UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

UNIDAD DE GESTIÓN DE TECNOLOGÍAS

DEPARTAMENTO DE ELECTRÓNICA Y COMPUTACIÓN

"IMPLEMENTACIÓN DE UN PROTOTIPO DE CONTROL PARA UN BRAZO ROBÓTICO EN BASE A UNA INTERFAZ ANDROID-ARDUINO PARA EL LABORATORIO DE INSTRUMENTACIÓN VIRTUAL DE LA UNIDAD DE GESTIÓN DE TECNOLOGÍAS UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE."

AUTOR: ZAPATA TOAPANTA EDWIN ISRAEL

Trabajo de Graduación para la obtención del título de:

TECNÓLOGO EN ELECTRÓNICA MENCIÓN INSTRUMENTACIÓN & AVIÓNICA

Latacunga, Mayo 2015

CERTIFICADO

Certifico que el presente Trabajo de Graduación fue realizado en su totalidad por el Sr. **ZAPATA TOAPANTA EDWIN ISRAEL**, como requerimiento parcial para la obtención del título de **TECNÓLOGO EN ELECTRÓNICA MENCIÓN INSTRUMENTACIÓN & AVIÓNICA.**

_____

SR. ING. JORGE PARDO
DIRECTOR DEL TRABAJO DE GRADUACIÓN

Latacunga, Mayo 2015

AUTORÍA DE RESPONSABILIDAD

Yo, Edwin Israel Zapata Toapanta

DECLARO QUE:

El trabajo de grado denominado “IMPLEMENTACIÓN DE UN PROTOTIPO DE CONTROL PARA UN BRAZO ROBÓTICO EN BASE A UNA INTERFAZ ANDROID-ARDUINO PARA EL LABORATORIO DE INSTRUMENTACIÓN VIRTUAL DE LA UNIDAD DE GESTIÓN DE TECNOLOGÍAS UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE”, ha sido desarrollado en base a una investigación científica exhaustiva, respetando derechos intelectuales de terceros conforme las citas constan al pie de las páginas correspondientes, cuyas fuentes se incorporan en la bibliografía.

Consecuentemente, este trabajo es de mi autoría.

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

Latacunga, Mayo 2015

_____

Edwin Israel Zapata Toapanta

C.I: 050309975-6

AUTORIZACIÓN

Yo, Edwin Israel Zapata Toapanta

Autorizo a la Universidad de las Fuerzas Armadas ESPE la publicación, en la biblioteca virtual de la Institución el trabajo “IMPLEMENTACIÓN DE UN PROTOTIPO DE CONTROL PARA UN BRAZO ROBÓTICO EN BASE A UNA INTERFAZ ANDROID-ARDUINO PARA EL LABORATORIO DE INSTRUMENTACIÓN VIRTUAL DE LA UNIDAD DE GESTIÓN DE TECNOLOGÍAS UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE”, cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y autoría.

Latacunga, Mayo 2015

_____

Edwin Israel Zapata Toapanta

C.I: 050309975-6

DEDICATORIA

Con mucho cariño y amor adjudico el esfuerzo y el sacrificio dedicado a este trabajo investigativo, a mis padres Edwin René y Blanca Azucena, por ser el motor de empuje para la realización y culminación de mis estudios y de muchos otros proyectos planteados en mi vida personal, estudiantil y profesional.

A mi hermano Jason Enrique quien pese a no estar físicamente conmigo, día a día ha estado presente en cada acción realizada por mí y quien sin duda seguirá a mi lado siempre guiando mis pasos y orientándome siempre. Algún día nos volveremos a encontrar.

Y de una manera muy especial quiero dedicar este trabajo a mi hermana Doménica Anahí para que sea un pequeño ejemplo a emular en su vida que apenas empieza a florecer.

Edwin Israel

AGRADECIMIENTO

Sin dudarlo, mi eterno agradecimiento a Dios por la salud y la vida que me brinda, por las oportunidades que ha puesto en mi vida y por darme la oportunidad de culminar con éxito un pequeño logro personal más. También mi eterno agradecimiento a mis padres René y Blanca por ser el pilar fundamental de mi vida, por demostrarme que la vida, sin sacrificios entrega y amor, no es vida; por tanta confianza depositada en mí, y sobre todo por cada sacrificio realizado para mi bienestar, en fin toda una vida no bastara para agradecerles todo lo que han hecho y seguirán haciendo por mí.

A mi hermana Doménica por llenarme la vida de ternura cariño y amor, por ser quién involuntariamente me motiva día a día a superarme.

De una manera especial quiero agradecer también al Ingeniero Guillermo Trujillo Jaramillo, por ser quien me oriento en momentos de confusión y por brindarme su apoyo incondicional y desinteresado día a día.

A cada uno de mis familiares y amigos, quienes de una u otra manera me han animado a seguir en momentos en los que necesitaba ayuda moral.

A todos y cada uno que Dios les pague por todo.

Edwin Israel

ÍNDICE DE CONTENIDOS

CERTIFICADO ii
AUTORÍA DE RESPONSABILIDAD iii
AUTORIZACIÓN iv
DEDICATORIA v
AGRADECIMIENTO vi
ÍNDICE DE CONTENIDOS vii
ÍNDICE DE TABLAS x
ÍNDICE DE FIGURAS xi
RESUMEN xiii
ABSTRACT xiv
CAPÍTULO I 15
1.1. ANTECEDENTES 15
1.2. PLANTEAMIENTO DEL PROBLEMA 15
1.3. JUSTIFICACIÓN E IMPORTANCIA 16
1.4. OBJETIVOS: 17
1.4.1. Objetivo General: 17
1.4.2. Objetivos Específicos: 17
1.5. ALCANCE 18
CAPÍTULO II 19
MARCO TEÓRICO 19
2.1. KITS DE BRAZOS ROBÓTICOS. 19
2.2 GENERALIDADES DE ARDUINO 20
2.2.1 ¿Qué es Arduino? 20
2.2.2 Hardware Arduino 20
2.2.2.1 Tipos de tarjetas Arduino. 22
2.2.3 Lenguaje de programación Arduino. 25
2.2.3.1 Sintaxis Básica. 27
2.2.3.2 Variables. 27
2.2.3.3 Constantes. 27
2.2.3.4 Tipos de datos. 28

2.2.3.5 Conversiones entre tipos. 28
2.2.3.6 Entradas/salidas digitales 28
2.2.3.7 Entradas/salidas analógicas 28
2.2.3.8 Tiempo 29
2.2.3.9 Comunicación por puerto serie 29
2.2.3.10 Manipulación de puertos 29
2.3 Módulo Bluetooth HC05 como dispositivo cliente en la comunicación. 30
2.3.1 Generalidades 30
2.3.2 Conexión básica entre el módulo bluetooth y la tarjeta Arduino. 31
2.4 Introducción a Android 32
2.4.1 Estructura de Android 33
2.4.2 Plataformas de desarrollo Android. 37
2.4.2.1 Basic4Android 37
2.4.2.2 Mono para Android. 38
2.4.2.3 App Inventor 38
2.4.3 App Inventor 2. 39
2.4.3.1 ¿Qué es MIT App Inventor 2? 39
2.4.3.2. Componentes. 43
2.4.3.3 Bloques. 47
CAPÍTULO III 48
DESARROLLO DEL TEMA. 48
3.1 ENSAMBLAJE DE LOS BRAZOS ROBÓTICOS 48
3.2 CONFIGURACIÓN DEL MÓDULO BLUETOOTH HC-06. 50
3.2.1 Compilar y cargar la configuración al módulo HC 06 51
3.3 PROGRAMACIÓN DEL SOFTWARE DE CONTROL ARDUINO. 52
3.3.1 Declaración de variables. 53
3.3.2 Configuración de comunicación, lectura de trama y asignación de entradas y salidas digitales. 54
3.3.3 Estructura del lazo de control. 56
3.4 PROGRAMACIÓN DE INTERFAZ DE CONTROL (APLICACIÓN ANDROID) 66
3.4.1 Diseño de la interfaz visual 66
3.4.1.1 Portada. 69

3.4.1.2 Menú....73
3.4.1.3 Control Brazo A y Brazo B. 76
3.4.1.4 Control Automático. 79
3.4.1.5 Ayuda.....81
3.4.1.5 Acerca de.....82
3.4.2 Diagramas de bloques de la aplicación. 83
3.4.2.1 Programación de la pantalla Portada.....84
3.4.2.2 Programación de la pantalla Menú. 85
3.4.2.3 Programación de la pantalla Control A y Control B.....86
3.4.2.4 Programación de la pantalla Control Automático. 88
3.4.2.5 Programación de la pantalla Ayuda. 89
3.4.2.6 Programación de la pantalla “Acerca de”.....90
3.5 CAMPO INDUSTRIAL A ESCALA EMPAREJAMIENTO Y PRUEBAS. .91
3.5.1 Implementación del prototipo.....92
CAPÍTULO IV....94
CONCLUSIONES Y RECOMENDACIONES....94
4.1 CONCLUSIONES 94
4.2 RECOMENDACIONES....95
GLOSARIO DE TERMINOS 97
Referencias bibliográficas....99
ANEXOS....100

ÍNDICE DE TABLAS

Tabla 1: Propiedades de la pantalla..... 69
Tabla 2: Propiedades del encabezado de la aplicación..... 71
Tabla 3: Parámetros de botones del menú de la aplicación. 74
Tabla 4: Parámetros de los botones e imágenes del Control A y B..... 77
Tabla 5: Botones de control de los led A y B 79
Tabla 6: Parametrización de pantalla de control. 80
Tabla 7: Parámetros de la ventana Ayuda..... 81

ÍNDICE DE FIGURAS

Figura 1: Kit Robot Arm 19
Figura 2: Interface de programación Arduino..... 20
Figura 3: Tarjeta Arduino UNO 22
Figura 4: Tarjeta Arduino Leonardo. 23
Figura 5: Tarjeta Arduino Yún. 24
Figura 6: Tarjeta Arduino 2560 25
Figura 7: Módulo HC06..... 30
Figura 8: Conexión entre Arduino y Bluetooth 31
Figura 9: Ícono Android 32
Figura 10: Versiones Android..... 33
Figura 11: Estructura Android 34
Figura 12: Ícono de Basic4Android, plataforma de programación. 37
Figura 13: Plataforma Mono para Android. 38
Figura 14: Plataforma de programación App Inventor. 39
Figura 15: Logo MIT App Inventor 40
Figura 16: Pantalla designer MIT APP INVENTOR 41
Figura 17: Pantalla Blocks MIT APP INVENTOR 42
Figura 18: Lista de componentes MIT APP INVENTOR 42
Figura 19: Kit Robot Arm 48
Figura 20: Manual de ensamblaje e instrucciones 49
Figura 21: Módulo HC06..... 50
Figura 22: Ventana de programación Arduino 1.6.0 50
Figura 23: Compilación satisfactoria..... 52
Figura 24: Abrir nuevo proyecto MIT APP INVENTOR 2..... 67
Figura 25: Componentes a utilizarse en la app..... 68
Figura 26: Añadir pantalla a la aplicación 68
Figura 27: Propiedades TableArrangment..... 70
Figura 28: Encabezado de la aplicación. 71
Figura 29: Conexión de la aplicación con el emulador..... 72
Figura 30: Compilación directamente en el celular de la aplicación 73
Figura 31: Renombrar a los componentes de la aplicación 74
Figura 32: Pantalla del menú principal 75
Figura 33: Componentes insertados en la pantalla del control A y B..... 76
Figura 34: Controles Brazo A y Brazo B 77
Figura 35: Controles del led de los brazos A y B 79
Figura 36: Componentes de la pantalla de control automático. 80
Figura 37: Ventana Ayuda de la aplicación..... 81
Figura 38: Pantalla Acerca De. 83
Figura 39: Ventana Blocks MIT APP INVENTOR2 84
Figura 40: Funciones de los componentes. 84
Figura 41: Bloque de función para los botones..... 85

Figura 42: Programación pantalla Portada. 85
Figura 43: Programación pantalla Menú. 86
Figura 44: Bloques para generar la fecha y hora actual. 86
Figura 45: Programación pantallas control inicializando la comunicación. .. 87
Figura 46: Programación de tramas de control. 87
Figura 47: Programación brazo B. 88
Figura 48: Programación Control Automático. 89
Figura 49: Regresar al menú principal. 89
Figura 50: Pantallas del menú Ayuda. 90
Figura 51: Programación de la pantalla Ayuda. 90
Figura 52: Programación de la pantalla Acerca de. 91
Figura 53: Esquema de conexión del motor de la banda transportadora..... 92

RESUMEN

El presente trabajo investigativo contempla la realización de un lazo de control bluetooth programado en Android y Arduino, el cual gobierna lógicamente un autómata representado por brazos robóticos por medio de un smartphone, los mismos que al simular un campo industrial a escala son capaces de transportar elementos e identificarlos físicamente para después clasificarlos. El control se realiza con la ayuda de una tarjeta Arduino Mega 2560 y con una interfaz hombre máquina representada en una aplicación desarrollada en la plataforma de software libre de Android e instalada en un Smartphone. En la programación Android se realiza una secuencia de envío de tramas de texto por comunicación bluetooth mientras que en la programación Arduino, se reconoce la trama del código recibido y se le asigna una secuencia a cada código de comunicación.

PALABRAS CLAVE: ANDROID, ARDUINO, APP INVENTOR, BLUETOOTH, BRAZOS ROBÓTICOS, SMARTPHONE.

ABSTRACT

This research work aims to develop a bond of bluetooth control scheduled Android and Arduino, which logically governs an automaton represented by robotic arms through a smartphone, thereof, to simulate an industrial scale field are capable transporting and identify physically elements and then classify them. The control is done with the help of an Arduino Mega 2560 board and a man machine interface displayed in an application developed in free software platform Android and installed in a Smartphone. In the Android programming is performed a sequence of sending text frames by bluetooth communication, while in the Arduino programming the received plot code is recognized and assigned a sequence to each communication code.

KEY WORDS: ANDROID, ARDUINO, APP INVENTOR, BLUETOOTH, ROBOTIC ARMS, SMARTPHONE.

CAPÍTULO I

1.1. ANTECEDENTES

Actualmente se vive una era de automatización industrial y general, en la cual el ser humano cada vez tiene menos actividad directa en procesos industriales tales como adquisición, manipulación y control de variables de proceso.

En los laboratorios de Electrónica e Instrumentación Virtual existen proyectos que incluyen el control de sensores y actuadores de diferentes aplicaciones controladas con software de adquisición de señales, es el caso del Robot Móvil Lego que adquiere señales y las proyecta en LabVIEW, este proyecto utiliza los sensores de Lego Mindstorm para realizar la adquisición de variables como temperatura, altitud y distancia para posteriormente monitorearlos en un sistema diseñado en LabView estableciendo una comunicación bluetooth entre el módulo interno del robto y un receptor incorporado en la PC. Otro ejemplo es la estación de monitoreo de sensores controlados inalámbricamente, en el cual vía bluetooth con la ayuda del módulo XBee se envía las señales de variables físicas adquiridas a la central de monitoreo diseñada en Labview, estos son los proyectos iniciales que han permitido a los estudiantes de la carrera de Electrónica mención Instrumentación y Aviónica incursionar en la robótica y en el monitoreo de variables, sensores o actuadores de manera inalámbrica.

1.2. PLANTEAMIENTO DEL PROBLEMA

Si bien es cierto se ha hablado de proyectos existentes en los laboratorios, pero aun no son una fuente de aprendizaje lo suficientemente amplios como para satisfacer las necesidades de innovación y emprendimiento tecnológico de un estudiante, aún más tomando en cuenta que ciertos softwares que permiten el control de variables necesitan

obligatoriamente el pago de una licencia para su utilización y correcta programación. Además de que para ciertas aplicabilidades se toma en cuenta un campo de interacción propiamente de laboratorio entre el estudiante y los equipos a utilizarse. La carrera de Electrónica mención Instrumentación y Aviónica no cuenta con un proyecto que permita inmiscuir en materias como Microcontroladores, Control Industrial, Automatización y Control utilizando última tecnología con la utilización de smartphones que serán los dispositivos que a futuro controlarán todo en nuestro ambiente y con hardware de costo considerablemente accesible.

Por esta razón, en los laboratorios es necesaria la implementación de equipos totalmente portátiles, de fácil acceso, y sobre todo que estén desarrollados bajo plataformas de software libre, en las cuales se permita al usuario ser totalmente independiente al momento de realizar proyectos y aplicarlos a sus necesidades y sin limitaciones de estructura física o electrónica, incentivando a estudiantes a profundizarse en el desarrollo de su propios proyectos electrónicos utilizando plataformas electrónicas de código abierto.

1.3. JUSTIFICACIÓN E IMPORTANCIA

El presente proyecto implementará un prototipo de proceso industrial que incluirán brazos robóticos, un módulo bluetooth, y una interfaz Android-Arduino compuesta por una tarjeta Arduino Mega 2560 controlada por una aplicación móvil diseñada bajo la plataforma de Andoid que servirá como el punto de partida para la actualización de plataformas electrónicas y equipos tecnológicos, permitiendo de esta manera renovar el conocimiento en estudiantes y docentes de la carrera de Electrónica mención Instrumentación y Aviónica. Es por esta misma razón que se ha optado por realizar el prototipo utilizando como plataforma de control a Arduino debido a su bajo costo y a su gama alta de aplicaciones prácticas y sencillas.

La finalidad de dicha implementación es sacar el mayor provecho a medios que rodean la vida normal de un estudiante, en este caso, un teléfono inteligente (Smartphone), el cual explotará los recursos cedidos por Google de software de programación libre para la creación de aplicaciones bajo la plataforma Android; que sean capaces de controlar motores, sensores, transductores, de manera totalmente fácil, rápida y gratuita. De esta manera se obviara en parte el uso de programas especializados en tareas similares, fomentando el desarrollo de software bajo necesidades personales y no bajo estructuras establecidas previamente en los programas de adquisición o control, además se incentivará al estudio de la robótica como parte del perfil estudiantil para futuros tecnólogos.

1.4. OBJETIVOS:

1.4.1. General:

- Implementar un prototipo de control para un brazo robótico en base a una interfaz Android-Arduino para el laboratorio de Instrumentación Virtual de la Unidad de Gestión de Tecnologías-Universidad de las Fuerzas Armadas-ESPE.

1.4.2. Específicos:

- Realizar un análisis de la tecnología Arduino y software de configuración que pueden ser utilizados en la implementación de robots.
- Establecer una comunicación bluetooth con la ayuda de la tarjeta Arduino MEGA y el módulo bluetooth HC-06 entre el teléfono móvil y los actuadores de los robots.
- Analizar los principales comandos de programación de App Inventor para diseñar una aplicación que al ejecutarla desde

un celular Android sea capaz de controlar a los brazos robóticos.

- Desarrollar un prototipo de proceso industrial que permita controlar un brazo robótico controlado desde una aplicación instalada en un smartphone.

1.5. ALCANCE

El proyecto de grado está compuesto por una tarjeta de control Arduino Mega 2560, dos brazos robóticos que serán los actuadores a controlar, una aplicación diseñada para el sistema operativo Android que hará las veces de una interfaz Hombre-Máquina, un módulo bluetooth con el cual se realizará la comunicación entre el módulo de control y los actuadores y una banda transportadora que simulará un proceso industrial básico y muy común en la industria.

CAPÍTULO II

MARCO TEÓRICO

2.1. KITS DE BRAZOS ROBÓTICOS.

Figura 1: Kit Robot Arm
Fuente; (Zapata, 2015)

Existen dos tipos de brazos robóticos dentro de esta rama de kit de robótica pertenecientes a la cadena OWI. Su principal diferencia es el modo de control utilizado en cada uno, es decir, a uno de los brazos es posible controlarlo mediante el mando físico que incluye el kit, el otro tipo de brazo tiene un control USB en el cual todos sus ejes son manipulados mediante la PC.

El kit USB incluye una tarjeta de comunicación serial la cuál le sirve para reconocimiento en la PC y para posterior control. Ambos cuentan con su manual de ensamblaje y operaciones. **Fuente:** (Zapata, 2015)

2.2 GENERALIDADES DE ARDUINO

2.2.1 ¿Qué es 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. **Fuente:** (Arduino, 2015)

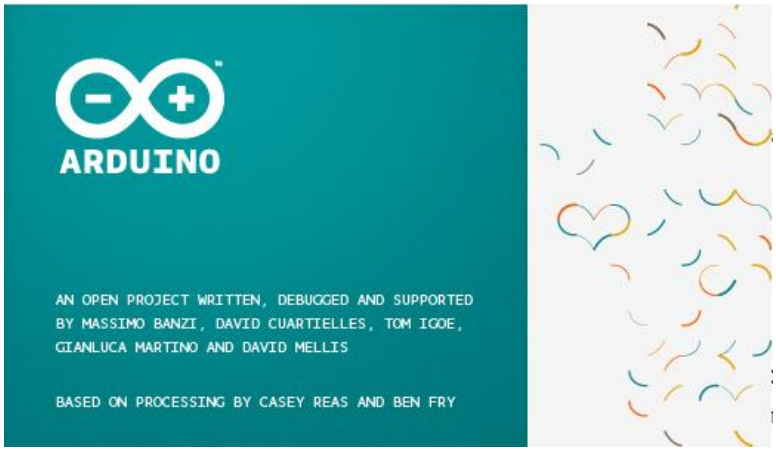


Figura 2: Interface de programación Arduino
Fuente: (Arduino, 2015)

2.2.2 Hardware Arduino

El hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida. Los microcontroladores más usados son el Atmega168, Atmega328, Atmega1280, ATmega8 por su sencillez y bajo coste que permiten el desarrollo de múltiples diseños. Por otro lado el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque que es ejecutado en la placa.

Desde octubre de 2012, Arduino se usa también con microcontroladoras CortexM3 de ARM de 32 bits, que coexistirán con las más limitadas, pero también económicas AVR de 8 bits. ARM y AVR no son plataformas compatibles a nivel binario, pero se pueden programar con el mismo IDE de Arduino y hacerse programas que compilen sin cambios en las dos plataformas. Eso sí, las microcontroladoras CortexM3 usan

3,3V, a diferencia de la mayoría de las placas con AVR que generalmente usan 5V. Sin embargo ya anteriormente se lanzaron placas Arduino con Atmel AVR a 3,3V como la Arduino Fio y existen compatibles de Arduino Nano y Pro como Meduino en que se puede conmutar el voltaje.

Arduino se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software tal como Adobe Flash, Processing, Max/MSP, Pure Data. Las placas se pueden montar a mano o adquirirse. El entorno de desarrollo integradolibre se puede descargar gratuitamente.

Arduino puede tomar información del entorno a través de sus entradas analógicas y digitales, puede controlar luces, motores y otros actuadores. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un computador. **Fuente:** (Wikipedia, 2014)

2.2.2.1 Tipos de tarjetas Arduino.

2.2.2.1.1 Arduino Uno.

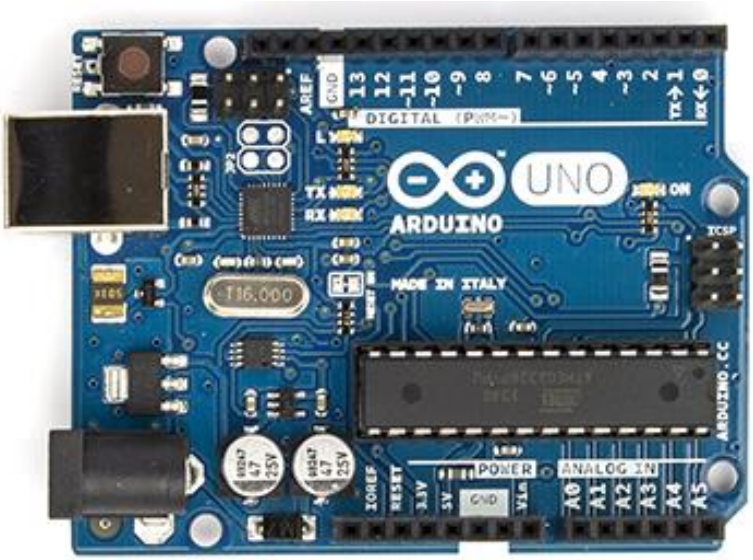


Figura 3: Tarjeta Arduino UNO
Fuente: (Arduino, 2015)

El Arduino Uno es una placa electronica basada en el ATmega328. Cuenta con 14 pines digitales de entrada/salida (de los cuales 6 pueden utilizarse para salidas PWM), 6 entradas analógicas, un resonador cerámico de 16 MHz, una puerto de conexión USB, un conector de alimentación, una cabecera ICSP, y un botón de reset. Contiene todo lo necesario para apoyar el microcontrolador; basta con conectarlo a un ordenador con un cable USB o a una batería de CC para inicializarlo.

La tarjeta Arduino Uno se diferencia de todas las placas anteriores en que no utiliza el chip controlador de USB a serial FTDI. En lugar de ello, cuenta con la Atmega16U2 (Atmega8U2 hasta la versión R2) programado como convertidor USB a serie.

2.1.2.1.2 Arduino Leonardo.

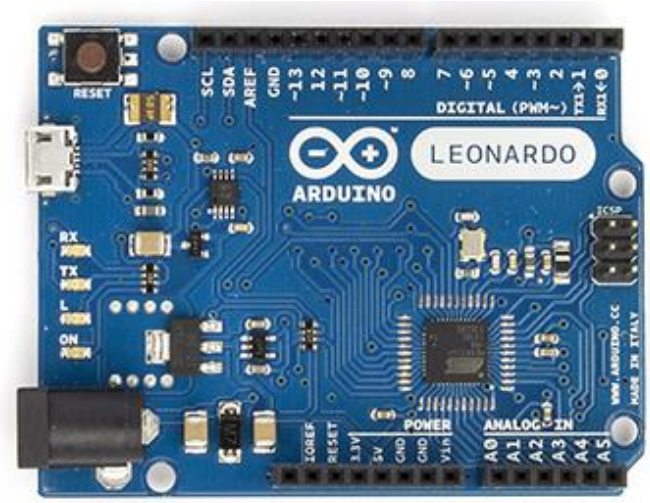


Figura 4: Tarjeta Arduino Leonardo.
Fuente: (Arduino, 2015)

La tarjeta Arduino Leonardo es una placa electrónica basada en el ATmega32u4. Cuenta con 20 pines digitales de entrada / salida (de los cuales 7 se pueden utilizar como salidas PWM y 12 entradas como analógicos), un oscilador de 16MHz, una conexión micro USB, un conector de alimentación, una cabecera ICSP, y un botón de reinicio. Contiene todo lo necesario para apoyar el microcontrolador; basta con conectarlo a un ordenador con un cable USB.

La placa Leonardo difiere de todas las placas anteriores en que el ATmega32u4 ha incorporado en la comunicación USB, eliminando la necesidad de un procesador secundario. Esto permite que el Leonardo aparezca a un ordenador conectado como un ratón y el teclado, además de un virtual (CDC) de puerto serie / COM.

2.1.2.1.3 Arduino Yún.

Figura 5: Tarjeta Arduino Yún.

Fuente: (Arduino, 2015)

El Arduino Yun es una placa electrónica basada en el ATmega32u4 y el Atheros AR9331. El procesador Atheros es compatible con una distribución Linux basada en OpenWrt llamado OpenWrt-Yun. La junta ha incorporado Ethernet y soporte WiFi, un puerto USB-A, ranura para tarjeta micro-SD, 20 entradas digitales / pines de salida (de los cuales 7 se pueden utilizar como salidas PWM y 12 como entradas analógicas), un cristal de 16 MHz oscilador, una conexión micro USB, un jefe de ICSP, y un 3 botones de reposición.

2.1.2.1.4 Arduino Mega.

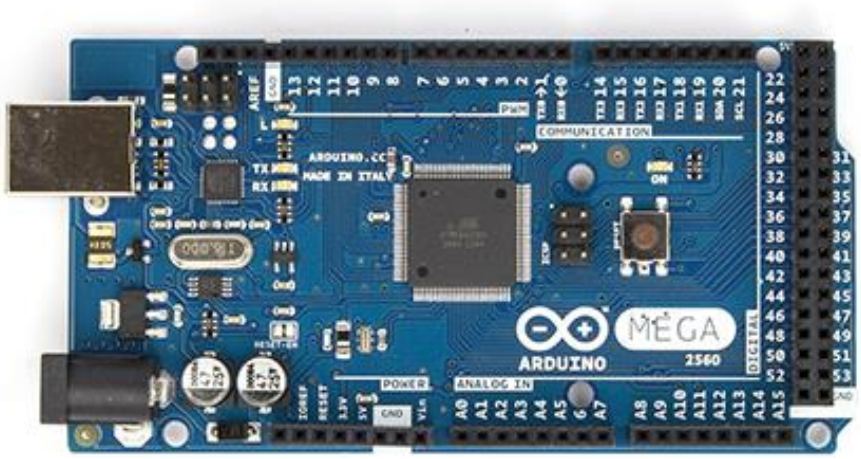


Figura 6: Tarjeta Arduino 2560
Fuente: (Arduino, 2015)

Arduino Mega 2560 es una placa electrónica basada en el Atmega2560. Cuenta con una interfaz de host USB para conectar con los teléfonos basados en Android, basado en el MAX3421E IC. Cuenta con 54 pines digitales de entrada / salida (de los cuales 15 se pueden utilizar como salidas PWM), 16 entradas analógicas, 4 UARTs (puertas seriales), un oscilador de 16MHz, una conexión USB, una cabecera ICSP, y un botón de reset. Cuenta con una ATmega8U2 programado como convertidor USB a serie.

2.2.3 Lenguaje de programación Arduino.

La plataforma Arduino se programa mediante el uso de un lenguaje propio basado en el lenguaje de programación de alto nivel Processing. Sin embargo, es posible utilizar otros lenguajes de programación y aplicaciones populares en Arduino,21 debido a que Arduino usa la transmisión serial de datos soportada por la mayoría de los lenguajes mencionados. Para los que no soportan el formato serie de forma nativa, es posible utilizar

software intermediario que traduzca los mensajes enviados por ambas partes para permitir una comunicación fluida. Algunos ejemplos son:

- 3DVIA Virtools: aplicaciones interactivas y de tiempo real.
- Adobe Director
- BlitzMax (con acceso restringido)
- C
- C++ (mediante libSerial o en Windows)
- C#
- Cocoa/Objective-C (para Mac OS X)
- Flash (mediante ActionScript)
- Gambas
- Isadora (Interactividad audiovisual en tiempo real)
- Instant Reality (X3D)
- Java
- Liberlab (software de medición y experimentación)
- Mathematica
- Matlab
- MaxMSP: Entorno gráfico de programación para aplicaciones musicales, de audio y multimedia
- Minibloq: Entorno gráfico de programación, corre también en las computadoras OLPC
- Perl
- Php
- Physical Etoys: Entorno gráfico de programación usado para proyectos de robótica educativa
- Processing
- Pure Data
- Python
- Ruby
- VBScript

- Visual Basic .NET
- VVVV: Síntesis de vídeo en tiempo real

2.2.3.1 Sintaxis Básica.

- Delimitadores:;, {}
- Comentarios: //, /* */
- Cabeceras: #define, #include
- Operadores aritméticos: +, -, *, /, %
- Asignación: =
- Operadores de comparación: ==, !=, <, >, <=, >=
- Operadores Booleanos: &&, ||, !
- Operadores de acceso a punteros: *, &
- Operadores de bits: &, |, ^, ~, <<, >>
- Operadores compuestos:
- Incremento y decremento de variables: ++, --
- Asignación y operación: +=, -=, *=, /=, &=, |=
- Estructuras de control
- Condicionales: if, if...else, switch case
- Bucles: for, while, do... while
- Bifurcaciones y saltos: break, continue, return, goto

2.2.3.2 Variables.

En cuanto al tratamiento de las variables también comparte un gran parecido con el lenguaje C.

2.2.3.3 Constantes.

- HIGH/LOW: representan los niveles alto y bajo de las señales de entrada y salida. Los niveles altos son aquellos de 3 voltios o más.
- INPUT/OUTPUT: entrada o salida.

- false (falso): Señal que representa al cero lógico. A diferencia de las señales HIGH/LOW, su nombre se escribe en letra minúscula.
- true (verdadero): Señal cuya definición es más amplia que la de false. Cualquier número entero diferente de cero es "verdadero", según el álgebra de Boole, como en el caso de -200, -1 o 1. Si es cero, es "falso".

2.2.3.4 Tipos de datos.

- void, boolean, char, unsigned char, byte, int, unsigned int, word, long, unsigned long, float, double, string, array.

2.2.3.5 Conversiones entre tipos.

Estas funciones reciben como argumento una variable de cualquier tipo y devuelven una variable convertida en el tipo deseado.

- char(), byte(), int(), word(), long(), float()

2.2.3.6 Entradas/salidas digitales

- pinMode(pin, modo)
- digitalWrite(pin, valor)
- int digitalRead(pin)

2.2.3.7 Entradas/salidas analógicas

- analogReference(tipo)
- int analogRead(pin)
- analogWrite(pin, valor)

2.2.3.8 Tiempo

- unsigned long millis()
- unsigned long micros()
- delay(ms)
- delayMicroseconds(microsegundos)

2.2.3.9 Comunicación por puerto serie

Las funciones de manejo del puerto serie deben ir precedidas de la palabra "Serial" aunque no necesitan ninguna declaración en la cabecera del programa. Por esto se consideran funciones base del lenguaje.m

Estas son las funciones para transmisión serial:

begin(), available(), read(), flush(), print(), println(), write()

2.2.3.10 Manipulación de puertos

Los registros de puertos permiten la manipulación a más bajo nivel y de forma más rápida de los contactos de entrada/salida del microcontrolador de las placas Arduino. Los contactos eléctricos de las placas Arduino están repartidos entre los registros B(0-7), C (analógicos) y D(8-13). Mediante estas variables ser observado y modificado su estado:

- DDR[B/C/D]: Data Direction Register (o dirección del registro de datos) del puerto B, C ó D. Es una variable de Lectura/Escritura que sirve para especificar cuáles contactos serán usados como entrada y salida.
- PORT[B/C/D]: Data Register (o registro de datos) del puerto B, C ó D. Es una variable de Lectura/Escritura.
- PIN[B/C/D]: Input Pins Register (o registro de pines de entrada) del puerto B, C ó D. Variable de sólo lectura.

2.3 Módulo Bluetooth HC05 como dispositivo cliente en la comunicación.

El futuro es inalámbrico y la tecnología Bluetooth es una de las favoritas en el mundo de los aficionados a la electrónica, donde el enlace de datos “sin vínculo físico” debe ser robusto, confiable y seguro. Luego de haber ensayado el módulo Bluetooth RN41, ahora le llega el turno a uno de los modelos más económicos del mercado. Distribuido en todo el mundo por Wavesen, el módulo Bluetooth HC-06 es un dispositivo muy fácil de obtener, económico y sencillo de utilizar. En este artículo veremos su implementación y uso dentro de una sencilla aplicación para Android. En esta primera entrega aprenderemos a conectar y poner en funcionamiento este popular y eficaz módulo Bluetooth HC-06. **Fuente:** (NT, 2011)

2.3.1 Generalidades

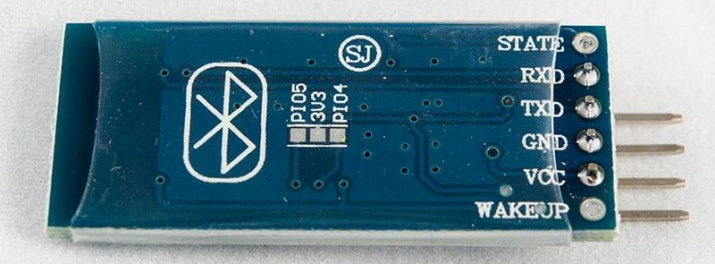


Figura 7: Módulo HC06
Fuente: (Zapata, 2015)

Una de las ventajas principales ventajas del módulo HC-06, además de su pequeño tamaño y sus buenas características de transmisión y recepción que le brindan un alcance muy amplio (por tratarse de un sistema local Bluetooth), es el bajo consumo de corriente que posee tanto en funcionamiento, como en modo de espera, es decir, alimentado con energía, pero sin conexión o enlace a otro dispositivo, por ejemplo, un móvil con SO Android.

Otra característica interesante de este módulo es que una vez que ha realizado un enlace con otro dispositivo es capaz de recordarlo en su memoria y no solicita validación alguna ("*1234*" por defecto), pero si se activa el pin 26 (*KEY*) hacia la tensión de alimentación, esta información se elimina y el módulo HC-06 solicitará nuevamente la validación del enlace. Otro detalle particular es que su tensión de alimentación de 3,3Volts y su bajo consumo (8mA en transmisión/recepción activa) lo transforman en un dispositivo ideal para trabajar con microcontroladores de la misma tensión de alimentación, logrando de este modo equipos portátiles que pueden ser alimentados durante muchas horas por baterías recargables o alcalinas AA, demostrando características excepcionales en aplicaciones médicas, o para actividades recreativas donde la fuente energética debe ser liviana y portátil. **Fuente:** (NT, 2011)

2.3.2 Conexión básica entre el módulo bluetooth y la tarjeta Arduino.

Las conexiones para realizar con arduino son bastante sencillas. Solamente requerimos colocar como mínimo la alimentación y conectar los pines de transmisión y recepción serial (TX y RX). Hay que recordar que en este caso los pines se debe conectar cruzados TX Bluetooth -> RX de Arduino y RX Bluetooth -> TX de Arduino. La siguiente imagen muestra las conexiones básicas para que funcione el módulo

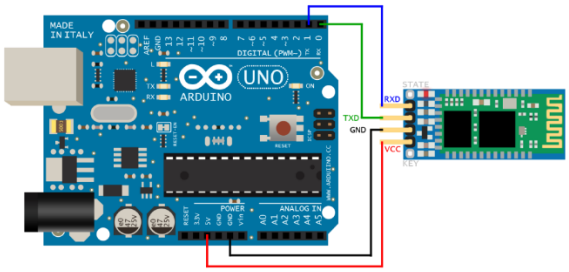


Figura 8: Conexión entre Arduino y Bluetooth
Fuente: (NT, 2011)

El código para la comunicación a través del bluetooth es idéntico al que utilizaríamos para comunicarnos con la PC vía USB. **Fuente:** (Rubén, s.f.)

2.4 Introducción a Android

Figura 9: Ícono Android
Fuente: (Android, s.f.)

La evolución de la tecnología va a paso veloz, Android es de las tecnologías que esta alcanzado a todos por el simple motivo de que se encuentra en los móviles. Android es un sistema operativo basado en Linux. La diferencia principal es que tiene módulos que responden a la pantalla táctil, eventos nativos del móvil. Se desarrolló por una compañía llamada Android, Inc. En 2005 Google adquiere la empresa para seguir trabajando en el mismo proyecto que después conociera la luz como un S.O. para móviles denominado finalmente como Android CAPÍTULO III.

Android tiene una característica peculiar: las versiones tienen nombre de postres en inglés y cada versión que cambia, continúa de forma incremental en el alfabeto, es decir que si el primer nombre inicio con A, el siguiente con B, el siguiente C y así sucesivamente; ya veremos que sucede cuando lleguen a la Z. **Fuente:** (Lujan, 2014)

2.3.1 Versiones de Android

Figura 10: Versiones Android
Fuente: (Press, s.f.)

- Versión 1.0 Apple Pie - Salió en septiembre del 2008.
- Versión 1.1 Banana Bread - Salió en febrero 2009.
- Versión 1.5 Cup Cake - Salió en abril 2009
- Versión 1.6 Donut - Salió en septiembre 2009
- Versión 2.0 Eclair - Salio en octubre 2009
- Versión 2.2 Froyo - Salió en mayo 2010
- Versión 2.3 Gingerbread - Salió en diciembre 2010
- Versión 3.0 Honeycomb - Salió en febrero 2011
- Versión 4 .0 Ice Cream Sandwich - Salió en octubre 2011
- Versión 4.1 Jelly Bean - Salió en julio 2012
- Versión 4.4 KitKat - Salió en octubre 2013 **Fuente:** (Lujan, 2014)

2.4.1 Estructura de Android

Ya mencionamos que Android está basado en Linux. Para ser más específicos, hablamos del kernel. Android utiliza como base el kernel de

Linux. Esto no significa que por estar basado en el algo que se desarrolló en Linux funcione para Android, por ejemplo Android no tiene soporte glibc.

Ahora vamos a darle un vistazo a la estructura:

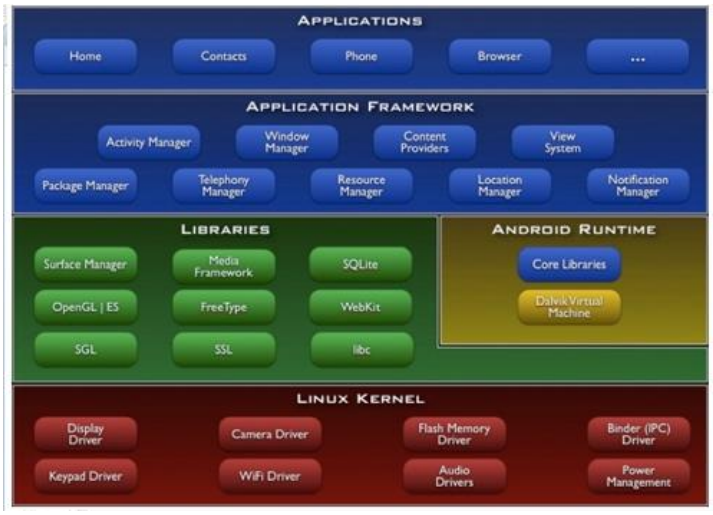


Figura 11: Estructura Android
Fuente: (Lujan, 2014)

Tenemos esta estructura:

- Capa Roja, Kernel.
- Capa Verde, Librerias.
- Capa Amarilla, Android runtime.
- Capa Azul, application Framework
- Capa Azul última, Application.

Capa del Kernel (Roja)

Aquí tenemos el corazón de Android: el manejo de memoria, procesos, drivers, etc. Aquí es donde se da la comunicación con el hardware. Esto nos sirve para no estar peleando con los fabricantes de cada móvil, nos ayuda a solo usar la “cámara” y no tener que saber cómo funciona la cámara del fabricante X, fabricante Y; solamente hacemos lo

que nos interesa, que sería usar la cámara y listo. Además de eso, aquí se administran los recursos del celular, memoria, energía.

Capa Librerías (Verde)

Esta capa tiene las librerías nativas de Android, están escritas en C o C++ y tienen tareas específicas.

- Surface manager: Gestión del acceso a la pantalla.
- Media Framework: Reproducción de imágenes, audio y vídeo.
- SQLite: BD
- Webkit, Navegador.
- SGL: Gráficos 2D.
- OpenGL: Gráficos 3D.
- Freetype: Renderizar vectores o imágenes.

Android Runtime (Capa Amarilla)

Esta capa amarilla no se considera al 100% una capa. Lo que es muy importante comentar es que aquí se encuentra Dalvik, la máquina virtual de Android, que no es lo misma que la Java Virtual Machine. Esto quiere decir que cuando compilamos en Java lo que se genera solamente va a funcionar en la JVM, porque Dalvik es una máquina virtual, pero de Android, así que el ByteCode que genera Java es inservible para Dalvik.

Algunas de las características de Dalvik son:

- Trabaja en entorno con restricción de memoria y procesador.
- Ejecuta el formato .dex.
- Convierte .class en .dx.

Application Framework (Capa azul)

Esta capa es la es más visible para el desarrollador, ya que la mayoría de los componentes que forman parte del desarrollo los vamos a encontrar aquí.

- Activity Manager- Administra las actividades de nuestra aplicación y el ciclo de vida.
- Windows Manager- Administra lo que se muestra en la pantalla.
- Content Provider-. Administra dependiendo de cómo le indiquemos algunos contenidos, puede ser información que necesitamos la encapsule para enviar o compartir.
- View- Las vistas de elementos que son parte de la interfaz gráfica, como los mapas, cuadros de texto, etc.
- Notification Manager- Administra las notificaciones.
- Package Manger- Administra los paquetes y nos permite el uso de archivos en otros paquetes.
- Telephony Manager- Administra lo que tiene que ver con la telefonía, llamadas, mensajes.
- Resource Manager- Administra recursos de la aplicación, como los xml, imágenes, sonido.
- Location Manager- Gestiona la posición geográfica.
- Sensor Manager- Gestiona los sensores que tenga el dispositivo.
- Cámara- Administra la cámara.
- Multimedia- Administra lo referente a audio, video y fotos.

Aplications (Capa Azul última)

Aquí tenemos las aplicaciones que vienen en el dispositivo, por ejemplo: el gestor de correos, los mensajes, el market, etc. **Fuente:** (Lujan, 2014)

2.4.2 Plataformas de desarrollo Android.

2.4.2.1 Basic4Android

Basic4Android es una plataforma de programación para aplicaciones Android cuyo lenguaje base de programación es VisualBasic, el eterno rival de Java, ese lenguaje que está orientado a aquellas personas que empezamos en el mundo de la programación de una manera más gráfica y no tan abstracta. No es el mismo lenguaje de Microsoft, pero su sintaxis es la misma, lo cual tiene sus mismas ventajas como algunos de sus inconvenientes.

Ésta plataforma no es gratuita, encontramos desde la versión mínima pago que es de 49 dólares hasta la versión máxima de 249 dólares americanos. **Fuente:** (León, 2012)

Figura 12: Ícono de Basic4Android, plataforma de programación.

Fuente: (León, 2012)

2.4.2.2 Mono para Android.

Otro de los lenguajes que Microsoft desarrollo para hacer aplicaciones fue C# y .NET, las cuales son muy usados en diferentes ambientes, por lo que no podría faltar que estos lenguajes tan comunes y opuestos a Java llegaran a Android.

Si tu ambiente de programación es Visual Studio lo único que debes instalar es el SDK de Android, la versión para Android de Mono y listo, se puede seguir desarrollando sin ningún inconveniente; además según Xamarin (la empresa creadora de Mono), se trabaja con un lenguaje nativo para Android ya que no tiene un intérprete con lo tendría Basic4Android, y su aprendizaje es relativamente sencillo en un tiempo prudente si lo que buscas es hacer esa aplicación tienes ya en mente y no tienes tiempo de aprender un nuevo lenguaje. Por otro lado está el tema del costo, que para la versión más económica de Mono es de 399 dólares americanos.

Figura 13: Plataforma Mono para Android.

Fuente: (León, 2012)

2.4.2.3 App Inventor

¿No se quiere Java, ni C#, ni C, ni .NET, ni VisualBasic, en resumidas cuentas, ningún programa de desarrollo tradicional? Perfecto, con App

Inventor es posible programar fácilmente sin necesidad de criterios técnicos.

Está basada en un lenguaje de desarrollo gráfico en donde no escribes ni una sola línea de código, tan solo arrastras bloques identificados con la acción que necesitas hacer y listo.

Esta plataforma de desarrollo fue impulsada por Google hace un tiempo con el fin de que más personas se unieran a la familia de Android; esta genial herramienta usa tu navegador como centro principal de trabajo, y almacena todo esto en servidores que están disponibles cada vez que entres a internet.

Figura 14: Plataforma de programación App Inventor.

Fuente: (León, 2012)

2.4.3 App Inventor 2.

2.4.3.1 ¿Qué es MIT App Inventor 2?

App Inventor es un lenguaje de programación visual basado en bloques que permite crear aplicaciones para Android. Desarrollado por Google, ahora lo mantiene el Massachusetts Institute of Technology (MIT). App

Inventor te permite desarrollar aplicaciones para los teléfonos Android mediante un navegador web y, bien un teléfono conectado o el emulador. Los servidores de App Inventor almacenan el trabajo y ayudarán a realizar el seguimiento de los proyectos. **Fuente** (Corín, s.f.)

Figura 15: Logo MIT App Inventor
Fuente: (Technology, s.f.)

Partes del IDE → Designer

Es donde elegimos los componentes de nuestra aplicación (botones, pantallas, imágenes, sonidos…) Arrastramos los componentes a la parte central, que es como si fuera la pantalla de nuestro teléfono. A la derecha se nos muestra el árbol de componentes. Pinchando sobre los componentes, podemos modificar sus propiedades en la parte más a la derecha de la pantalla. También podemos añadir ficheros (audio, video….)

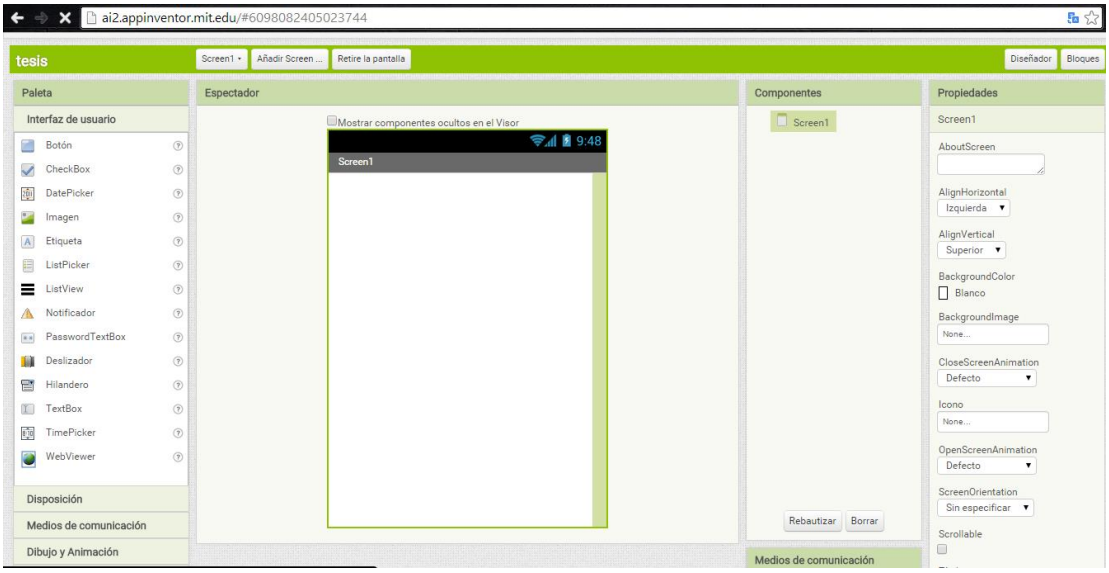


Figura 16: Pantalla designer MIT APP INVENTOR
Fuente: (Google, s.f.)

Partes del IDE → Blocks

Es donde hacemos que los componentes hagan cosas.

Diferentes tipos de bloques según lo que queramos hacer:

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

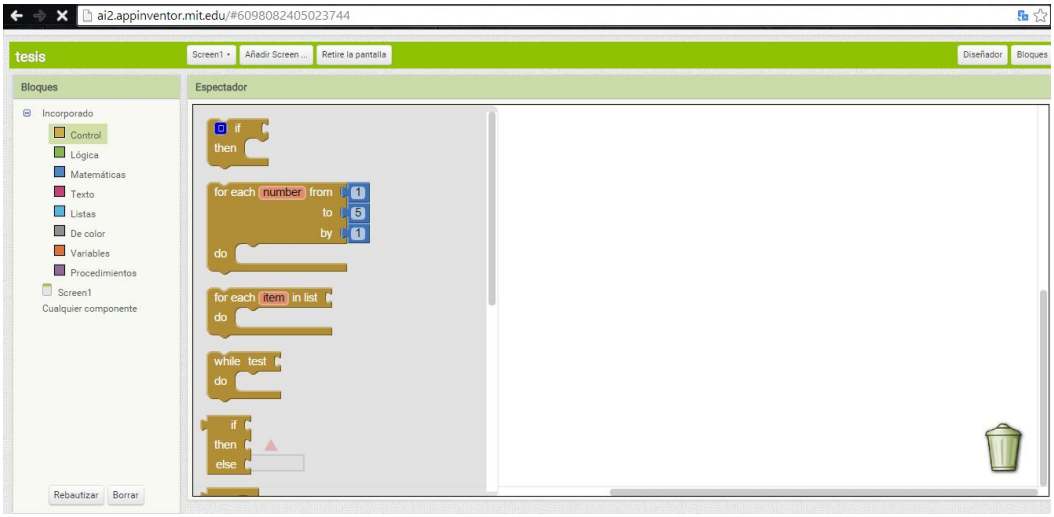


Figura 17: Pantalla Blocks MIT APP INVENTOR
Fuente: (Google, s.f.)

Además tenemos los bloques de los componentes. **Fuente:** (Martínez, 2014)

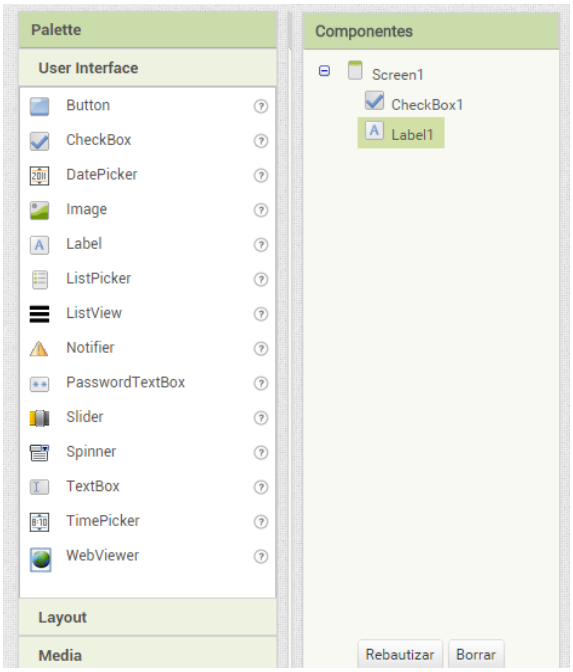


Figura 18: Lista de componentes MIT APP INVENTOR
Fuente: (Google, s.f.)

2.4.3.2. Componentes.

Cada componente puede tener métodos, eventos y propiedades. La mayoría de las propiedades se pueden cambiar por las aplicaciones - estas propiedades tienen bloques que puede utilizar para obtener y establecer los valores.

2.4.3.2.1 Button.

Los botones son componentes que los usuarios tocan para realizar alguna acción en su aplicación.

Botones detectan cuando los usuarios tocan ellos. Muchos aspectos de la apariencia de un botón se pueden cambiar.

Propiedades

- `BackgroundColor`
  - Color de fondo del botón.
- `Activado`
  - Si se activa, el usuario puede tocar el botón para causar la acción.
- `FontBold`
  - Si se establece, el texto del botón se muestra en negrita.
- `FontItalic`
  - Si se establece, el texto del botón aparece en cursiva.
- `Tamaño de Letra`
  - El tamaño en puntos para el texto del botón.
- `FontTypeface`
  - Familia de fuentes de texto del botón.
- `Altura`
  - Altura de botón (de tamaño y).

- `Ancho`
    - Ancho Button (de tamaño x).
- `Imagen`
    - Imagen para mostrar el botón.
- `Texto`
    - Texto a mostrar en el botón.

2.4.3.2.2 Image.

Un componente de imagen muestra una imagen. Puede especificar la imagen para mostrar y otros aspectos de la apariencia de la imagen en el Diseñador o en el Editor de bloques.

Propiedades

- `Imagen`
    - Imagen que muestra esta imagen.
- `Visible`
    - Si es verdad, se muestra una imagen.
- `Altura`
    - Altura de la imagen (tamaño y).
- `Ancho`
    - Imagen anchura (de tamaño x).

2.4.3.2.3 Label.

Las etiquetas son componentes que se utilizan para mostrar texto.

Un texto muestra la etiqueta que se especifica en el texto propiedad. Otras propiedades, todos los cuales se pueden establecer en el Diseñador o Editor de bloques, controlan el aspecto y la colocación del texto.

Propiedades

- `BackgroundColor`
  - Color de fondo de la etiqueta.
- `FontBold`
  - Si se establece, texto de la etiqueta se muestra en negrita.
- `FontItalic`
  - Si se establece, texto de la etiqueta aparece en cursiva.
- `Tamaño de Letra`
  - El tamaño en puntos para el texto de la etiqueta.
- `FontTypeface`
  - Familia de fuentes de texto de la etiqueta.
- `Altura`
  - Alto de etiqueta (de tamaño y).
- `Ancho`
  - Etiquetar ancho (x-size).
- `Texto`
  - Texto a mostrar en la etiqueta.
- `TextAlignment`
  - Izquierda, centro o derecha.
- `TextColor`
  - Color para el texto de la etiqueta.
- `Visible`
  - Si se establece, la etiqueta es visible.

2.4.3.2.4 Screen

La pantalla no aparece en la paleta al igual que otros componentes, pero si viene automáticamente con el proyecto. Cada proyecto tiene exactamente una pantalla, llamada Screen1. Este nombre no se puede cambiar.

Propiedades

- `BackgroundColor`
  - Color de fondo de la pantalla.
- `BackgroundImage`
  - Una imagen que forma el fondo de la pantalla.
- *`Altura`*
  - Altura de la pantalla (de tamaño y).
- `icono`
  - Una imagen que se utilizará como el icono de la aplicación instalada en el teléfono. Esto debería ser una imagen JPG o PNG; 48x48 es un buen tamaño. *Advertencia* : Especificación de las imágenes de otro tipo que PNG o JPG, por ejemplo .ico archivos, pueden impedir el App Inventor de la posibilidad de empaquetar la aplicación.
- `Scrollable`
  - Esto es establecido por una casilla de verificación en el diseñador. Cuando se activa, habrá una barra de desplazamiento vertical en la pantalla, y la altura de la demanda puede exceder la altura física del dispositivo. Cuando no se controla, la altura de aplicación está limitado a la altura del dispositivo.
- `Título`
  - Título para la pantalla (texto). Este aparecerá en la parte superior izquierda del teléfono cuando la aplicación se ejecuta. Una elección natural para el título es el título de la

aplicación, pero se podía hacer otra cosa, o incluso cambiar mientras la aplicación se está ejecutando.

- *Ancho*
  - Anchura de la pantalla (de tamaño x).

2.4.3.3 Bloques.

2.4.3.3.1 Texto.

Contiene una cadena de texto.

2.4.3.3.2 Igual a.

Comprueba si dos valores dados son iguales. Si es así, devuelve true; de lo contrario, devuelve false.

2.4.3.3.3 If.

Prueba una condición dada. Si la condición es verdadera, realiza las acciones en una determinada secuencia de bloques; de lo contrario, se ignoran los bloques.

2.4.3.3.4 Close Screen.

Cierra la pantalla

2.4.3.3.5 True, False

Son estados lógicos con propiedades booleanas. **Fuente:** (Corín, s.f.)

CAPÍTULO III

DESARROLLO DEL TEMA

3.1 ENSAMBLAJE DE LOS BRAZOS ROBÓTICOS

Figura 19: Kit Robot Arm
Fuente; (Zapata, 2015)

Para el ensamblaje de los brazos, se necesita la ayuda de un destornillador de punta estrella pequeño, un corta fríos o cortadora de alambre, y la guía de ensamblaje que se incluye en el kit Robot Arm, y que será adjuntado en el **Anexo A**.

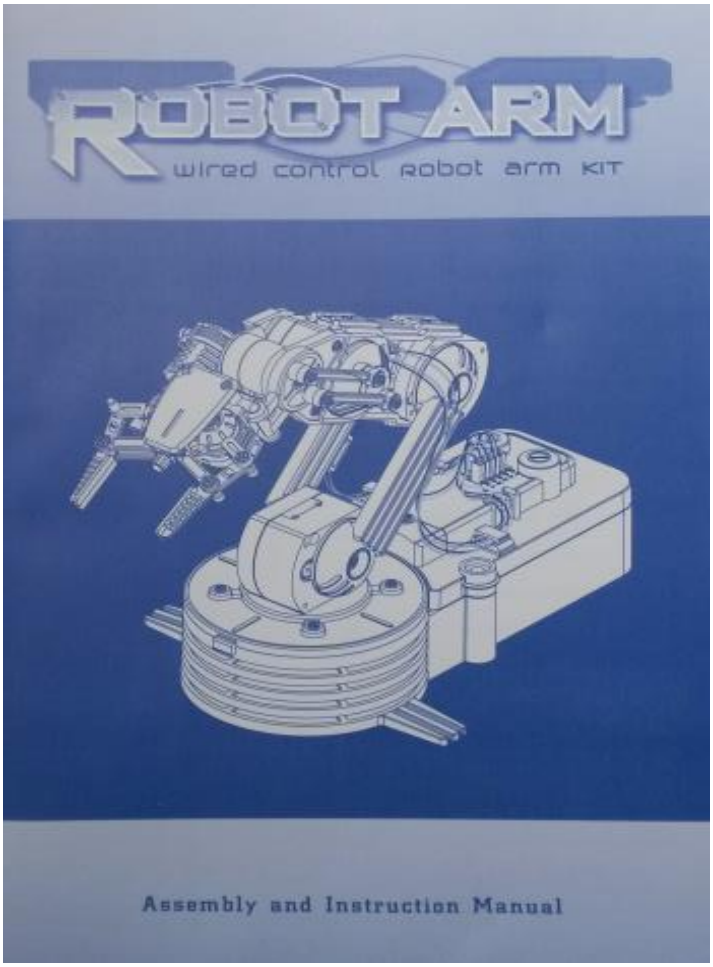


Figura 20: Manual de ensamblaje e instrucciones
Fuente: (Zapata, 2015)

El ensamblaje es prácticamente rápido, por lo cual no tomará más de 2 horas construirlo en su totalidad, desde el ensamblaje de los motoreductores hasta su funcionamiento propiamente dicho.

Cabe destacar que es necesario una alimentación de voltaje comprendida por cuatro baterías tipo D.

3.2 CONFIGURACIÓN DEL MÓDULO BLUETOOTH HC-06.

Antes de empezar a desarrollar el proyecto dentro de la plataforma de programación, se debe configurar el dispositivo bluetooth con el que se va a trabajar (datasheet en el **Anexo D**), para que el dispositivo sea reconocido por el bluetooth del teléfono móvil y de esta manera se conozca cuál será su nombre y su contraseña de conexión.

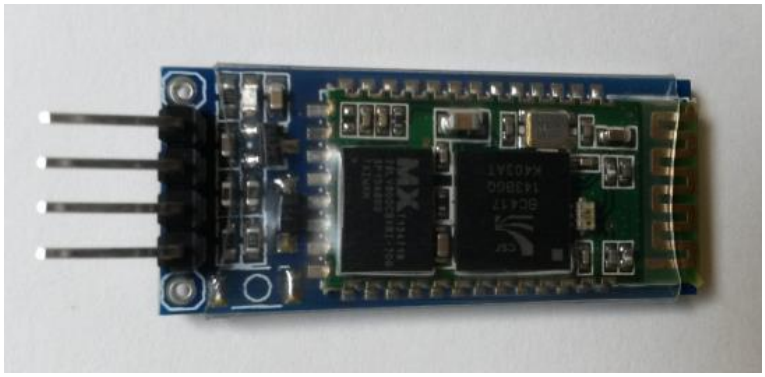


Figura 21: Módulo HC06
Fuente: (Zapata, 2015)

La configuración del módulo bluetooth se la realiza dentro de la ventana de programación Arduino 1.6.0 que se puede descargar directamente desde la página de Arduino: www.arduino.cc.

Figura 22: Ventana de programación Arduino 1.6.0
Fuente: (Arduino, 2015)

Una vez dentro de la interfaz de programación se debe configurar el nombre de reconocimiento del dispositivo, la contraseña de conexión y la velocidad a la cual se realizara la comunicación.

El código a utilizarse es el siguiente:

```
char NOMBRE[10]="TESIS_IZ";
char BPS = '4';
char PASS[10]="1234";
```

La primera línea de código especifica el nombre que se le dará al dispositivo, en este caso: TESIS_IZ, se le puede configurar para que aparezca el nombre que el usuario desee.

En la segunda línea de programa esta especificada la velocidad de la comunicación, en donde, 1=1200, 2=2400, 3=4800, 4=9600. Utilizaremos una velocidad de 9600bps.

La tercera línea ayuda a definir la contraseña con la cual el teléfono móvil se comunicara con los brazos robot.

3.2.1 Compilar y cargar la configuración al módulo HC 06

Una vez escrito el código, es necesario compilarlo para verificar si tiene errores y de esta manera evitar daños en los equipos.

Al lado superior izquierdo de la ventana se encuentra el ícono de un “check” que en su descripción se nombra como Verificar, para fines prácticos es igual a compilar.

Una vez compilado el código, es necesario conectar la tarjeta Arduino Mega 2560 al ordenador a través de un puerto USB previamente instalados

los drives de reconocimiento Arduino (en el caso de no poseerlos descargar desde la página de Arduino).

Hay que tomar en cuenta que en el programador debe reconocerse automáticamente el puerto de comunicación y el tipo de tarjeta que se utilizará:

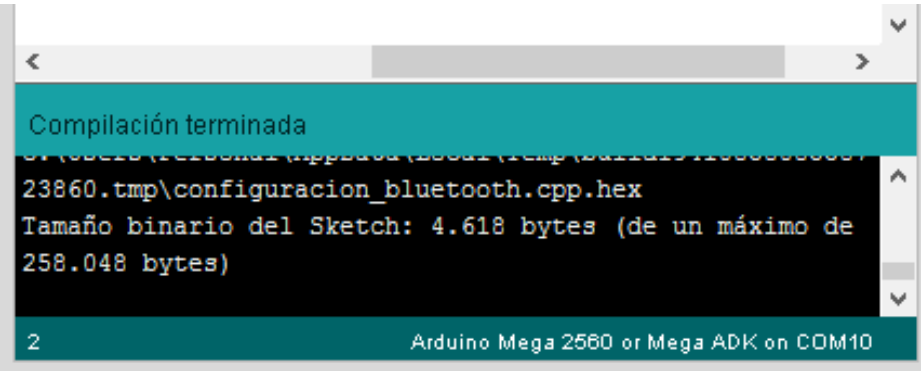


Figura 23: Compilación satisfactoria
Fuente: (Arduino, 2015)

Cuando la compilación haya finalizado se procederá a cargar el programa al módulo HC 06 con la ayuda de la tarjeta Arduino Mega 2560.

Para cargar el programa se da un click en el ícono cargar de la barra de herramientas. 

Una vez cargado el programa se puede ya visualizar en un teléfono móvil su conexión.

3.3 PROGRAMACIÓN DEL SOFTWARE DE CONTROL ARDUINO.

Una vez configurado el módulo bluetooth, es preciso establecer el tipo de control que se va a dar al robot, es decir si se enviarán tramas completas de mensajes o si solo enviando letras se realizará el reconocimiento de las instrucciones.

El lenguaje de programación que utiliza Arduino es muy similar al utilizado por el lenguaje C.

3.3.1 Declaración de variables.

Al inicializar las constantes de control, deben definirse con una nomenclatura fácil de entender y de recordar por el programador.

Es importante aclarar también que el programa está diseñado para funcionar con una tarjeta Arduino Mega, por lo tanto utilizaremos las entradas digitales partiendo desde la entrada 22.

```
#include <EEPROM.h>

const int HA = 22;  //HOMBRO
const int HB = 23;  //HOMBRO
const int BA = 24;  //ANTEBRAZO
const int BB = 25;  //ANTEBRAZO
const int AA = 26;  //BRAZO
const int AB = 27;  //BRAZO
const int MA = 28;  //MUÃ‘ECA ARRIBA
const int MB = 29;  //MUÃ‘ECA
const int PA = 30;  //PINZA
const int PC = 31;  //PINZA

//BRAZO 2
const int HA_B = 32;  //HOMBRO
const int HB_B = 33;  //HOMBRO
const int BA_B = 34;  //ANTEBRAZO
const int BB_B = 35;  //ANTEBRAZO
const int AA_B = 36;  //BRAZO
const int AB_B = 37;  //BRAZO
const int MA_B = 38;  //MUÃ‘ECA ARRIBA
const int MB_B = 39;  //MUÃ‘ECA
const int PA_B = 40;  //PINZA
const int PC_B = 41;  //PINZA
```

```
const int CONTROL = 42;

int SENSOR1 = 43;
int SENSOR2 = 44;
int SENSOR3 = 45;
int SENSOR4 = 49;

int I1;
int I2;
int I3;
int I4;

int LEDA = 46;
int LEDB = 47;

const int MOTORBANDA = 48; //ENCIENDE LA BANDA
TRANSPORTADORA

char dato;
```

3.3.2 Configuración de comunicación, lectura de trama y asignación de entradas y salidas digitales.

Para comenzar la comunicación entre el dispositivo Android y el módulo de control es necesario leer la trama de texto que se envíe en la comunicación y establecer la velocidad a la cual se realizara la misma:

```
String readString;
void setup(){
Serial.begin(9600);
```

También es necesario definir que pines de la tarjeta serán configuradas como entradas y cuales como salidas digitales:

```
  pinMode(HA, OUTPUT);
  pinMode(HB, OUTPUT);
  pinMode(BA, OUTPUT);
  pinMode(BB, OUTPUT);
  pinMode(AA, OUTPUT);
```

```
pinMode(AB, OUTPUT);
pinMode(MA, OUTPUT);
pinMode(MB, OUTPUT);
pinMode(PA, OUTPUT);
pinMode(PC, OUTPUT);
//LED A
pinMode(LEDA, OUTPUT);
//BRAZO B
pinMode(HA_B, OUTPUT);
pinMode(HB_B, OUTPUT);
pinMode(BA_B, OUTPUT);
pinMode(BB_B, OUTPUT);
pinMode(AA_B, OUTPUT);
pinMode(AB_B, OUTPUT);
pinMode(MA_B, OUTPUT);
pinMode(MB_B, OUTPUT);
pinMode(PA_B, OUTPUT);
pinMode(PC_B, OUTPUT);

//LED B
pinMode(LEDB, OUTPUT);
//SENSORES
pinMode(SENSOR1, INPUT);
pinMode(SENSOR2, INPUT);
pinMode(SENSOR3, INPUT);
pinMode(SENSOR4, INPUT);

pinMode(CONTROL, OUTPUT);

// BANDA TRANSPORTADORA
pinMode(MOTORBANDA, OUTPUT);
```

```
}
```

3.3.3 Estructura del lazo de control.

La estructura del lazo de control, es el conjunto de instrucciones que el programa ejecutará mientras corra la aplicación. Cabe destacar que el control se realizará enviando letras mediante los botones de la app.

```
void loop() {
  leer_dato();
  //CONTROL HOMBR0 PINES 22 Y 23

  if (readString.length() > 0) {
    if (readString == "A") {
      Serial.println("Derecha");
      digitalWrite(HA, HIGH);
      digitalWrite(HB, LOW);
      delay(100);
      digitalWrite(HA, LOW);
    }
    else if (readString == "B") {
      Serial.println("Izquierda");
      digitalWrite(HA, LOW);
      digitalWrite(HB, HIGH);
      delay(100);
      digitalWrite(HB, LOW);

    }
  }

  // CONTROL BRAZO PINES 24 Y 25
  if (readString == "C") {
    Serial.println("Arriba");
    digitalWrite(BA, HIGH);
```

```
  digitalWrite(BB, LOW);
  delay(100);
  digitalWrite(BA, LOW);
}
else if (readString == "D") {
  Serial.println("ABAJO");
  digitalWrite(BA, LOW);
  digitalWrite(BB, HIGH);
  delay(100);
  digitalWrite(BB, LOW);
}

// CONTROL ANTEBRAZO PINES 26 Y 27
if (readString == "E") {
  Serial.println("ARRIBA");
  digitalWrite(AA, HIGH);
  digitalWrite(AB, LOW);
  delay(100);
  digitalWrite(AA, LOW);
}
else if (readString == "F") {
  Serial.println("Abajo");
  digitalWrite(AA, LOW);
  digitalWrite(AB, HIGH);
  delay(100);
  digitalWrite(AB, LOW);
}

// CONTROL MUÃ‘ECA PINES 28 Y 29
if (readString.length() > 0) {
  if (readString == "G") {
    Serial.println("ARRIBA");
```

```
    digitalWrite(MA, HIGH);
    digitalWrite(MB, LOW);
    delay(100);
    digitalWrite(MA, LOW);
  }
  else if (readString == "H") {
    Serial.println("ABAJO");
    digitalWrite(MA, LOW);
    digitalWrite(MB, HIGH);
    delay(100);
    digitalWrite(MB, LOW);
  }
}


// CONTROL PINZA PINES 30 Y 31
if (readString == "I") {
  Serial.println("Abrir");
  digitalWrite(PA, HIGH);
  digitalWrite(PC, LOW);
  delay(100);
  digitalWrite(PA, LOW);
}
else if (readString == "J") {
  Serial.println("Cerrar");
  digitalWrite(PA, LOW);
  digitalWrite(PC, HIGH);
  delay(100);
  digitalWrite(PC, LOW);
}
```

```
////////////////////////////////////BRAZO B ////////////////////////////////////////

//32 Y 33
if (readString.length() > 0) {
  if (readString == "K") {
    Serial.println("Derecha");
    digitalWrite(HA_B, HIGH);
    digitalWrite(HB_B, LOW);
    delay(100);
    digitalWrite(HA_B, LOW);
  }
  else if (readString == "L") {
    Serial.println("Izquierda");
    digitalWrite(HA_B, LOW);
    digitalWrite(HB_B, HIGH);
    delay(100);
    digitalWrite(HB_B, LOW);
  }
}

// CONTROL BRAZO PINES 34 Y 35
if (readString == "M") {
  Serial.println("Arriba");
  digitalWrite(BA_B, HIGH);
  digitalWrite(BB_B, LOW);
  delay(100);
  digitalWrite(BA_B, LOW);
}
else if (readString == "N") {
  Serial.println("ABAJO");
  digitalWrite(BA_B, LOW);
  digitalWrite(BB_B, HIGH);
```

```
  delay(100);
  digitalWrite(BB_B, LOW);
}

// CONTROL ANTEBRAZO PINES 36 Y 37
if (readString == "O") {
  Serial.println("ARRIBA");
  digitalWrite(AA_B, HIGH);
  digitalWrite(AB_B, LOW);
  delay(100);
  digitalWrite(AA_B, LOW);
}
else if (readString == "P") {
  Serial.println("Abajo");
  digitalWrite(AA_B, LOW);
  digitalWrite(AB_B, HIGH);
  delay(100);
  digitalWrite(AB_B, LOW);
}

// CONTROL MUÃ‘ECA PINES 38 Y 39
if (readString.length() > 0) {
  if (readString == "Q") {
    Serial.println("ARRIBA");
    digitalWrite(MA_B, HIGH);
    digitalWrite(MB_B, LOW);
    delay(100);
    digitalWrite(MA_B, LOW);
  }
  else if (readString == "R") {
    Serial.println("ABAJO");
    digitalWrite(MA_B, LOW);
```

```
    digitalWrite(MB_B, HIGH);
    delay(100);
    digitalWrite(MB_B, LOW);
  }
}


// CONTROL PINZA PINES 40 Y 41
if (readString == "S") {
  Serial.println("Abrir");
  digitalWrite(PA_B, HIGH);
  digitalWrite(PC_B, LOW);
  delay(100);
  digitalWrite(PA_B, LOW);
}
else if (readString == "T") {
  Serial.println("Cerrar");
  digitalWrite(PA_B, LOW);
  digitalWrite(PC_B, HIGH);
  delay(100);
  digitalWrite(PC_B, LOW);
}

//////////////////LEDS///////
if (readString == "W") {
  Serial.println("APAGAR");
  digitalWrite(LEDA, LOW);
  digitalWrite(LEDB, LOW);
}
else if (readString == "X") {
  Serial.println("ENCENDER");
  digitalWrite(LEDA, HIGH);
```

```
  digitalWrite(LEDB, LOW);
  delay(20);
  digitalWrite(LEDA, LOW);
  digitalWrite(LEDB, HIGH);
  delay(100);
  digitalWrite(LEDA, HIGH);
  digitalWrite(LEDB, LOW);
  delay(100);
  digitalWrite(LEDA, LOW);
  digitalWrite(LEDB, HIGH);
  delay(100);
  digitalWrite(LEDA, HIGH);
  digitalWrite(LEDB, LOW);
  delay(100);
  digitalWrite(LEDA, LOW);
  digitalWrite(LEDB, HIGH);
  delay(100);
  digitalWrite(LEDA, HIGH);
  digitalWrite(LEDB, LOW);
  delay(100);
  digitalWrite(LEDA, LOW);
  delay(75);
  digitalWrite(LEDA, HIGH);
  digitalWrite(LEDB, HIGH);
}

//demostracion de ejes
if (readString == "U") {
  digitalWrite(BB, HIGH);
  delay(750);
  digitalWrite(BB, LOW);
  digitalWrite(AB, HIGH);
```

```
delay(500);
digitalWrite(AB, LOW);
digitalWrite(MB, HIGH);
delay(1150);
digitalWrite(MB, LOW);
digitalWrite(PC, HIGH);
delay(1200);

//retroceso
digitalWrite(PC, LOW);
digitalWrite(MA, HIGH);
delay(1150);
digitalWrite(MA, LOW);
digitalWrite(AA, HIGH);
delay(2000);
digitalWrite(AA, LOW);
digitalWrite(BA, HIGH);
delay(2000);
digitalWrite(BA, LOW);
digitalWrite(HA, HIGH);
delay(8000);
digitalWrite(HA, LOW);
digitalWrite(MB, HIGH);
delay(500);
digitalWrite(HB, LOW);


//DEJAR EL OBJETO
digitalWrite(BB, LOW);
digitalWrite(AB, HIGH);
delay(300);
digitalWrite(AB, LOW);
```

```
  digitalWrite(MB, HIGH);
  delay(2000);
  digitalWrite(MB, LOW);
  digitalWrite(PA, HIGH);
  delay(1000);
  digitalWrite(PA, LOW);

  //regresar
  digitalWrite(MA, HIGH);
  delay(3000);
  digitalWrite(MA, LOW);
  digitalWrite(HB, HIGH);
  digitalWrite(BB, LOW);
  delay(8000);
  digitalWrite(HB, LOW);
  digitalWrite(BB, LOW);
  digitalWrite(AB, HIGH);
  delay(675);
  digitalWrite(AB, LOW);
}


if (readString == "V") {
  Serial.println("PARAR");
  digitalWrite(HA, LOW);
  digitalWrite(HB, LOW);
  digitalWrite(BA, LOW);
  digitalWrite(BB, LOW);
  digitalWrite(AA, LOW);
  digitalWrite(AB, LOW);
  digitalWrite(MA, LOW);
  digitalWrite(MB, LOW);
```

```
  digitalWrite(PA, LOW);
  digitalWrite(PC, LOW);
  digitalWrite(HA_B, LOW);
  digitalWrite(HB_B, LOW);
  digitalWrite(BA_B, LOW);
  digitalWrite(BB_B, LOW);
  digitalWrite(AA_B, LOW);
  digitalWrite(AB_B, LOW);
  digitalWrite(MA_B, LOW);
  digitalWrite(MB_B, LOW);
  digitalWrite(PA_B, LOW);
  digitalWrite(PC_B, LOW);
  digitalWrite(LEDA, LOW);
  digitalWrite(LEDB, LOW);
  delay(100);
}

I1 = digitalRead(SENSOR1);
I2 = digitalRead(SENSOR2);
I3 = digitalRead(SENSOR3);
I4 = digitalRead(SENSOR4);


if (I1 == HIGH) {
  if (I2 == HIGH) {
    digitalWrite(HA, HIGH);
    delay(200);
    digitalWrite(HA, LOW);
  }
  if (I2 == LOW) {
    digitalWrite(HA_B, HIGH);
    delay(200);
```

```
    digitalWrite(HA_B, LOW);
  }
}     readString = "";
  }
```

3.4 PROGRAMACIÓN DE INTERFAZ DE CONTROL (APLICACIÓN ANDROID)

Una vez realizado el software de control, ya es posible realizar la interfaz de usuario, la cual enviará la trama de texto que activara o desactivara los sensores del robot. Para diseñar la aplicación es de vital importancia tener una cuenta activada en Google, en caso no contar con una hay que proceder a crearla desde Google+. En el **Anexo B** podemos encontrar una guía de inicialización rápida**,** en la cual de manera generalizada aprenderemos a crear una aplicación rápida que servirá como entrenamiento antes de crear la aplicación definitiva de control

3.4.1 Diseño de la interfaz visual

Para el desarrollo de la aplicación hay que ingresar en la página de MIT App Inventor, accediendo en la siguiente dirección:
http://ai2.appinventor.mit.edu/

Se desplegará una pantalla en la cual se debe asignar un nombre al proyecto que se va a diseñar.

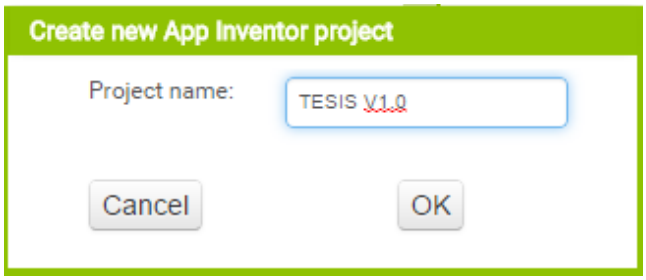


Figura 24: Abrir nuevo proyecto MIT APP INVENTOR 2
Fuente: (Google, s.f.)

Una vez creado el proyecto, hay que tomar en cuenta que la pantalla que se presenta primero es la pantalla de Designer, en donde se creara la interfaz visual y estética de la aplicación.

Existe otra pantalla en la cual se desarrollara el programa netamente dicho, denomiada Blocks, en donde se dará funciones a los botones que se inserten en la pantalla de Designer.

Al lado izquierdo de la pantalla de Designer se encuentran ordenados por tipo todos los componentes que podemos insertar a la app. Para el diseño de esta aplicación se necesitarán los siguientes componentes:

- Button
- Image
- Label
- Horizontal Arrangement
- Table Arrangement
- Bluetooth Client
- Screens

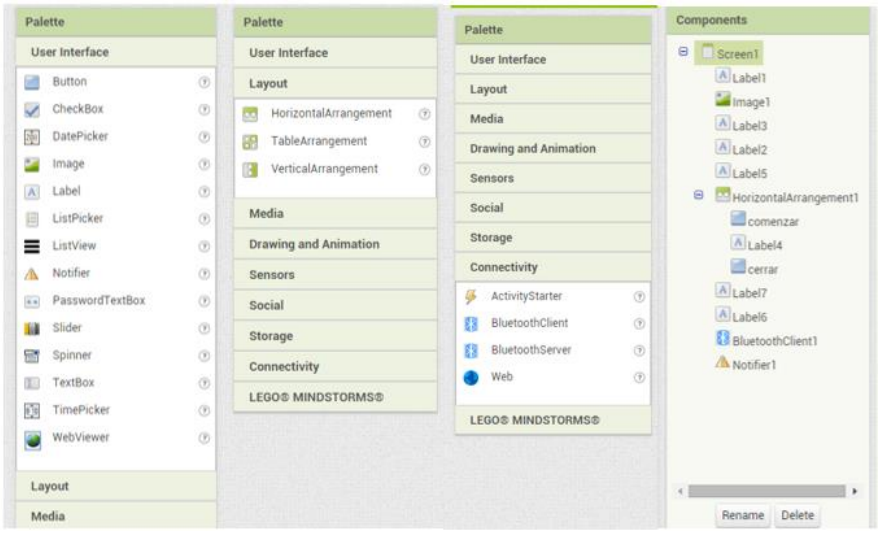


Figura 25: Componentes a utilizarse en la app
Fuente: (Google, s.f.)

Además de los elementos de interacción también se necesitan de más de una sola pantalla. Para agregarla se debe dar un click izquierdo en la pestaña de “Add Screen” que se muestra en la barra de navegación superior de la interfaz de App Inventor:

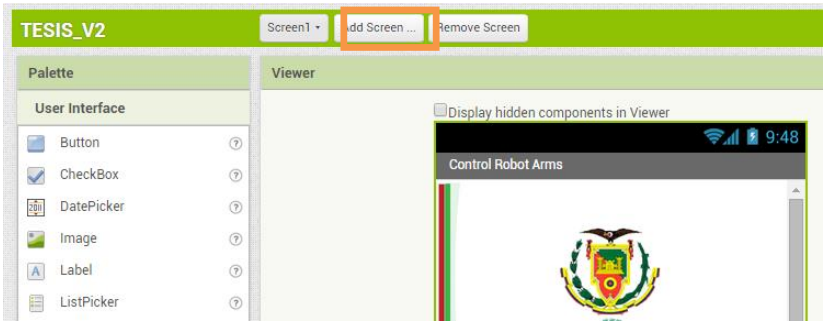


Figura 26: Añadir pantalla a la aplicación
Fuente: **(Google, s.f.)**

Antes de comenzar a seleccionar los componentes que se utilizarán en el diseño de la aplicación es necesario crear la base de datos de los archivos multimedia que se vayan a utilizar, es decir imágenes, video o archivos de sonido. Para la aplicación de control se necesitan solamente imágenes.

Para subir los archivos a la base de datos digital, se debe ir a la ventana de nombre “Media”, dar click en “Upload File” y seleccionar todos los archivos que se deseen. De esta manera al momento de insertar imágenes será necesario únicamente seleccionarlas y ya no cargarlas. Las imágenes a utilizarse se encuentran en el **Anexo C**

3.4.1.1 Portada.

La primera pantalla añadida será la pantalla que primero se muestre al abrir la aplicación, por lo tanto es aquí donde se realizara la portada de bienvenida de la app. La misma que se configurará con alineación central para los elementos, se quitará el color de fondo y se añadirá una imagen de fondo. Para todas las pantallas se realizará la misma acción de configuración visual.

Esta configuración se la realiza en la parte derecha de la pantalla, en una ventana de edición de nombre “Properties”, en donde al seleccionar un componente se despliega una lista de propiedades modificables tales como: color de fondo, imagen de fondo, altura en pixeles, ancho en pixeles, se puede definir si es visible o no visible al arrancar la aplicación, etc.

Propiedades y componentes de la pantalla principal:

En la Tabla 1 se muestran los parámetros que necesitan ser editados con su valor por defecto y con los valores que necesitamos que estén fijados.

Tabla 1: Propiedades de la pantalla

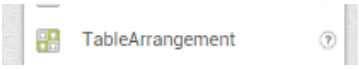
Properties		
Screen 1		
Propiedad	**Valor por defecto**	**Valor necesario**
Align Horizontal	Left	Center

Continúa

AppName	Nombre que se le asignó al proyecto	Tesis
BackgroundColor	White	None
BackgroundImage	None	Fondo.png
CloseScreenAnimation	Default	SlideVertical
Icon	None	Icon.png
OpenScreenAnimation	Default	SlideHorizontal
ScreenOrientation	Unspecified	Sensor
Tittle	Screen1	Portada

Fuente: (Zapata, 2015)

Una vez editada la pantalla, es necesario implementar los componentes a utilizarse, es decir los botones, las etiquetas de texto, las imágenes etc. En esta pantalla se insertará un encabezado compuesto por un Table Arrangement en donde estará insertado en este mismo orden un Label, un Button, y dos Image.

De la paleta de componentes seleccionamos el componente TableArrangement 

lo arrastramos hasta la pantalla y en sus propiedades editamos el número de columnas de 2 a 4, el número de filas de 2 a 1 y el parámetro Width de Automátic a Fill Parent.

Figura 27: Propiedades TableArrangment

Fuente: (Google, s.f.)

Dentro del TableArrangment colocamos los elementos que se mencionaron con anterioridad para el encabezado.

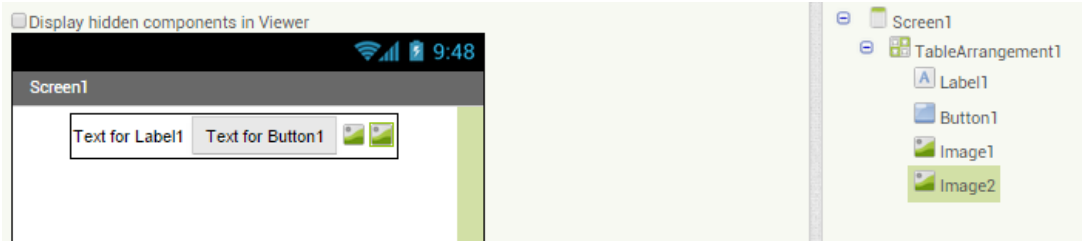


Figura 28: Encabezado de la aplicación.
Fuente: (Google, s.f.)

En la Tabla 2 se puede observar los parámetros de cada componente en base a los requerimientos de la aplicación que se está diseñando.

Tabla 2: Propiedades del encabezado de la aplicación

Label		
Propiedad	**Valor por defecto**	**Valor necesario**
Text	Text for Label 1	Vacío
Width	Automatic	10 pixels
Heigth	Automatic	10 pixels
Button		
Propiedad	**Valor por defecto**	**Valor necesario**
Text	Text for Button 1	Vacío
Width	Automatic	30 pixels
Heigth	Automatic	33 pixels
Image 1, Image 2		
Propiedad	**Valor por defecto**	**Valor necesario**
Image	None	111.png
Width	Automatic	30
Heigth	Automatic	33

Fuente: (Zapata, 2015)

En la parte central de la pantalla irá una imagen de portada, la cuál será editada de la misma manera que los elementos anteriores, cargando la imagen en la opción de “Image” y con un ancho de Fill Parent y altura de Automatic. Además ira un Label con el texto “Control Android Arduino”.

De esta manera está ya diseñada la pantalla de portada y se puede ir emulando paso a paso la edición de la aplicación, al momento de crear una cuenta en MIT App Inventor nos permite descargar un emulador de nuestro dispositivo Android para en el ir compilando y corrigiendo errores antes de pasarlo directamente al celular. Para ello vamos a la pestaña Connection y seleccionamos la opción Emulator:

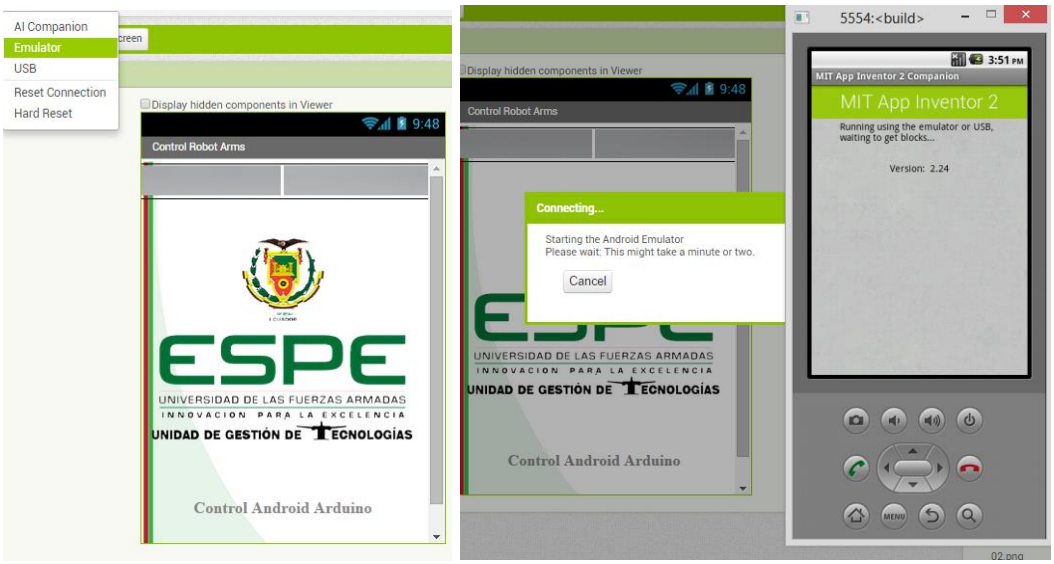


Figura 29: Conexión de la aplicación con el emulador.
Fuente: (Google, s.f.)

Otra manera de compilar la aplicación es directamente en el celular con una aplicación de nombre MIT AI2 Companion disponible de forma gratuita en el PlayStore de Android. Para esta conexión es necesario leer un código QR generado automáticamente o mediante un código de pariedad ingresado en la aplicación antes descargada:

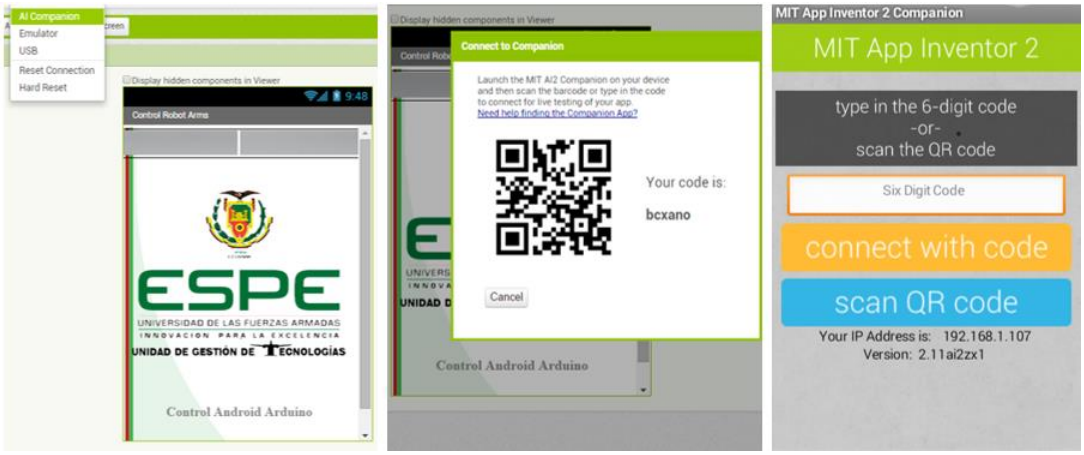


Figura 30: Compilación directamente en el celular de la aplicación
Fuente: (Google, s.f.)

3.4.1.2 Menú

Como se había mencionado, la configuración visual de la pantalla será la misma en cada pantalla, es decir se repetirá el procedimiento mostrado en la Tabla 1, incluyendo la configuración del encabezado.

Una vez configurado el aspecto de la pantalla, es necesario arrastrar los componentes a la misma:

- 1 TableArrangment de 2 columnas y 6 filas en el cual se insertarán:
  - 6 Buttons de 155x135 pixels cada uno.
- 2 Labels, en donde se presentará la hora y fecha en tiempo real.
- Clock (componente no visible)

A cada botón se le cambiará el nombre para identificarlos de mejor manera, en la parte inferior de la ventana Components se encuentra el botón “rename” en el cual al seleccionar cualquier componente previamente insertado en la pantalla le asignará el nombre que aquí se edite.

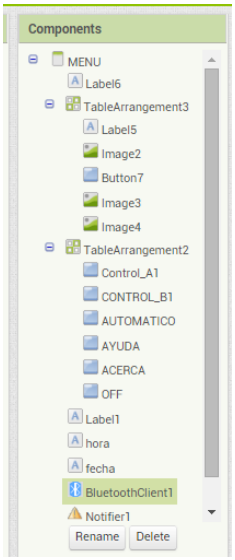


Figura 31: Renombrar a los componentes de la aplicación
Fuente: (Google, s.f.)

Cada button y label contará con los parámetros descritos en la Tabla 3:

Tabla 3: Parámetros de botones del menú de la aplicación.

	Control_A1	
Propiedad	**Valor por defecto**	**Valor necesario**
Image	None	ConA.png
Width	Automatic	155 pixels
Heigth	Automatic	135 pixels
	Control_B1	
Propiedad	**Valor por defecto**	**Valor necesario**
Image	None	ConB.png
Width	Automatic	155 pixels
Heigth	Automatic	135 pixels
	Automático	
Propiedad	**Valor por defecto**	**Valor necesario**
Image	None	AUTO.png
Width	Automatic	155 pixels
Heigth	Automatic	135 pixels
	Ayuda	
Propiedad	**Valor por defecto**	**Valor necesario**
Image	None	AYUDA.png
Width	Automatic	155 pixels
Heigth	Automatic	135 pixels
	Acerca	

Continúa

Propiedad	Valor por defecto	Valor necesario
Image	None	ACERCA DE.png
Width	Automatic	155 pixels
Heigth	Automatic	135 pixels
	Off	
Propiedad	**Valor por defecto**	**Valor necesario**
Image	None	SALIR.png
Width	Automatic	155 pixels
Heigth	Automatic	135 pixels
	Label1	
Propiedad	**Valor por defecto**	**Valor necesario**
TextAlignment	Left	Right
TextColor	White	Gray
Width	Automatic	Fill Parent
	Label2	
Propiedad	**Valor por defecto**	**Valor necesario**
TextAlignment	Left	Right
TextColor	White	Gray
Width	Automatic	Fill Parent

Fuente: (Zapata, 2015)

Al finalizar esta configuración el aspecto visual será el siguiente:

Figura 32: Pantalla del menú principal
Fuente: (Zapata, 2015)

3.4.1.3 Control Brazo A y Brazo B.

Estas pantallas contendrán los siguientes componentes:

- Encabezado (el mismo descrito previamente).
- Image en la cual se cargará un banner para la ventana.
- Label
- TableArrangment (columnas 3 filas 5) que agrupará:
  - 10 Buttons.
  - 5 Labels.
- TableArrangmente (columnas 3, filas 1) que agrupar:
  - 2 Buttons
  - 1 Label.

De igual manera cada botón será renombrado de acuerdo a la acción que ejecutarán:

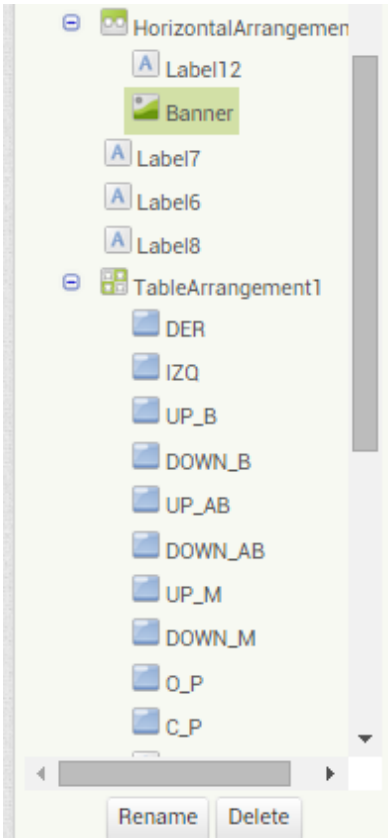


Figura 33: Componentes insertados en la pantalla del control A y B
Fuente: (Google, s.f.)

En el TableArrangment de 3x5 irá en cada fila un Button, un Label, y otro Button en ese orden específico, los Label insertados en medio de los botones servirá de identificación del eje que controla cada pulsador, es decir Base, Brazo, Antebrazo, Muñeca y Pinza en el mismo orden descrito.

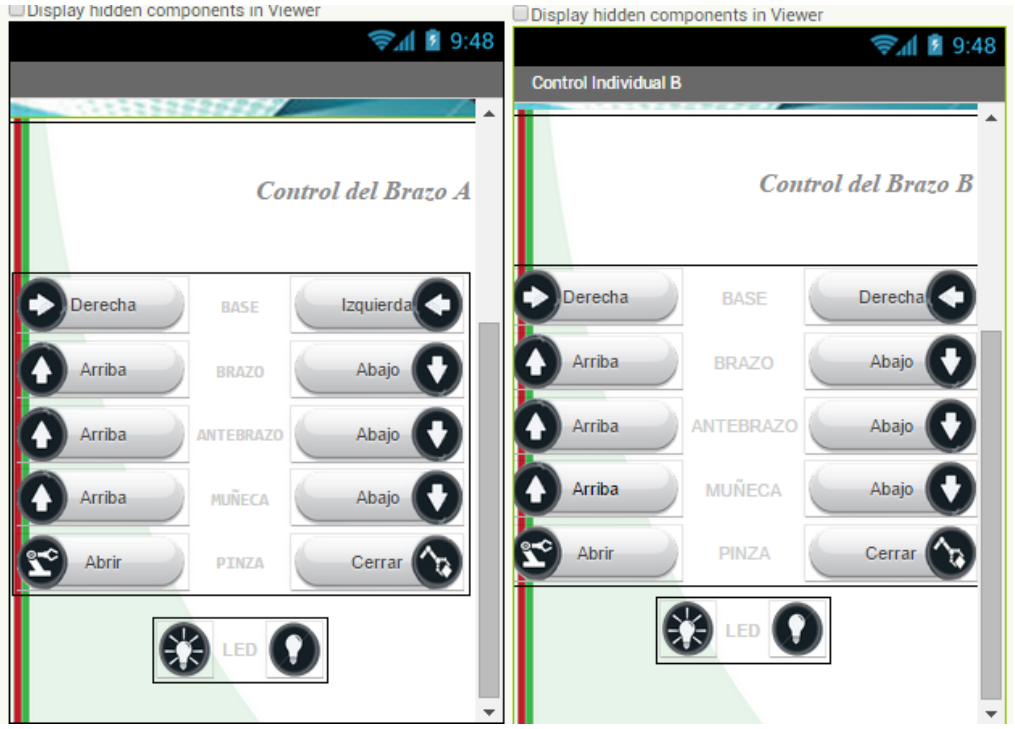


Figura 34: Controles Brazo A y Brazo B
Fuente: (Google, s.f.)

Sus parámetros se muestran en la Tabla 4:

Tabla 4: Parámetros de los botones e imágenes del Control A y B

		Botón: DER	
	Propiedad	**Valor por defecto**	**Valor necesario**
BASE	Image	None	Der.png
	Width	Automatic	155 pixels
	Heigth	Automatic	135 pixels
		Botón: IZQ	
	Propiedad	**Valor por defecto**	**Valor necesario**
	Image	None	Izq.png
	Width	Automatic	155 pixels

Continúa

	Heigth	Automatic	135 pixels
		Botón: UP_B	
	Propiedad	**Valor por defecto**	**Valor necesario**
BRAZO	Image	None	Arriba.png
	Width	Automatic	155 pixels
	Heigth	Automatic	135 pixels
		Botón: DOWN_B	
	Propiedad	**Valor por defecto**	**Valor necesario**
	Image	None	Abajo.png
	Width	Automatic	155 pixels
	Heigth	Automatic	135 pixels
		Botón: UP_AB	
	Propiedad	**Valor por defecto**	**Valor necesario**
ANTEBRAZO	Image	None	Arriba.png
	Width	Automatic	155 pixels
	Heigth	Automatic	135 pixels
		Botón: DOWN_AB	
	Propiedad	**Valor por defecto**	**Valor necesario**
	Image	None	Abajo.png
	Width	Automatic	155 pixels
	Heigth	Automatic	135 pixels
		Botón: UP_M	
	Propiedad	**Valor por defecto**	**Valor necesario**
MUÑECA	Image	None	Arriba.png
	Width	Automatic	155 pixels
	Heigth	Automatic	135 pixels
		Botón: DOWN_M	
	Propiedad	**Valor por defecto**	**Valor necesario**
	Image	None	Abajo.png
	Width	Automatic	155 pixels
	Heigth	Automatic	135 pixels
		Botón: O_P	
	Propiedad	**Valor por defecto**	**Valor necesario**
PINZA	Image	None	AbriPinza.png
	Width	Automatic	155 pixels
	Heigth	Automatic	135 pixels
		Botón: C_P	
	Propiedad	**Valor por defecto**	**Valor necesario**
	Image	None	CerrarPinza.png
	Width	Automatic	155 pixels
	Heigth	Automatic	135 pixels
		Image: Banner	
	Propiedad	**Valor por defecto**	**Valor necesario**
	Width	Automatic	Automatic

Continúa

Heigth	Automatic	Automatic

Fuente: (Zapata, 2015)

Para el control de los led's en el control A y B se inserta en la pantalla un nuevo TableArrangment de 3 columnas x 1 fila

Figura 35: Controles del led de los brazos A y B
Fuente: (Google, s.f.)

Los componentes que muestra la Figura 27 son 2 botones y un label insertados en el TableArrangment. Su parametrización se muestra en la Tabla 5.

Tabla 5: Botones de control de los led A y B

		Botón: ON	
	Propiedad	**Valor por defecto**	**Valor necesario**
LED'S	Image	None	ON.png
	Width	Automatic	155 pixels
	Heigth	Automatic	135 pixels
		Botón: OFF	
	Propiedad	**Valor por defecto**	**Valor necesario**
	Image	None	OFF.png
	Width	Automatic	155 pixels
	Heigth	Automatic	135 pixels

Fuente: (Zapata, 2015)

3.4.1.4 Control Automático.

Esta pantalla contendrá los siguientes componentes:

- Encabezado (el mismo descrito previamente).
- Image en la cual se cargará un banner para la ventana.
- Image de descripción
- Label
- TableArrangment (columnas 2 filas 2) que agrupará:
  - 2 Buttons.

- BluetoothClient (componente no visible).

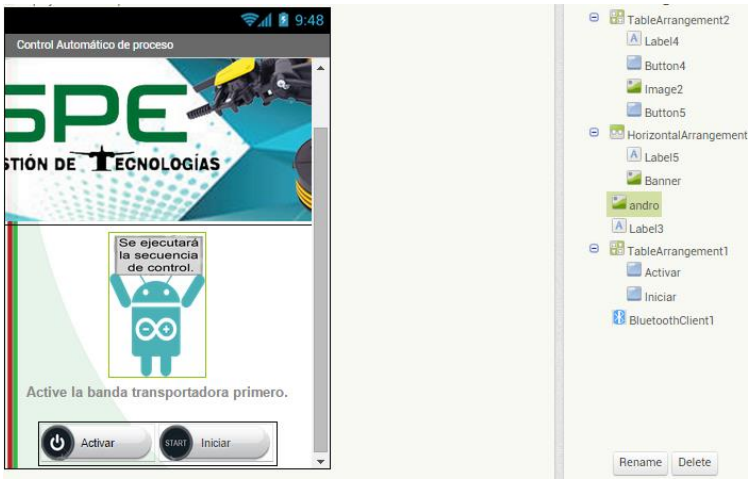


Figura 36: Componentes de la pantalla de control automático.
Fuente: (Google, s.f.)

En la tabla 6 se muestra su respectiva parametrización.

Tabla 6: Parametrización de pantalla de control.

Image: Banner		
Propiedad	**Valor por defecto**	**Valor necesario**
Image	None	Bannertesis.png
Width	Automatic	Automatic
Heigth	Automatic	Automatic
Image: Android		
Propiedad	**Valor por defecto**	**Valor necesario**
Image	None	asd.png
Width	Automatic	155 pixels
Heigth	Automatic	135 pixels
Label		
Propiedad	**Valor por defecto**	**Valor necesario**
Text	TextForLabel1	Active la banda transportadora primero.
Width	Automatic	Automatic
Heigth	Automatic	30 pixels
Botón: ACTIVAR		
Propiedad	**Valor por defecto**	**Valor necesario**

Continúa

Image	None	mb.png
Width	Automatic	120 pixels
Heigth	Automatic	40 pixels
Botón: INICIAR		
Propiedad	**Valor por defecto**	**Valor necesario**
Image	None	iniciar.png
Width	Automatic	120 pixels
Heigth	Automatic	40 pixels

Fuente: (Google, s.f.)

3.4.1.5 Ayuda.

Para la ventana de Ayuda se crea un nuevo esquema de menú con un TableArrangment de 2 columnas x 3 filas y en cada fila irá una imagen y un botón:

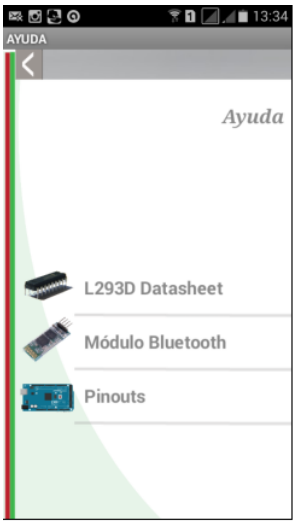


Figura 37: Ventana Ayuda de la aplicación
Fuente: (Zapata, 2015)

La parametrización se la encuentra en la Tabla 7:

Tabla 7: Parámetros de la ventana Ayuda

Image: CI		
Propiedad	**Valor por defecto**	**Valor necesario**
Image	None	L2.png

Continúa

Width	Automatic	Automatic
Heigth	Automatic	Automatic
	Image: HC	
Propiedad	**Valor por defecto**	**Valor necesario**
Image	None	hca.png
Width	Automatic	155 pixels
Heigth	Automatic	135 pixels
	Image: Tarjeta	
Propiedad	**Valor por defecto**	**Valor necesario**
Image	None	Arduino_Mega.png
Width	Automatic	Automatic
Heigth	Automatic	30 pixels
	Botón: L293D	
Propiedad	**Valor por defecto**	**Valor necesario**
Image	None	W.PNG
Width	Automatic	Automatic
Heigth	Automatic	Automatic
	Botón: MODULOBT	
Propiedad	**Valor por defecto**	**Valor necesario**
Image	None	W.PNG
Width	Automatic	Automatic
Heigth	Automatic	Automatic
	Botón: PINOUTS	
Propiedad	**Valor por defecto**	**Valor necesario**
Image	None	W.PNG
Width	Automatic	Automatic
Heigth	Automatic	Automatic

Fuente: (Zapata, 2015)

3.4.1.5 Acerca de.

La ventana Acerca de es un plues adicional en el cual se describe la herramienta de programación utilizada, el sitio web en el cuál se encuentra dicho software de manera gratuita y datos de la aplicación como la versión y el autor.

Realmente esta ventana no es necesaria para la funcionalidad de la aplicación en el control propiamente dicho de los brazos robóticos.

Figura 38: Pantalla Acerca De.
Fuente: (Google, s.f.)

3.4.2 Diagramas de bloques de la aplicación.

Una vez terminado el diseño en la ventana “designer”, se procederá a darle las respectivas acciones que tendrán que ejecutar cada botón insertado en la pantalla de diseño.

Cada pantalla será programada por separado, por lo tanto se debe tomar en cuenta que funciones se le asigna a cada botón para evitar estados ambiguos en la activación de los actuadores de los autómatas.

Los bloques extras están definidos y clasificados por colores de acuerdo a su utilidad.

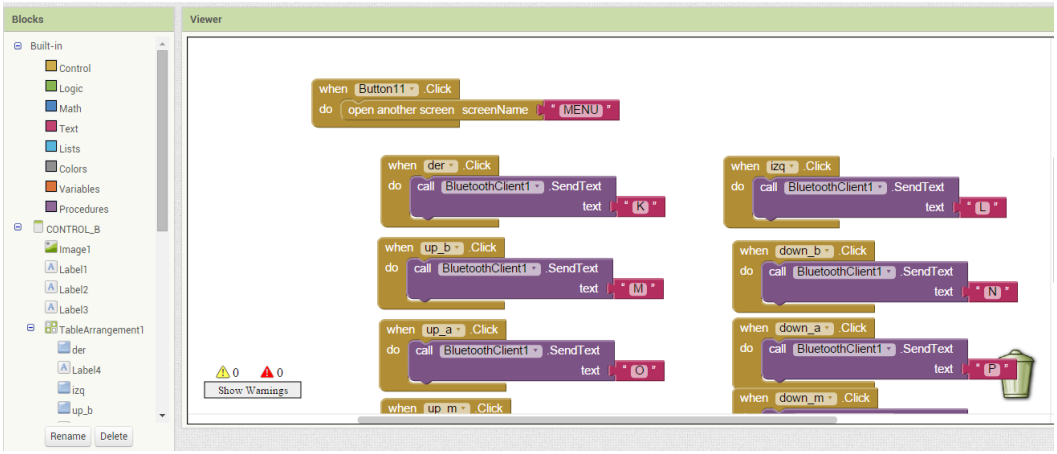


Figura 39: Ventana Blocks MIT APP INVENTOR2
Fuente: (Google, s.f.)

3.4.2.1 Programación de la pantalla Portada.

Para darle su función a los componentes se debe seleccionar en el bloque propio de cada uno de ellos mostrados en la parte izquierda de la pantalla Blocks. Aquí se puede definir qué acción va a realizar cada botón al pulsarlo una sola vez, al mantenerlo pulsado durante un tiempo de retardo (en segundos), etc.

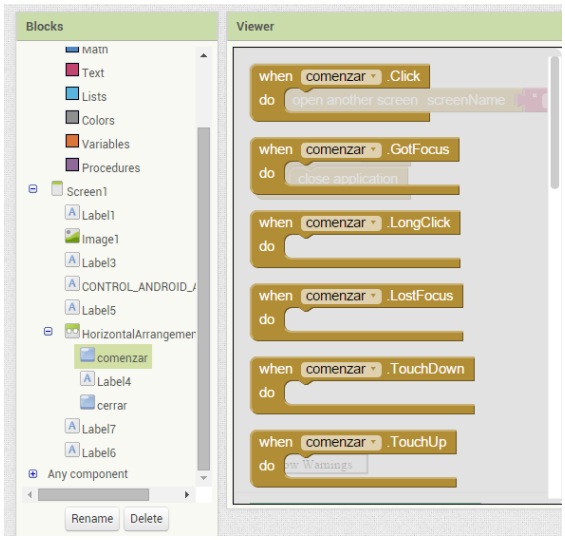


Figura 40: Funciones de los componentes.
Fuente: (Google, s.f.)

Cada función está programada para un pulso normal en cada botón asi que se utilizara el bloque “When ___ Click, Do”:

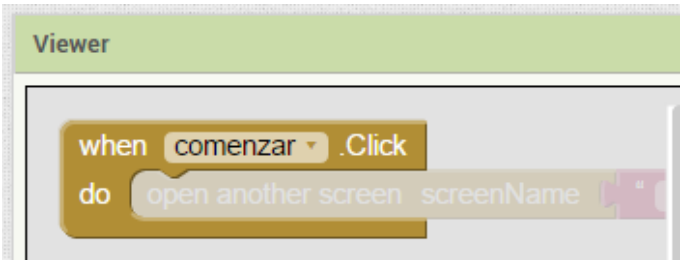


Figura 41: Bloque de función para los botones
Fuente: (Google, s.f.)

La programación de la pantalla de portada está diseñada para re direccionar con el botón menú a la pantalla de nombre MENU. En el editor de bloques sacamos el bloque de click único del botón y hacemos un llamado a abrir otra pantalla:

Figura 42: Programación pantalla Portada.
Fuente: (Google, s.f.)

3.4.2.2 Programación de la pantalla Menú.

En la pantalla MENU también se hará un llamado a distintas ventanas al presionar los botones principales y al presionar el botón emergente SALIR se cerrará la aplicación.

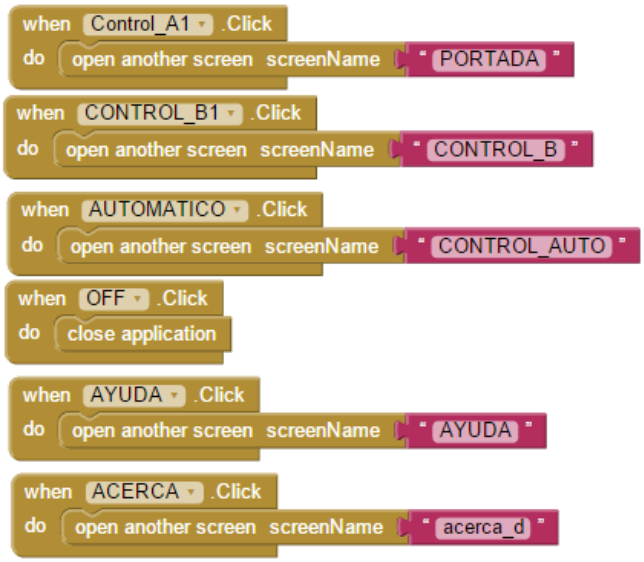


Figura 43: Programación pantalla Menú.
Fuente: (Google, s.f.)

Para insertar la fecha y hora actual en los labels del final de la pantalla es necesario hacer un llamado al componente no visible “clock” para de esta manera adquirir los datos en tiempo real del dispositivo en el cual se ejecuta la aplicación.

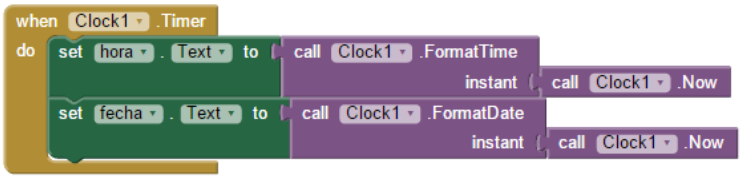


Figura 44: Bloques para generar la fecha y hora actual.
Fuente: (Google, s.f.)

3.4.2.3 Programación de la pantalla Control A y Control B.

En las pantallas de Control, se realizará el envío de los códigos de programación hacia el módulo bluetooth. Aquí ya entra en programación el

BluetoothClient insertado en la pantalla, el cuál primeramente debe ser inicializado insertando una variable local en la que se especifique la dirección Mac de dicho módulo de comunicación.

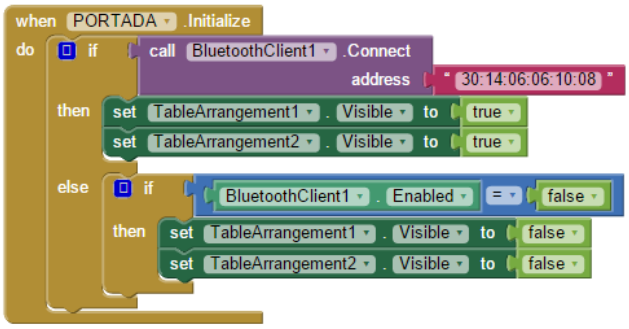


Figura 45: Programación pantallas de control inicializando la comunicación.
Fuente: (Google, s.f.)

En la Figura 38 se encuentra la función de inicialización de la comunicación, en donde cuando inicialice la pantalla se hace el llamado al módulo y mientras no se encuentre una conexión no se habilitan los enable de cada componente insertado.

Una vez conectado se habilitaran los componentes y se puede proceder a enviar los códigos que en este caso son letras:

Figura 46: Programación de tramas de control.
Fuente: (Google, s.f.)

En cada pantalla de control se encuentra la misma raíz de programación con la única diferencia de que las letras que se envían (todas mayúsculas) son diferentes para cada botón, el control va desde la base hasta la muñeca del robot, es decir para la base van las letras A y B hasta llegar a la muñeca que se controla con las letras I y J. Para los leds el control utiliza números 1=on 2=off.

Para el control del brazo B las letras de control van desde la K hasta la T y los leds se controlan con los números 3 y 4 para on y off respectivamente.

Figura 47: Programación brazo B.
Fuente: (Google, s.f.)

3.4.2.4 Programación de la pantalla Control Automático.

En la pantalla de control automático se utilizará el número 5 para activar la banda transportadora, el número 6 para desactivarla y la letra U para el control.

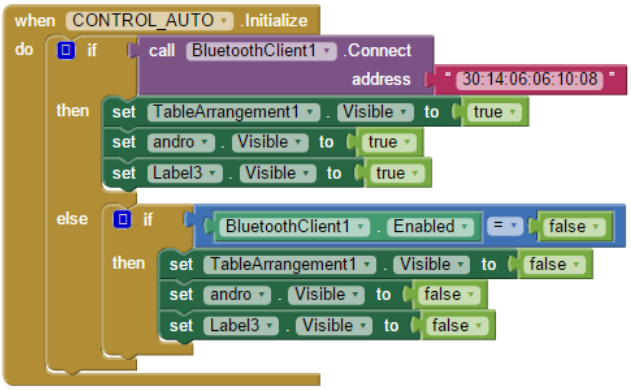


Figura 48: Programación Control Automático.
Fuente: (Google, s.f.)

Cada pantalla al inicializarse se conectará al bluetooth, y contará con un botón que redirecciona directamente hasta la pantalla del menú, desconectándose del bluetooth y permitiendo realizar el control de una nueva etapa.

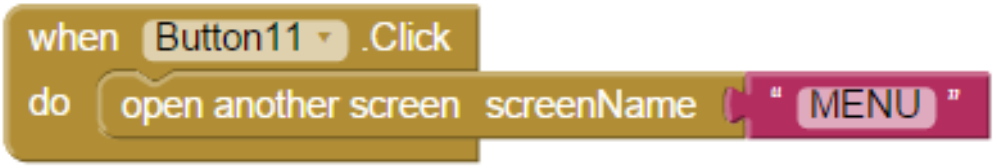


Figura 49: Regresar al menú principal.
Fuente: (Google, s.f.)

3.4.2.5 Programación de la pantalla Ayuda.

En esta pantalla existen tres botones que redireccionan a nuevas pantallas en las cuales se proyectan simplemente imágenes de los datasheet de los integrados de control L293D y del módulo bluetooth. Y el botón Pinouts redirecciona a una pantalla que muestra esquemáticamente la conexión entre arduino y los motores de los brazos, pasando por el control en los integrados L293D.

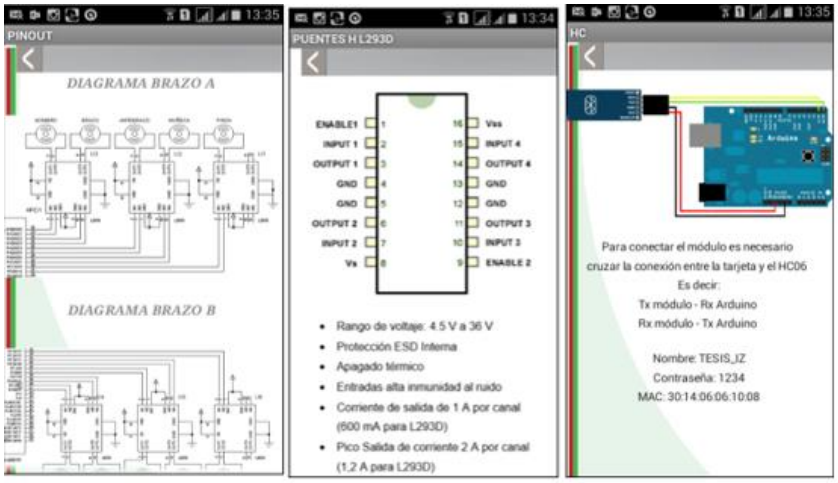


Figura 50: Pantallas del menú Ayuda.
Fuente: (Google, s.f.)

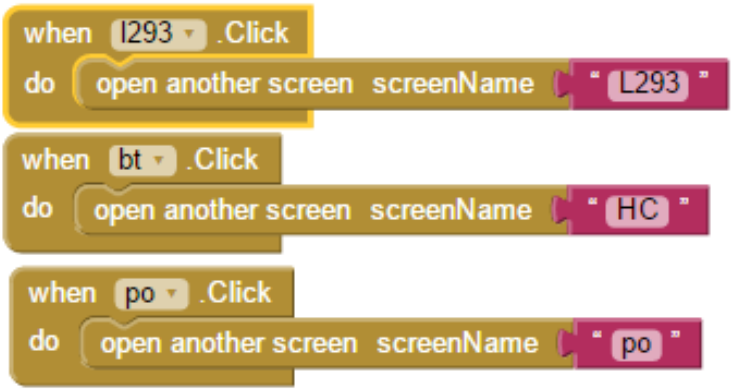


Figura 51: Programación de la pantalla Ayuda.
Fuente: (Google, s.f.)

En cada pantalla existe el banner con un botón de retorno al menú Ayuda.

3.4.2.6 Programación de la pantalla “Acerca de”.

Esta pantalla, al igual que todas, cuenta con el banner de retorno, por lo tanto su única programación es de retorno al menú principal.

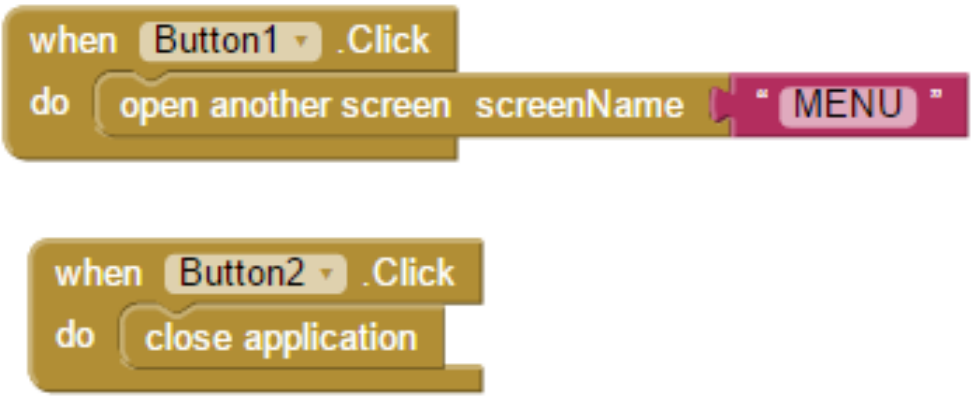


Figura 52: Programación de la pantalla Acerca de.
Fuente: (Google, s.f.)

3.5 CAMPO INDUSTRIAL A ESCALA EMPAREJAMIENTO Y PRUEBAS.

Para realizar la prueba operacional de la eficiencia del control en los autómatas se ha optado por la simulación de un mini proceso industrializado en donde la función principal de los brazos robot es tomar desde una posición cero a unos cubos de metal, ponerlos en una banda transportadora también controlada por bluetooth y seleccionar si cumple con una u otra exigencia física, es decir, al final de la banda existirán sensores CNY70 (para visualizar su hoja técnica diríjase al **Anexo E**) que detectará, franjas de color blanco en los cubos, de esta manera al existir dos franjas en el objeto, el brazo robótico lo llevará hacia una posición "aprobada" mientras que de lo contrario al existir una sola franja en el cubo, el brazo lo llevará a una posición "desaprobada", cumpliendo de esta manera un ciclo selectivo en el cual la automatización se la realizara totalmente independiente de la manipulación manual del usuario. Cabe recalcar que para comenzar con el control se debe primero movilizar al robot hacia su posición cero con la ayuda del control individual de cada autómata utilizando la aplicación Android cargada en el celular.
La banda transportadora se activará desde la pantalla de Control Automático, en donde el botón enviará un número 5 mediante bluetooth, lo

que activará una entrada del puente H (CI L293D). El circuito de control del motor se muestra en la figura 47.

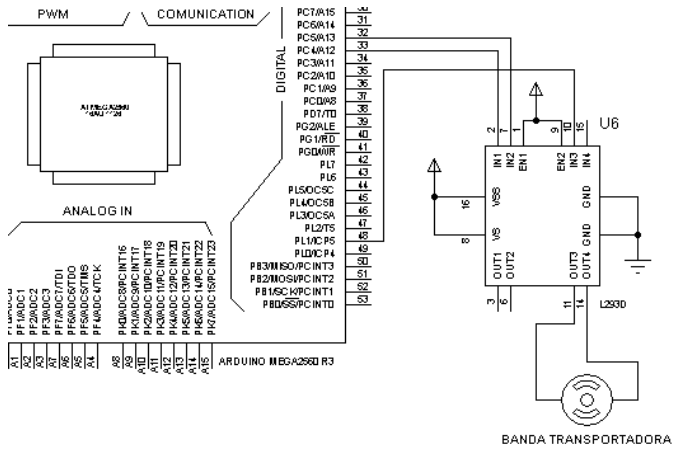


Figura 53: Esquema de conexión del motor de la banda transportadora.
Fuente: (Zapata, 2015)

Se eligió como elemento central del campo industrial a una banda transportadora debido a que en el campo industrial es el elemento más común en todo proceso, desde el transporte de materia prima hasta la salida de conversión y posterior empaquetado de casi cualquier producto elaborado industrialmente.

3.5.1 Implementación del prototipo.

Una vez diseñado cada elemento que comprende el conjunto del prototipo de control, es necesario establecer una unión física entre ellos para poder crear una interconexión secuencial en el control.

De esta manera se necesita la implementación previa de tarjetas de control de giro e inversión del mismo para los motores ejes del robot, lo cual se realizará con la ayuda de los circuitos integrados L293D (diríjase al **Anexo F** para visualizar su datasheet) los cuales tendrán una habilitación de

estado lógico enviada mediante los puertos digitales de la tarjeta Arduino. Con la ayuda de la aplicación se activarán dichos puertos de Arduino que enlazan vía Bluetooth el autómata y el control electrónico (app).

El diagrama del circuito general del prototipo se lo puede encontrar en el **ANEXO G,** que contiene el pinout de cada brazo, de los sensores, y de la banda transportadora.

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1 **CONCLUSIONES**

- Posterior a un análisis detallado de los tipos de tarjetas Arduino, se pudo seleccionar a la adecuada para esta aplicación, tomando en cuenta detalles como el número de puertos disponibles en cada tarjeta y la capacidad de su procesador central.
- La comunicación bluetooth facilita el control inalámbrico de procesos debido a la velocidad de su comunicación, y dicha comunicación se realizó de manera correcta y efectiva enlazando el módulo HC 06 y la tarjeta Arduino mediante líneas de programación básicas estableciendo así una velocidad de transmisión de 9600bps.
- El desarrollo de la aplicación para el sistema operativo Android permite evidenciar que este lenguaje de programación es cada vez más fácil de aprender y sobre todo capaz de cumplir con casi cualquier requerimiento de control adquisición y manipulación de variables de cualquier tipo, de esta manera se hace posible una adaptación a cada necesidad de automatización y control requerido por un estudiante.
- El prototipo de proceso industrial sin duda es el medio de interacción perfecto para poner a prueba a los brazos guiados por la aplicación móvil instalada en el Smartphone, recalcando también que se han realizado pruebas instalando en diferentes dispositivos móviles de SO Android la aplicación de control.

4.2 **RECOMENDACIONES**

- Trabajar cuidadosamente con los voltajes de alimentación de los brazos robóticos, de la tarjeta Arduino y de los integrados de control de giro L293D, ya que para un mejor resultado sería mejor alimentar a los brazos con su fuente independiente del sistema, es decir con cuatro baterías tipo D, se evidenció que al alimentarlos con la fuente que incorpora Arduino existe un corto circuito en una de las salidas y es posible incluso llegar a quemar los motores de los ejes del robot o incluso a misma tarjeta. Los demás componentes si pueden ser alimentados con la fuente de 5V de Arduino y el módulo Bluetooth incluso puede ser alimentado con la fuente de 3V.
- Tomar en consideración el alcance y las limitaciones que tiene MIT App Inventor ya que este software en su aspecto visual es muy delimitante y se debe optar por implementar varias imágenes para mejorar la interfaz de la aplicación, es recomendable utilizar programas de edición de imágenes especializados de tal manera que mejore aún más la estética de la aplicación. Además que es un desarrollador exclusivo para Android.
- Verificar la conexión entre las entradas y las salidas de la tarjeta Arduino hacia los integrados L293D y de éstos hacia los actuadores del robot, ya que si el sentido de giro está contrario al especificado es porque dos de las entradas o salidas están conectadas erróneamente, la solución más fácil sería intercambiar físicamente los cables de comunicación.
- Tener muy presente que aunque el trabajo investigativo se desarrolló en base a una tarjeta Arduino Mega esto no quiere decir que sea una aplicación de uso exclusivo por esta tarjeta, en el caso de querer desarrollarla sobre una tarjeta Arduino diferente, es necesario cambiar la programación Arduino específicamente en la

asignación de entradas y salidas, tomando en cuenta la limitación de puertos de cada tarjeta. En la aplicación no hace falta ningún otro cambio o modificación.

GLOSARIO DE TERMINOS

A

Android: Sistema operativo para dispositivos móviles de software libre basado en Linux.

Aplicación: Interfaz programada para ejecutarse en celulares inteligentes o tablets.

Arduino: Plataforma de programación de software libre basada en una tarjeta con un microcontrolador AVR

AVR: Microcontroladores fabricados por Atmel. Núcleo de una yarjeta Arduino.

Atmega XXXX: Microcontroladores AVR con memoria flash programable.

B

Bluetooth: Método de comunicación inalámbrico basado en el protocolo RS232.

Bloque de programación: Elemento compacto que contiene un conjunto de sentencias e instrucciones.

Brazo robot: Brazo electrónico programable capaz de emular las acciones de un brazo humano.

C

Campo Industrial: Lugar en donde se desarrollan procesos industrializados.

Cargar: Enviar la información desde el software de programación hasta la memoria del microcontrolador.

Compilar: Depurar errores en un código de programación.

E

Enable: Habilitación de un estado lógico.

Ensamblar: Armar o unir dos o más partes de un solo elemento

Entrada/salida analógica: Entrada/salida de estados lógicos 1, 0.

Entrada/salida digital: Entrada/salida de variables analógicas.

H

Hardware: Partes físicas de un conjunto tecnológico.

I

Interfaz: Conexión entre hardware y software de un sistema.

L

Lazo de control: Control y manipulación de variables de un proceso.

Linux: Sistema operativo de software libre.

M

Mit App Inventor: Plataforma de Google que sirve de herramienta de programación de aplicaciones Android

Módulo: Unión de elementos que al agruparse hace más fácil su ubicación.

P

Plataforma: Sistema que sirve como base para la realización de una estructura lógica.

Processing/wiring: Lenguaje de programación de código abierto basado en Java

Puerto Serie: Medio de comunicación industrial.

R

Rs 232. Norma de intercambio de comunicación serial o serie.

S

Sensor: Dispositivo electrónico capaz de transformar magnitudes físicas en variables eléctricas.

Software: Parte intangible de un Sistema informático comprende componentes lógicos.

Software Libre: Software que permite la interacción libre y gratuita de los usuarios para modificaciones y mejoras.

Referencias bibliográficas.

Android. (s.f.). *Android*. (Google Inc.) Obtenido de https://www.android.com

Arduino. (2015). *Arduino*. Obtenido de http://arduino.cc

Corín, P. (s.f.). *App Inventor Spanish*. (Creative Commons Attributtion 3.0)

Google. (s.f.). *Mit app inventor 2*. (Massachusetts Institute of Technology (MIT)) Obtenido de http://ai2.appinventor.mit.edu/

León, J. (17 de 7 de 2012). *Androideity*. (Creative Commons) Obtenido de http://androideity.com/2012/07/16/5-lenguajes-para-programar-en-android/

Lujan, J. D. (01 de Enero de 2014). *Desarrollo web. Introducción a Android.* Obtenido de http://www.desarrolloweb.com/articulos/introduccion-android.html

Martínez, C. L. (2014). *Docs. Google: Introduccióna a App Inventor.* Obtenido de https://docs.google.com

NT, M. (20 de 11 de 2011). *Módulo Bluetooth HC 06*. (NeoTeo) Obtenido de http://www.neoteo.com

Press, W. (s.f.). *La esquina positiva*. (WordPress.com) Obtenido de https://laesquinapositiva.wordpress.com/2013/11/06/los-postres-de-android/

Rubén, J. (s.f.). *Bluetooth HC-05 y HC-06 Tutorial de Configuración*. (Geek Factory) Obtenido de http://www.geekfactory.mx

Technology, M. I. (s.f.). *MIT App Inventor 2*. (massachusetts institute of technology) Obtenido de http://ai2.appinventor.mit.edu/

Wikipedia. (2014). *Wikipedia*. Obtenido de http://es.wikipedia.org/wiki/Arduino

Zapata, I. (2015). Investigación de campo. Latacunga.

ANEXOS