración central para la aplicación. En él se definen las capacidades y los permisos de su aplicación, así como la forma en que se ejecuta.

Este último es uno de los más importantes ficheros de nuestra aplicación. Como se menciona, es aquí donde se definen los permisos para acceder a cada uno de los elementos de hardware del dispositivo móvil (GPS, Cámara, Red de Datos, Red WiFi, etc).

"La interfaz gráfica de usuario para una aplicación para Android está construido usando una jerarquía de View y ViewGroup Objects. Los ViewObjects son usualmente Widgets de interfaz de usuario como Buttons (botones) o TextFields (Cuadros de Texto)." (Google Inc, 2015)

Toda esta jerarquía en la estructura de la interfaz gráfica es definida a través de un archivo XML, el cual es manipulable y permite establecer la interfaz de la aplicación de acuerdo a las necesidades establecidas.

Es así que para este diseño en modo jerárquico identificamos los siguientes principales elementos dentro de la interfaz gráfica de una aplicación típica Android:

2.1.2 *"**Android Layout.-** Un control Layout sigue siendo un View Object pero en realidad no dibuja nada especifico en su pantalla. En su lugar es un contenedor (parent) que organiza los demás controles (childs). Un control Layout determina como y donde son dibujados los controles child dentro de la pantalla."* (Annuzzi, J.Jr., Darcey, L., Conder, S., 2013)

En resumen, es en el control Layout donde es dibujada toda la interfaz de usuario, es aquí donde son ubicados los botones, textviews, togglebuttons y demás elementos que forman parte del diseño de interfaz de una aplicación Android.

2.1.3 Botones Android

En el entorno de desarrollo Android, los botones se constituyen en herramientas que ejecutan una acción programada al ser presionados. De aquí que en Android tenemos los siguientes tipos de botones:

Basic button.- Es utilizado muy comúnmente para ejecutar algún tipo de acción de tipo confirmación, por ejemplo. Este puede contener texto o imágenes.

CheckBox.- Este elemento se define en dos estados, marcado y desmarcado. Es comúnmente utilizado en selecciones de elementos en listas.

ToggleButton.- Tiene semejanza con el CheckBox, con la diferencia de que permite, por ejemplo, mostrar el estado del mismo (presionado, no presionado)

Switch.- Se puede definir como un botón de dos estados. Por ejemplo es utilizado para definir estados "on" y "off".

RadioButton.- Sirve como elemento de selección, por ejemplo, para seleccionar una opción entre varias.

Figura 2.3 Tipos de Botones

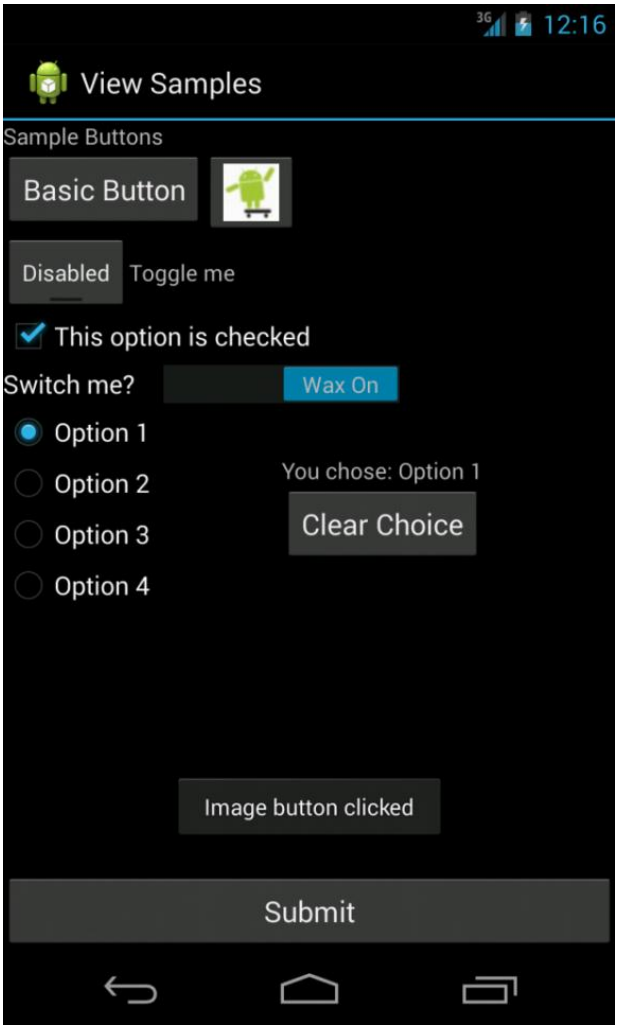


Figura: Interfaz Android con tipos de botones disponibles (Annuzzi, J.Jr., Darcey, L., Conder, S., 2013)

2.1.4 TextView

Muestra el texto para el usuario y opcionalmente les permite editarlo. Un TextView es un editor de texto completo, sin embargo la clase básica está configurada para no permitir la edición. (Google Inc, 2015)

2.2 Arduino

"Arduino es una plataforma electrónica de código abierto basado en hardware y software fácil de usar. Está dirigido a cualquier persona que hace proyectos interactivos." (Arduino SA, 2015)

En otras palabras, es una plataforma interactiva para la programación y conexión de soluciones electrónicas. Puede adquirir datos del entorno a través de una gama de sensores y elementos compatibles mediante sus pines de entrada. Ejemplo: Encendido de luces, control de motores, captura de temperatura y humedad.

2.2.1 La Tarjeta Arduino

Arduino SA (2015), describe a Arduino UNO como una tarjeta con microcontrolador el cual está constituido por un Microprocesador AT-MEGA 328. Posee 14 pines digitales de entrada / salida, de las cuales 6 son PWM, 6 entradas analógicas, conexión a USB y un botón de reinicio que permite, como a cualquier computadora, realizarle un reset cuando nuestro Arduino empieza a comportarse de manera errónea.

Figura 2.4 Arduino Uno

Figura: Tarjeta Arduino con microcontrolador ATMEGA (Arduino SA, 2015)

TABLA 2. 1 Características Arduino UNO

RESUMEN	
Microcontroladores	*ATmega328*
Tensión de funcionamiento	*5V*
Voltaje de entrada (recomendado)	*7-12V*
Voltaje de entrada (límites)	*6-20V*
Digital pines I / O	*14 (de los cuales 6 proporcionan PWM)*
Pines de entrada analógica	*6*
Corriente DC por Pin I / O	*40 mA*
Corriente DC de 3.3V Pin	*50 mA*
Memoria Flash	*32 KB (ATmega328) de los cuales 0,5 KB utilizado por gestor de arranque*
SRAM	*2 KB (ATmega328)*
EEPROM	*1 KB (ATmega328)*
Velocidad de reloj	*16 MHz*

NOTA: Tabla de características de Board Arduino UNO (Arduino SA, 2015)

2.2.2 Estructura de Programación Arduino

A manera básica, el lenguaje de programación arduino es bastante simple y muy familiar al lenguaje de programación C, por lo cual muchos de los desarrolladores y expertos que utilizan este hardware lo siente bastante familiar.

Esta estructura se compone de dos funciones principales que son requeridas para que el sketch funcione:

voidsetup()

{ declaraciones;}

voidloop()

{declaraciones;}

*"**setup()***

La función setup debería contener la declaración de cualquier variable al comienzo del programa. Es la primera función a ejecutar en el programa, es ejecutada una vez y es usada para asignar pinMode o inicializar las comunicaciones serie." (Enriquez, R., 2009)

```
voidsetup()
{
pinMode(pin,OUTPUT); // ajusta 'pin' como salida
}
```

*"**loop()***

La función loop se ejecuta a continuación e incluye el código que se ejecuta continuamente leyendo entradas, activando salidas, etc. Esta función es el núcleo de todos los programas Arduino y hace la mayor parte del trabajo." (Enriquez, R., 2009)

```
voidloop()
{
digitalWrite(pin,HIGH);//Activa'pin'
delay(1000);//esperaunsegundo
digitalWrite(pin,LOW);//Desactiva'pin'
delay(1000);//esperaunsegundo
}
```

Funciones

Se puede definir a las funciones como un grupo de comandos que realizan tareas cada vez que la función es invocada.

"Las funciones se escriben para ejecutar tareas repetitivas y reducir el desorden en un programa. " (Enriquez, R., 2009)

E /S Digital

. pinMode (pin,mode) Configura el pin especificado a comportarse ya sea como una entrada o una salida. Modo: INPUT, OUTPUT, INPUT_PULLUP

. digitalRead(pin) Lee valores constantes y variables desde pines establecidos como HIGH o como LOW

. digitalWrite (pin,value) Pone en nivel HIGH o LOW el pin especificado por ´pin´. Admite valores de variable o constante.

E /S Analógica

. analogRead (pin) Lee el valor ingresado en un pin analógico especificado por 'pin'. Los valores entre 0v y 5v representan un rango de resolución de números enteros entre 0 y 1023 en tiempos de lectura de 100 ms (microsegundos).

. analogWrite (pin,value) Escribe un valor analógico en el pin 'pin' utilizando Modulación por Ancho de Pulso (PWM) exclusivamente en los pines marcados como PWM en la tarjeta arduino. Este es utilizado comúnmente en la variación de velocidad de motores y en controles de luminosidad de luces.

TABLA 2. 2 Relación Valor / Salida con analogWrite ()

Valor	Nivel de salida
0	0 V (t)
64	0 V (3/4 de t) y 5 V (1/4 de t)
128	0 V (1/2 de t) y 5 V (1/2 de t)
192	0 V (1/4 de t) y 5 v (3/4 de t)
255	5 V (t)

NOTA: Niveles y valores de en los pines de salida con la función analogWrite

(Enriquez, R., 2009)

2.2.3 Comunicación Serial

Se utiliza para la comunicación entre la placa Arduino a un ordenador u otros dispositivos. Todas las placas Arduino tienen al menos un puerto serie.

Se comunica a los pines digitales 0 (RX) y 1 (TX), así como con el ordenador a través de USB. Si se usan estos pines no pueden ser usados como entrada o salida

Se puede utilizar el monitor incorporado en el entorno Arduino para comunicarse con una placa Arduino.

. Serial.begin (rate) Realiza la apertura del puerto y establece la tasa de transmisión en baudios (9600 la más común).

. "Serial.println(data)

Imprime datos al puerto serie, seguido de un retorno de carro y avance de línea automáticos." (Enriquez, R., 2009)

CAPÍTULO III

DISEÑO Y CONSTRUCCIÓN DEL MÓDULO DIDÁCTICO

3. Módulo didáctico

3.1 Diseño y Construcción del Módulo

El modulo didáctico se ha diseñado con el objetivo de que sea resistente, fácil de transportar y a su vez fácil de interpretar y maniobrar por los alumnos que lo utilizaran para la implementación y simulación de las prácticas.

La construcción del módulo ha sido dividida en 5 secciones principales e identificables del siguiente modo:

a) Base Principal
b) Tapa A
c) Tapa B
d) Panel
e) Bandeja

a) Base Principal

La base principal sirve de soporte al resto de la estructura, además de proveer el espacio para alojar las fuentes de poder y las conexiones que de estas se derivan.

Los materiales utilizados en la construcción de la base principal se detallan en la siguiente tabla:

Tabla 3.1 Inventario Base Principal

BASE PRINCIPAL	COMPONENTES
Descripción	**Cantidad**
Remaches	28 und
Agarraderas	2 und
Portafusibles	1 und
Fusibles	1 und
Interruptor	1 und
Toma de cable de poder	1 unf
Bisagras	6 und
Tornillos	86 und
Tuercas	86 und
Ángulos de Aluminio 3/4 (31.3 x 2.3)	16 pza
"T" de Aluminio 3/4 (24.7 x 1.5)	2 pza
Alucobond (15.7 x 30.3)	2 pza
Alucobond (29.4 x 49.2)	1 pza
Alucobond (15.7 x 50.4)	2 pza
Fibra Sintética (30.3 x 50.4)	1 pza

NOTA: Detalle de materiales Base principal. Los Autores

Figura 3.1 Base principal, Medidas

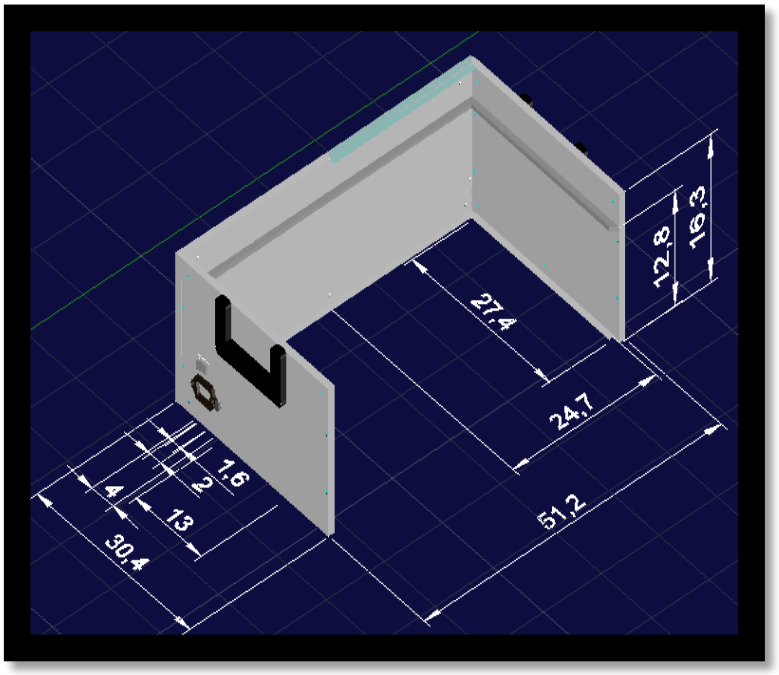


Figura: Vista general base principal. Los autores

b) Tapa A

Esta sección ubicada en la parte superior contiene el soporte sobre el cual será montada la Tablet (dispositivo móvil). Las partes que la componen se detallan a continuación:

Tabla 3.2 Inventario Tapa A

Tapa A	**Componentes**
Descripción	**Cantidad**
Agarraderas	1 und
Soporte Tablet	1 und
Tornillos	25 und
Tuercas	25 und
Remaches	23 und
HEBILLA METÁLICA (lado B)	2 und
Ángulos de Aluminio 3/4 (13.5 x 1.8)	8 pza
"T" de Aluminio 3/4 (2.3 x 15.5)	3 pza
Alucobond (14.7 x 15.2)	3 pza
Fibra Sintética (30.3 x 50.4)	3 pza

NOTA: Elementos Tapa A. Los Autores

Figura 3.2 Tapa A, Medidas

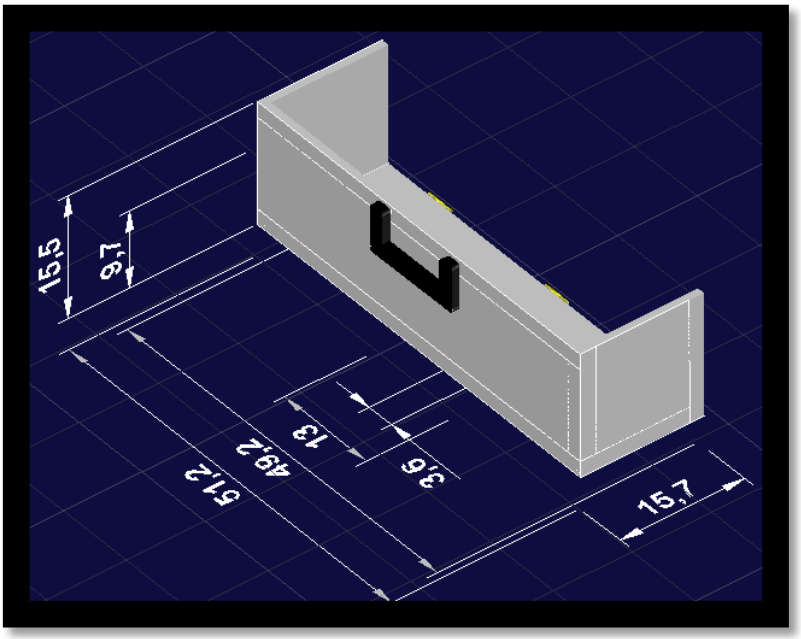


Figura: Vista general Tapa A. Los autores

c) Tapa B

Cumple la función de soporte para el computador portátil cuando el modulo didáctico es abierto. Los elementos que la constituyen se enlistan a continuación:

Tabla 3.3 Inventario Tapa B

Tapa B	**Componentes**
Descripción	**Cantidad**
Agarraderas	1 und
Tornillos	24 und
Tuercas	24 und
Remaches	19 und
HEBILLA METÁLICA (lado A)	2 und
Ángulos de Aluminio 3/4 (47.5 x 2.5)	9 pza
Alucobond (15 x 15)	3 pza
Fibra Sintética (13.5 x 14)	5 pza

NOTA: Elementos Tapa B. Los Autores

Figura 3.3 Tapa B, Medidas

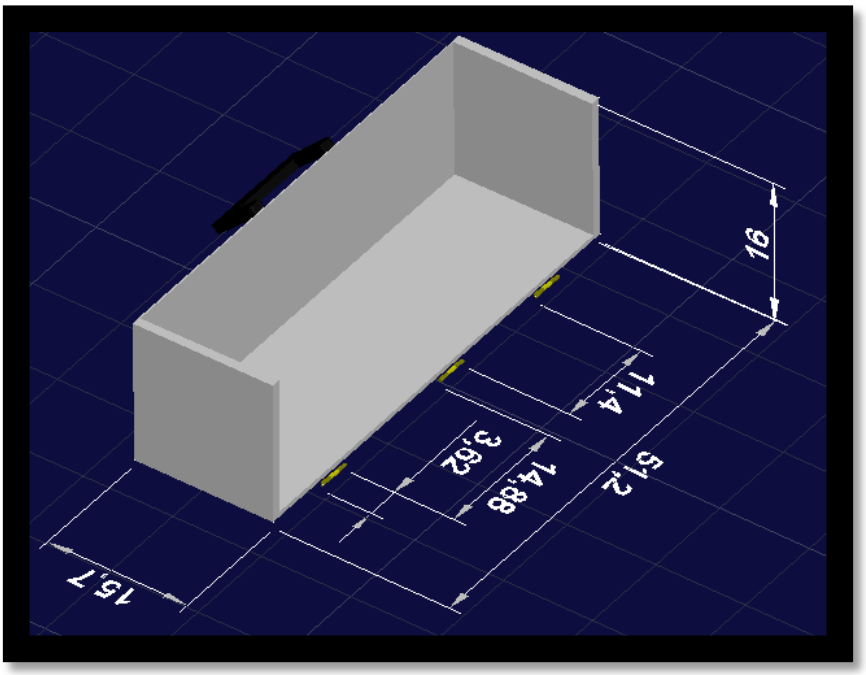


Figura: Vista general Tapa B. Los autores

d) Panel

El panel fue diseñado para proveer al estudiante los elementos y herramientas que le permitan desarrollar de manera correcta las prácticas que se proponen en el presente proyecto. Consta de borneras para alimentación de voltaje (DC y AC) y de líneas de protoboard en las que el estudiante puede simular practicas a su preferencia. A continuación los elementos que constituyen el panel:

Tabla 3.4 Inventario Panel

PANEL	**COMPONENTES**
Descripción	**Cantidad**
Borneras rojas	6 und
Borneras negras	3 und
Tomacorrientes	1 und
Perilla (Potenciómetro)	2 und
Display	1 und
Arduino Uno	1 und
Protoboard	6 und
Agarradera	1 und
Tornillos	14 und
Tuercas	14 und
Alucobond	1 pza
Fibra	1 pza

NOTA: Elementos Panel principal. Los Autores

El panel consta de tomas de voltaje de las siguientes características:

- Voltaje DC variable : 12 VDC
- Voltaje AC variable: 12 VAC
- Voltaje DC fijo: +5, -5, +12, -12
- Tomacorriente para alimentación: 120 VAC

Figura 3.4 Panel, Medidas

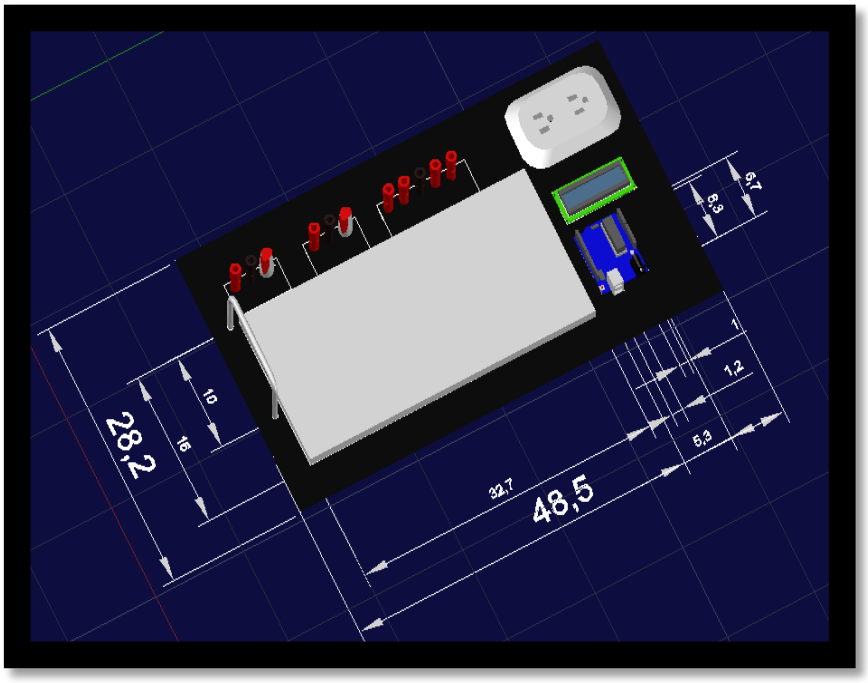


Figura: Vista general del Panel principal. Los autores

e) Bandeja

Tiene la función de almacenar de manera ordenada los elementos que serán utilizados en cada una de las prácticas propuestas en el presente proyecto.

Tabla 3.5 Inventario bandeja

Bandeja	**COMPONENTES**
Descripción	**Cantidad**
Bisagras	2 und
Tornillos	10 und
Tuercas	10 und
Remaches	27 und
Ángulos de Aluminio 3/4 (6 x 1.2)	11 pzas
"T" de Aluminio 3/4 (3.7 x 1.5)	2 pzas
Alucobond (27.5 x 23.3)	12 pzas
Fibra Sintética (23.6 x 28)	1

NOTA: Elementos Bandeja de materiales. Los autores

Figura 3.5 Bandeja, Medidas

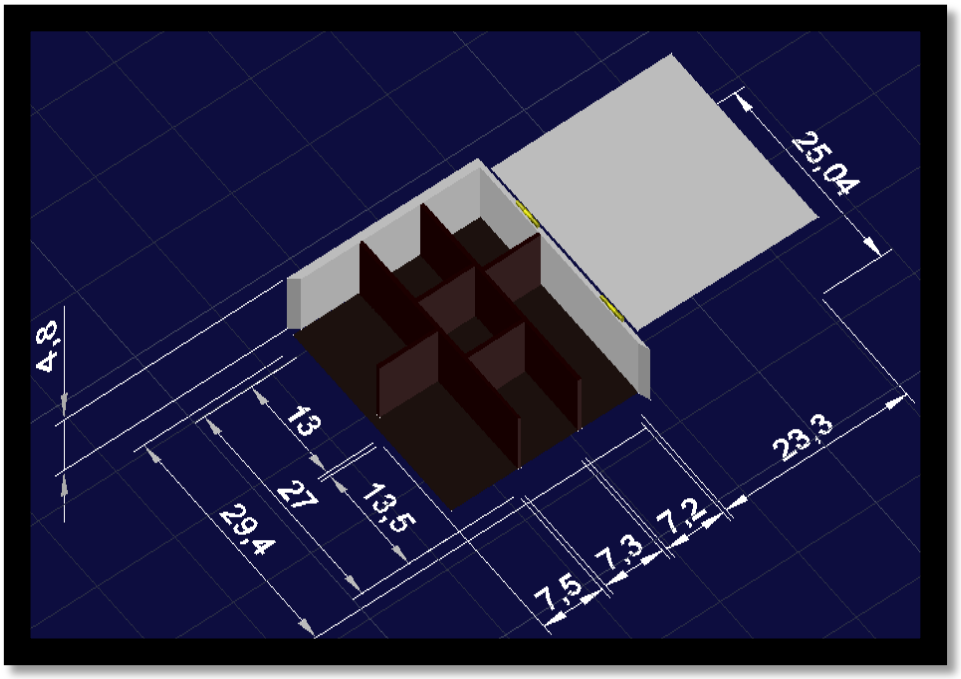


Figura: Vista general de bandeja de elementos. Los autores

Figura 3.6 Maletín Ensamblado

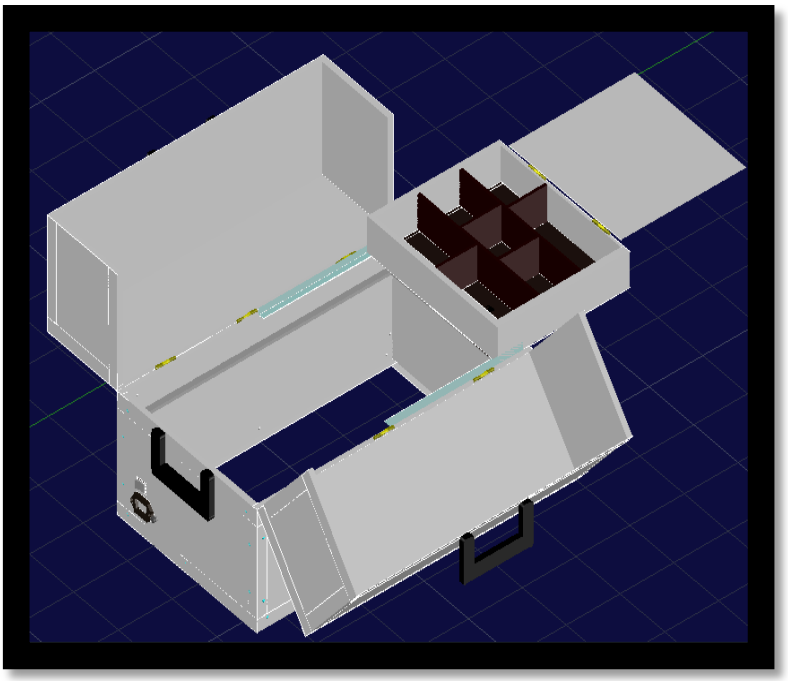


Figura: Vista general del módulo ensamblado. Los autores

Figura 3.7 Panel

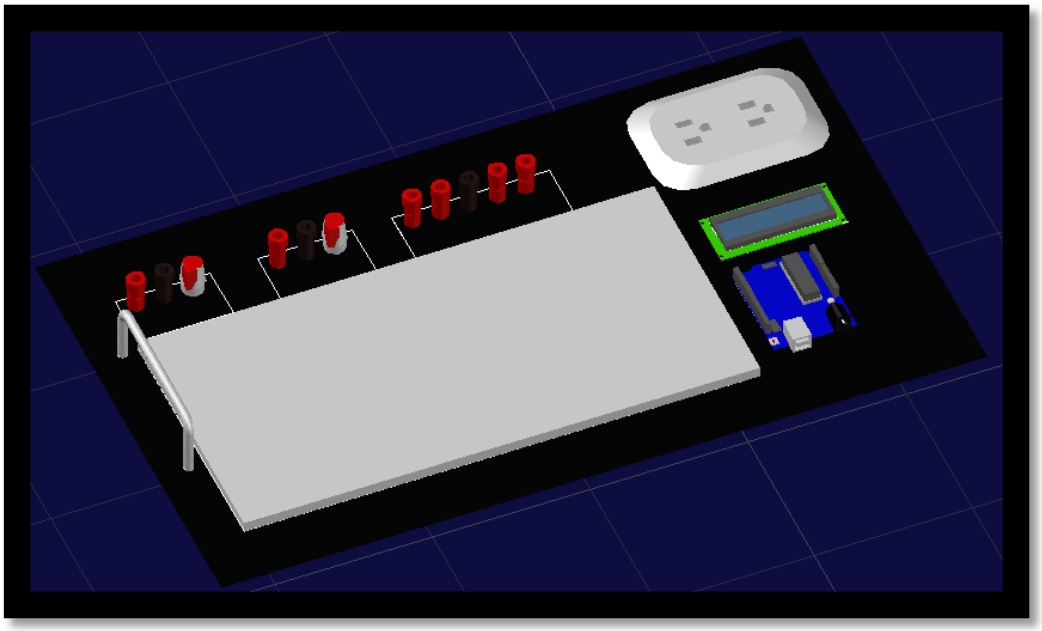


Figura: Vista final del panel de prácticas. Los autores

3.2 Diseño y Construcción de las Fuentes de Poder

El modulo didáctico está equipado con fuentes de poder de diferentes características que ayudaran y facilitarán al estudiante la alimentación de poder para los diferentes elementos que conforman las prácticas. Las fuentes de poder construidas son las siguientes:

a) Fuente AC variable 0 – 12 VAC
b) Fuente DC variable 1.25 – 12 VDC
c) Fuente fija DC + 5 VDC -5 VDC
d) Fuente fija DC +12 VDC -12 VDC

a) Fuente AC variable 0 – 12 VAC

Para el diseño y construcción de la fuente AC se utilizó el principio del divisor de voltaje.

$$V_{out} = \frac{V1 \,.\, R2}{R1 + R2}$$

Estableciendo R1 con un valor de 100 K, y conociendo el valor de V1 24 v, que es el voltaje que nos suministra el transformador; definimos R2 con un potenciómetro de un valor similar de 100k, con lo cual la variación provocada por el divisor de voltaje nos permitirá establecer valores de voltaje en la salida de entre 0 y 12 Voltios AC.

$$\textbf{Vout (50\%)} = \frac{\textbf{24 v. * 50k}}{\textbf{100k + 50k}} = \textbf{8 v}$$

$$\textbf{Vout (75\%)} = \frac{\textbf{24 v. * 75k ohm}}{\textbf{100k + 75k}} = \textbf{10.28 v}$$

$$\textbf{Vout (100\%)} = \frac{\textbf{24 v. * 100k ohm}}{\textbf{100k + 100k}} = \textbf{12 v}$$

Elementos utilizados:

- Resistencia 100k (1 und)
- Potenciómetro 100k (1 und)
- Transformador 24 – 0 – 24 (1 und)

Figura 3.8 Fuente Ac Variable diagrama Esquemático

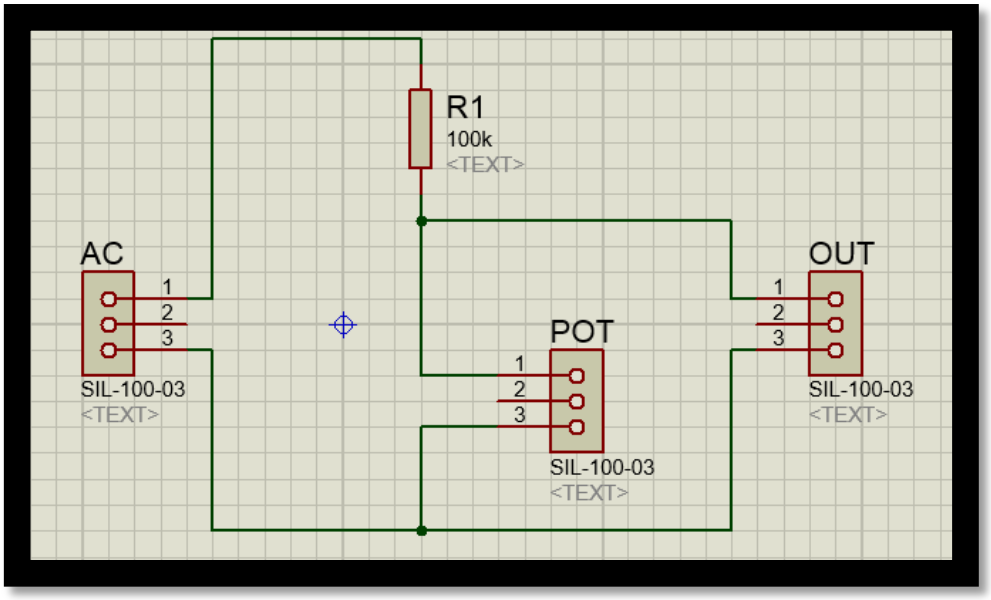


Figura: Diseño esquemático fuente AC variable. Los autores

Figura 3.9 Fuente Ac Variable Layout

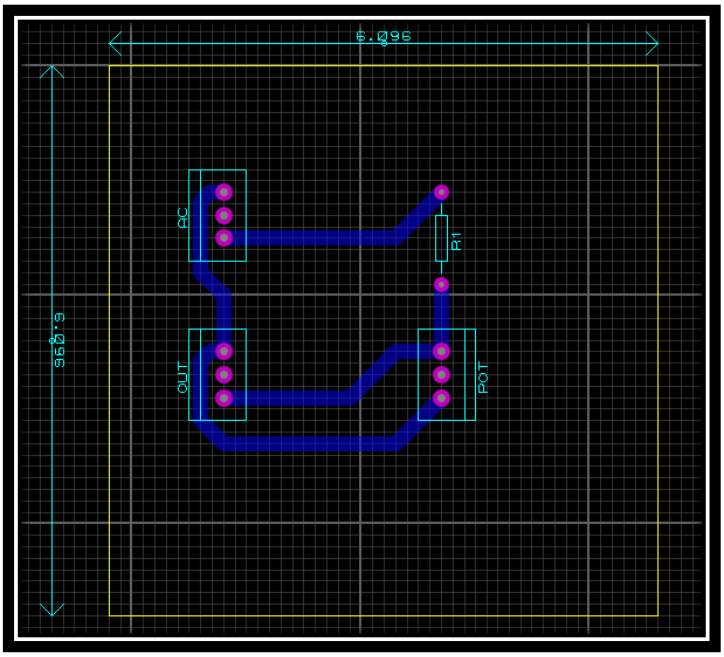


Figura: Vista circuito impreso fuente AC. Los autores

Figura 3.10 Fuente Ac Variable 3D

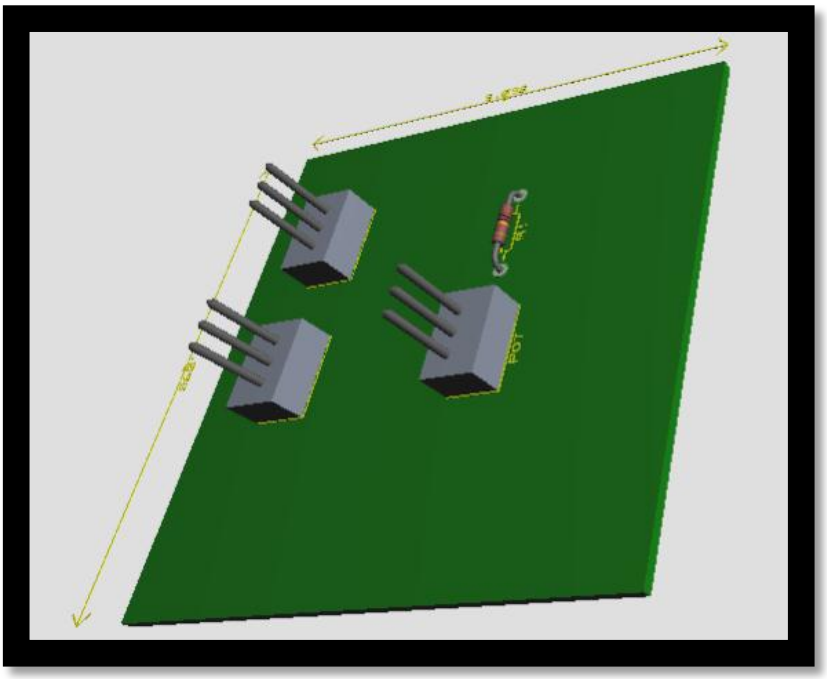


Figura: Vista 3D de fuente AC variable. Los autores

b) Fuente DC variable 1.25 – 12 VDC

Para la fuente de voltaje DC variable utilizamos un circuito integrado regulador de la serie LM31X.

El CI LM317 es un regulador de voltaje positivo de tres terminales, el cual puede proporcionar 1.5 A de corriente en su salida, y voltajes variables desde 1.25 a 37 Voltios. La hoja de característica de este regulador nos da la siguiente ecuación para establecer el voltaje de salida:

$$V_{out} = \frac{R1 + R2}{R1} V_{ref} + I_{ADJ} R2$$

En esta ecuación V_{ref} tiene un valor de 1.25 Voltios e I_{ADJ} un valor de 50 uA, lo que reduce la ecuación anterior a lo siguiente:

$$V_{out} = \frac{R1 + R2}{R1} V_{ref}$$

Y para nuestro circuito hemos establecido R1 a 100k y el Voltaje de salida Vout máximo a 12 voltios. Lo cual nos da como resultado un Potenciómetro de 5 K. a modo de protección contra sobrecorrientes (cortocircuito) hemos implementado diodos rectificadores a las salidas del circuito.

Elementos utilizados:

- Resistencia 100k (1 und)
- Potenciómetro 5k (1 und)
- Regulador de Voltaje positivo LM317 (1 und)
- Transformador 24 – 0 – 24 (1 und)
- Capacitor 100uf (1und)
- Capacitor 100pf (1 und)
- Diodo rectificador 1N4007 (2 und)

Figura 3.11 Fuente DC Variable diagrama esquemático

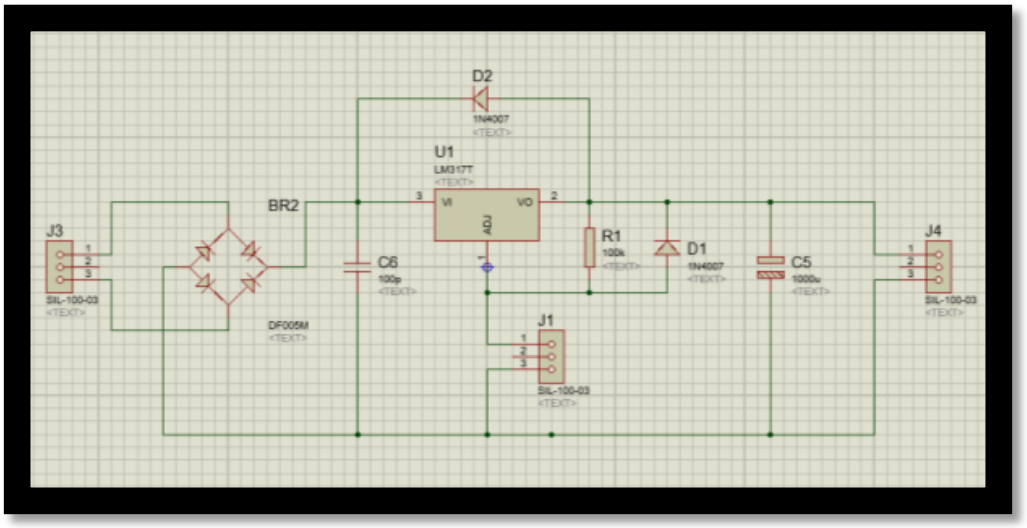


Figura: Vista esquemática fuente DC variable. Los autores

Figura 3.12 Fuente Dc Variable Layout

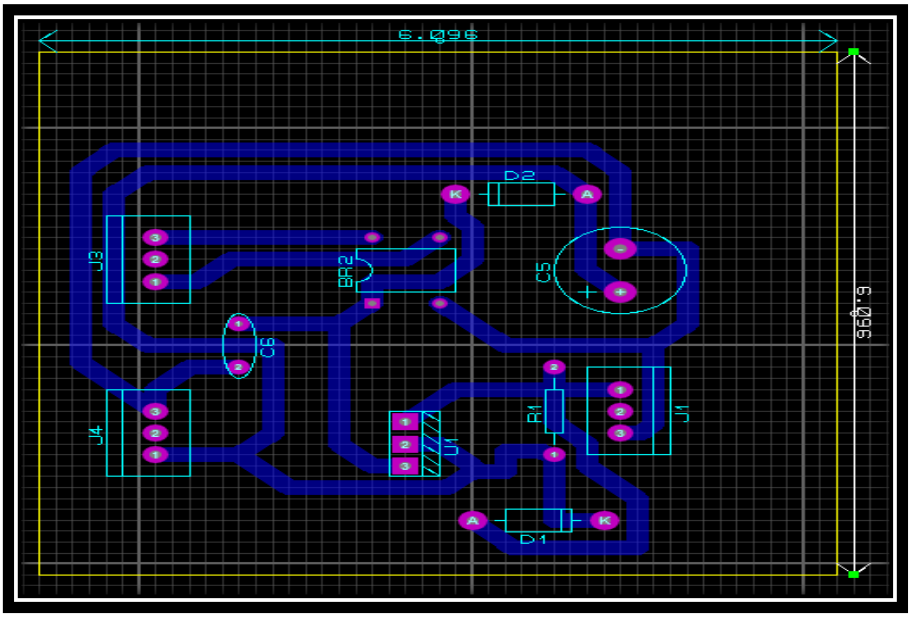


Figura: Vista circuito impreso fuente DC. Los autores

Figura 3.13 Fuente Dc Variable 3D

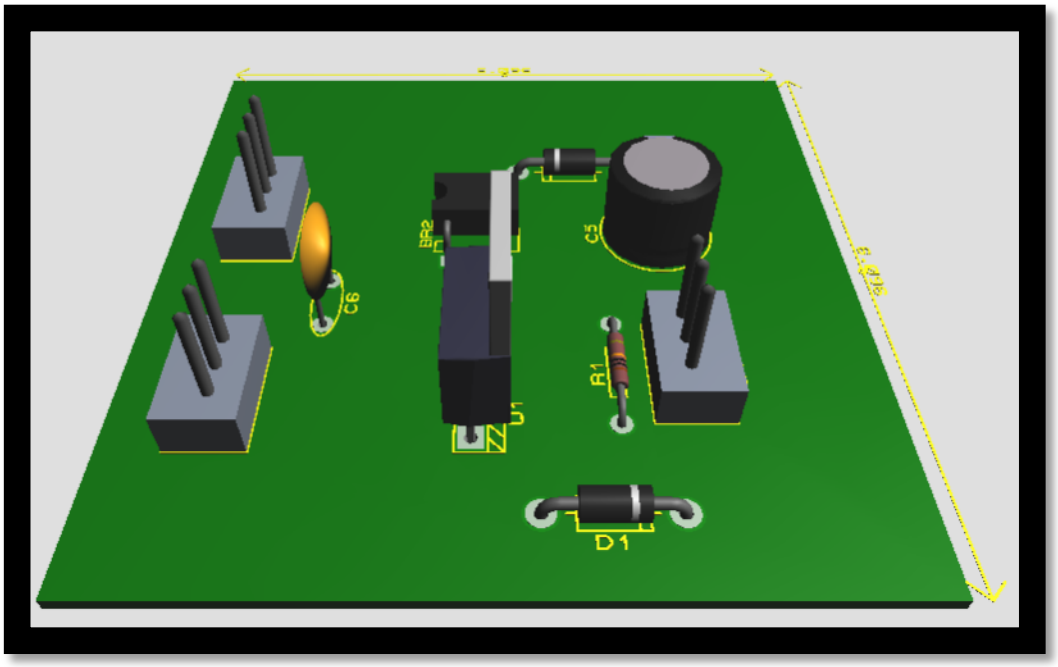


Figura: Vista 3D fuente DC variable. Los autores

c) Fuente fija DC + 5 VDC +12 VDC

Para la construcción de las fuentes fijas positivas, hemos utilizado reguladores fijos de voltaje de la serie LM78XX, el regulador LM7805 produce una salida de +5v, el LM7812 produce una salida de +12v y serán utilizados en nuestras fuentes de poder.

Cuando un CI está conectado a unos cuantos centímetros del filtro capacitivo de la fuente de regulación se pueden producir oscilaciones, por lo cual utilizamos la recomendación del fabricante de utilizar condensadores de desacoplo en la entrada y en la salida, con valores menores a 1uf.

Elementos utilizados:

- Regulador de Voltaje no regulable positivo LM7805 (1 und)
- Regulador de Voltaje no regulable positivo LM7812 (1und)
- Transformador 12 – 0 – 12 (1 und)
- Capacitor 0.1uf (2und)
- Capacitor 0.22uf (2 und)

Figura 3.14 Fuentes 5v 12v diagrama esquemático

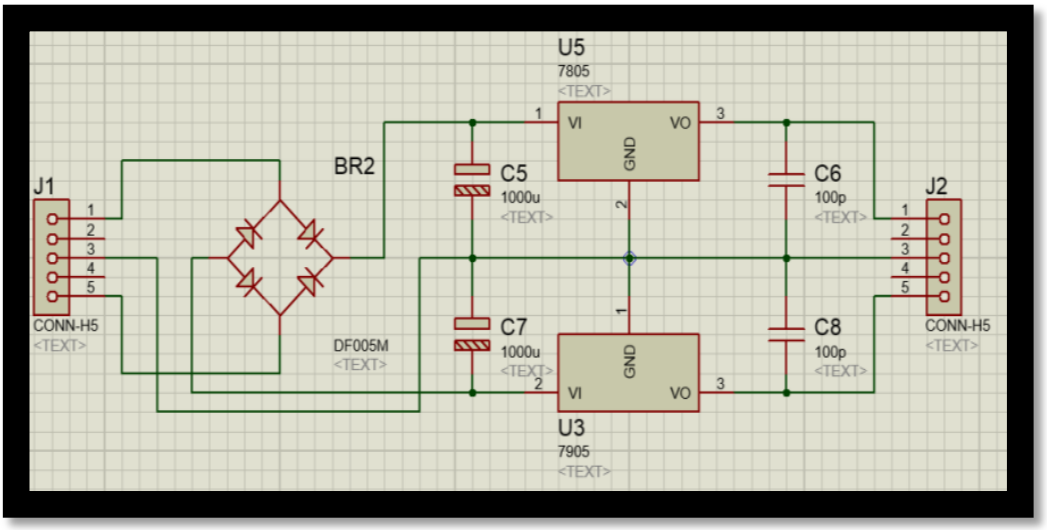


Figura: Vista esquemática fuentes fijas 5 y 12v. Los autores

Figura 3.15 Fuentes 5v 12v Layout

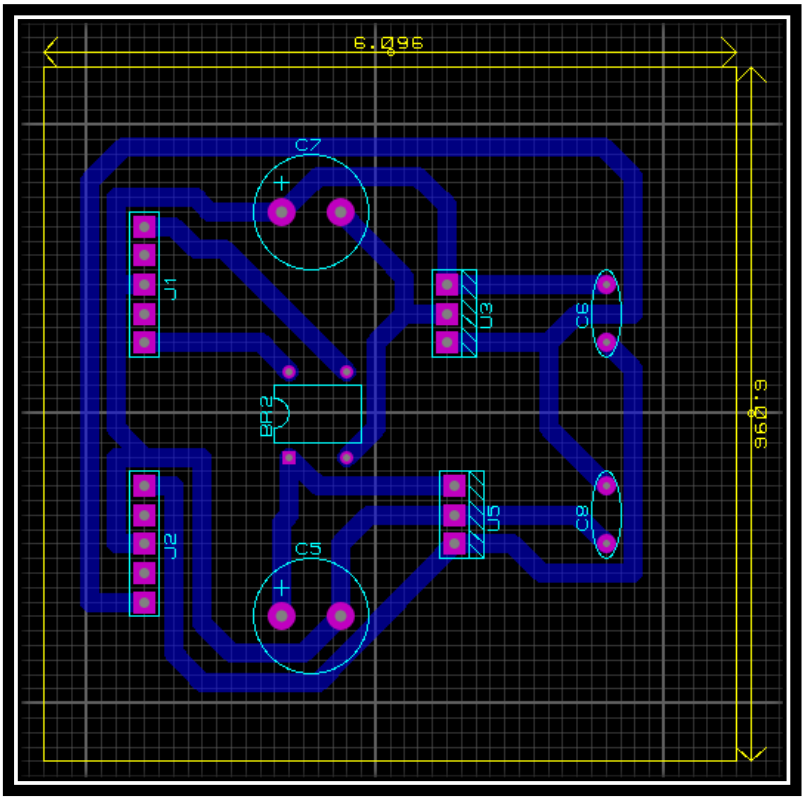


Figura: Vista circuito impreso fuentes fijas positivas. Los autores

Figura 3.16 Fuentes 5v 12v 3D

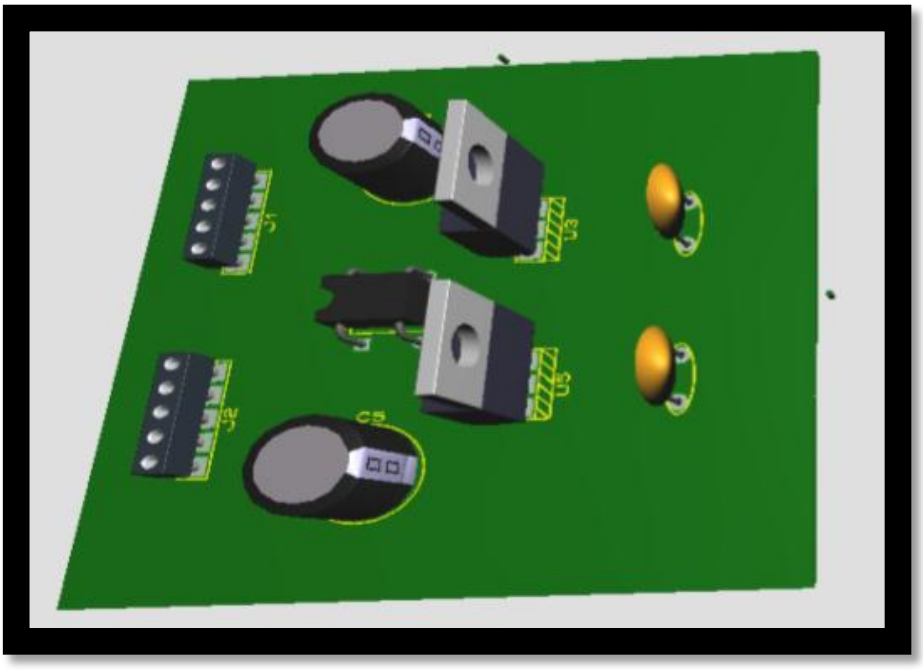


Figura: Vista 3D fuentes fijas. Los autores

d) Fuente fija DC -5 VDC -12 VDC

Para la construcción de las fuentes fijas negativas, hemos utilizado reguladores fijos de voltaje de la serie LM79XX, el regulador LM7905 produce una salida de -5v, el LM7912 produce una salida de -12v.

Los elementos utilizados para las fuentes fijas negativas son:

- Regulador de Voltaje no regulable negativo LM7905 (1 und)
- Regulador de Voltaje no regulable negativo LM7912 (1und)
- Transformador 12 – 0 – 12 (1 und)
- Capacitor 0.1uf (2und)
- Capacitor 0.22uf (2 und)

Figura 3.17 Fuentes (Uniones) Diagrama Esquemático

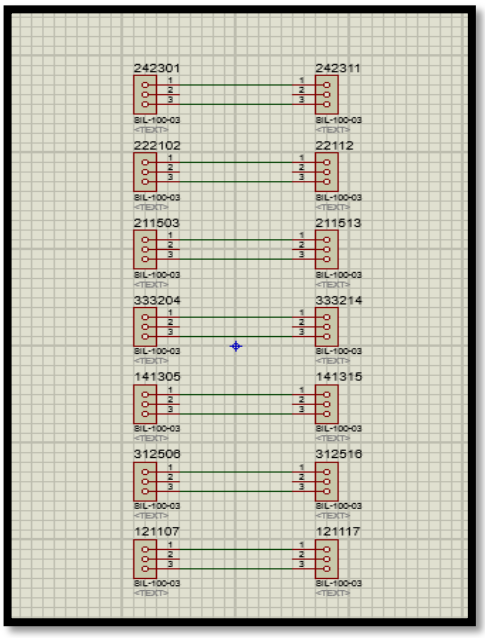


Figura: Vista esquemática circuito de uniones. Los autores

Figura 3.18 Fuentes (Uniones) Layout

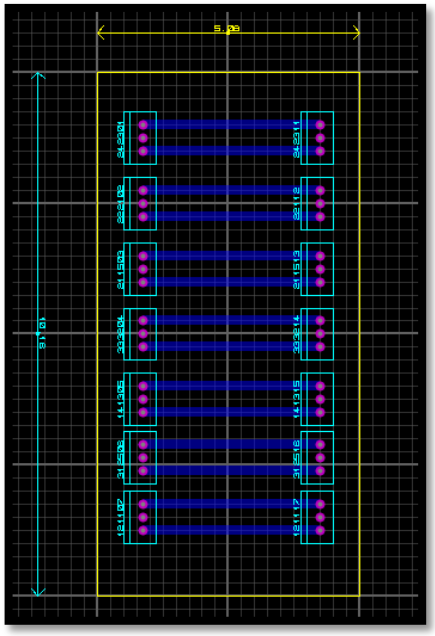


Figura: Vista circuito impreso uniones. Los autores

Figura 3.19 Fuentes (Uniones) 3D

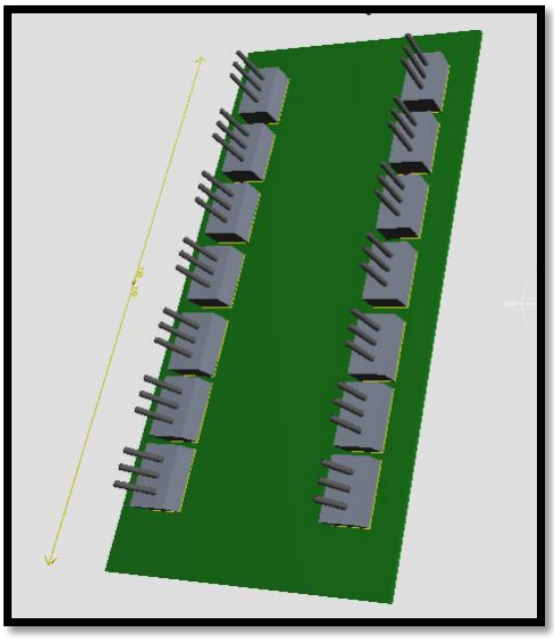


Figura: Vista 3D uniones. Los autores

3.3 Conexiones

Tabla 3.6 Conexiones fuentes de poder modulo

Tabla de Conexiones		
Tarjeta de Conexión	**Pin**	**Identificador**
Tarjeta fuente +5V -5V	Fuente -5V	[21]
	GND	[22]
	Fuente +5V	[15]
Tarjeta fuente +12V -12V	Fuente+12V	[23]
	GND	[22]
	Fuente -12V	[24]
Tarjeta fuente Variable Dc +12V	- Fuente Variable Dc 12V	[12]
	+ Fuente Variable Dc 12V	[11]
	Potenciómetro pin 1 Fuente Variable Dc 12V	[25]
	Potenciómetro pin 2 Fuente Variable Dc 12V	[31]
Tarjeta fuente Variable Ac +12V	- Fuente Variable Ac 12V	[14]
	+ Fuente Variable Ac 12V	[13]
	Potenciómetro pin 1 Fuente Variable Ac 12V	[32]
	Potenciómetro pin 2 Fuente Variable Ac 12V	[33]
Tarjeta de Conexiones 1	Fuente -5V	[21]
	GND	[22]
	Fuente +5V	[15]
	Fuente+12V	[23]

	GND	[22]
	Fuente -12V	[24]
	- Fuente Variable Dc 12V	[12]
	+ Fuente Variable Dc 12V	[11]
	Potenciómetro pin 1 Fuente Variable Dc 12V	[25]
	Potenciómetro pin 2 Fuente Variable Dc 12V	[31]
	- Fuente Variable Ac 12V	[14]
	+ Fuente Variable Ac 12V	[13]
	Potenciómetro pin 1 Fuente Variable Ac 12V	[32]
	Potenciómetro pin 2 Fuente Variable Ac 12V	[33]
Tarjeta de Conexiones 2	Fuente -5V	[21]
	GND	[22]
	Fuente +5V	[15]
	Fuente+12V	[23]
	GND	[22]
	Fuente -12V	[24]
	- Fuente Variable Dc 12V	[12]
	+ Fuente Variable Dc 12V	[11]
	Potenciómetro pin 1 Fuente Variable Dc 12V	[25]
	Potenciómetro pin 2 Fuente Variable Dc 12V	[31]
	- Fuente Variable Ac 12V	[14]
	+ Fuente Variable Ac 12V	[13]
	Potenciómetro pin 1 Fuente Variable Ac 12V	[32]
	Potenciómetro pin 2 Fuente Variable Ac 12V	[33]

NOTA: Detalle de las conexiones de las fuentes de poder. Los Autores

CAPÍTULO IV

MÓVIL EXPLORADOR

4. Móvil Explorador

4.1 Diseño

En la construcción de nuestro móvil explorador hemos utilizado la configuración de robot móvil tipo diferencial, en la cual cada rueda es controlada por un motor siendo así que el giro del robot queda determinado por la diferencia de velocidad y sentido de giro de las ruedas.

A continuación detallamos cada uno de los elementos que constituyen el móvil explorador:

Motor reductor

Permitirán el desplazamiento del móvil, siendo instalado uno en cada rueda y conectando en paralelo los correspondientes a cada lado del móvil explorador.

Figura 4.1 Motor reductor Medidas

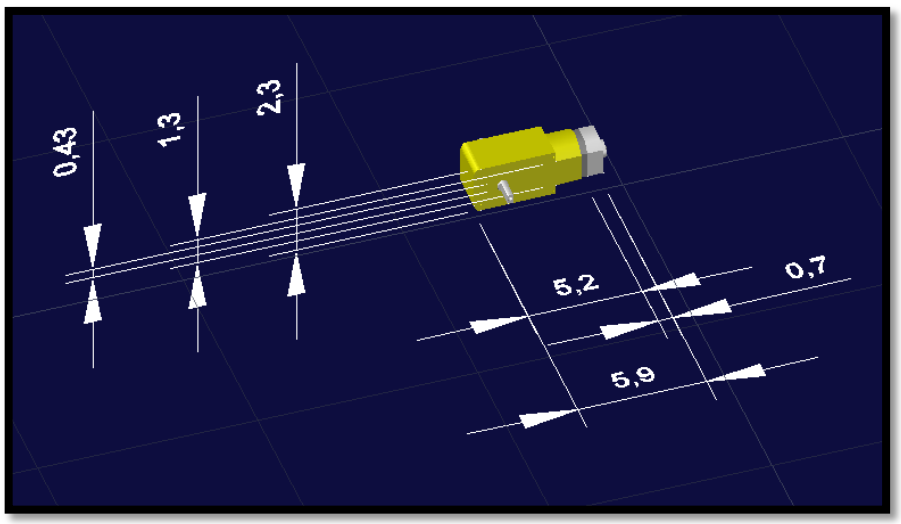


Figura: Vista general de motor reductor. Los autores

Arduino

Es el encargado del control de cada elemento que conforma el móvil.

Figura 4.2 Arduino Uno Medidas

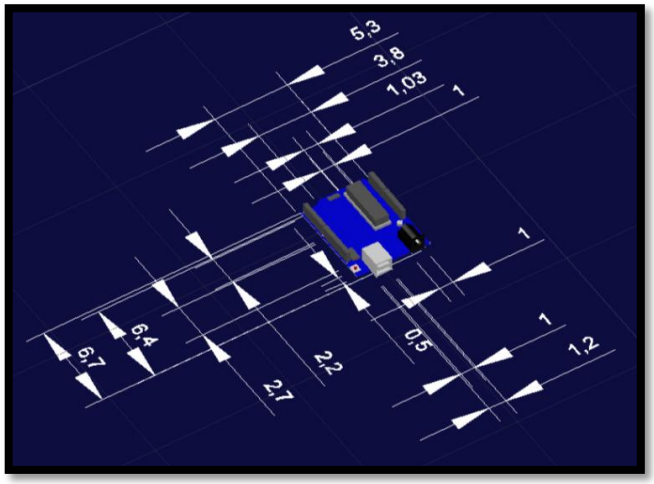


Figura: Detalle gráfico de Arduino UNO. Los autores

Batería

Es la encargada de suministrar la potencia (voltaje) que alimentará los elementos a través de la fuente de poder (circuito regulador de voltaje) establecido para cada uno de ellos.

Figura 4.3 Batería (6V-4Amp) Medidas

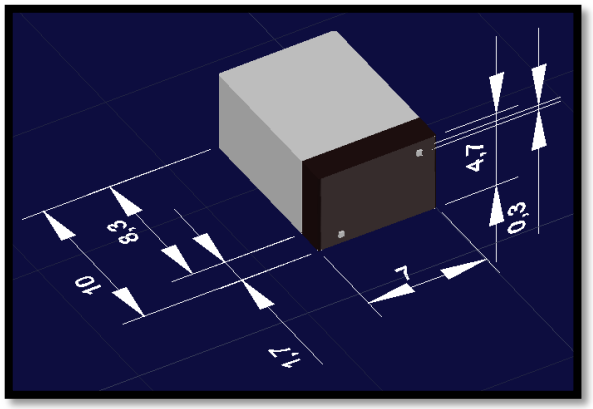


Figura: Vista general batería DC. Los autores

Servomotor

El servomotor de movimiento angular permitirá desplazar la cámara IP para poder obtener una visualización de 180° del entorno recorrido.

Figura 4.4 Servomotor Futaba S3003 Medidas

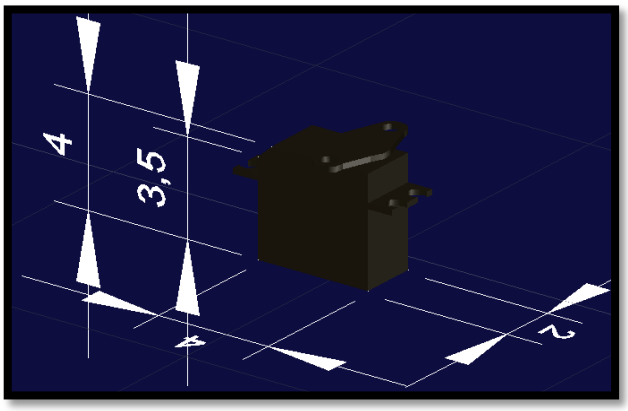


Figura: Vista detallada servomotor s300. Los autores

Cámara IP

El elemento que permite la visualización en tiempo real de las imágenes obtenidas por el móvil explorador.

Figura 4.5 Cámara D-Link Dcs-930L Medidas

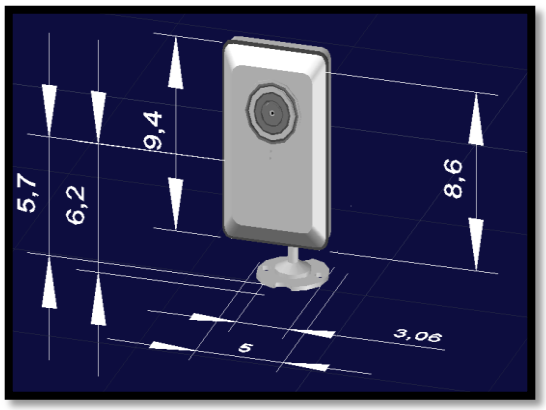


Figura: Vista general Cámara IP. Los autores

Panel Posterior

Cumple la función de activar o desactivar las funcionalidades del móvil explorador. Del mismo modo permite realizar la carga de las baterías que suministran el voltaje a los elementos.

Figura 4.6 Panel Posterior

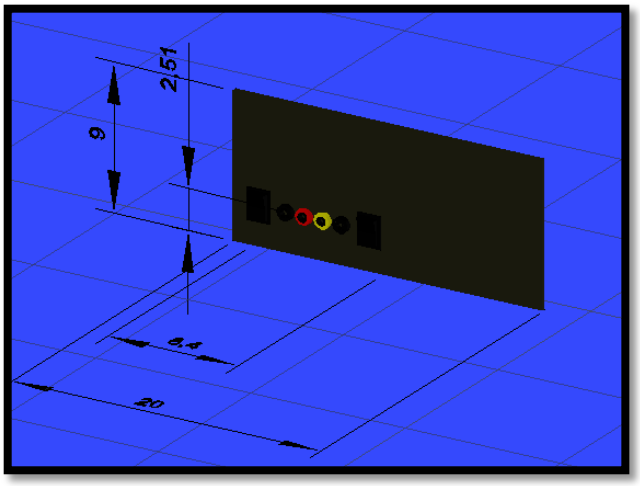


Figura: Vista general panel posterior móvil explorador. Los autores

Fuentes y Tarjetas de Control

Se diseñaron e implementaron fuentes reguladas de voltaje para cada elemento del móvil explorador, con lo cual se distribuye de manera eficiente el voltaje suministrado por la batería.

Se utilizaron para esto CI 7805 para crear una fuente regulada de voltaje para cada elemento, los circuitos y sus diagramas esquemáticos se detallan a continuación:

- Fuente regulada 5v para servomotor
- Fuente regulada 5v para Arduino
- Fuente regulada 5v para HC-06 y DHT-11
- Fuente regulada 5v para luces

Figura 4.7 Robot1 Diagrama Esquemático

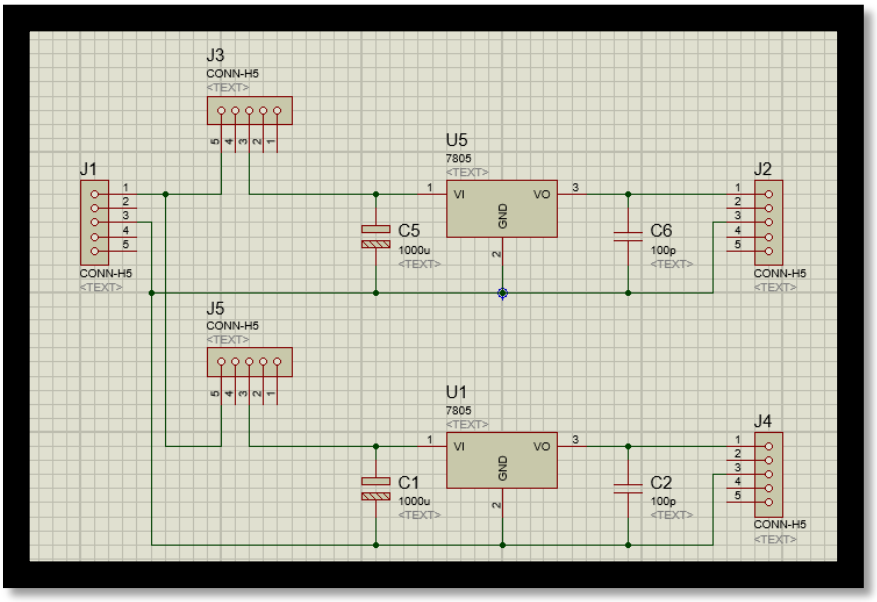


Figura: Vista esquemática de circuitos de robot. Los autores

Figura 4.8 Robot1 Layout

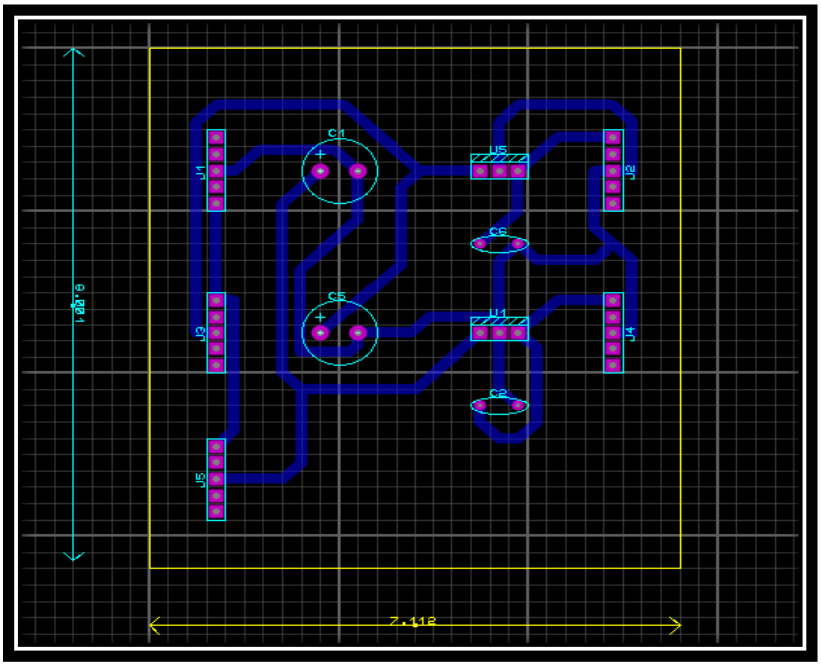


Figura: Vista circuito impreso fuentes robot. Los autores

Figura 4.9 Robot1 3D

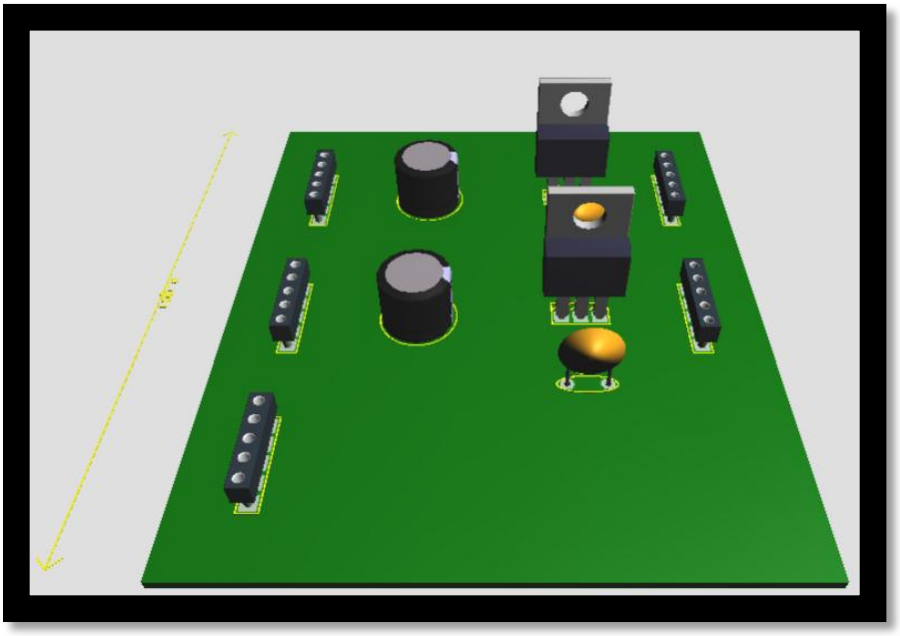


Figura: Vista 3D de fuentes robot. Los autores

Figura 4.10 Robot2 Diagrama Esquemático

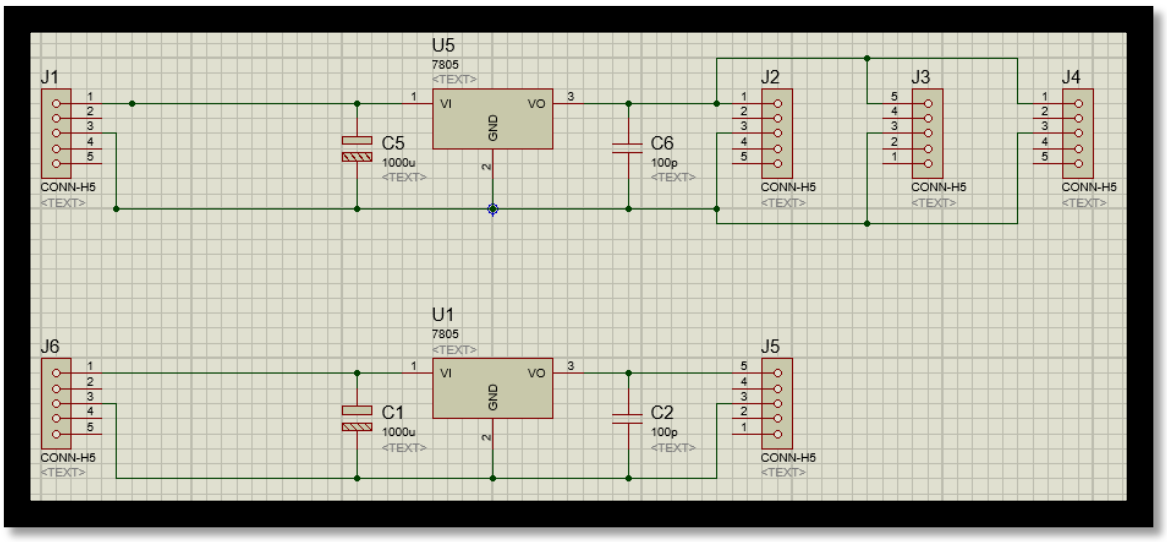


Figura: Vista esquemática circuito fuentes 2 robot. Los autores

Figura 4.11 Robot2 Layout

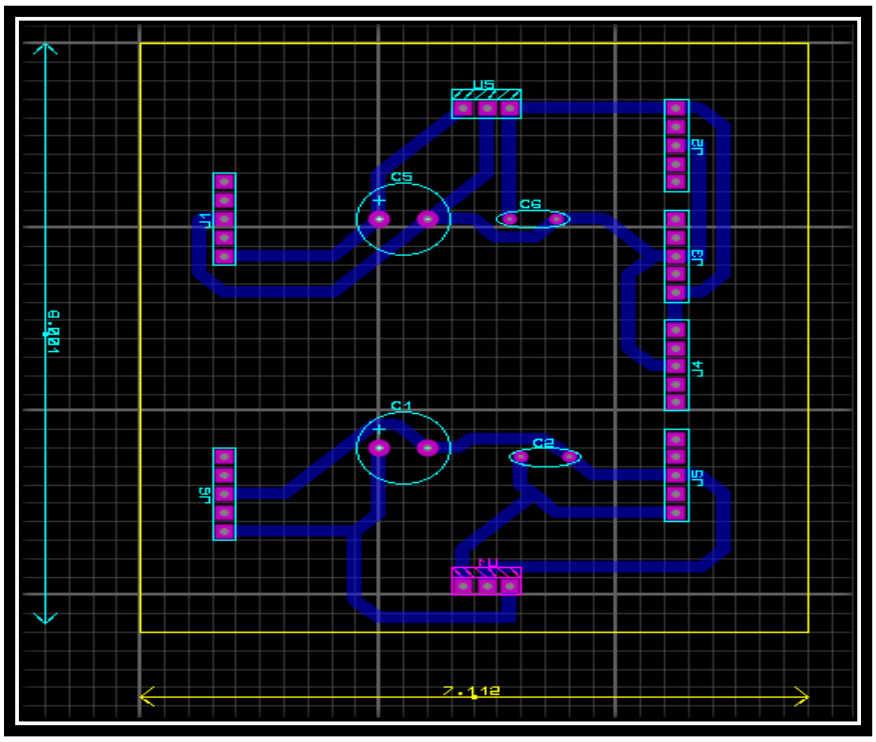


Figura: Vista circuito impreso fuente 2 robot. Los autores

Figura 4.12 Robot2 3D

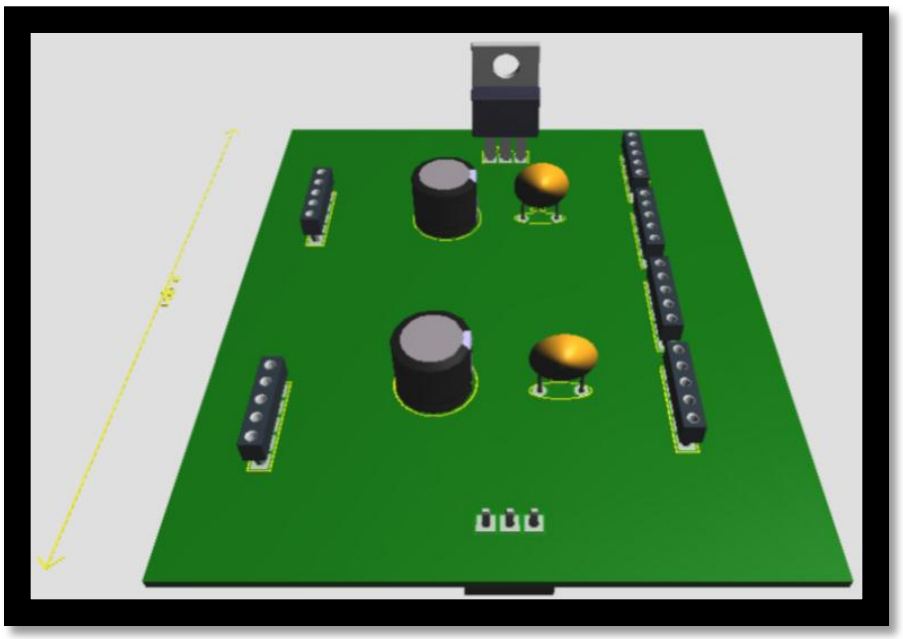


Figura: Vista 3D fuente 2 Robot. Los autores

Para el manejo de los motores se utilizó un circuito de control con el CI L293D, el cual suministra de manera eficiente el voltaje a cada motor reductor a la vez que establece el sentido de giro y velocidad:

Figura 4.13 Robot3 Diagrama Esquemático

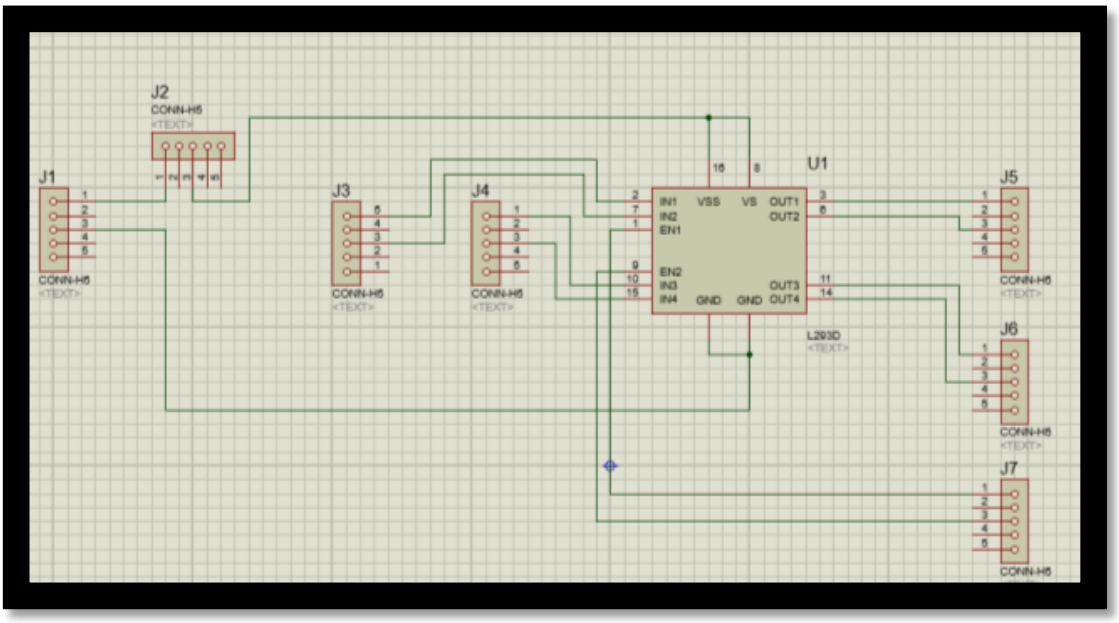


Figura: Vista esquemática drive motores robot. Los autores

Figura 4.14 Robot3 Layout

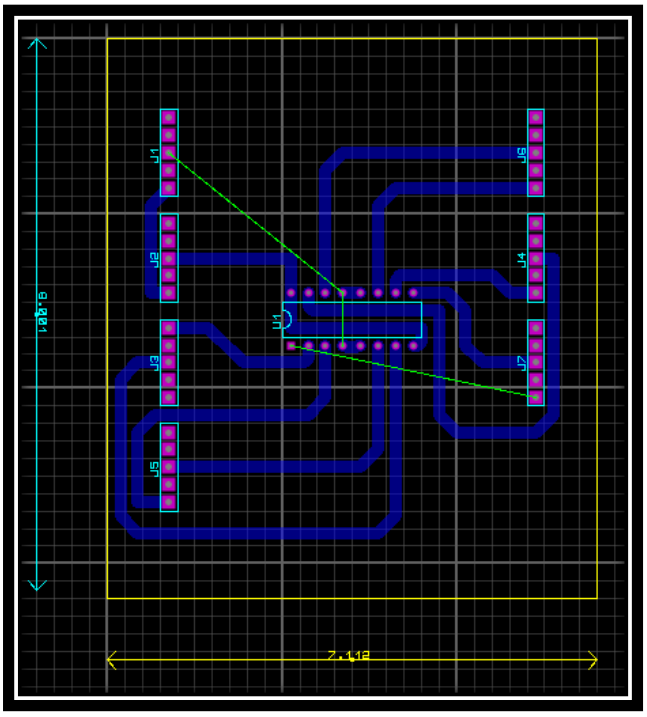


Figura: Vista circuito impreso drive motores. Los autores

Figura 4.15 Robot3 3D

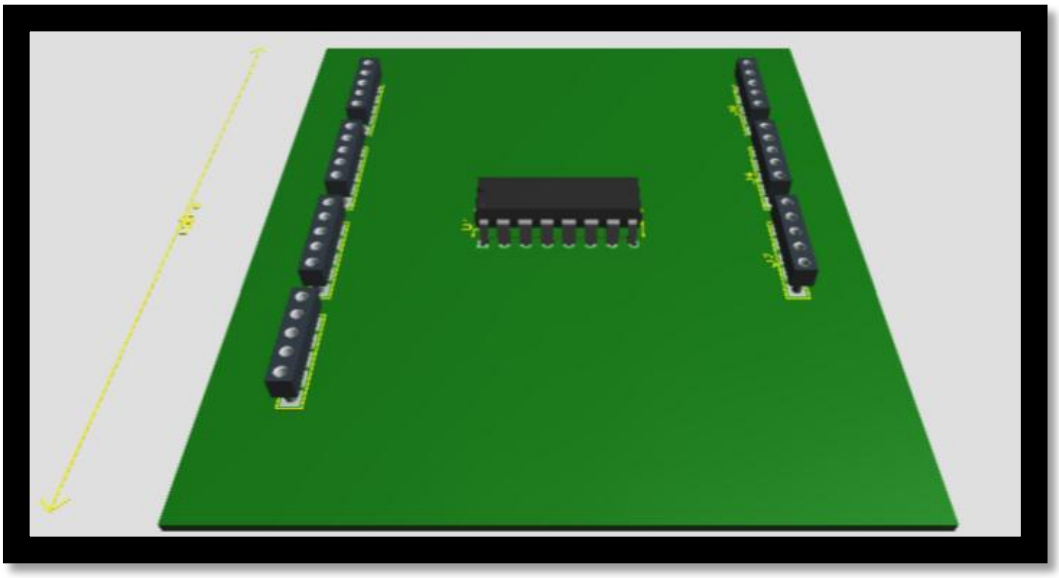


Figura: Vista 3D drive motores robot. Los autores

Figura 4.16 Ensamblaje del robot

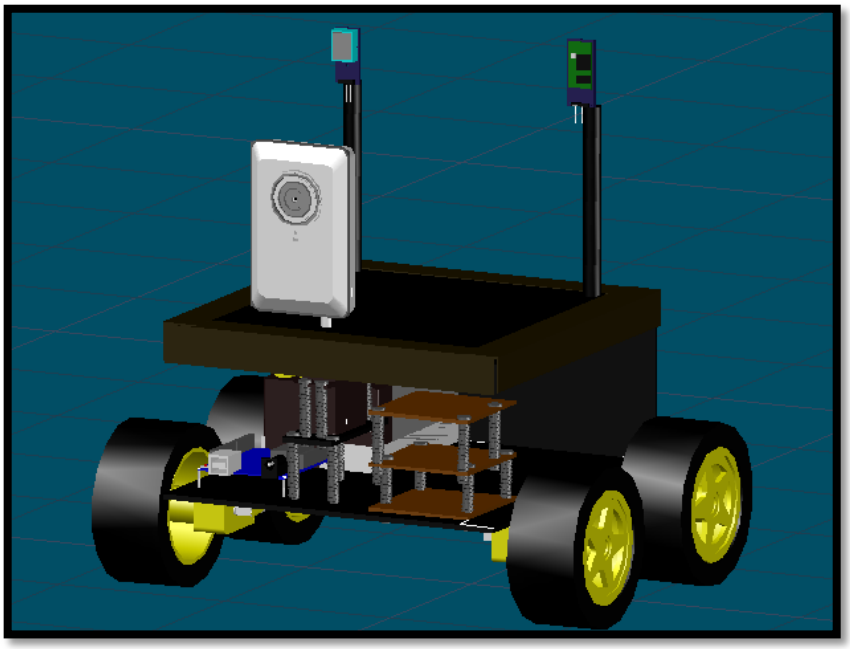


Figura: Vista final robot. Los autores

4.2 Materiales

En la siguiente tabla se detallan los elementos utilizados en el móvil explorador:

Tabla 4.1 Materiales Robot

Robot	**Componentes**
Descripción	**Cantidad**
Moto reductores 6v	4
Llantas (motor reductores)	4
Arduino UNO	1
Batería LRCD NI-CD AA700mhA 9-6V	1
Batería RB 4-6 4Amp 6Volt Rechargeable	1
Servomotor Futaba s3003	1
Network Camera D-Link	1
Faros Lámpara Led	2
borneras color negro	2
borneras color amarillo	1
borneras color rojo	1
Tarjeta controladora de ruedas	1
Tarjeta fuentes reguladoras de voltaje	2
Tarjetas de borneras para conexión	3
Switch on/off	2
Antenas de madera cilíndrica (10,5x 1)	2
Bluetooth HC 06	1
Sensor de Temperatura y Humedad DHT11	1
Tornillos	59
Tuercas	59
acrílico de 20x20	2
acrílico de 20 x 9	4
Ángulos de Aluminio 3/4 (1.8x20)	8
Conector voltaje	2

NOTA: Detalle de materiales utilizados en la construcción del robot. Los Autores

4.3 Conexiones

A modo de guía para entender la forma en que se interconectan las diferentes tarjetas electrónicas de nuestro móvil explorador, detallamos a continuación las conexiones que lo conforman con sus respectivos identificadores.

Tabla 4.2 Conexiones Robot

Tabla de Conexiones Móvil Explorador		
Tarjeta de Conexión	**Pin**	**Identificador**
Tarjeta Motor	GND	14
	VCC	15
	IN1	16
	IN2	17
	OUT1	18
	OUT2	19
	IN3	20
	IN4	21
	OUT3	22
	OUT4	23
	ENABLE1	24
	ENABLE2	25
Tarjeta fuente 1	GND	26
	VCC	27
	GND	28
	VCC	29
	Bluetooth GND	30
	Bluetooth VCC	31
	DHT-11 VCC	32
	DHT-11 GND	33
	CAM GND	34
	CAM VCC	35
Tarjeta fuente 2	GND	36
	VCC	37
	SERVO VCC	38
	SERVO GND	39
	ARDUINO VCC	40
	ARDUINO GND	41

Tarjeta conexiones	Luz izquierda VCC	42
	Luz izquierda GND	43
	DHT11 GND	44
	DHT11 DATO	45
	DHT11 VCC	46
	OUT2	47
	OUT1	48
	ARD11 (SERVO SEÑAL)	49
	SERVO VCC	50
	SERVO GND	51
	LUZ IZQUIERDA GND	52
	LUZ IZQUIERDA VCC	53
	ARD2	54
	ARD1	55
	ARD0	56
	Bluetooth GND	57
	Bluetooth VCC	58
	OUT3	59
	OUT4	60
Arduino UNO	ARD0 (RX)	0
	ARD1 (TX)	1
	ARD2 (LUCES)	2
	ARD3 (IN1)	3
	ARD4 (IN2)	4
	ARD5 (IN3)	5
	ARD6 (ENABLE)	6
	ARD7 (IN4)	7
	ARD8 (DHT11 DATO)	8
	ARD11 (SERVO SEÑAL)	9
	ARD14 (GND)	10

NOTA: Detalle de conexiones al interior del robot para guía. Los Autores

CAPÍTULO V

DISEÑO E IMPLEMENTACIÓN DE PRÁCTICAS

5.1 Práctica 1: De compilación aplicaciones pre-existentes

UNIVERSIDAD POLITÉCNICA SALESIANA

ESCUELA DE INGENIERÍA ELECTRÓNICA

PRÁCTICA: Nº1

TÍTULO: De compilación de aplicaciones Android pre-existentes

OBJETIVOS:

- Utilizar las aplicaciones pre-existentes, su código y estructura mediante la Decompilación de programas Android
- Identificar las partes que componen el código Android desarrollado sobre java y la nomenclatura que se utiliza para cada uno de sus componentes.

PROCEDIMIENTO:

Para el desarrollo de esta práctica se utiliza herramientas disponibles en la web para:

1. Descargar las aplicaciones de interés.
2. Extraer el código Android y las clases de Java que componen la aplicación.

DESARROLLO:

1. Ingrese en la página: http://apps.evozi.com/apk-downloader/

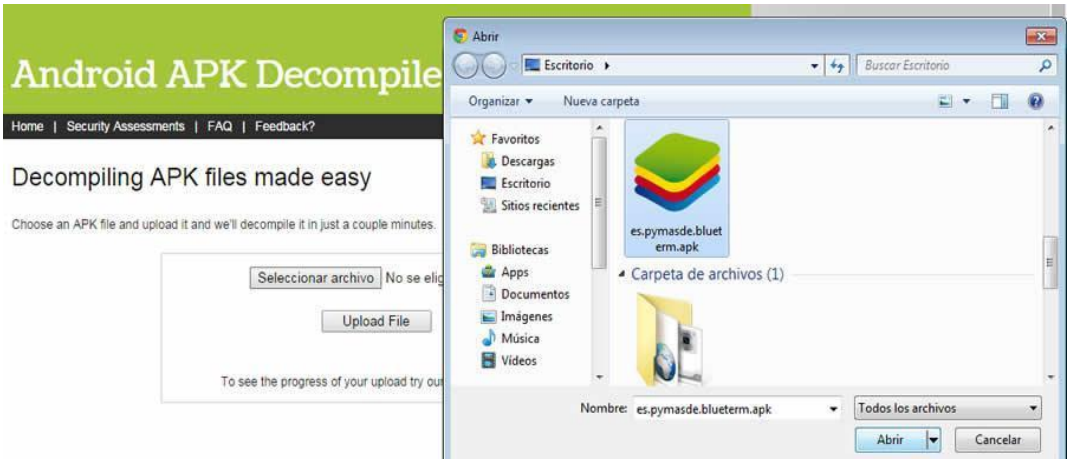
2. En el recuadro copiar la dirección electrónica (url) de la aplicación a descargar. Y de clic en Generar Link de Descarga(Generate download link)

Figura 5.1. Descarga de Apk

Nota: Vista página de descarga de apk. Los autores

3. Generado el link de enlace damos clic en "Click here to download" para este ejemplo usamos la aplicación de Google Play llamada Blueterm. El archivo descargado para este caso se llama **es.pymasde.blueterm.apk**
4. Para la Decompilación del apk Android (Extracción de archivos de la aplicación) utilizamos http://www.decompileandroid.com/ y seleccionamos *static versión* donde seleccionamos subimos el apk descargado mediante *upload file*.

Figura 5.2. Decompilación de Apk



Nota: Proceso de Decompilación de un apk. Los autores

5. Al decompilar se obtiene parte del código Android para revisar su estructura, archivos. En general el fichero RES y el AndroidManifest.xml

Figura 5.3. Decompilación de Apk

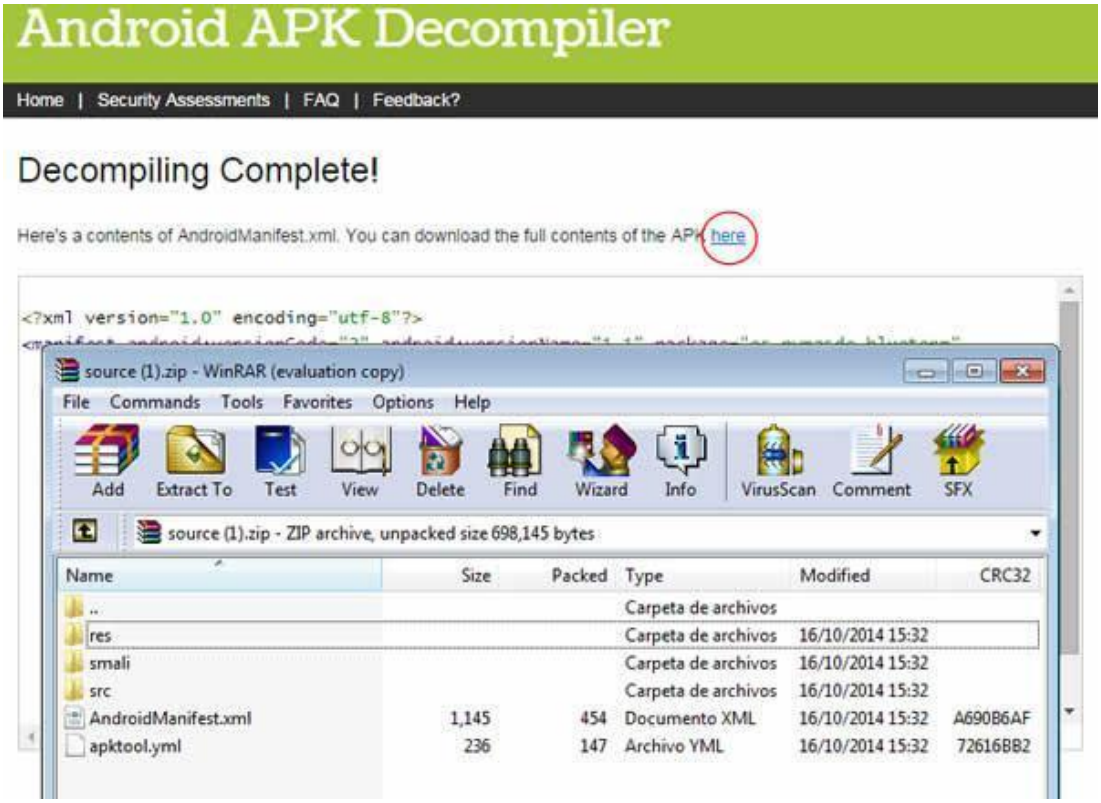


Figura: Proceso de Decompilación de un apk equipo. Los autores

6. Para obtener el código Java descargamos las aplicaciones:

- **Dex2jar.** Genera un .jar a partir de un .apk https://code.google.com/p/dex2jar/downloads/detail?name=dex2jar-0.0.9.15.zip&can=2&q=

Figura 5.4. Descarga dex2jar

Figura: Ubicación de descarga dex2jar. Los autores

- **jd-gui.** Visualiza el código fuente que se encuentra el .jar http://jd.benow.ca/

Figura 5.5. Descarga jd-gui

Figura: Ubicación de descarga jd-gui. Los autores

7. Convertimos el apk en un .jar con dex2jar siguiendo los siguientes pasos:

7.1 Descomprimimos el archivo dex2jar

7.2 Copiamos la apk (es.pymasde.blueterm.apk) en la carpeta dex2jar

7.3 Arrastramos la apk al archivo **dex2jar.bat**

Figura 5.6. Convertir APK to JAR

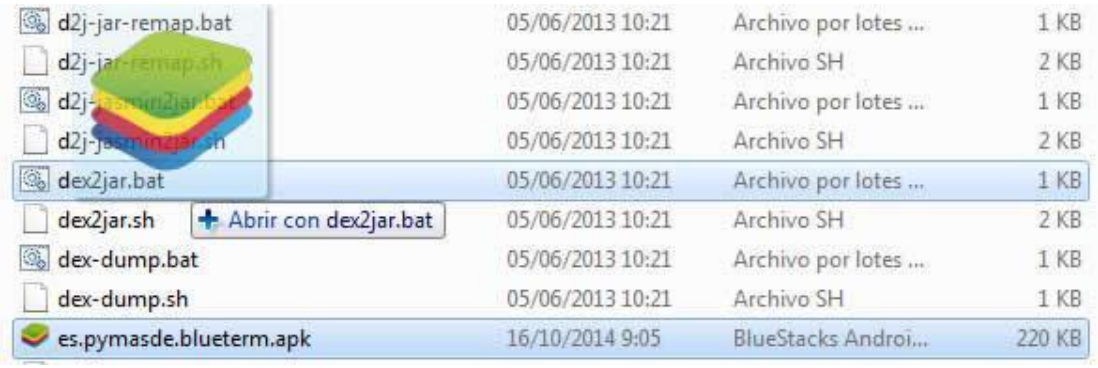


Figura: Conversión apk vista gráfica. Los autores

7.4 Se tiene como resultado un archivo **es.pymasde.blueterm_dex2jar.jar**

8. Descomprimimos el jd-gui descargado y abrimos el **jd-gui.exe** con este programa abrimos **es.pymasde.blueterm_dex2jar.jar** y obtenemos las clases utilizadas en la aplicación **blueterm** de Android

Figura 5.7. Decompilación JAR con JD-GUI

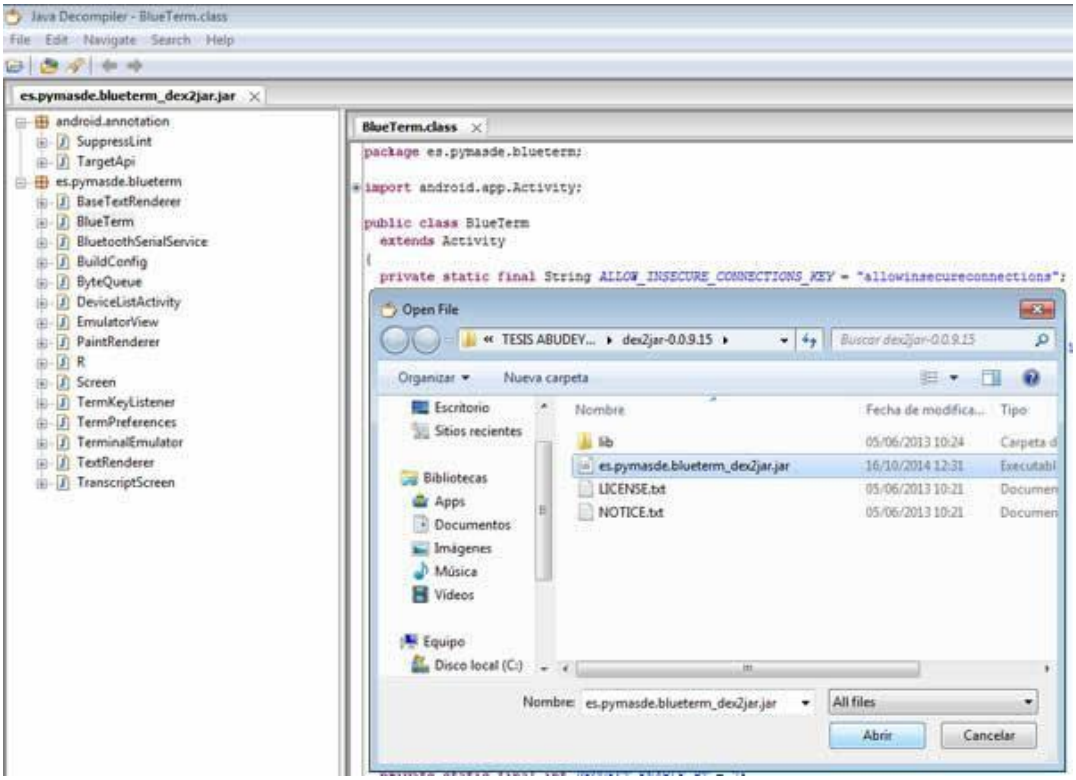


Figura: Proceso de Decompilación JAR. Los autores

CONCLUSIONES:

- Para la Decompilación de una aplicación Android debemos en primer lugar descargar la aplicación con extensión .apk
- La Decompilación se realiza en dos partes:
  1. Extraer el androidManifest.xml y el fichero RES que podemos visualizar en eclipse.
  2. Extraer el código JAVA convirtiendo el APK en extensión .jar para luego decompilar dicha extensión para acceder a las clases usadas en la aplicación Android.

5.2 Práctica 2: Comunicación con interfaz arduino- Ethernet

UNIVERSIDAD POLITÉCNICA SALESIANA

ESCUELA DE INGENIERÍA ELECTRÓNICA

PRÁCTICA N° 2

TÍTULO: Comunicación con Interfaz Arduino - Ethernet

OBJETIVOS:

- Entender los diferentes protocolos e interfaces con las que Arduino es capaz de comunicarse hacia el exterior.
- Desarrollar un programa en código Arduino para realizar la comunicación con dispositivos de tipo Ethernet.
- Desarrollar e implementar una aplicación java sobre eclipse que capture los datos transmitidos por Arduino a través de la red Ethernet.

PROCEDIMIENTO:

Arduino ha sido desarrollado con la capacidad de comunicarse con diferentes dispositivos y a nivel de diferentes protocolos, en esta práctica haremos énfasis en la comunicación con dispositivos Ethernet, para lo cual utilizaremos los siguientes elementos:

Materiales Utilizados

- Arduino Ethernet Shield
- Arduino UNO
- Router inalámbrico con tecnología 802.1 b/g/n
- Dispositivo móvil con OS Android
- Cable Serial USB
- Cable de Red UTP de conexión directa

DESARROLLO:

Vamos a crear con el Ethernet Shield un servidor Web, el cual nos proporcionará el código HTML para poder verlo en nuestro navegador y poder, así, interactuar con él. Para acceder a la página Web que vamos a crear, tendremos que acceder con una dirección IP perteneciente a nuestra red, que será la que le hayamos asignado al Ethernet Shield. Para nuestra práctica hemos asignado al Ethernet Shield la dirección IP 192.168.0.110/24.

De este modo el dispositivo ANDROID se comunicara a través de la red LAN hacia nuestro web server creado en el Ethernet Shield de ARDUINO. La configuración lógica del proceso de conexión y comunicación se vería del siguiente modo:

a. Conexiones

Figura 5.8. Diagrama de conexión lógica y física Ethernet Shield-ANDROID

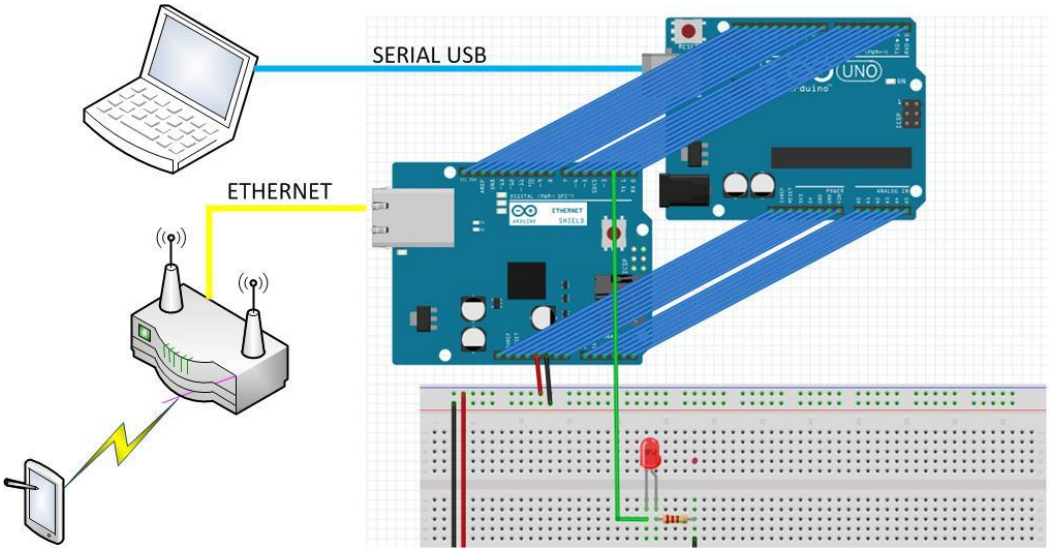


Figura: Vista lógica y física conexión Ethernet. Los autores

b. Aplicación Arduino

Creamos una aplicación en ARDUINO que sea capaz de captar la lectura de los datos recibidos vía HTTP:

```
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; //Dirección MAC física
byte ip[] = { 192, 168, 0, 110 }; // direccion IP que asignaremos a Arduino en la red LAN
byte gateway[] = { 192, 168, 0, 1 }; // Direccion Gateway de mi red LAN
byte subnet[] = { 255, 255, 255, 0 }; //Mascara de red

EthernetServer server(80); //Puerto a utilizar
int ledPin = 2; // LED pin
String readString = String(30); //Cadena de String para almacenar datos desde la dirección IP
boolean LEDON = false; //Flag de estado del LED


void setup(){
//Iniciamos captura ethernet
Ethernet.begin(mac, ip, gateway, subnet);
//Establecemos el LEDpin en estatus output
pinMode(ledPin, OUTPUT);
Serial.begin(9600);
}

void loop(){
// Creamos una conexión con el cliente ethernet
EthernetClient client = server.available();
if (client) {
while (client.connected()) {
if (client.available()) {
char c = client.read();
//Lectura del carácter char desde la captura de HTTP
if (readString.length() < 100)
{
//Almacenamos el caracter como string
readString += c;
}
//Mostramos el caracter en puerto serial
Serial.print(c);
//si la trama HTTP ha finalizado
if (c == '\n') {
if (readString.indexOf("?") <0)
{
//descartar todo
}
```

```
else
if(readString.indexOf("L=1") >0)
{
digitalWrite(ledPin, HIGH); // Establecemos LED en ON
LEDON = true;
}else{
digitalWrite(ledPin, LOW); // Establecemos LED en OFF
LEDON = false;
}

// A continuación se muestra el HTML resultante en nuestro Arduino con un header standard
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println();
client.print("<body>");
client.println("<h1>LED control</h1>");
if (LEDON)
client.println("<form method=get name=LED><input type=checkbox name=L value=1 CHECKED>LED<br><input type=submit value=submit></form>");
else
client.println("<form method=get name=LED><input type=checkbox name=L value=1>LED<br><input type=submit value=submit></form>");
client.println("<br />");
client.println("</body></html>");
readString="";
client.stop();
}
}
}
}
}
```

c. Aplicación Android

Nuestra aplicación en ANDROID, desarrollada en eclipse consta de dos botones en su Main Activity, los cuales cumplen la función de encender y apagar el LED enviando distintas tramas de HTTP a nuestro web server creado en ARDUINO:

AndroidManifest.xml

Establecemos las versiones compatibles y los permisos necesarios para la aplicación:

```
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="8" />
    <uses-permission android:name="android.permission.INTERNET" />
```

Diseño del Layout (Interfaz)

Definimos los elementos que forman parte de nuestra interfaz gráfica con sus respectivos parámetros: Layout, TextView y Buttons.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />
    <Button
        android:id="@+id/sendButton"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:onClick="send"
        android:text="Encender LED" />
    <Button
        android:id="@+id/sendButtonOff"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:onClick="sendOff"
        android:text="Apagar LED" />
</LinearLayout>
```

Main Activity

Declaramos las acciones que se ejecutaran al presionar cada botón de encendido o apagado, y el comportamiento del envío de esta información vía http al arduino y su Ethernet shield.

```
    sendButton = (Button) findViewById(R.id.sendButton);
    sendButtonO = (Button) findViewById(R.id.sendButtonOff);
}

public void send(View v)
{
    HttpClient httpclient = new DefaultHttpClient();
    HttpPost httppost = new HttpPost("http://192.168.0.110/?L=1");

public void sendOff(View v)
     {
     HttpClient httpclient = new DefaultHttpClient();
     HttpPost httppost = new HttpPost("http://192.168.0.110/?");
```

d. Resultados

Figura 5.9 Aplicación Android Practica 2

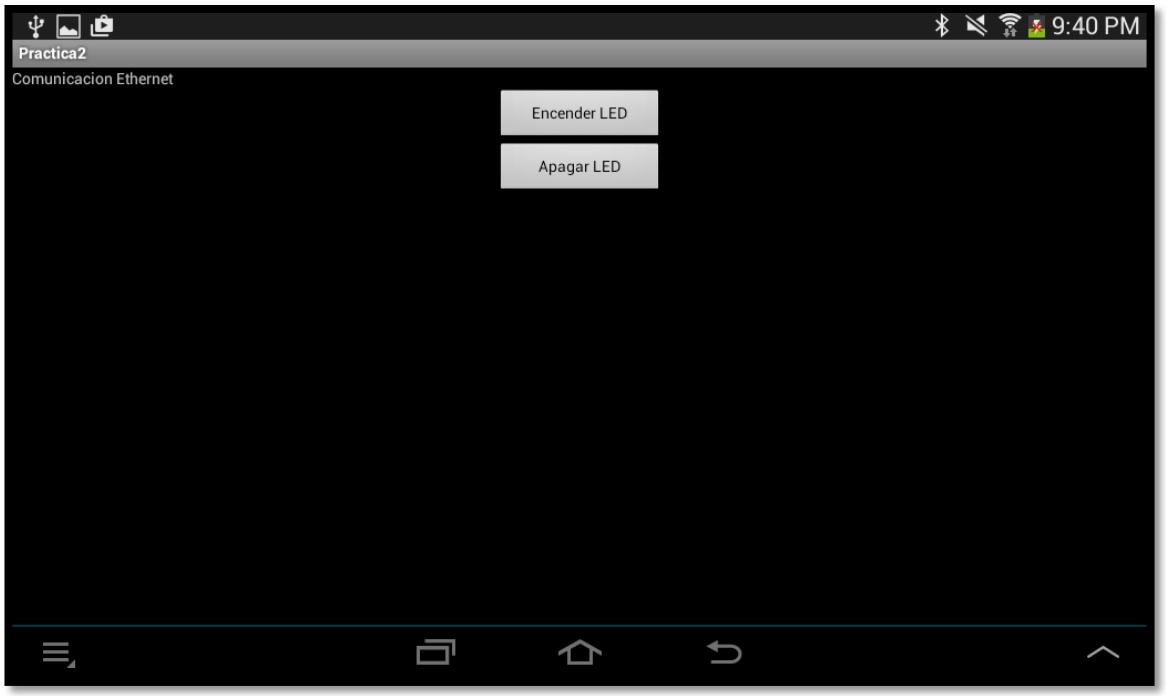


Figura: Vista interfaz práctica 2. Los autores

Se obtuvo como resultado la comunicación entre nuestro dispositivo Android y el Ethernet Shield de Arduino, enviando datos satisfactoriamente y recibiendo como respuesta la ejecución de la actividad solicitada, encender el LED con comandos http.

CONCLUSIONES:

- Entendimos los diferentes protocolos e interfaces con las que Arduino es capaz de comunicarse con el exterior.
- Hemos desarrollado un programa en código Arduino que estableció la comunicación con dispositivos Ethernet.
- Desarrollamos e implementamos una aplicación java sobre eclipse que capturó los datos transmitidos por Arduino a través de la red Ethernet.

5.3 Práctica 3: Comunicación con interfaz arduino- USB

UNIVERSIDAD POLITÉCNICA SALESIANA

ESCUELA DE INGENIERÍA ELECTRÓNICA

PRÁCTICA: Nº3

TÍTULO: Comunicación con interfaz arduino- USB

OBJETIVOS:

- Establecer los principios de comunicación serial a través del puerto serie.
- Interpretar la información obtenida en lenguaje comprendido por el usuario.

PROCEDIMIENTO:

La comunicación arduino USB nos permite ver la información obtenida por la tarjeta controladora a través de comandos interpretados por el usuario. Se obtiene del monitor serial que incorpora el programa Arduino o de medios físicos como displays o sensores.

ELEMENTOS UTILIZADOS

- Tarjeta Arduino UNO
- Cable USB
- Potenciómetro 1kΩ
- Cables de Conexión protoboard

DESARROLLO:

a) Código Arduino

```
//Libreria utilizada
#include <SoftwareSerial.h>

void setup()
{
  //Establecemos la velocidad de comunicacion serial
  Serial.begin(9600);
  //Imprimimos nuestro "Comunicación Serial" en el monitor serial
  Serial.println("Comunicación Serial");
}

void loop()
{
   int valorPotenciometro = analogRead(A0); //Se establece el pin del potenciometro
   Serial.println("El valor del potenciometro es:");
   Serial.println(valorPotenciometro);// Según la posición del potenciometro se
                                      //imprimee el valor
   delay(1000);
}
```

b) Conexiones

Figura 5.10 Conexiones Practica 3

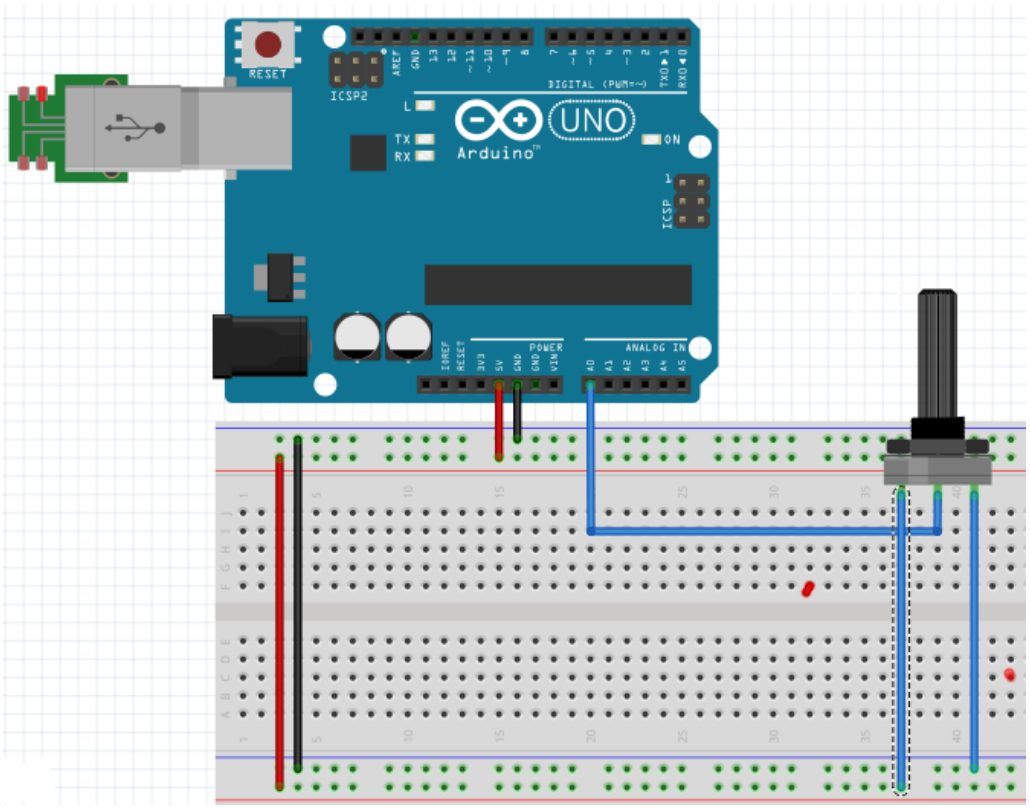


Figura: Vista física conexiones práctica 3. Los Autores

c) Resultado

Figura 5.11 Pruebas serial

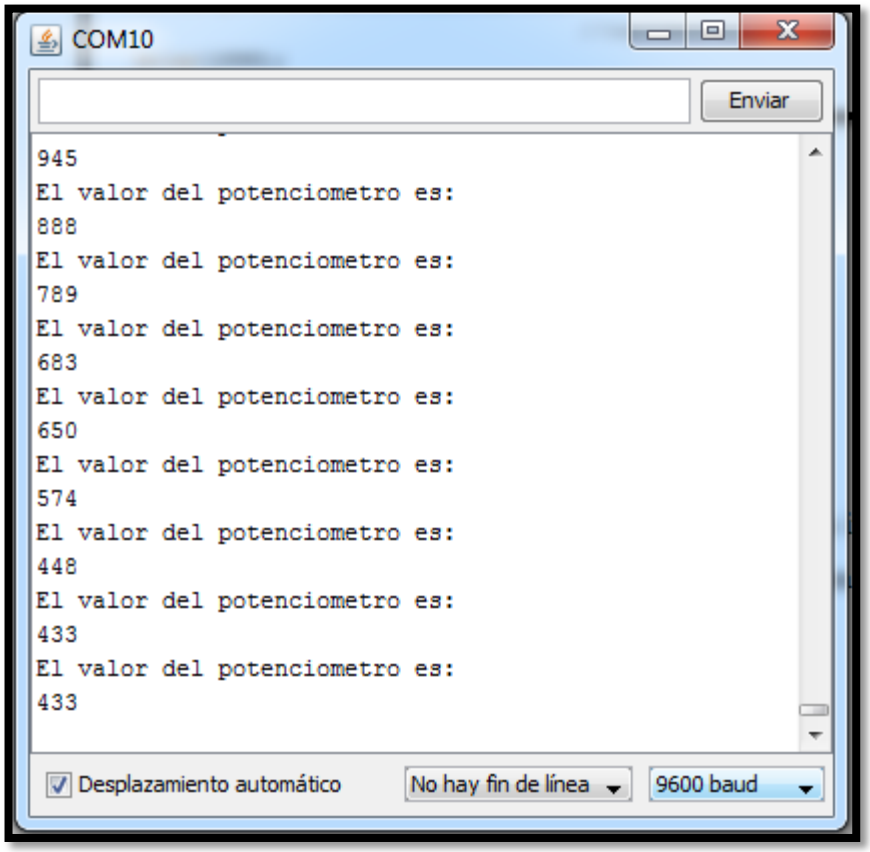


Figura: Monitor serial y pruebas. Los Autores

CONCLUSIONES:

La comunicación serial es la forma de comunicación más importante en Arduino porque nos permite visualizar, interpretar, controlar y obtener los datos necesarios para la resolución de una aplicación electrónica.

5.4 Práctica 4: Comunicación con interfaz arduino- Bluetooth

UNIVERSIDAD POLITÉCNICA SALESIANA

ESCUELA DE INGENIERÍA ELECTRÓNICA

PRÁCTICA: Nº4

5.4.1 TÍTULO: Comunicación con interfaz arduino- Bluetooth

OBJETIVOS:

- Implementar una aplicación Android que permita establecer comunicación entre los dispositivos bluetooth de Arduino y Android.

PROCEDIMIENTO:

1. Realizamos la conexión del dispositivo bluetooth en los pines Rx(0) y Tx(1), sin olvidar proveer el voltaje de alimentación (VCC y Gnd) desde los pines correspondientes en la placa arduino hacia el dispositivo bluetooth el mismo que permitirá establecer la comunicación entre el dispositivo Android y nuestra tarjeta Arduino.
2. Generamos el código de programación en Arduino, que capture datos recibidos a través del dispositivo bluetooth.
3. Realizamos la aplicación para Android, sobre eclipse, la cual permita verificar el estado de conexión del dispositivo bluetooth del dispositivo móvil.
4. Establecemos la conexión bluetooth entre arduino y el dispositivo en Android.

Módulo Bluetooth HC-06 Serial Rs232.

Utiliza el protocolo UART RS 232 serial. Se utiliza para aplicaciones inalámbricas, con PC, Microcontroladores o módulos Arduino.

Características del HC-06

- Compatible con el protocolo Bluetooth V2.0.
- Voltaje de alimentación: 3.3VDC – 6VDC.
- Voltaje de operación: 3.3VDC.
- Baud rate ajustable: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.
- Tamaño: 1.73 in x 0.63 in x 0.28 in (4.4 cm x 1.6 cm x 0.7 cm)
- Corriente de operación: < 40 mA
- Corriente modo sleep: < 1mA

Materiales Utilizados

- Tarjeta Arduino UNO
- Cable USB
- Diodo Led
- Cables de Conexión protoboard
- Modulo Bluetooth HC-06
- Tablet Android 7”

DESARROLLO:

d) Conexiones

Figura 5.12 Conexión Práctica 4

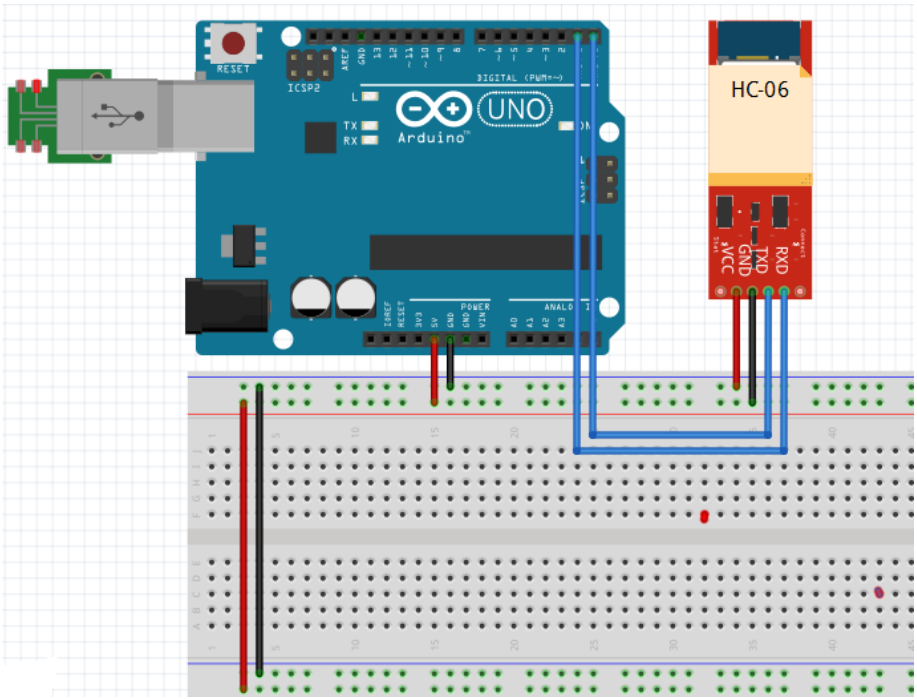


Figura: Vista física conexión practica 4. Los Autores

e) Aplicación Android

AndroidManifest.xml

Establecemos las versiones de Android compatibles así como los permisos necesarios para la aplicación:

```
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="21" />

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

Layout

activitymainpractica4.xml

Definimos los elementos que forman parte de la interfaz gráfica de nuestra aplicación en Android:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.practica4.Mainpractica4" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="234dp"
        android:text="Conexión Bluetooth" />

</RelativeLayout>
```

MainActivity.java

Declaramos los parámetros de conexión al enlazarse el dispositivo Android con el HC-06:

```
public  void sendMessage(String message) {
            if (Servicio_BT.getState() == ConexionBT.STATE_CONNECTED)
{//checa si estamos conectados a BT
            if (message.length() > 0) {   // checa si hay algo que enviar
            byte[] send = message.getBytes();//Obtenemos bytes del
mensaje
            if(D) Log.e(TAG, "Mensaje enviado:"+ message);
            Servicio_BT.write(send);     //Mandamos a escribir el mensaje
            }
            } else Toast.makeText(this, "No conectado",
Toast.LENGTH_SHORT).show();
            }//fin de sendMessage
            case Mensaje_Nombre_Dispositivo:
            mConnectedDeviceName = msg.getData().getString(DEVICE_NAME);
//Guardamos nombre del dispositivo
            Toast.makeText(getApplicationContext(), "Conectado con "+
mConnectedDeviceName, Toast.LENGTH_SHORT).show();
            seleccionador=true;
            break;
```

f) Resultado

Figura 5.13 Aplicación Android Practica 4

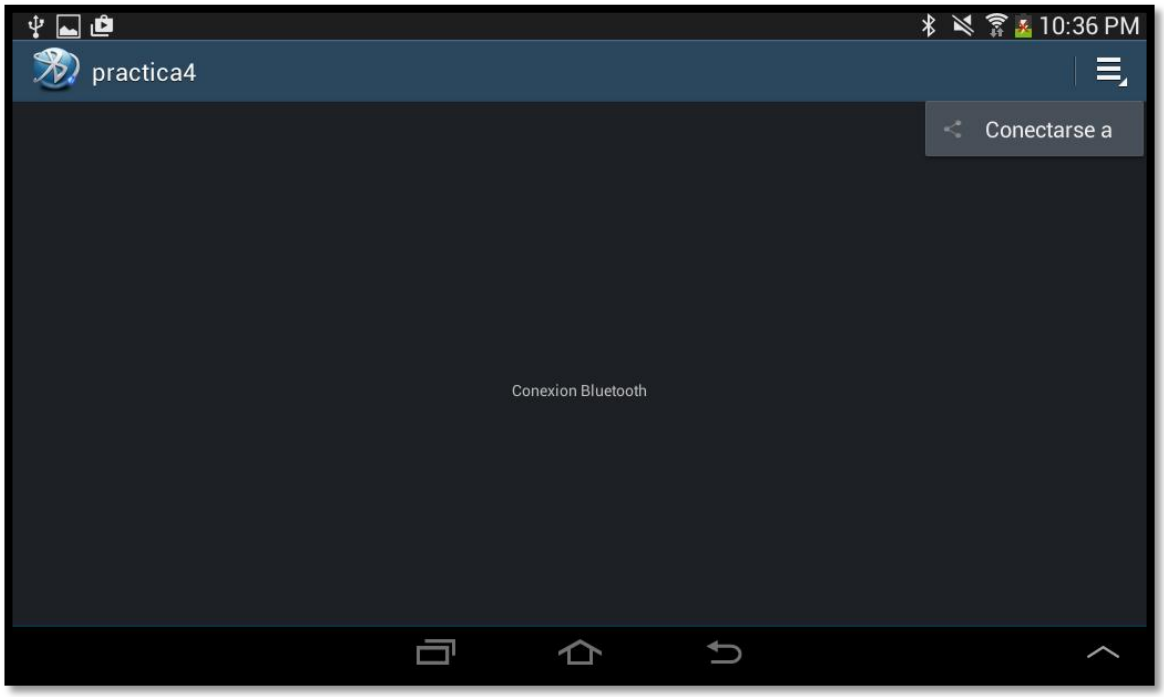


Figura: Vista interfaz práctica 4. Los autores

CONCLUSIONES:

- Hemos implementado una aplicación Android que permite establecer comunicación entre los dispositivos bluetooth de Arduino y Android.

5.5 Práctica 5: Leds

UNIVERSIDAD POLITÉCNICA SALESIANA

ESCUELA DE INGENIERÍA ELECTRÓNICA

PRÁCTICA: Nº5

TÍTULO: Leds

OBJETIVOS:

- Implementar una aplicación Android para el encendido de un led a través de Arduino.

PROCEDIMIENTO:

1. Realizamos las conexiones físicas entre el diodo led y la placa de entrenamiento Arduino UNO, para el cual hemos definido como salida el PIN #2. Del mismo modo realizamos la conexión del dispositivo bluetooth en los pines Rx(0) y Tx(1), sin olvidar proveer el voltaje de alimentación (VCC y Gnd) desde los pines correspondientes en la placa arduino hacia el dispositivo bluetooth el mismo que permitirá establecer la comunicación entre el dispositivo Android y nuestra tarjeta Arduino.
2. Generamos el código de programación en Arduino, que capture datos recibidos a través del dispositivo bluetooth y dependiendo de las condiciones establecidas generar un pulso de Encendido o Apagado en la salida (Pin #2).
3. Realizamos la aplicación para Android, sobre eclipse, la cual a través de su interfaz simule el encendido o apagado de un diodo led, enviando a través de la conexión bluetooth una señal para cada estado.
4. Establecemos la conexión bluetooth entre la placa arduino y el dispositivo en Android.

Materiales Utilizados

- Tarjeta Arduino UNO
- Cable USB
- Diodo Led
- Resistencia 330 ohm
- Cables de Conexión protoboard
- Modulo Bluetooth HC-06
- Tablet Android 7”

DESARROLLO:

a) Código Arduino

```
/* Práctica 5:Presiona la imagen en el dispositivo Android para encender el led
                y vuelva a presionar para apagarlo
*/
#include <SoftwareSerial.h> //Librería que permite establecer comunicación serie en otros pines
SoftwareSerial BT(0,1); // Conecta el módulo bluetooth en los pines 10 RX, 11 TX.

int green=2; // Establece el nombre y número de PIN de la salida al diodo LED
char cadena[255]; //Crea un arreglo de 255 caracteres
int i=0; //Tamaño actual del arreglo

void setup()
{
  /*Inicializa la comunicación PC y bluetooth*/
  BT.begin(9600);
  Serial.begin(9600);
  pinMode(green,OUTPUT); //Defina el PIN GREEN como salida
}

void loop()
{

*Pregunta si existen datos en la comunicación bluetooth*/
  if(BT.available())
  {
    char dato=BT.read(); //Guarda los datos carácter a carácter en la variable "dato"
    cadena[i++]=dato; //Coloca cada carácter recibido en el arreglo "cadena"

    /*Cuando reciba una nueva línea al pulsar la imagen de la aplicación entra en la función*/
```

```
    if(dato=='\r')
    {
      Serial.print(cadena); //Visualiza el comando recibido en el Monitor Serial
      /*Verifica la cadena. 1 para encender 2 para apagar*/
      if(strstr(cadena,"1")!=0)
      {
        digitalWrite(green,HIGH);
      }
      if(strstr(cadena,"2")!=0)
      {
        digitalWrite(green,LOW);
      }
      BT.write("\r"); //Envía un retorno de carro de la aplicación.
      clean(); //Ejecuta la función clean() para limpiar el arreglo
    }
  }
}

/*Limpia el arreglo*/
void clean()
{
  for (int cl=0; cl<=i; cl++)
  {    cadena[cl]=0;
  }
  i=0;
}
```

b) Conexiones

Figura 5.14 Conexión práctica 5

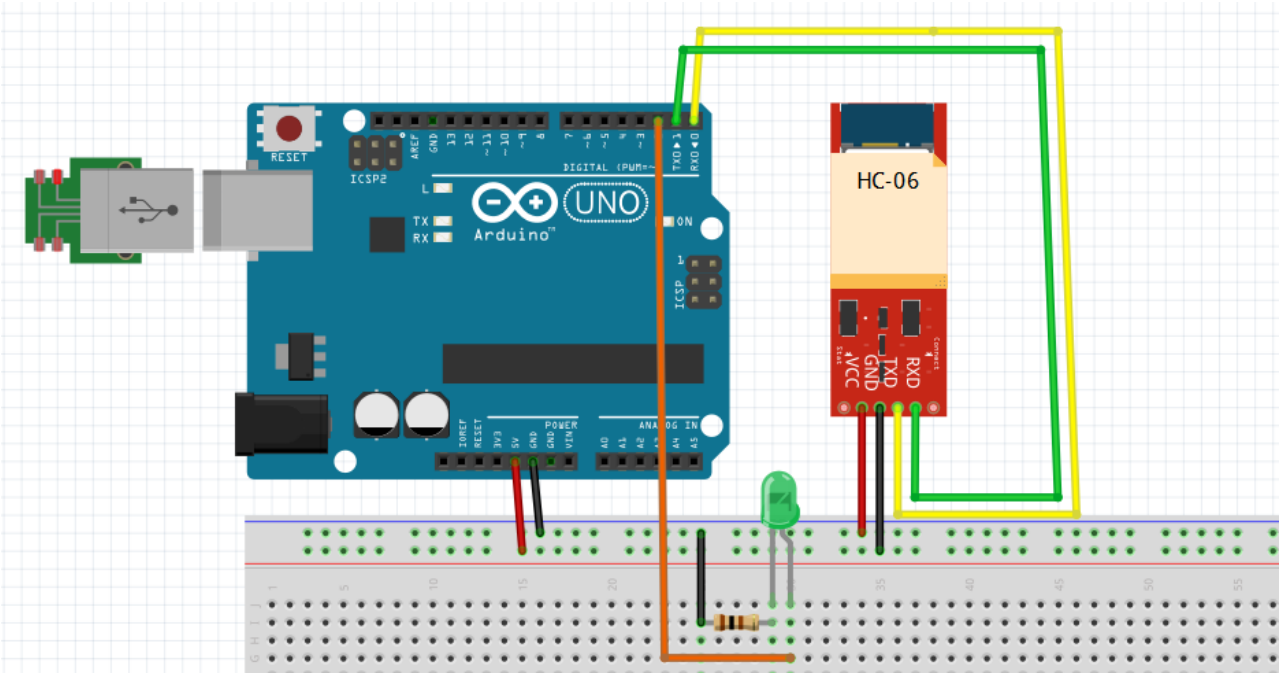


Figura: Conexión física practica 5. Los Autores

c) Aplicación Android

Para establecer la conexión desde el modulo bluetooth hacia el dispositivo Android utilizaremos la clase java *ConexionBT.java*, la misma que fue ya previamente desarrollada e implementada.

AndroidManifest.xml

Establecemos los permisos de la aplicación y las versiones compatibles:

```
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="21"
    <uses-permission android:name="android.permission.BLUETOOTH"/>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
    <uses-permission android:name="android.permission.VIBRATE"/>
    </uses-sdk>
```

activity_MainPractica5.xml

Definimos los elementos de nuestra interfaz gráfica en la aplicación:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal"
    tools:context="com.example.practica_5.Mainpractica5"
    <ToggleButton
        android:id="@+id/Led1"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:background="@drawable/ledoff"
        android:layout_gravity="center"
        android:layout_marginLeft="400dp" />
</LinearLayout>
```

MainActivity.java

Declaramos la acción que ejecutará la aplicación al presionar el botón “ON/OFF” dentro de la aplicación:

```
        public void onClick(View vv)
        {
              if(BotonLed.isChecked())
                {
                     if(D) Log.e("BotonLed", "Encendiendo..");
                           sendMessage("1\r");
        findViewById(R.id.Led1).setBackgroundResource(R.drawable.ledon); }
                     else {
                     if(D) Log.e("BotonLed", "Apagando..");
                           sendMessage("2\r");
        findViewById(R.id.Led1).setBackgroundResource(R.drawable.ledoff);}
```

d) Resultado

Figura 5.15 Aplicación Android Práctica 5

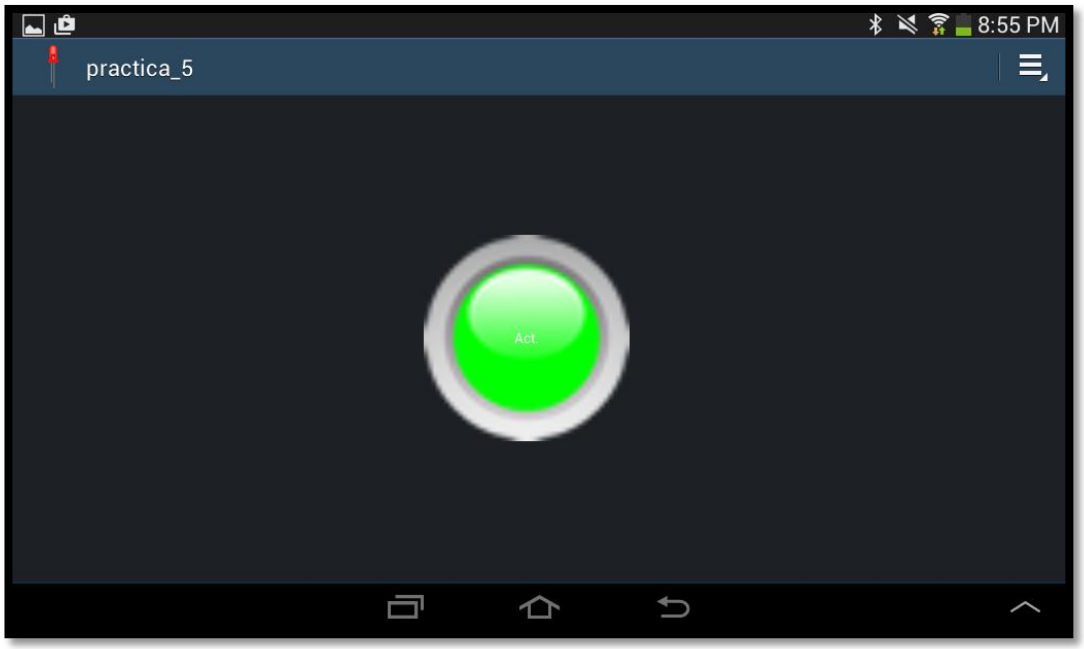


Figura: Vista interfaz gráfica practica 5. Los autores

Se logra de manera exitosa la conexión del módulo bluetooth con el dispositivo Android, permitiendo así que la interacción desde la aplicación, activando y desactivando el ToggleButton utilizado, encienda y apague el diodo led en el Protoboard.

El código arduino interpreta a través de los datos recibidos por el modulo bluetooth desde el dispositivo Android un *"1"* para estado ENCENDIDO, y un *"2"* para estado APAGADO, con estas condiciones establece el estado HIGH o LOW en la salida del pin# 2.

CONCLUSIONES:

Hemos implementado una aplicación en Android que nos ha permitido el encendido de un LED a través de arduino.

5.6 Práctica 6: Semáforo

UNIVERSIDAD POLITÉCNICA SALESIANA

ESCUELA DE INGENIERÍA ELECTRÓNICA

PRÁCTICA: Nº6

TÍTULO: Semáforo

OBJETIVOS:

- Implementar una aplicación Android para el encendido led en configuración semáforo a través de arduino.

PROCEDIMIENTO:

1. Realizamos las conexiones físicas entre los diodos led (rojo, amarillo, verde) y la placa de entrenamiento Arduino UNO, para el cual hemos definido como salida los Pines #2,3 y 4. Del mismo modo realizamos la conexión del dispositivo bluetooth en los pines Rx(0) y Tx(1), sin olvidar proveer el voltaje de alimentación (VCC y Gnd) desde los pines correspondientes en la placa arduino hacia el dispositivo bluetooth el mismo que permitirá establecer la comunicación entre el dispositivo Android y nuestra tarjeta Arduino.
2. Generamos el código de programación en Arduino, que capture datos recibidos a través del dispositivo bluetooth y dependiendo de las condiciones establecidas para la secuencia del semáforo, generar un pulso de Encendido o Apagado en las salida (Pines #2, 3 y 4).
3. Realizamos la aplicación para Android, sobre eclipse, la cual a través de su interfaz simule la secuencia de encendido de las luces de un semáforo, enviando a través de la conexión bluetooth una señal para cada estado.

4. Establecemos la conexión bluetooth entre la placa arduino y el dispositivo en Android.

Materiales Utilizados

- Tarjeta Arduino UNO
- Cable USB
- Diodos Led (x 3)
- Resistencia 330 ohm (x 3)
- Cables de Conexión protoboard
- Modulo Bluetooth HC-06
- Tablet Android 7”

DESARROLLO:

a) Código Arduino

```
#include <SoftwareSerial.h> //Librería que permite establecer comunicación serie en otros pins
//Aquí conectamos los pins RXD,TDX del módulo Bluetooth.
SoftwareSerial BT(0,1); //10 RX, 11 TX.

int green=4;
int yellow=3;
int red=2;
char cadena[255]; //Creamos un array de caracteres de 256 posiciones
int i=0; //Tamaño actual del array

void setup()
{
  BT.begin(9600);
  Serial.begin(9600);
  pinMode(green,OUTPUT);
  pinMode(yellow,OUTPUT);
```

```
  pinMode(red,OUTPUT);
}

void loop()
{
  //Cuando haya datos disponibles
  if(BT.available())
  {
    char dato=BT.read(); //Guarda los datos carácter a carácter en la variable "dato"

    cadena[i++]=dato; //Vamos colocando cada carácter recibido en el array "cadena"

    //Cuando reciba una nueva línea (al pulsar enter en la app) entra en la función
    if(dato=='\r')
    {
      Serial.print(cadena); //Visualizamos el comando recibido en el Monitor Serial

      if(strstr(cadena,"0")!=0)
      {
        digitalWrite(red,LOW);
        digitalWrite(yellow,LOW);
        digitalWrite(green,LOW);

      }
      if(strstr(cadena,"1")!=0)
      {
        delay(400);
        digitalWrite(green,HIGH);
        digitalWrite(yellow,LOW);
        digitalWrite(red,LOW);
      }
      if(strstr(cadena,"2")!=0)
      {
```

```
        delay(200);
        digitalWrite(red,LOW);
        digitalWrite(yellow,HIGH);
        digitalWrite(green,LOW);
      }
       if(strstr(cadena,"3")!=0)
      {
        delay(200);
        digitalWrite(green,LOW);
        digitalWrite(yellow,LOW);
        digitalWrite(red,HIGH);
      }

      BT.write("\r"); //Enviamos un retorno de carro de la app. La app ya crea una línea nueva
      clean(); //Ejecutamos la función clean() para limpiar el array
    }
  }
}

//Limpia el array
void clean()
{
  for (int cl=0; cl<=i; cl++)
  {
    cadena[cl]=0;
  }
  i=0;
}
```

b) Conexiones

Figura 5.16 Conexión Práctica 6

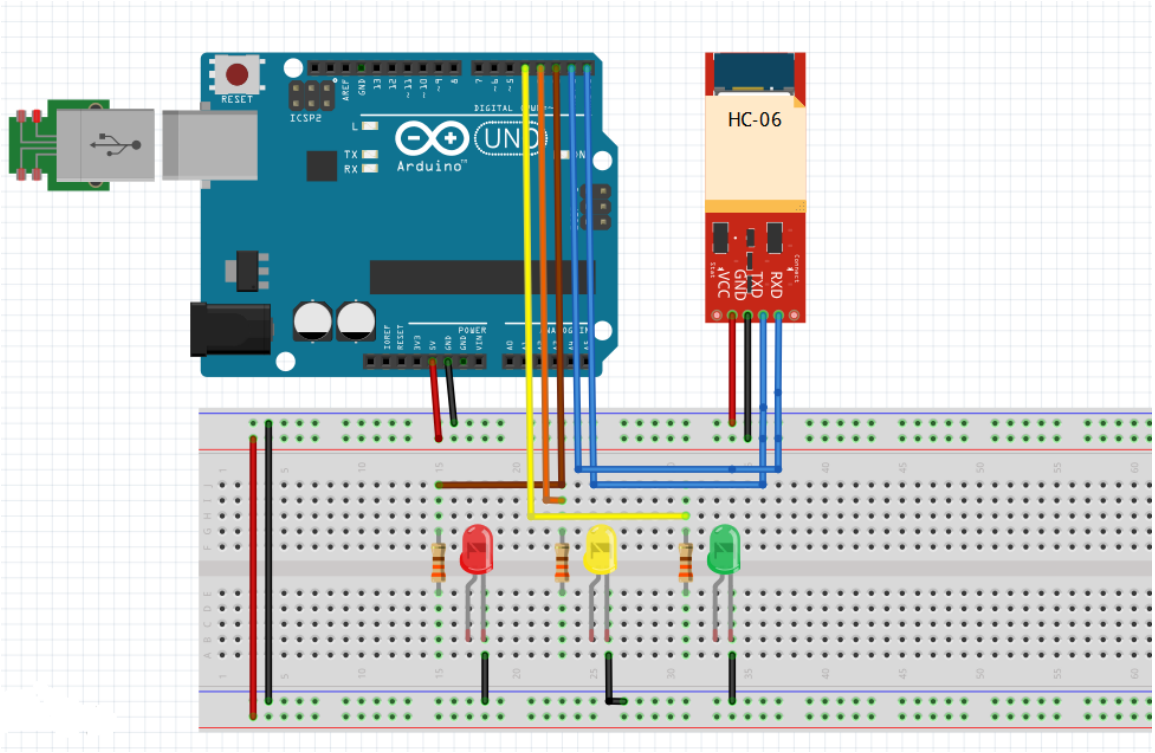


Figura: Vista conexión física practica 6. Los Autores

c) Aplicación Android

Para establecer la conexión desde el modulo bluetooth hacia el dispositivo Android utilizaremos la clase java *ConexionBT.java*, la misma que fue ya previamente desarrollada e implementada.

Manifest.xml

Establecemos la compatibilidad y permisos sobre nuestra aplicación:

```
<uses-sdk
      android:minSdkVersion="8"
      android:targetSdkVersion="21" />
  <uses-permission android:name="android.permission.BLUETOOTH"/>
  <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
  <uses-permission android:name="android.permission.VIBRATE"/>
```

Diseño del Layout: activity_practica6.xml

Definimos la interfaz, los elementos que la conforman y la forma en que estos se ajustan en la pantalla del dispositivo móvil:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@android:color/white"
    >
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_above="@+id/buttons"
        android:gravity="center" >
        <ImageSwitcher
            android:id="@+id/imageSwitcher"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" >
        </ImageSwitcher>
    </LinearLayout>

    <LinearLayout
        android:id="@+id/buttons"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true" >
        <Button
            android:id="@+id/buttonStart"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:onClick="start"
            android:text="@string/start"  />
        <Button
            android:id="@+id/buttonStop"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:onClick="stop"
            android:text="@string/stop" />
    </LinearLayout>

</RelativeLayout>
```

MainActivity.java

Aquí declaramos el comportamiento de la aplicación al presionar el botón START/STOP, el cual ejecutará la secuencia del semáforo en la interfaz gráfica y consecuentemente enviará estas instrucciones vía bluetooth a arduino:

```
public void start(View button)
    {
      timer = new Timer();
      timer.scheduleAtFixedRate(new TimerTask() {
      public void run() {
            runOnUiThread(new Runnable() {
            public void run() {
                  if(position == 0)
                     { //verde
                     imageSwitcher.setImageResource(gallery[position]);
                     sendMessage("1\r");
                           tiempo1++;
                           if(tiempo1==10)
                                 {position++;}
                     }

                     if(position == 1)
                     {//amarillo
                     imageSwitcher.setImageResource(gallery[position]);
                     sendMessage("2\r");
                           tiempo2++;
                           if(tiempo2==5)
                                 {position++;}
                     }
                     if(position == 2)
                     {//rojo
                     imageSwitcher.setImageResource(gallery[position]);
                     sendMessage("3\r");
                           tiempo3++;
                           if(tiempo3==20)
                                 {position++;}
                     }
                     if (position == 3)
                     {
                           position = 0;
                           tiempo1=0;
                           tiempo2=0;
                           tiempo3=0;
                     }
          }
       });
       }
       }, 0, 500);   }

public void stop(View button)
{
       sendMessage("0\r");
       timer.cancel();
       imageSwitcher.setImageResource(gallery[3]);}
```

d) Resultado

Figura 5.17 Aplicación Android Practica 6

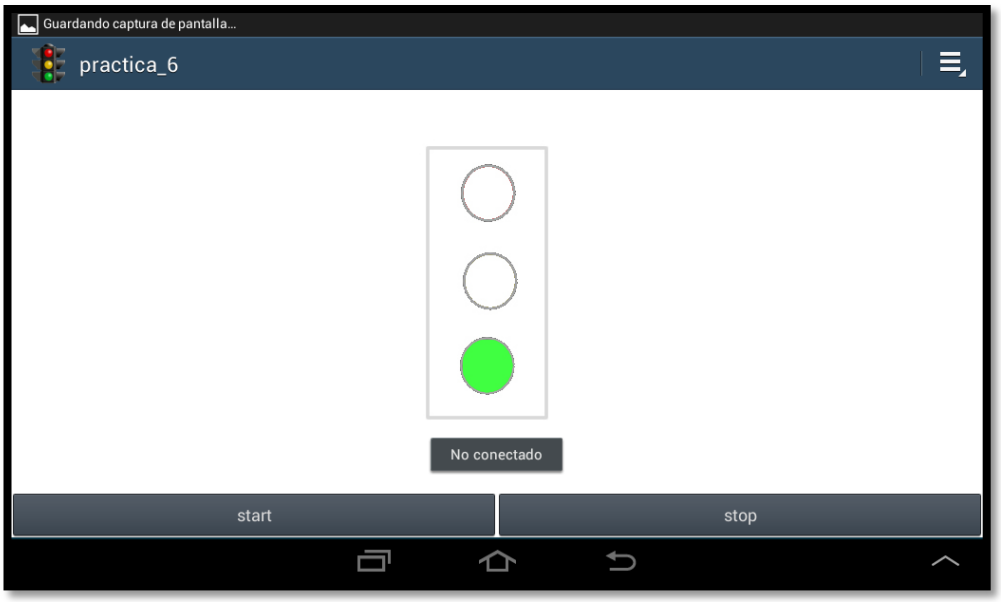


Figura: Vista interfaz práctica 6. Los autores

Se logra de manera exitosa la conexión del módulo bluetooth con el dispositivo Android, permitiendo así que la interacción desde la aplicación, el botón START inicia la secuencia de encendido de los leds que simulan el semáforo y en forma sincronizada este encendido secuencial se realiza en los diodos led.

El tiempo establecido, para cada estado es el siguiente:

Rojo: 20 seg

Verde: 10 seg.

Amarillo: 2 seg.

El código arduino interpreta a través de los datos recibidos por el modulo bluetooth desde el dispositivo Android un *"1"* para estado *VERDE*, un *"2"* para estado *YELLOW*, y un "*3"* para el estado *RED;* con estas condiciones establece el estado HIGH o LOW en la salida de los pines# 2,3 y 4, respectivamente.

CONCLUSIONES:

Hemos implementado una aplicación Android para el encendido de leds en configuración semáforo a través de arduino.

5.7 Práctica 7: Matriz

UNIVERSIDAD POLITÉCNICA SALESIANA

ESCUELA DE INGENIERÍA ELECTRÓNICA

PRÁCTICA: Nº7

TÍTULO: Matriz

OBJETIVOS:

- Implementar una aplicación Android para el control y visualización de caracteres en una matriz led a través de arduino.

PROCEDIMIENTO:

1. Realizamos la conexión del dispositivo bluetooth en los pines Rx(0) y Tx(1), sin olvidar proveer el voltaje de alimentación (VCC y Gnd) desde los pines correspondientes en la placa arduino hacia el dispositivo bluetooth el mismo que permitirá establecer la comunicación entre el dispositivo Android y nuestra tarjeta Arduino.
2. Procedemos a realizar la conexión de la matriz LED, obteniendo VCC y GND desde los pines que suministra el arduino. Para los pines DATA IN, CLK y LOAD utilizamos los pines 2, 3 y 4 del arduino, respectivamente.
3. Generamos el código de programación en Arduino, que capture datos recibidos a través del dispositivo bluetooth y envíe la señalización correspondiente a la matriz LCD que permita visualizar los caracteres enviados desde la aplicación en Android.
4. Realizamos la aplicación para Android, sobre eclipse, la cual permita la escritura de una frase o palabra, la cual sea enviada a través de la conexión bluetooth del dispositivo.
5. Establecemos la conexión bluetooth entre la placa arduino y el dispositivo en Android.

Materiales Utilizados

- Tarjeta Arduino UNO
- Cable USB
- Matriz LED
- Cables de Conexión protoboard
- Modulo Bluetooth HC-06
- Tablet Android 7”

DESARROLLO:

a) Código Arduino

```
#include <SoftwareSerial.h>

//Inclusión de la librería para el manejo de la matriz
#include "LedControlMS.h"

/*
Variable LedControl para que realice el control de la matriz
 pin 2 : DataIn
 pin 3 : CLK
 pin 4 : LOAD
*/
#define NBR_MTX 2
LedControl lc=LedControl(2,3,4, NBR_MTX);
String digits= "1234567890";
int digitCounter=0;

unsigned long delaytime=300;
char cadena;
```

```
void setup() {
  Serial.begin (9600);
  Serial.println("Ready");
  digitCounter=0;
  for (int i=0; i< NBR_MTX; i++){
    lc.shutdown(i,false);
    lc.setIntensity(i,8);
    lc.clearDisplay(i);
  }

void loop()
{
  delay(1000);
  Serial.print("\n Ingrese el texto: \n");
  while(Serial.available())
  {
  cadena = Serial.read();
  delay(500);
  Serial.print(cadena);
  scrollLeft(cadena);
  }

  lc.clearAll();
  delay(200);
}

//desplazar izquierda
void scrollLeft(char ch){
  int pos =lc.getCharArrayPosition(ch);
  for (int scroll =0; scroll<6; scroll++) {
     for (int i=scroll; i<6;i++) {
        lc.setRow(0,i-scroll, alphabetBitmap[pos][i]);
    }
```

```
    delay(300);
    lc.clearDisplay(0);
  }
}
//desplazar derecha
void scrollRight(char ch){
  int pos =lc.getCharArrayPosition(ch);
  for (int scroll =0; scroll<8; scroll++) {
     for (int i=0; i<6;i++) {
        if (scroll+i<8) lc.setRow(0, scroll+i, alphabetBitmap[pos][i]);
    }
    delay(300);
    lc.clearDisplay(0);
  }
}
```

b) Conexiones

Figura 5.18 Conexión practica 7

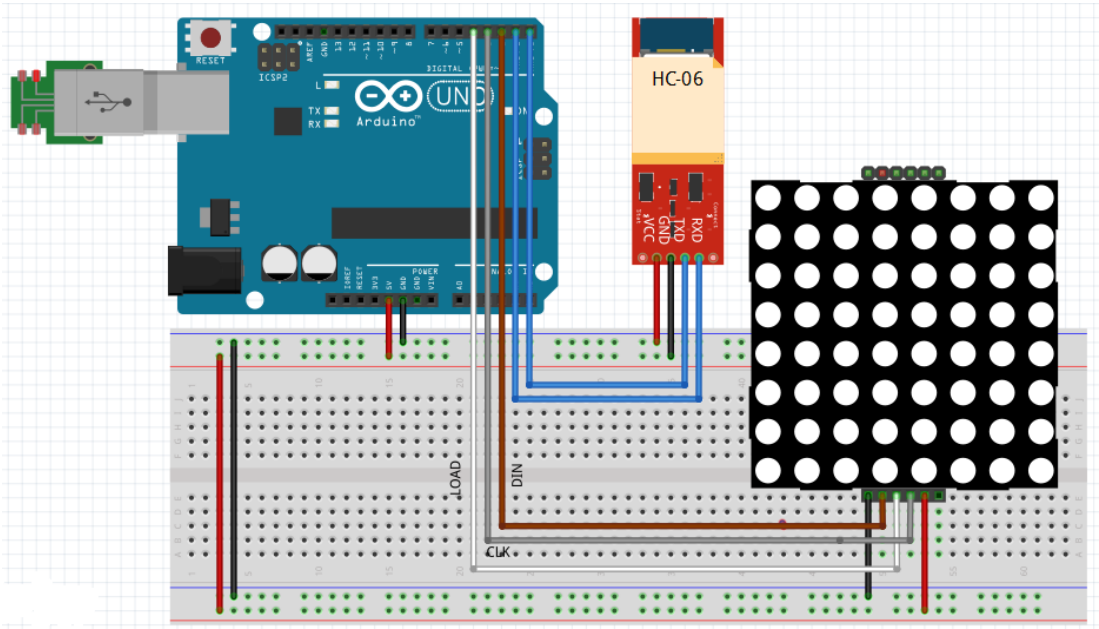


Figura: Vista conexión física practica 7. Los Autores

c) Aplicación Android

Para establecer la conexión desde el modulo bluetooth hacia el dispositivo Android utilizaremos la clase java *ConexionBT.java*, la misma que fue ya previamente desarrollada e implementada.

A continuación definimos el archivo Manifest y Layout para nuestra aplicación, los cuales delimitaran la interfaz y compatibilidad de la misma.

Manifest.xml

Establecemos la compatibilidad de la aplicación y los permisos de la misma sobre el dispositivo Android:

```
  <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="21" />
    <uses-permission android:name="android.permission.BLUETOOTH"/>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
      <uses-permission android:name="android.permission.VIBRATE"/>
```

Diseño del Layout: activity_practica7.xml

Definimos los elementos que conforman la interfaz gráfica de la aplicación:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.practica7.MatrixActivity" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="14dp"
        android:text="Ingrese el texto para la matrix"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="42dp"
        android:ems="10"
        android:inputType="textMultiLine" >

        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/btEnviar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editText1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="25dp"
        android:text="Enviar" />

    <TextView
        android:id="@+id/tvMensaje"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/btEnviar"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="57dp"
        android:text="Texto Enviado"
        android:textAppearance="?android:attr/textAppearanceLarge" />

</RelativeLayout>
```

Programación Botón "Enviar": MainActivity.xml

Declaramos la sentencia que se ejecutará en la aplicación al pulsar el botón "Enviar":

```
            switch(arg0.getId()){
            case R.id.btEnviar:
             String textoenviar=editText1.getText().toString();
             tvMensaje.setText("" + textoenviar);
             sendMessage(textoenviar);
            break;
}
```

d) Resultado

Figura 5.19 Aplicación Android Practica 7

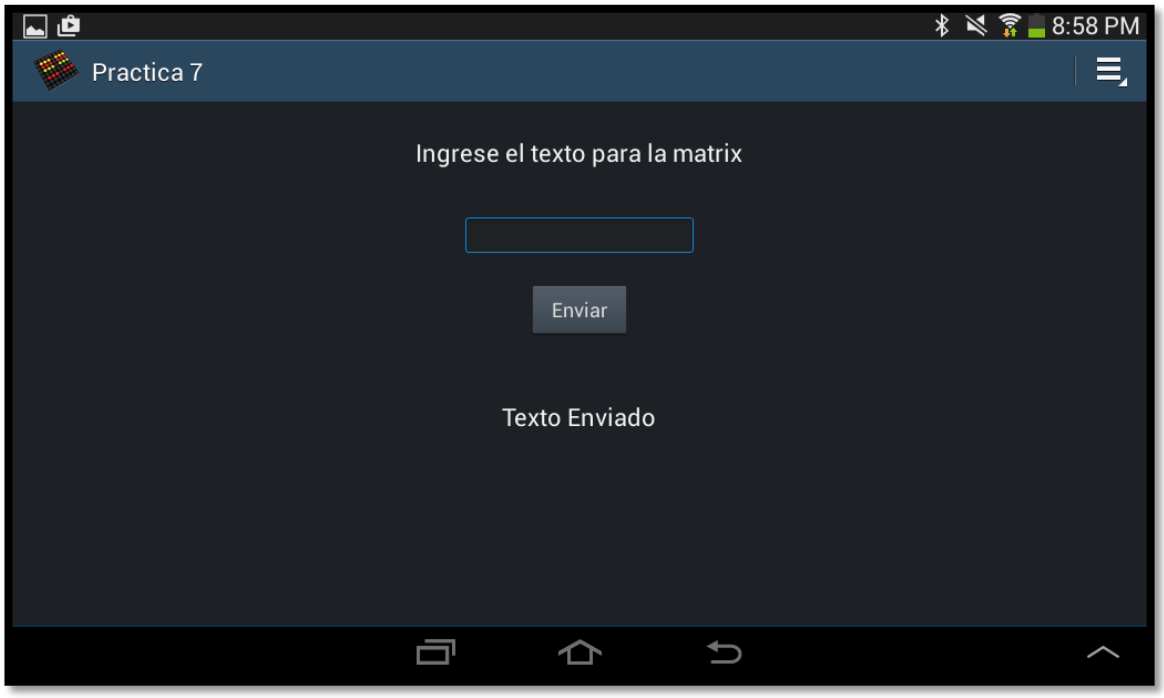


Figura: Vista interfaz práctica 7. Los autores

Se logra de manera exitosa la conexión del módulo bluetooth con el dispositivo Android, permitiendo así que la interacción desde la aplicación, el botón ENVIAR captura los caracteres digitados en el "edittext1" uno por uno y los transmite al arduino.

El código arduino interpreta a través de los datos recibidos por el modulo bluetooth desde el dispositivo Android los caracteres y los envía al drive de la matriz, la que a su vez los transfiere a esta última para que sean visualizados y desplazados a través de los diodos led que la conforman.

CONCLUSIONES:

Hemos implementado una aplicación Android para el control y visualización de caracteres en una matriz led a través de arduino.

5.8 Práctica 8: Mando de un servomotor

UNIVERSIDAD POLITÉCNICA SALESIANA

ESCUELA DE INGENIERÍA ELECTRÓNICA

PRÁCTICA: Nº8

TÍTULO: Mando de un servomotor

OBJETIVOS:

- Implementar una aplicación Android que permita controlar el movimiento de un servomotor de desplazamiento angular a través de arduino.

PROCEDIMIENTO:

1. Realizamos la conexión del dispositivo bluetooth en los pines Rx(0) y Tx(1), sin olvidar proveer el voltaje de alimentación (VCC y Gnd) desde los pines correspondientes en la placa arduino hacia el dispositivo bluetooth el mismo que permitirá establecer la comunicación entre el dispositivo Android y nuestra tarjeta Arduino.
2. Procedemos a realizar la conexión del servo motor, obteniendo VCC y GND (Red / Black) desde los pines que suministra el arduino. Para el pin de señal (White) utilizaremos el pin #9 del Servomotor.
3. Generamos el código de programación en Arduino, que capture datos recibidos a través del dispositivo bluetooth y envíe la señal correspondiente al Servomotor para que realice el desplazamiento al ángulo deseado.
4. Realizamos la aplicación para Android, sobre eclipse, la cual permita el control y desplazamiento del servomotor indicando el ángulo al cual debe este desplazarse entre 0° y 180°, el ángulo resultante seleccionado será enviado a través de la conexión bluetooth del dispositivo.
5. Establecemos la conexión bluetooth entre la placa arduino y el dispositivo en Android.

Materiales Utilizados

- Tarjeta Arduino UNO
- Cable USB
- Servo Motor Futaba S3003
- Cables de Conexión protoboard
- Modulo Bluetooth HC-06
- Tablet Android 7”

DESARROLLO:

a) Código Arduino

```
#include <Servo.h>
Servo miServo;
int angulo=90;

void setup()
{
  miServo.attach(9);
  Serial.begin(9600);
}

void loop(){
  unsigned char comando=0;
  if(Serial.available())
  {//solo leeremos si hay un byte en el buffer
    comando=Serial.read();//leemos el byte
    if(comando=='a')angulo=0;
    if(comando=='s')angulo=10;
    if(comando=='d')angulo=20;
    if(comando=='f')angulo=30;
    if(comando=='g')angulo=40;
    if(comando=='h')angulo=50;
```

```
  if(comando=='j')angulo=60;
  if(comando=='k')angulo=70;
  if(comando=='l')angulo=80;
  if(comando=='z')angulo=90;
  if(comando=='x')angulo=100;
  if(comando=='c')angulo=110;
  if(comando=='v')angulo=120;
  if(comando=='b')angulo=130;
  if(comando=='n')angulo=140;
  if(comando=='m')angulo=150;
  if(comando==',')angulo=160;
  if(comando=='.')angulo=170;
  if(comando=='-')angulo=180;
  angulo=constrain(angulo,0,180);//restringimos el valor de 0 a 180
 }
 miServo.write(angulo);
 Serial.print("Angulo:");Serial.println(angulo);
 delay(100);
}
```

b) Conexiones

Figura 5.20 Conexión practica 8

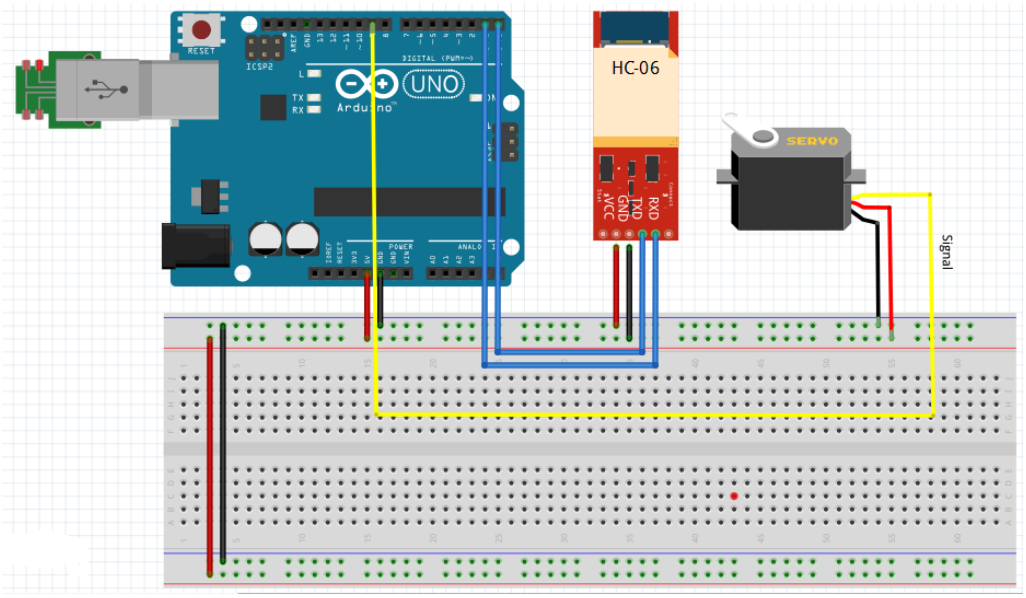


Figura: Vista física conexión practica 8. Los Autores

c) Aplicación Android

Para establecer la conexión desde el modulo bluetooth hacia el dispositivo Android utilizaremos la clase java *ConexionBT.java*, la misma que fue ya previamente desarrollada e implementada.

Manifest.xml

Establecemos los permisos de la aplicación sobre el dispositivo móvil y las versiones de Android que serán soportadas:

```
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="21" />
    <uses-permission android:name="android.permission.BLUETOOTH"/>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
    <uses-permission android:name="android.permission.VIBRATE"/>
```

Diseño del Layout: activity_practica8.xml

Definimos los elementos que serán parte de la interfaz gráfica de la aplicación:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.herramientas.HerramientasActivity" >
    <TextView
        android:id="@+id/tvgrados"
        android:text="Servomotor Grados"
        android:textAppearance="?android:attr/textAppearanceLarge" />
    <Button
        android:id="@+id/bt0"
        android:text=" 0º " />
    <Button
        android:id="@+id/bt90"
        android:text=" 90º" />
    <Button
        android:id="@+id/bt180"
        android:text="180º" />
    <Button
        android:id="@+id/bt60"
        android:text=" 60º" />
    <Button
        android:id="@+id/bt150"
```

```
        android:text="150º" />
    <Button
        android:id="@+id/bt30"
        android:text=" 30º" />
    <Button
        android:id="@+id/bt120"
        android:text="120º" />
    <SeekBar
        android:id="@+id/seekBar1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/bt180"
        android:layout_marginTop="46dp" />
</RelativeLayout>
```

Programación botones y seekbar: MainActivity.xml

Se declaran las acciones que serán ejecutadas por la aplicación al presionar un botón o desplazar el "seekbar":

```
public void onClick(View arg0) {
      switch(arg0.getId()){
      //se agrega un switch para escuchar en caso de que este pulsado
alguno
      case R.id.bt0:
      tvgrados.setText("0");
      sendMessage("a");
      break;
      case R.id.bt30:
      tvgrados.setText("30");
      sendMessage("f");
      break;
      case R.id.bt60:
      tvgrados.setText("60");
      sendMessage("j");
      break;
      case R.id.bt90:
      tvgrados.setText("90");
      sendMessage("z");
      break;
      case R.id.bt120:
      tvgrados.setText("120");
      sendMessage("v");
      break;
      case R.id.bt150:
      tvgrados.setText("150");
      sendMessage("m");
      break;
      case R.id.bt180:
      tvgrados.setText("180");
      sendMessage("-");
      break;
      }
```

```
        }

        public void onProgressChanged(SeekBar seekBar, int progress,
                      boolean fromUser) {
               tvgrados.setText(""+progress*10);
               switch(progress){
               //se agrega un switch para escuchar el progreso
               case 0:
                      tvgrados.setText("0");
                      sendMessage("a");
                      break;
               case 1:
                      tvgrados.setText("10");
                      sendMessage("s");
                      break;
               case 2:
                      tvgrados.setText("20");
                      sendMessage("d");
                      break;
               case 3:
                      tvgrados.setText("30");
                      sendMessage("f");
                      break;
               case 4:
                      tvgrados.setText("40");
                      sendMessage("g");
                      break;
               case 5:
                      tvgrados.setText("50");
                      sendMessage("h");
                      break;
               case 6:
                      tvgrados.setText("60");
                      sendMessage("j");
                      break;
               case 7:
                      tvgrados.setText("70");
                      sendMessage("k");
                      break;
               case 8:
                      tvgrados.setText("80");
                      sendMessage("l");
                      break;
               case 9:
                      tvgrados.setText("90");
                      sendMessage("z");
                      break;
               case 10:
                      tvgrados.setText("100");
                      sendMessage("x");
                      break;
               case 11:
                      tvgrados.setText("110");
                      sendMessage("c");
                      break;
               case 12:
                      tvgrados.setText("120");
                      sendMessage("v");
                      break;
```

```
            case 13:
                  tvgrados.setText("130");
                  sendMessage("b");
                  break;
            case 14:
                  tvgrados.setText("140");
                  sendMessage("n");
                  break;
            case 15:
                  tvgrados.setText("150");
                  sendMessage("m");
                  break;
            case 16:
                  tvgrados.setText("160");
                  sendMessage(",");
                  break;
            case 17:
                  tvgrados.setText("170");
                  sendMessage(".");
                  break;
            case 18:
            tvgrados.setText("180");
            sendMessage("-");
            break;
            }

            }
```

d) Resultado

Figura 5.21 Aplicación Android Practica 8

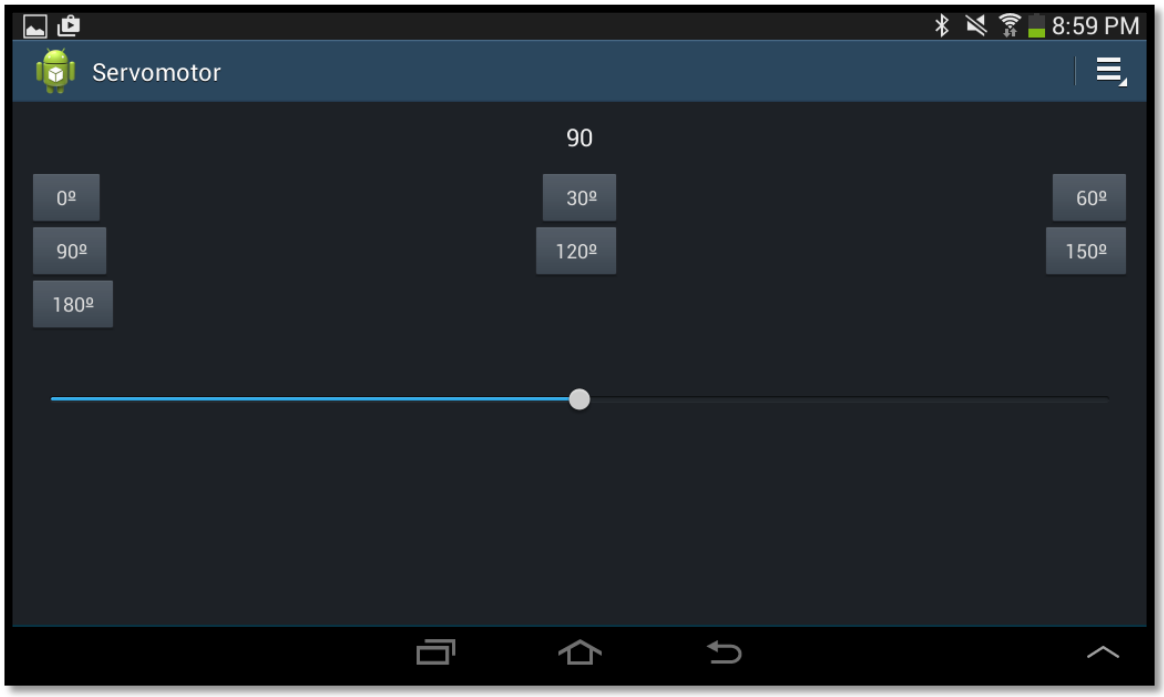


Figura: Vista interfaz práctica 8. Los autores

Se logra de manera exitosa la interacción desde la aplicación con los botones que definen los grados de ubicación y movimiento del Servomotor y del mismo modo la ubicación definida por el desplazamiento del seekbar. Estas posiciones son transmitidas al arduino a través de la conexión bluetooth establecida.
El código arduino interpreta a través de los datos recibidos por el modulo bluetooth desde el dispositivo Android y envía la señal de movimiento hacia el servomotor, ejecutando este último el movimiento hacia la posición definida desde la aplicación.

CONCLUSIONES:

Hemos implementado una aplicación Android que permite controlar el movimiento de un servomotor de movimiento angular a través de arduino.

5.9 Práctica 9: Cambio de sentido de giro y velocidad de un motor DC

UNIVERSIDAD POLITÉCNICA SALESIANA

ESCUELA DE INGENIERÍA ELECTRÓNICA

PRÁCTICA: Nº9

TÍTULO: Cambio de sentido de giro y velocidad de un motor DC

OBJETIVOS:

- Implementar una aplicación Android que permita controlar el sentido de giro y velocidad de un motor DC a través de arduino.

PROCEDIMIENTO:

1. Realizamos la conexión del dispositivo bluetooth en los pines Rx(0) y Tx(1), sin olvidar proveer el voltaje de alimentación (VCC y Gnd) desde los pines correspondientes en la placa arduino hacia el dispositivo bluetooth el mismo que permitirá establecer la comunicación entre el dispositivo Android y nuestra tarjeta Arduino.
2. Procedemos a realizar la conexión del motor DC a través del integrado controlador L293D. Se utilizaran los pines 4,5 y 6 para enviar las señales de Enable y polarización del motor.
3. Generamos el código de programación en Arduino, que capture datos recibidos a través del dispositivo bluetooth y envíe la señal correspondiente al motor DC para que realice el movimiento en el sentido y velocidad indicados en la aplicación.

4. Realizamos la aplicación para Android, sobre eclipse, la cual permita establecer el sentido de giro y velocidad del motor DC, estos datos serán enviado a través de la conexión bluetooth del dispositivo.
5. Establecemos la conexión bluetooth entre la placa arduino y el dispositivo en Android.

Materiales Utilizados

- Tarjeta Arduino UNO
- Cable USB
- Controlador de motor L293D
- Motor DC
- Cables de Conexión protoboard
- Modulo Bluetooth HC-06
- Tablet Android 7”
- Batería 5 V.

DESARROLLO:

a) Código Arduino

```
#include <SoftwareSerial.h>
SoftwareSerial Serial2(10, 11); // RX, TX
int motorPin1 = 3; // pin 2 on L293D IC
int motorPin2 = 4; // pin 7 on L293D IC
int enablePin = 5; // pin 1 on L293D IC

void setup() //Inicializa y prepara los valores
{
  Serial.begin(9600); //Abre el puerto serie a 9600 bps
  Serial.println("Speed (0-9) or + – to set direction"); //Imprime en la
consola puerto serie
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
  pinMode(enablePin, OUTPUT);
  digitalWrite(enablePin,HIGH);
}

void loop() //La función loop se ejecuta consecutivamente
{ //permitiéndole al programa variar y responder
  if ( Serial2.available())
  { //Si el puerto serie está disponible?
    char ch = Serial2.read(); //variable tipo char desde puerto serial
    if(ch >= '0' && ch <= '9') //Si ch es un numero?
    {
```

```
      int speed = map(ch, '0', '9', 0, 255); //Re-mapea un numero desde un
rango a otro
      analogWrite(enablePin, speed); //'0' a 0 '9' a 255
    }
    else if (ch == '+') //Si el caracter es '+'
    { //el motor avanza Clockwise
      digitalWrite(motorPin1,LOW);
      digitalWrite(motorPin2,HIGH);
    }
    else if (ch == '-') //Si el caracter es -
    { //el motor avanza CounterClockwise
      digitalWrite(motorPin1,HIGH);
      digitalWrite(motorPin2,LOW);
    }
    else //Cualquier otro caracter es omitido
    {
    }
```

b) Conexiones

Figura 5.22 Conexión Practica 9

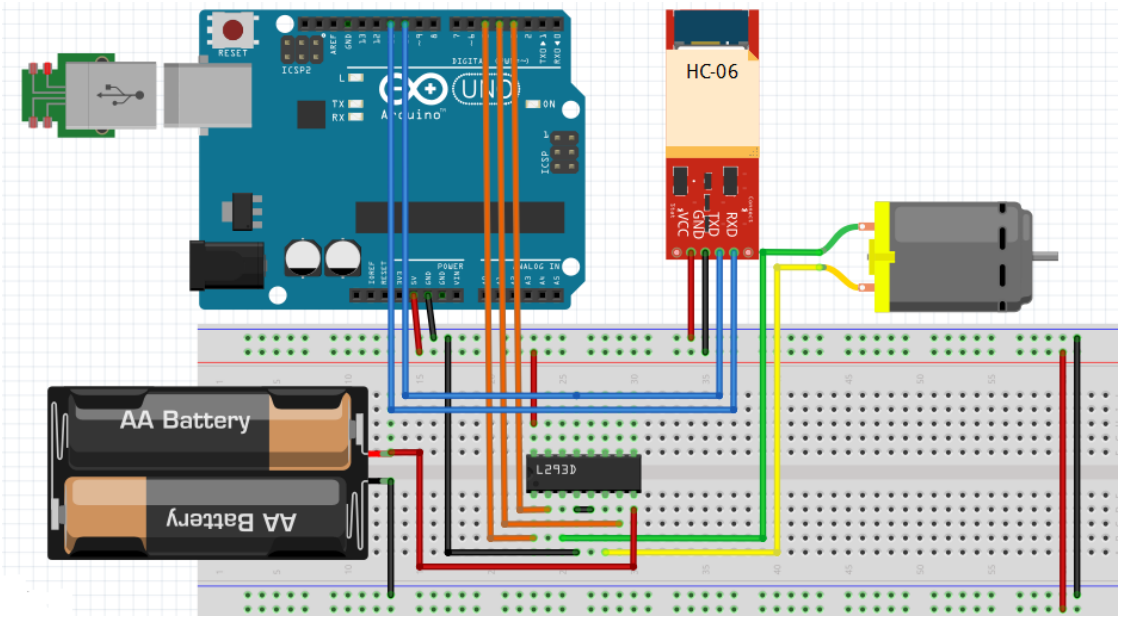


Figura: Vista conexión física practica 9. Los Autores

c) Aplicación Android

AndroidManifest.xml

Se establecen los permisos de la aplicación y las versiones soportadas:

```
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="21" />
    <uses-permission android:name="android.permission.BLUETOOTH"/>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
    <uses-permission android:name="android.permission.VIBRATE"/>
```

Layout: activity_mainpractica9.xml

Se definen los elementos de la interfaz gráfica para la aplicación:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.practica9.MainActivity">

    <TextView
        android:id="@+id/tvTitulo"
        android:text="Velocidad y Giro"/>

     <EditText
         android:id="@+id/etVelocidad"
         android:text=""
         />

    <Button
        android:id="@+id/btEnviar"
        android:text="Enviar"
        />

     <TextView
        android:id="@+id/tvMensaje"
        android:text=""/>

     <Button
        android:id="@+id/btIzquierda"
        android:text="Izquierda"/>

     <Button
        android:id="@+id/btDerecha"
        android:text="Derecha"/>
      <TextView
        android:id="@+id/tvDireccion"
        android:text=""/>
</RelativeLayout>
```

MainActivityPractica9.xml

Se declaran las acciones que serán ejecutadas por la aplicación y enviadas a través del Bluetooth cuando sean pulsados los botones de velocidad y sentido de giro, de acuerdo al valor de velocidad que se establezca en el TextView:

```
case R.id.btEnviar:
 String numeroEnv=etVelocidad.getText().toString();
 tvMensaje.setText("La velocidad es " + numeroEnv);
 sendMessage(numeroEnv);
break;
case R.id.btIzquierda:
 String giroEnvi=btIzquierda.getText().toString();
 tvDireccion.setText("La direccion es " + giroEnvi);
 sendMessage("+\r");
break;
case R.id.btDerecha:
 String giroEnvd=btDerecha.getText().toString();
 tvDireccion.setText("La direccion es " + giroEnvd);
 sendMessage("-\r");
break;
```

d) Resultado

Figura 5.23 Aplicación Android Practica 9

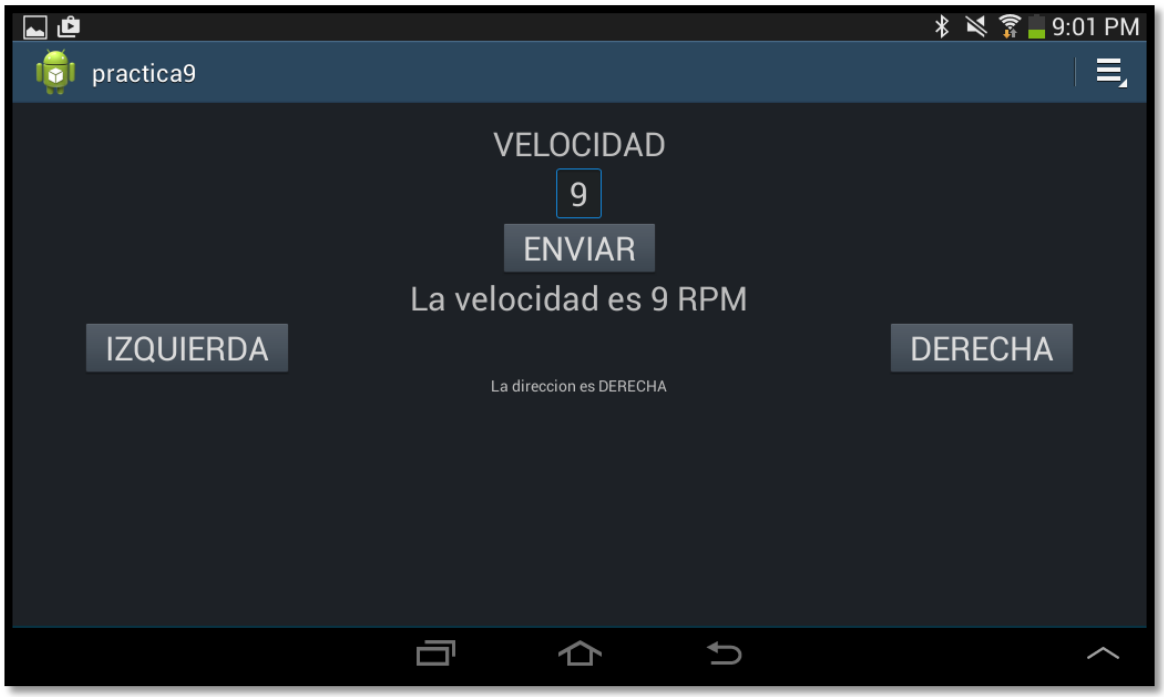


Figura: Vista interfaz de aplicación práctica 9. Los autores

CONCLUSIONES:

Hemos implementado una aplicación Android que permite controlar el sentido de giro y velocidad de un motor DC a través de arduino.

5.10 Práctica 10: Censado de temperatura y humedad

UNIVERSIDAD POLITÉCNICA SALESIANA

ESCUELA DE INGENIERÍA ELECTRÓNICA

PRÁCTICA 10

TÍTULO: Censado de temperatura y humedad

OBJETIVOS:

- Implementar una aplicación Android que permita el censado de la temperatura y humedad de un ambiente determinado a través de arduino.

PROCEDIMIENTO:

1. Realizamos la conexión del dispositivo bluetooth en los pines Rx(0) y Tx(1), sin olvidar proveer el voltaje de alimentación (VCC y Gnd) desde los pines correspondientes en la placa arduino hacia el dispositivo bluetooth el mismo que permitirá establecer la comunicación entre el dispositivo Android y nuestra tarjeta Arduino.
2. Procedemos con la conexión del DHT11, que es el dispositivo que realizará la captura de los valores de temperatura y humedad del ambiente donde se realizará el censado de los datos, utilizando para la entrega de los datos al arduino el pin de entrada A0..
3. Generamos el código de programación en Arduino, que capture datos recibidos a través del sensor de temperatura y humedad y los transmita a través del puerto serial bluetooth los valores capturados.
4. Realizamos la aplicación para Android, sobre eclipse, la cual permita interpretar a través de su conexión bluetooth los valores de temperatura y humedad recibidos desde el Android, para presentarlos en su interfaz.

DHT11

El DHT11 es un sensor de temperatura y humedad compatible con Arduino. Muestra los datos mediante una señal digital en el pin de datos. Este sensor obtiene nuevos datos una vez cada 2 segundos.

Materiales Utilizados

- Tarjeta Arduino UNO
- Cable USB
- Sensor de Temperatura y humedad DHT11
- Cables de Conexión protoboard
- Potenciómetro 5 K
- Display LCD de 2 filas
- Modulo Bluetooth HC-06
- Tablet Android 7”

DESARROLLO:

a) Código Arduino

```
#include <SoftwareSerial.h>
#include "DHT.h"
#include <LiquidCrystal.h>
#define DHTPIN A0
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
//int five_volts_Pin=12;

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup() {
                Serial.begin(9600);
   Serial.println("DHT test");
                //pinMode(five_volts_Pin, OUTPUT);
  // digitalWrite(five_volts_Pin,HIGH);
   lcd.begin(16, 2);
   lcd.print("hello, world!");
                dht.begin();
}

void loop() {
                float h = dht.readHumidity();
```

```
                float t = dht.readTemperature();
                if (isnan(t) || isnan(h))
    {
                Serial.println("Fallo al leer DHT11");
     lcd.setCursor(0,0);
                 lcd.print("Fallo al leer DHT11");
                }
    else
    {
                        lcd.setCursor(0,1);
                        lcd.print("Temperatura ");
                        lcd.print(t,0);
                        lcd.print("C");
                        lcd.setCursor(0,1);
                        lcd.print("Humedad ");
                        lcd.print(h,0);
                        lcd.print("%");
                        delay(1200);
          Serial.print("Humedad: ");
                        Serial.print(h);
                        Serial.print(" %\t");
          Serial.print("Temperatura: ");
                        Serial.print(t);
                        Serial.println(" *C");
                        lcd.setCursor(0,1);
                        lcd.print("Humedad ");
                        lcd.print(h,0);
                        lcd.print("%");
                        delay(1200);
                }
    delay(2000);
}
```

b) Conexiones

Figura 5.24 Conexión practica 10

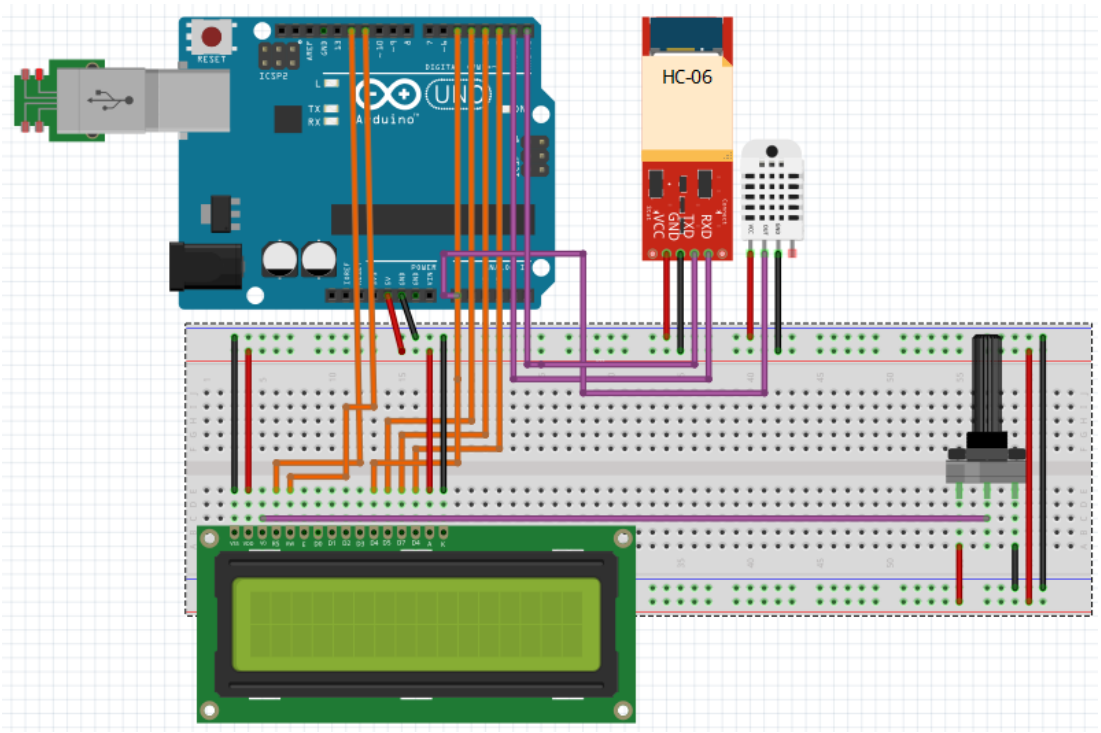


Figura: Conexión física practica 10. Los Autores

c) Aplicación Android

AndroidManifest.xml

Se establecen permisos para la aplicación y las versiones soportadas por esta:

```
    <uses-sdk
        android:minSdkVersion="7"
        android:targetSdkVersion="21" />
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.WAKE_LOCK" />
```

Layout: acitivity_main_practica10.xml

Definimos los elementos que serán parte de la interfaz gráfica, y como estos se organizan en la pantalla:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <TextView
        android:id="@+id/txtTemp"
        android:text="sensor"/>
    <Button
        android:id="@+id/bpConectar"
        android:text="Conectar" />
</RelativeLayout>
```

MainActivity.java

Declaramos la tarea que se ejecutará inmediatamente después de establecer la conexión en el dispositivo Bluetooth, esto es la lectura de datos:

```
  case MESSAGE_READ:
             int data_length;
                byte[] readBuf = (byte[]) msg.obj;
                data_length = msg.arg1;
                if (data_length>=35) {
                    String readMessage = new String(readBuf);
                    readMessage=readMessage.substring(0,data_length );
                    String sEnt=new String(readMessage.substring(0,35));
                    sEnt+="";
                    txtTemp.setText(sEnt); }
```

d) Resultado

Figura 5.25 Aplicación Android Practica 10

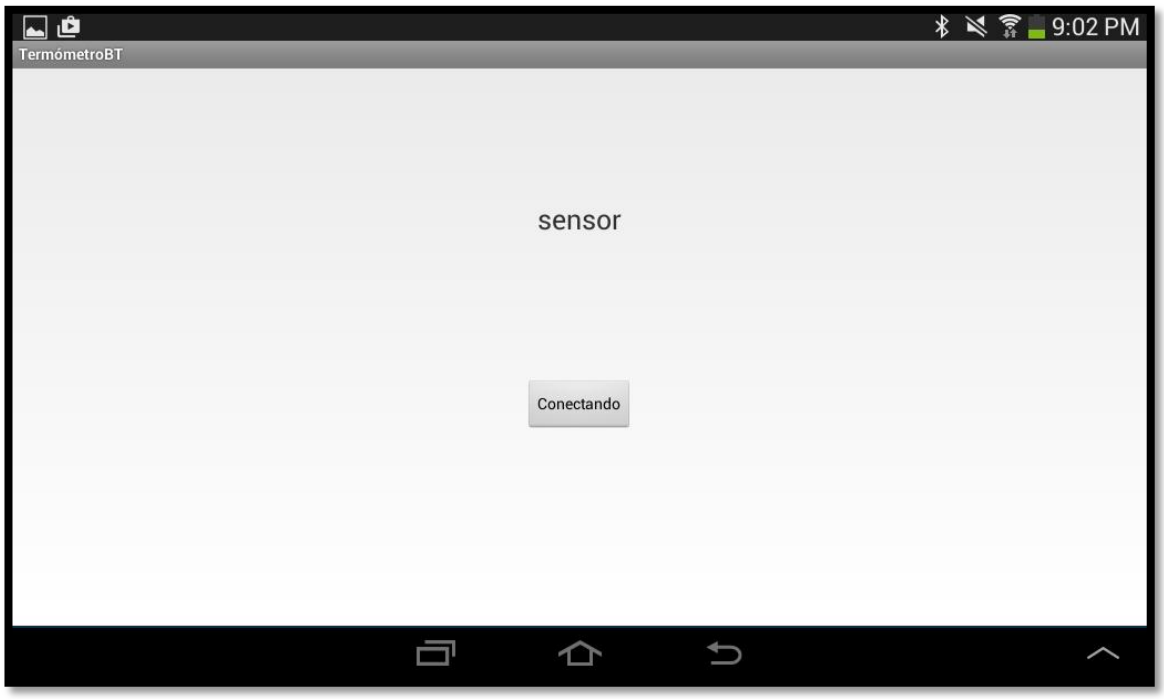


Figura: Vista interfaz práctica 10. Los autores

Al conectar el dispositivo Android con el bluetooth preestablecido, esto luego de presionar el botón de "Conectando" se empieza con la lectura de datos recopilados por el Sensor DHT-11.

Los datos son interpretados por la aplicación y mostrados en el TextView.

CONCLUSIONES

Se implementó de manera exitosa una aplicación Android que permite controlar la temperatura y humedad de un ambiente determinado a través de arduino.

5.11 Práctica 11: Control PWM de luminosidad DC y AC

UNIVERSIDAD POLITÉCNICA SALESIANA

ESCUELA DE INGENIERÍA ELECTRÓNICA

PRÁCTICA: Nº11

TÍTULO: Control PWM de luminosidad DC y AC

OBJETIVOS:

- Implementar una aplicación en Android que permita demostrar y simular la variación de luminosidad de elementos AC y DC, a través de su interfaz con arduino.

PROCEDIMIENTO:

Para el desarrollo de esta práctica se utiliza las siguientes herramientas disponibles:

Materiales Utilizados

- Tarjeta Arduino UNO
- Cable USB
- Diodo Led
- Resistencia 1K (x3)
- Resistencia 56 K (x1)
- Resistencia 120 K (x1)
- Resistencia 270 K (x1)
- Resistencia 5.5K
- Resistencia 39 ohm
- Triac BT 137
- Radiador T0220

- Diac BR100
- MOC 321
- Cables de Conexión protoboard
- Modulo Bluetooth HC-06
- Tablet Android 7”

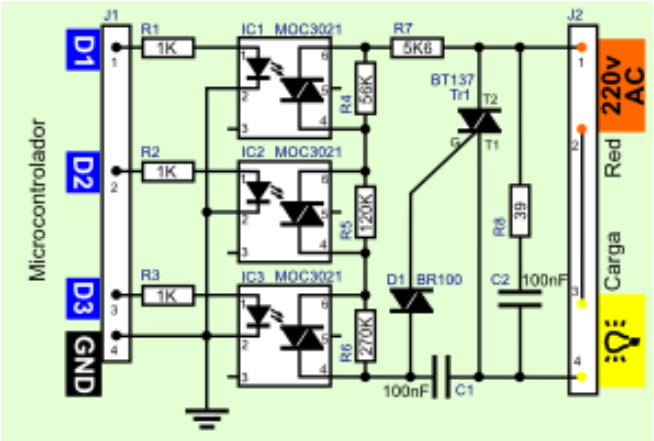
DESARROLLO:

a) Código Arduino

```
#define flag_inicio_char '*'

String inText;

void setup() {
  Serial.begin(9600);
  Serial.flush();
}

void loop()
{
  Serial.flush();
  int num_pin = 0;
  int value_pin = 0;

  char get_char = ' ';  // Lectura de datos desde puerto Serial (bluetooth)

  if (Serial.available() < 1) return; // Si no hay datos en puerto serial retorna al loop().
  // de lo contrario se lee el dato serial y se lo almacena en variable get_char.
  get_char = Serial.read();
  if (get_char != flag_inicio_char) return; // Si el valor leído NO contienen flag de inicio retorna al loop().

  // de lo contrario analiza pin# and valor
  num_pin = Serial.parseInt(); // lectura de valor pin
  value_pin = Serial.parseInt();  // lectura de valor value

    analogWrite( num_pin, value_pin ); //envío de valor a pin PWM seleccionado
    return;  // Acción completada, retorno al loop();
}
```

b) Conexiones

Figura 5.25 Conexiones AC pwm



Nota: Diagrama de conexión AC pwm. Los autores

c) Aplicación Android

AndroidMAnifest.xml

Establecemos los permisos para la aplicación y las versiones de Android que la soportarán:

```
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="21" />
    <uses-permission android:name="android.permission.BLUETOOTH"/>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
    <uses-permission android:name="android.permission.VIBRATE"/>
```

Layout: activity_mainpractica11.xml

Definimos los elementos que forman parte la interfaz gráfica de la aplicación en el Layout:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
```

```
    tools:context="com.example.practica11.Pwm" >
    <SeekBar
        android:id="@+id/seekBar1"
        android:max="255"
        android:progress="0"" />
    <SeekBar
        android:id="@+id/seekBar2"
        android:max="255"
        android:progress="0"/>
    <TextView
        android:id="@+id/textView2"
        android:text="@string/Led DC" />
    <TextView
        android:id="@+id/textView3"
        android:text="@string/Luz AC" />
    <TextView
        android:id="@+id/textView1"
        android:text="@string/titulo" />
</RelativeLayout>
```

MainActivity.java: Variación de Luminosidad

Declaramos las acciones que ejecutará la aplicación al desplazar el "seekbar" correspondiente a cada salida: DC y AC.

```
seek1.setOnSeekBarChangeListener(new OnSeekBarChangeListener(){
       public void onStartTrackingTouch(SeekBar seek1) {
       }
       public void onStopTrackingTouch(SeekBar seek1) {
       }
         public void onProgressChanged(SeekBar seek1, int progress1,
boolean fromUser) {
              sendMessage("d");
              sendMessage(Integer.toString(progress1));
       }
         });

         seek2.setOnSeekBarChangeListener(new OnSeekBarChangeListener(){
         public void onStartTrackingTouch(SeekBar seek2) {
         }
         public void onStopTrackingTouch(SeekBar seek2) {
         }
         public void onProgressChanged(SeekBar seek2, int progress2,
boolean fromUser) {
              sendMessage("a");
              sendMessage(Integer.toString(progress2));
       } });
```

d) Resultados

Figura 5.26 Aplicación Android Práctica 10

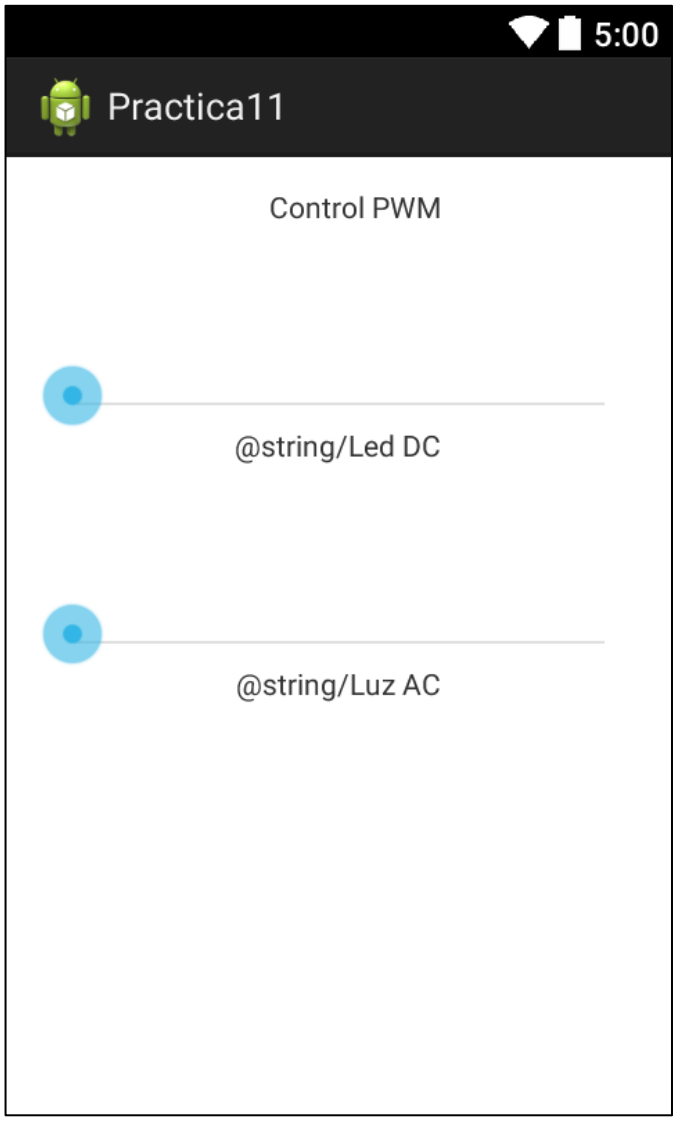


Figura: Vista interfaz aplicación parcita 11. Los Autores

Luego de establecer la conexión bluetooth, mientras se realiza la variación de los seekbar dentro de la aplicación se puede ver la variación física de la luminosidad en ambas salidas, DC y AC, en el protoboard.

CONCLUSIONES

Se iimplementó una aplicación en Android que permita demostrar y simular la variación de luminosidad de elementos AC y DC, a través de su interfaz con arduino.

5.12 Práctica 12: Control y recepción de imágenes vía cámara IP

UNIVERSIDAD POLITÉCNICA SALESIANA

ESCUELA DE INGENIERÍA ELECTRÓNICA

PRÁCTICA: Nº12

TÍTULO: Control y recepción de imágenes vía cámara IP

OBJETIVOS:

- Implementar una aplicación Android que permita visualizar las imágenes obtenidas por una cámara IP.

PROCEDIMIENTO:

Esta práctica consiste en desarrollar una aplicación sencilla en Android, que permita visualizar las imágenes / video capturado por una cámara IP en tiempo real.

1. Realizamos la aplicación para Android, sobre eclipse, la cual a través de su interfaz permita visualizar en tiempo real las imágenes / video capturados por la cámara IP.

Materiales Utilizados

- Tablet Android 7”
- Cámara IP “apexis”

DESARROLLO:

a) Conexiones

Figura 5.27 Conexión practica 12

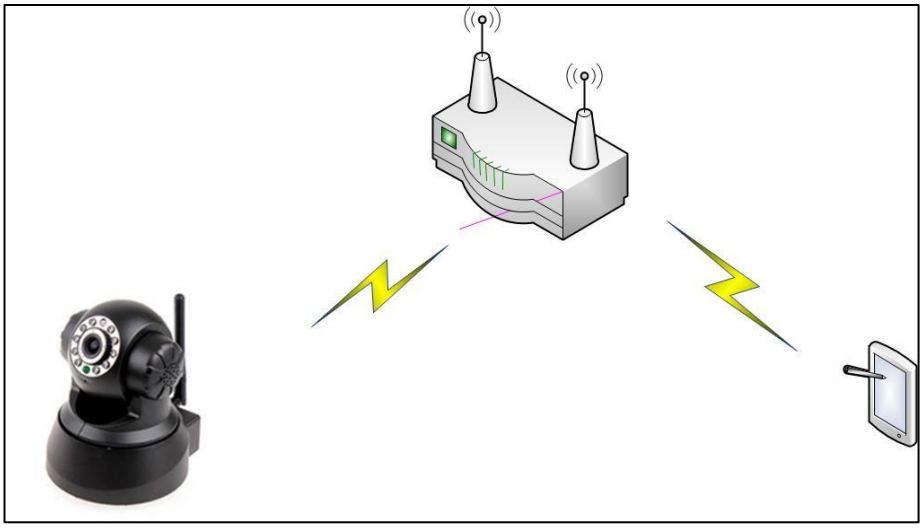


Figura: Vista esquemática conexión práctica 12. Los Autores

b) Aplicación Android

A continuación detallaremos los elementos principales que componen la aplicación desarrollada en eclipse:

AndroidManifest.xml

Se establecen los permisos para la aplicación y las versiones que la soportarán:

```
<uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="20" />
     <uses-permission android:name="android.permission.INTERNET" />
```

Diseño del Layout (Interfaz):activity_main.xml

Definimos los elementos que se utilizan en la interfaz gráfica de la aplicación:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.practica12.MainActivityPractica12" >

    <WebView
        android:id="@+id/webView1"/>

    <TextView
        android:id="@+id/textView1"
        android:text="@string/IP Cam" />

</RelativeLayout>
```

Main Acitivity.java

Se define las sentencias de java que permitirán la captura y visualización de imágenes obtenidas de la cámara IP, a través del control "Webview":

```
        public void onResume() {
         super.onResume();
         autoUpdate = new Timer();
         autoUpdate.schedule(new TimerTask() {

          public void run() {
           runOnUiThread(new Runnable() {
            public void run() {
             updateHTML();
            }
           });
          }
         }, 0, 200); // actualiza cada 200 ms
       }

       private void updateHTML(){
              mWebview =(WebView) findViewById(R.id.webView1);
              mWebview.getSettings().setJavaScriptEnabled(true);
mWebview.loadUrl("http://192.168.0.20:8860/snapshot.cgi?user=admin&pwd=&co
unt=2208");
              mWebview.reload();
              }
```

c) Resultado

Figura 5.28 Aplicación Android Practica 12

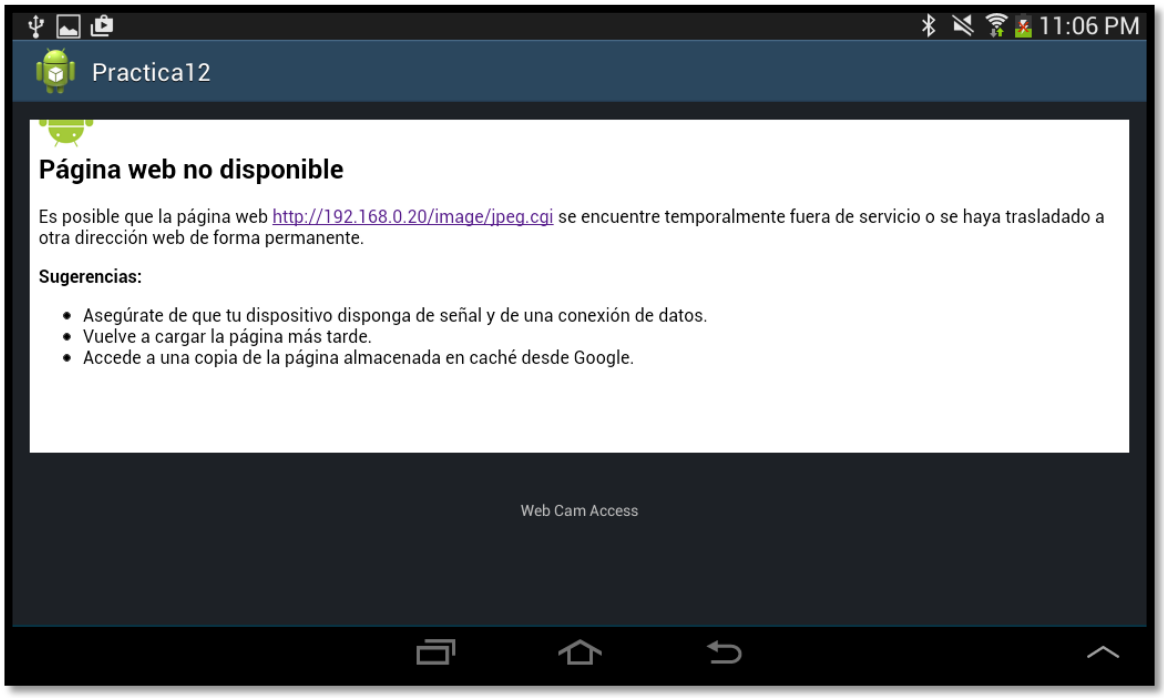


Figura: Interfaz aplicación práctica 12. Los autores

Logramos de manera exitosa la visualización de las imágenes capturadas por una cámara IP en nuestra aplicación Android desarrollada en eclipse. Para esto hicimos uso de un elemento importante de eclipse que es el timer, el cual nos permite realizar un update a la imagen capturada cada 200ms para así mostrarla en tiempo real en la interfaz de la aplicación

CONCLUSIONES

Hemos implementado una aplicación en Android que nos permitió la visualización de imágenes de video capturadas por una cámara IP.

5.13 Práctica 13: Mando remoto de un robot móvil explorador (control de luces, control de velocidad, sensor de temperatura, exploración vía cámara IP.

UNIVERSIDAD POLITÉCNICA SALESIANA

ESCUELA DE INGENIERÍA ELECTRÓNICA

PRÁCTICA: Nº13

TÍTULO: Mando remoto de un robot móvil explorador (control de luces, control de velocidad, sensor de temperatura, exploración vía cámara IP.

OBJETIVOS:

- Implementar una aplicación en Android, la cual a través de una conexión con Arduino permita el control remoto de un robot móvil explorador.

PROCEDIMIENTO:

Materiales Utilizados

- Tarjeta Arduino UNO
- Cable USB
- Diodo Led
- Resistencia 330 ohm
- Cables de Conexión protoboard
- Modulo Bluetooth HC-06
- Tablet Android 7”

DESARROLLO:

a) Código Arduino

```
//LIBRERÍAS UTILIZADAS
#include "DHT.h"
#include <SoftwareSerial.h>
#include <Servo.h>
#define DHTPIN 8
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

//PINES UTILIZADOS
int luces = 2;
int motorPin1 = 3; // pin 2 en L293D ICA
int motorPin2 = 4; // pin 7 en L293D IC
int motorPin3 = 5; // pin 15 en L293D IC
int motorPin4 = 7; // pin 10 en L293D IC
int enablePin = 6; // pin 1 en L293D IC
Servo miServo;
int angulo=90;
char comando;

void setup()
{
  miServo.attach(11); //PIN 11 DATOS DEL SERVOMOTOR
  Serial.begin(9600);
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
  pinMode(enablePin, OUTPUT);
  pinMode(motorPin3, OUTPUT);
  pinMode(motorPin4, OUTPUT);
  digitalWrite(enablePin,HIGH);
  pinMode(luces, OUTPUT);
  miServo.write(angulo);
  dht.begin();
}

void loop()
{
  if(Serial.available())
  {
    char comando=Serial.read();//leemos el byte
    //LUCES
    if(comando=='e')digitalWrite(luces,HIGH);
    if(comando=='a')digitalWrite(luces,LOW);

    //VELOCIDAD DEL CARRO
    if(comando >= '0' && comando <= '9')
    {
      int speed = map(comando, '0', '9', 0, 255); //Re-mapea un numero desde un rango a otro
      analogWrite(enablePin, speed); //'0' a 0 '9' a 255
      Serial.println(speed);
    }
    //DERECHA
    if(comando=='d')
    {
      digitalWrite(motorPin1,HIGH);
      digitalWrite(motorPin2,LOW);
```

```
   digitalWrite(motorPin3,LOW);
   digitalWrite(motorPin4,HIGH);
  }
  //IZQUIERDA
  if(comando=='i')
  {
   digitalWrite(motorPin4,LOW);
   digitalWrite(motorPin3,HIGH);
   digitalWrite(motorPin2,HIGH);
   digitalWrite(motorPin1,LOW);
  }
  //ADELANTE
  if(comando=='b')
  {
   digitalWrite(motorPin1,LOW);
   digitalWrite(motorPin2,HIGH);
   digitalWrite(motorPin3,LOW);
   digitalWrite(motorPin4,HIGH);
  }
  //ATRAS
  if(comando=='f')
  {
   digitalWrite(motorPin1,HIGH);
   digitalWrite(motorPin2,LOW);
   digitalWrite(motorPin3,HIGH);
   digitalWrite(motorPin4,LOW);
  }
    //PARAR
  if(comando=='s')
  {
   digitalWrite(motorPin1,LOW);
   digitalWrite(motorPin2,LOW);
   digitalWrite(motorPin3,LOW);
   digitalWrite(motorPin4,LOW);
  }
      //POSICIÓN CÁMARA
   if(comando=='t')angulo=0;//GIRO A LA DERECHA
   if(comando=='y')angulo=90;//GIRO A LA IZQUIERDA
   if(comando=='u')angulo=180;//MOVER AL CENTRO
   angulo=constrain(angulo,0,180);//restringimos el valor de 0 a 180

 }
miServo.write(angulo);

 float h = dht.readHumidity();
 float t = dht.readTemperature();
 if (isnan(t) || isnan(h))
 {
          Serial.println("Fallo al leer DHT11");
 }
 else
 {
  delay(100);
  Serial.print(" Humedad: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperatura: ");
  Serial.print(t);
  Serial.println(" *C");
  }}
```

b) Aplicación Android

AndroidManifest.xml

Aquí establecemos los permisos de la aplicación para los diferentes elementos que se utilizan, y las versiones de Android que soportarán la aplicación:

```
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="21" />
    <uses-permission android:name="android.permission.BLUETOOTH"/>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
    <uses-permission android:name="android.permission.VIBRATE"/>
    <uses-permission android:name="android.permission.INTERNET" />
```

activity_Main.xml

Se definen los elementos de la interfaz gráfica de la aplicación, se incluyen los elementos que permitirán visualizar imágenes de la Cámara IP y para el desplazamiento y dirección del móvil explorador y el servomotor.

```
            <TextView
                android:id="@+id/textView1"
                android:text="Temperatura"/>

            <TextView
                android:id="@+id/txTemperatura"
                android:text="Centigrados" />

            <TextView
                android:id="@+id/textView6"
                android:text="Posición Cámara"/>

            <Button
                android:id="@+id/bt180"
                android:text="180º" />

            <Button
                android:id="@+id/bt90"
                android:text="90º" />

            <Button
                android:id="@+id/bt0"
                android:text="0º" />
            <TextView
                android:id="@+id/txGrados"
                android:text="ºGrados" />
```

```
        <TextView
            android:id="@+id/textView7"
            android:text="Luces"/>

        <ToggleButton
            android:id="@+id/Led1"
            android:background="@drawable/ledoff" />
<WebView
    android:id="@+id/webvw"/>

        <TextView
            android:id="@+id/textView2"
            android:text="Control Motores"/>

        <Button
            android:id="@+id/button6"
            android:text="Button"/>

        <Button
            android:id="@+id/btAdelante"
            android:text="ADELANTE" />

        <Button
            android:id="@+id/button7"
            android:text="Button"/>

        <Button
            android:id="@+id/btIzquierda"
            android:text="IZQUIERDA" />

        <Button
            android:id="@+id/btParar"
            android:text="PARAR" />

        <Button
            android:id="@+id/btDerecha"
            android:text="DERECHA" />

        <Button
            android:id="@+id/button8"
            android:text="Button"" />

        <Button
            android:id="@+id/btAtras"
            android:text="  ATRAS  " />

        <Button
            android:id="@+id/button9"
            android:text="Button"/>

        <TextView
            android:id="@+id/textView3"
            android:text="Velocidad" />


        <SeekBar
            android:id="@+id/seekVelocidad"/>
```

```
            <TextView
                android:id="@+id/txVelocidad"
                android:text="RPM" />
```

MainActivity.java

Definimos la acción que ejecutará cada elemento de la aplicación al ser presionado o seleccionado, así como la captura de imágenes a través del control Wevbiew.

```
public void onClick(View arg0) {

      switch(arg0.getId()){
      //se agrega un switch para escuchar en caso de que este pulsado
alguno
      case R.id.btParar:
       sendMessage("s\r");
      break;
      case R.id.btIzquierda:
       sendMessage("i\r");
      break;
      case R.id.btDerecha:
       sendMessage("d\r");
      break;
      case R.id.btAdelante:
      sendMessage("f\r");
      break;
      case R.id.btAtras:
     sendMessage("b\r");
      break;

     case R.id.bt180:
      sendMessage("t\r");
      txGrados.setText("180");
      break;
      case R.id.bt90:
      sendMessage("y\r");
      txGrados.setText("90");
      break;
      case R.id.bt0:
      sendMessage("u\r");
      txGrados.setText("0");
      break;

public void onProgressChanged(SeekBar seekBar, int progress,
            boolean fromUser) {
      txVelocidad.setText(""+progress);

      switch(progress){
      case 0:
            txVelocidad.setText("0");
            sendMessage("0\r");
            break;
      case 1:
            txVelocidad.setText("1");
            sendMessage("1\r");
```

```
                break;
        case 2:
                txVelocidad.setText("2");
                sendMessage("2\r");
                break;
        case 3:
                txVelocidad.setText("3");
                sendMessage("3\r");
                break;
        case 4:
                txVelocidad.setText("4");
                sendMessage("4\r");
                break;
        case 5:
                txVelocidad.setText("5");
                sendMessage("5\r");
                break;
        case 6:
                txVelocidad.setText("6");
                sendMessage("6\r");
                break;
        case 7:
                txVelocidad.setText("7");
                sendMessage("7\r");
                break;
        case 8:
                txVelocidad.setText("8");
                sendMessage("8\r");
                break;
        case 9:
                txVelocidad.setText("9");
                sendMessage("9\r");
                break;}
}
 //Cámara
@Override
public void onResume() {
 super.onResume();
 autoUpdate = new Timer();
 autoUpdate.schedule(new TimerTask() {
  @Override
  public void run() {
   runOnUiThread(new Runnable() {
    public void run() {
     updateHTML();
    }
   });
  }
 }, 0, 200); // actualiza cada 200 ms
}

private void updateHTML(){
       mWebview =(WebView) findViewById(R.id.webvw);
       mWebview.getSettings().setJavaScriptEnabled(true);
       mWebview.loadUrl("http://192.168.0.20/image/jpeg.cgi");
       mWebview.reload();}
```

c) Resultado

Figura 5.29 Aplicación Android Practica 13

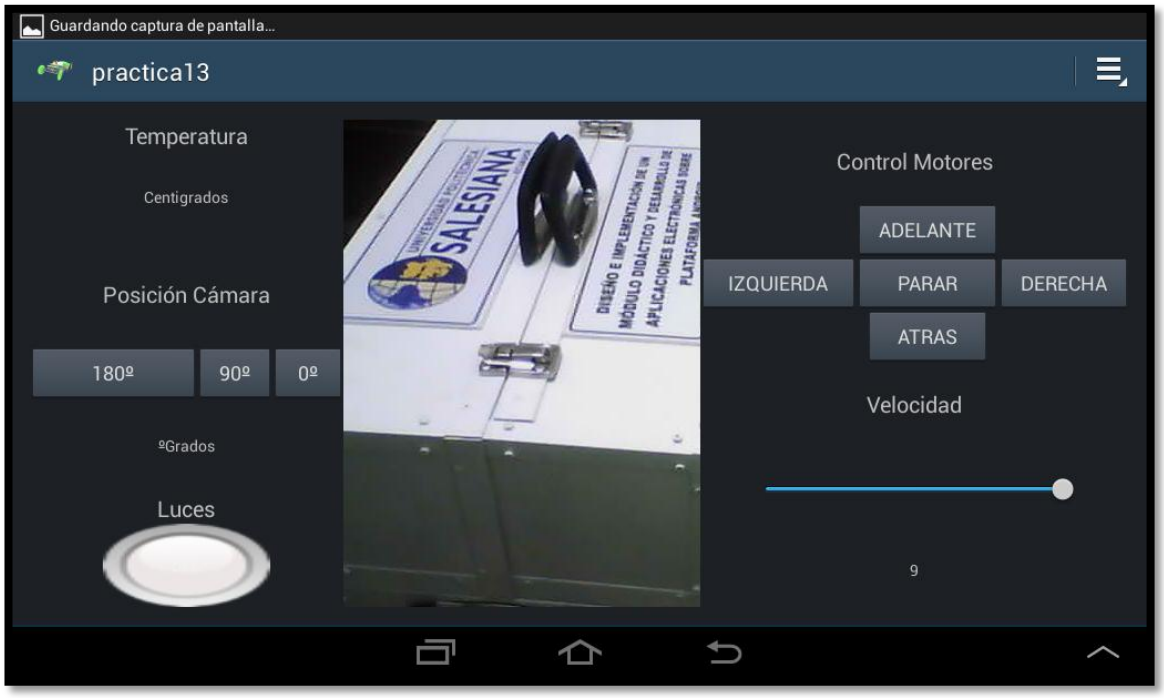


Figura: Vista interfaz aplicación práctica 13. Los autores

Esta práctica se compone de una compilación de todas las prácticas propuestas en el presente proyecto, lo cual se logra de manera exitosa gracias a los diferentes eventos y que pueden ser manejados por ambas plataformas (Android y Arduino). Así podemos tener todos los parámetros y controles en una misma interfaz sobre una misma aplicación.

CONCLUSIONES

Se consigue implementar de manera exitosa una aplicación en Android, la cual a través de una conexión con Arduino permita el control remoto de un robot móvil explorador.

PRESUPUESTO

ELEMENTO	CANTIDAD	PRECIO UNITARIO	SUBTOTAL
Samsung Galaxy tab 2 de 7"	1.00	$ 200.00	$ 200.00
Laptop Hp 455	1.00	$ 500.00	$ 500.00
Protoboard	6.00	$ 5.00	$ 30.00
batería 6v 4amp	1.00	$ 1.00	$ 1.00
cargador de batería	1.00	$ 1.00	$ 1.00
Pinzas de ropa de madera	24.00	$ 1.00	$ 24.00
arduino uno	4.00	$ 25.00	$ 100.00
displays	2.00	$ 7.00	$ 14.00
Cámara D-Link DCS-930L	1.00	$ 1.00	$ 1.00
Cámara Apexis	1.00	$ 1.00	$ 1.00
Bluetooth Hc 06	4.00	$ 16.00	$ 64.00
Sensor de Temperatura DHT11	2.00	$ 15.00	$ 30.00
Servomotor Futaba s3003	2.00	$ 22.00	$ 44.00
Motor reductor	8.00	$ 10.00	$ 80.00
Sócalos 14 pines	30.00	$ 0.08	$ 2.40
Sócalos 40 pines	3.00	$ 0.50	$ 1.50
Espadines	10.00	$ 0.50	$ 5.00
Borneras	10.00	$ 0.25	$ 2.50
Jack Banana	9.00	$ 0.65	$ 5.85
Tip 41c	8.00	$ 0.60	$ 4.80
Tip 42c	8.00	$ 0.60	$ 4.80
Resistencias 10K	100.00	$ 0.04	$ 4.00
7805	1.00	$ 0.60	$ 0.60
Lm324	5.00	$ 0.50	$ 2.50
Espadines	6.00	$ 0.50	$ 3.00
2n3904	10.00	$ 0.08	$ 0.80
2n3906	10.00	$ 0.08	$ 0.80
Diodos led	6.00	$ 0.10	$ 0.60
Diodos 1Amp	8.00	$ 0.08	$ 0.64
2200 - 16V	2.00	$ 0.25	$ 0.50
Cable telefónico	4.00	$ 0.25	$ 1.00
BTA 12 - 600B	1.00	$ 1.00	$ 1.00
Resistencias 47K	2.00	$ 0.05	$ 0.10
Resistencias de 100ohm	2.00	$ 0.05	$ 0.10
3904	4.00	$ 0.08	$ 0.32
3906	4.00	$ 0.08	$ 0.32
Transformador 120-12V 1amp	1.00	$ 7.80	$ 7.80
Transformador 9-9V 1amp	1.00	$ 7.80	$ 7.80
Baquelita 20x30	1.00	$ 2.80	$ 2.80
Fundas de percloruro	5.00	$ 0.60	$ 3.00
Switch on-off 3Pin Negro	2.00	$ 0.30	$ 0.60

ELEMENTO	CANTIDAD	PRECIO UNITARIO	SUBTOTAL
Punta Broca 1mm	1.00	$ 0.75	$ 0.75
Borneras 12 pines	20.00	$ 0.20	$ 4.00
Fusibles 8amp	8.00	$ 0.05	$ 0.40
Portafusibles 4 cables	6.00	$ 0.30	$ 1.80
Portafusibles rosca	2.00	$ 0.25	$ 0.50
Baquelita 10x16 perforada	1.00	$ 2.00	$ 2.00
Percloruro	3.00	$ 0.60	$ 1.80
Papel Térmico	3.00	$ 1.00	$ 3.00
Transformador 12v - 12V - 1A	2.00	$ 6.00	$ 12.00
Transformador 9v - 9V - 1A	1.00	$ 5.80	$ 5.80
Capacitor 10000uf 25v	2.00	$ 1.50	$ 3.00
Lm324	19.00	$ 0.35	$ 6.65
Resistencia 1/2w- 10k	102.00	$ 0.03	$ 3.06
Resistencia 1/2w- 100k	20.00	$ 0.03	$ 0.60
Resistencia 1/2w- 220ohm	30.00	$ 0.03	$ 0.90
Resistencia 1/2w- 0.47ohm	20.00	$ 0.10	$ 2.00
Resistencia 1/2w- 1k	35.00	$ 0.03	$ 1.05
2N3904	16.00	$ 0.10	$ 1.60
Lm317-T	4.00	$ 0.50	$ 2.00
TIP 41	6.00	$ 0.50	$ 3.00
TIP 42	6.00	$ 0.50	$ 3.00
Marquillas 1	1.00	$ 4.06	$ 4.06
Marquillas 2	1.00	$ 4.06	$ 4.06
Marquillas 3	1.00	$ 4.06	$ 4.06
Marquillas 4	1.00	$ 4.06	$ 4.06
Marquillas 5	1.00	$ 4.06	$ 4.06
agarraderas			
soporte Tablet			
Fibra			
		GRAN TOTAL	**$1,234.34**

CONCLUSIONES

Hemos diseñado e implementado un módulo didáctico que permitirá a los estudiantes desarrollar aplicaciones sobre plataforma Android para la construcción de soluciones con diferentes dispositivos electrónicos.

Logramos desarrollar aplicaciones Android que permiten establecer comunicación bluetooth con el módulo de entrenamiento Arduino.

Interactuamos con el medio físico a través de sensores compatibles con el módulo de entrenamiento Arduino bluetooth, Ethernet, USB, temperatura y humedad.

Desarrollamos un módulo para el aprendizaje, comprensión y aplicación de elementos electrónicos con la plataforma Android.

Implementamos y demostramos, de modo práctico-experimental, las aplicaciones que puede ofrecer la plataforma Android, en conjunto con los diferentes elementos que ayudan a la interacción con el exterior.

Conseguimos demostrar en aplicaciones reales la interacción de los dispositivos y elementos electrónicos con la plataforma Android.

RECOMENDACIONES

Los autores recomiendan para un mejor provecho de los elementos y desarrollo de aplicaciones tener una base conceptual de programación, puesto que en Android su estructura de control se basa en sentencias JAVA que permiten a los elementos interactuar con el medio físico.

Se recomienda la consecución del proyecto en nuevas prácticas ya que el uso de dispositivos con sistema operativo Android aumenta día a día permitiendo el desarrollo de aplicaciones con mayor grado de complejidad y solución a mayor cantidad de problemas tecnológicos simplificando el uso de dispositivos con elementos de la vida cotidiana.

Es posible la modificación del maletín educativo debido a que se realizó el proyecto pensando en futuras mejoras, dependiendo de las necesidades que presenten los docentes o estudiantes en el uso continuo del mismo. Diseñado de manera modular y etiquetado de cada una de las conexiones que conforman el maletín para su respectiva identificación o corrección de problemas que se puedan ocasionar con el debido desgaste en el tiempo de uso.

BIBLIOGRAFÍA

- ACM Company C. A., (2015) recuperado el 09 de enero de 2015, de http://www.acmcompanyca.com/especificaciones.php
- Antony García, Kiara Navarro y su Equipo Creativo, (2015) recuperado el 09 de enero de 2015, de http://panamahitek.com/dht22-sensor-de-humedadtemperatura-de-precision-para-arduino/
- Arduino SA, 2015. Arduino Guide Recuperado de http://arduino.cc/en/Main/ArduinoBoardUno
- Arduino SA, 2015. Arduino UNO Recuperado de http://arduino.cc/en/Main/ArduinoBoardUno
- Arduino, (2015) recuperado el 09 de enero de 2015, de http://arduino.cc/en/pmwiki.php?n=Main/ArduinoBoardUno
- Arduino, (2015) recuperado el 09 de enero de 2015, de http://arduino.cc/en/pmwiki.php?n=Main%2FArduinoEthernetShield
- Arduino, (2015) recuperado el 09 de enero de 2015, de http://arduino.cc/en/uploads/Main/arduino-ethernet-shield-06-schematic.pdf
- Arduino, (2015) recuperado el 09 de enero de 2015, de http://www.ardublog.com/library-for-arduino-ultrasonic-ranging-hc-sr04/
- Automatización y Robótica Industrial, (2015) recuperado el 09 de enero de 2015, de http://www.esi2.us.es/~vivas/ayr2iaei/LOC_MOV.pdf
- Eduardo J. Carletti, Motores paso a paso Características básicas, (2015) recuperado el 09 de enero de 2015, de http://robots-argentina.com.ar/MotorPP_basico.htm
- electronicaestudio.com, (2015) recuperado el 09 de enero de 2015, de http://www.electronicaestudio.com/arduino_productos.htm
- Electronilab.co, (2015) recuperado el 09 de enero de 2015, de http://electronilab.co/tienda/sensor-de-distancia-de-ultrasonido-hc-sr04/
- Enríquez, R., 2009, "Guía de Usuario de Arduino"
- Google Inc, 2015. Android APIs Recuperado de http://developer.android.com/reference/android/widget/TextView.html
- Google Inc, 2015. Fundamentos de Aplicaciones Recuperado de http://developer.android.com/intl/es/guide/components/fundamentals.html

- Google Inc, 2015. Introducción a Android Recuperado de http://developer.android.com/guide/index.html
- Google Inc, 2015. Training for Android Developers Recuperado de http://developer.android.com/intl/es/training/basics/firstapp/building-ui.html
- Introducción al desarrollo en Android **http://www.ibm.com/developerworks/ssa/library/os-android-devel/**
- Irving M. Gottlieb Motores eléctricos y técnicas de control, (2015) recuperado el 09 de enero de 2015, de http://www.goodreads.com/book/show/1006167.Electric_Motors_and_Control_Techniques
- J.Rodrigo, (2015) recuperado el 09 de enero de 2015, de http://playground.arduino.cc/Main/MAX72XXHardware#WiringLedMatrix
- Joseph Annuzzi, Jr.; Lauren Darcey; Shane Conder, 2013, "Introduction to Android™ Application Development: Android Essentials", Fourth edition
- muchotrasto.com, (2015) recuperado el 09 de enero de 2015, de http://www.muchotrasto.com/TiposDePlataformas.php
- Nikos Drakos, Unidad Computer Based Learning de la Universidad de Leeds, (2015) recuperado el 09 de enero de 2015, de http://www.tamps.cinvestav.mx/~mgomez/Odometria/node4.html
- Potenciaelectromecanica.com, (2015) recuperado el 09 de enero de 2015, de http://www.potenciaelectromecanica.com/calculo-de-un-motorreductor/
- Regata, (2015) recuperado el 09 de enero de 2015, de http://tallerarduino.com/2012/12/24/sensor-dht11-humedad-y-temperatura-con-arduino/
- Tecnófilo,(2015) recuperado el 09 de enero de 2015, de http://www.tecnofilo.es/tienda/comunicaciones/58-modulo-inalambrico-bluetooth-hc-06-con-base-de-conexion.html
- Todoelectronica.com, (2015) recuperado el 09 de enero de 2015, de http://www.todoelectronica.com/motorreductor-6-vdc-180-rpm-p-89051.html
- Unicrom.com, (2015) recuperado el 09 de enero de 2015, de http://www.unicrom.com/Tut_MotorAC.asp
- Wavesen, (2015) recuperado el 09 de enero de 2015, de http://www.neoteo.com/modulo-bluetooth-hc-06-android/

ANEXOS

ANEXO 1. Programación Android Práctica 5

ANEXO 2. Programación Android Práctica 6

ANEXO 3. Programación Android Práctica 7

ANEXO 4. Programación Android Práctica 8

ANEXO 5. Programación Android Práctica 9

ANEXO 6. Programación Android Práctica 10

ANEXO 7. Programación Android Práctica 11

ANEXO 8. Programación Android Práctica 12

ANEXO 9. Programación Android Práctica 13

ANEXO 10. Max7219/Max7221

ANEXO 11. BR 100/04

ANEXO 12. BT137 Series

ANEXO 13. Dlink DCS930L

ANEXO 14. S3003 FUTABA SERVO

ANEXO 15. L293, L293D

ANEXO 16. MOC3020 Optocouplers

ANEXO 17. Arduino Ethernet Shield

ANEXO 18. Arduino UNO

ANEXO 19. DHT11

ANEXO 20. Bluetooth HC-06

ANEXO 21. SG90 Micro Servo

ANEXO 22. Alucobond