ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

CONSTRUCCIÓN DE MÓDULOS CON BLUETOOTH, MOTORES Y SENSORES UTILIZANDO ARDUINO MEGA PARA EL LABORATORIO DE MICROPROCESADORES DE LA ESFOT

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN ELECTRÓNICA Y TELECOMUNICACIONES

EDUARDO PAÚL CASA CABEZAS
epaulcasa@gmail.com

ANDERSON ROBERTO QUINALOA RAMÍREZ
anderson.quinaloa@epn.edu.ec

DIRECTOR: Ing. FANNY PAULINA FLORES ESTÉVEZ MSc.
fanny.flores@epn.edu.ec

CODIRECTOR: Ing. MÓNICA DE LOURDES VINUEZA RHOR MSc.
monica.vinueza@epn.edu.ec

Quito, noviembre de 2018

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por Eduardo Paúl Casa Cabezas y Anderson Roberto Quinaloa Ramírez, bajo nuestra supervisión.

Ing. Fanny Flores MSc.	Ing. Mónica Vinueza Rhor MSc.
DIRECTOR DE PROYECTO	**CODIRECTOR DEL PROYECTO**

DECLARACIÓN

Nosotros, Eduardo Paúl Casa Cabezas y Anderson Roberto Quinaloa Ramírez, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado en ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

EDUARDO PAÚL CASA CABEZAS	**ANDERSON ROBERTO QUINALOA RAMÍREZ**

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente proyecto apunta a diseñar y construir módulos didácticos basados en plataformas *Arduino* Mega. Estos módulos serán utilizados para renovar el Laboratorio de Microprocesadores y mejorar la formación académica de los estudiantes de la Escuela de Formación de Tecnólogos.

Con este propósito se realizó un análisis de los elementos necesarios que integran un módulo didáctico, para ello se tomaron en cuenta las prácticas de laboratorio anteriores y las prácticas vigentes en la materia de Microprocesadores. Basado en este análisis, se seleccionó una estructura móvil que permita realizar aplicaciones de robótica básica, además de permitir otras aplicaciones con sensores y otros elementos electrónicos.

Se diseñó el módulo didáctico de tal forma que permita alimentación eléctrica a través del computador utilizando cable USB, para aplicaciones que requieran un bajo consumo de corriente. Adicionalmente, alimentación externa a través de baterías para aplicaciones con alto consumo y aplicaciones inalámbricas.

La tarjeta base del módulo didáctico contiene *Driver*s para control de motores DC, *Driver* para control de motor a pasos unipolar o bipolar, diodos emisores de luz, *Buzzer*, borneras, sensores de temperatura, humedad, módulo *Bluetooth*, sensor ultrasónico y conectores *header* que permiten realizar las prácticas de laboratorio propuestas en este documento.

Además, el diseño del módulo didáctico se realizó con la idea de dejar abierto dicho módulo para otras aplicaciones, sin que éste se vea limitado a las que se presentan en este documento.

Finalmente, se han realizado pruebas de funcionamiento de todas las prácticas de laboratorio propuestas, tanto en software como en hardware. Estas pruebas demostraron que el módulo funciona correctamente y cumple con los objetivos planteados.

ABSTRACT

The present project aims to design and build didactic modules on Arduino Mega platforms. These modules will be used to renovate the Microprocessors Laboratory and improve the academic training of the students of the Escuela de Formación de Tecnólogos.

With this objective, an analysis of the required elements that make up a didactic module was carried out, for which the previous laboratory practices and current practices of the Microprocessors subject were considered. Based on this analysis, a mobile structure was selected. This structure allows the deployment of applications based on robotics, as well as other applications with sensors and other electronic elements.

The didactic module was designed in such a way that it allows electric feeding through the computer using the USB cable, for applications that require a low power consumption. Additionally, it is designed for external power through batteries for applications with high consumption and wireless applications. The base card of the didactic module contains controllers for DC motor control, controller for unipolar or bipolar step motor control, light emitting diodes, Buzzer, terminals, temperature sensor, humidity sensor, Bluetooth module, ultrasonic sensor and connectors that allow to perform the laboratory practices proposed in this document.

In addition, the design of the didactic module was made with the idea of leaving the module open for other applications. Finally, functional tests of all laboratory practices have been performed, both in software and in hardware. These tests showed that the module works correctly and meets the purposes outlined.

1. INTRODUCCIÓN

Con el auge de *Arduino*, se ha logrado visualizar la manera de facilitar la implementación de cualquier aplicación mediante varios módulos o interfaces que se encuentran en el mercado. Para la construcción del módulo didáctico, se han tomado en cuenta estas interfaces existentes a manera de minimizar espacio y familiarizar al estudiante con las mismas.

La adquisición de un módulo didáctico que preste características de movilidad y además cubra todas las necesidades relacionadas con sensores, resulta un costo económico muy elevado; razón por la cual se ha considerado implementar interfaces para aplicaciones específicas de motores, comunicación inalámbrica *Bluetooth* y sensores para temperatura, luz y distancia. Adicionalmente, se han considerado elementos indicadores como son: diodos emisores de luz y *Buzzer*.

Al incluir *Visual Studio* dentro las prácticas de este documento se han realizado interfaces de usuario sencillas, las cuales no requieren un conocimiento elevado en lenguaje de programación Basic. Además, el tiempo que lleve la programación de esta interfaz, se verá compensado con el código de *Arduino* y el armado del diagrama establecido para la práctica.

Finalmente, se tiene un manual de prácticas establecidas que abarcarán todos los temas previamente dichos. El manual contendrá una guía para el estudiante y para el profesor; siendo este último una guía paso a paso para lograr implementar la práctica con los resultados deseados, considerando posibles inconvenientes.

El presente proyecto plantea el desarrollo de diez módulos didácticos para actualizar el Laboratorio de Microprocesadores de la Escuela de Formación de Tecnólogos. El proyecto posee herramientas, las cuales permitirán familiarizar al estudiante con nuevas tecnologías y técnicas. Además, se ofrece un rápido aprendizaje y una optimización de tiempo en la elaboración de prácticas de laboratorio, ya que son módulos que permiten flexibilidad y fácil uso. Esto se debe principalmente, a que su unidad principal de control es una placa de desarrollo *Arduino* Mega [1].

El personal docente encargado del Laboratorio de Microprocesadores también se verá beneficiado, dado que el módulo permite una explicación más sencilla de la práctica y una fácil detección de errores, que el alumno pudiera cometer durante la implementación.

Por otra parte, *Arduino* ofrece un entorno de desarrollo, el cual es distribuido gratuitamente por la misma compañía y cuyo lenguaje de programación es de alto nivel, lo que permite una programación relativamente sencilla, rápida y comprensible. Además, permite realizar aplicaciones interactivas autónomas o dependientes de un PC a través de una comunicación serial [2], [3].

Cabe destacar que un módulo didáctico beneficia a los estudiantes, permitiéndoles utilizar y explotar al máximo las herramientas tecnológicas actuales, puesto que hoy en día la tendencia consiste en utilizar sistemas embebidos para el desarrollo de aplicaciones de domótica, robótica y sensores en general, temas que serán abordados en el desarrollo de los módulos propuestos.

Finalmente, el aspecto económico es otro factor importante para el uso de este módulo, ya que constituye un ahorro para los estudiantes al no requerir elementos electrónicos adicionales para su funcionamiento.

1.1. Marco Teórico

A continuación, se procede a describir las partes involucradas en este trabajo.

Introducción a *Arduino*

Arduino es una plataforma Open Source, tanto en hardware y software, la cual presenta facilidades para distintas aplicaciones de entornos interactivos. Al ser una plataforma open source, presenta el soporte para una amplia gama de sensores, actuadores e indicadores que pueden ser fácilmente conectados a la misma.

La placa de una plataforma *Arduino* se basa en un microcontrolador ATMEL, en sus distintas versiones, dependiendo de la plataforma *Arduino* que se disponga. El microcontrolador se programa mediante el uso del *Arduino* Language Programming, el cual se encuentra basado en *Wiring*; y junto con esto, el *Arduino* Development Environment, basado en *Processing* [2].

Las facilidades que brinda *Arduino* al momento de implementar proyectos electrónicos se ven enfocadas principalmente en que estos proyectos pueden ser autónomos o a su vez controlarse desde un ordenador con la utilización de *Processing Maxmsp*. Además, se puede interactuar desde el mismo IDE de *Arduino*.

Las características previamente descritas, hacen de esta plataforma una opción más que viable en la actualidad, al poder acoplar su código y diseño a las necesidades del usuario. Para la realización del presente proyecto se utilizado la plataforma *Arduino* Mega 2560.

IDE *Arduino*

Para el desarrollo e integración del código en la placa, se maneja un entorno interactivo IDE propio de *Arduino*, el cual facilita la manera de escribir un código y subirlo a la plataforma, teniendo una compatibilidad con las distintas versiones de sistemas operativos disponibles en la actualidad como son Mac Os, Windows y Linux. Constituye un software que puede ser utilizado con cualquiera de las plataformas de *Arduino* [4].

Como un entorno de desarrollo interactivo, *Arduino* IDE posee una cantidad impresionante de librerías, las cuales se encuentran disponibles en foros oficiales de *Arduino*; existiendo una librería específica para cada sensor o módulo compatible con *Arduino*. Además, permite la inclusión de hardware hacia la plataforma *Arduino*, de manera que este hardware funcione sobre *Arduino* o poder programarlo desde el IDE.

Finalmente, se presenta un monitor serie, el cual permite enviar y recibir datos de la plataforma *Arduino* sin la necesidad de utilizar algún software o elemento de hardware adicional; únicamente, mediante la conexión del cable USB [5].

***Arduino* Mega 2560**

Esta versión de la plataforma *Arduino* presenta un microcontrolador Atmega2560. Además, tiene 54 pines, los cuales pueden ser usados como entradas o salidas digitales; de estos a su vez, 14 permiten realizar aplicaciones con PWM (*Pulse*

Width Modulation). A estos 54 pines, se suman 16 que funcionan como entradas análogas, 4 UARTS, un oscilador de 16MHz, conector USB, Jack de alimentación, conector ICSP, como se muestra en la Figura 1.1, de manera que se integran todos estos componentes en una plataforma de 101.52x53.3 mm [6] [7].

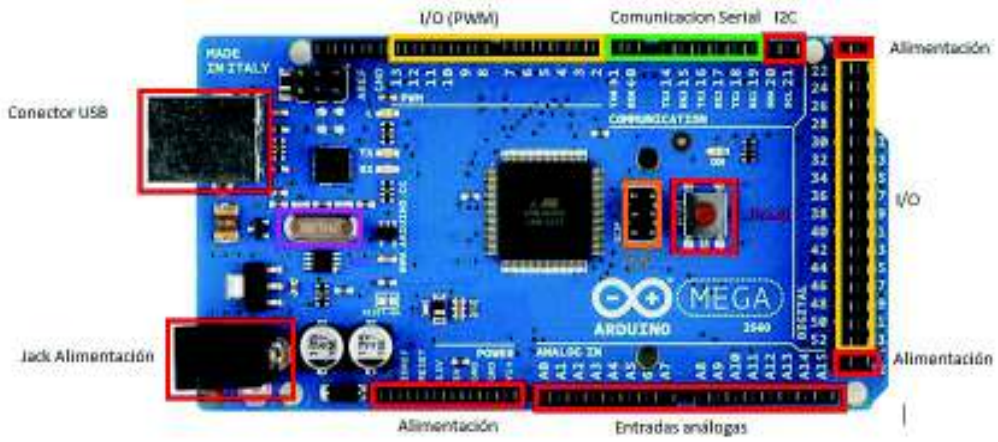


Figura 1.1: Distribución de pines de *Arduino* Mega 2560 [7].

En lo que concierne a características operacionales para el *Arduino* Mega 2560, se tienen las siguientes:

- Voltaje de operación 5V
- Voltaje de entrada recomendado 7-12V
- Voltaje de entrada límite 6-20V
- Corriente DC por pin entrada/salida 20mA
- Corriente DC para pin de 3.3V 50mA
- Memoria Flash 256KB
- SRAM 8KB
- EEPROM 4KB

Para conocer más parámetros de hardware y software correspondientes a *Arduino* Mega se adjunta la hoja técnica en el Anexo C-1.

Comunicación serie

Los puertos serie solo requieren de un único conector para poder enviar un caracter de datos hacia un dispositivo. Los datos del sistema transmisor están dispuestos en un formato paralelo que, para ser enviados, se convierten en un formato serie en la que los bits se encuentran organizados uno tras de otro. Los bits se transmitirán al dispositivo receptor, de tal manera que el bit menos significativo llegará primero [8].

Para este tipo de comunicación, es importante que tanto emisor como receptor, se encuentren sincronizados.

Con *Arduino*, al acoplar el conector USB al computador, se establecerá la comunicación serie con la plataforma. Además, es importante considerar que los pines asociados al USB están en las posiciones 0 y 1 de la plataforma *Arduino*, por lo que son pines destinados, los cuales no pueden utilizarse como entradas o salidas digitales [9].

Como se aprecia en la Figura 1.1, el *Arduino* Mega posee puertos adicionales en los pines 19-18 (RX-TX), 17-16 (RX-TX) y 15-14 (RX-TX), los cuales no están conectados a la interfaz USB del *Arduino*.

Comunicación *Bluetooth* con *Arduino*

Bluetooth representa una especificación industrial para redes inalámbricas de área personal o también llamadas (WPAN), la cual permite la transmisión de voz y datos entre distintos dispositivos utilizando un enlace de radiofrecuencia a 2.4Ghz o también llamada banda libre (ver Anexo A-4). Dentro de los beneficios que se consiguen al implementar comunicación mediante *Bluetooth* se tienen:

- Facilitar las comunicaciones entre equipos móviles.
- Eliminar los cables y conectores entre dispositivos.
- Facilitar sincronización de datos entre equipos móviles personales, en forma de una red inalámbrica pequeña, como se muestra en la Figura 1.2.

Bluetooth presenta la característica de salto de frecuencia adaptativa o también llamada AFH (*Adaptative Frequency Hopping*), lo que le permite detectar otras tecnologías inalámbricas dentro de su alcance con el fin de evitar la interferencia, haciendo que el canal se registre en una lista negra temporal de canales inutilizables. Sin embargo, la conexión se vuelve a intentar para determinar si la interferencia ha cesado y así considerar este canal nuevamente utilizable o no.

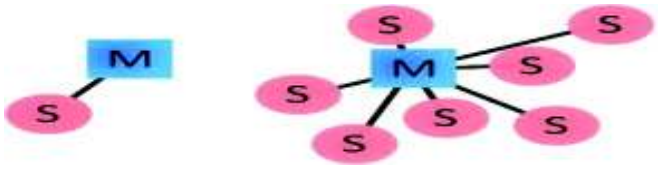


Figura 1.2: Red inalámbrica con maestros y esclavos [10].

Dentro de *Arduino* se cuenta con módulos dedicados para comunicación *Bluetooth*, los cuales son el módulo Hc-05 y Hc-06. Cabe recalcar que la principal diferencia entre estos dos módulos radica en el rol que pueden desempeñar (maestro o esclavo). El módulo HC-05 puede ser configurado tanto como maestro o esclavo, por su lado, el HC-06 únicamente puede usarse como esclavo.

Los dispositivos *Bluetooth* se pueden clasificar teniendo en cuenta la Potencia de Transmisión, siendo estos de Clase 1, 2 o 3 y la Capacidad de Canal (ver tabla 1.1).

Tabla 1.1: Clasificación por Capacidad de Canal [10].

VERSIÓN DE BLUETOOTH	ANCHO DE BANDA
Versión 1.2	1 Mbps
Versión 2.0 + EDR	3 Mbps
Versión 3.0 + HS	24 Mbps
Versión 4.0	32 Mbps

EDR: Mayor velocidad de datos (*Enhanced Data Rate*).

HS: Incorporación de enlace 802.11 de alta velocidad de Transferencia de datos.

Módulo HC-05

El módulo *Bluetooth* HC 05 maneja el protocolo base de *Bluetooth* 802.15 con una velocidad de 3Mbps siendo este un *Bluetooth* V2 (ver Tabla 1.1). Puede ser configurado como maestro o esclavo, según la aplicación [11].

Tabla 1.2: Comandos AT [16]

LISTA DE COMANDOS AT	
COMANDO	**FUNCIÓN**
AT	Test de conexión UART
AT+RESET	Reiniciar configuración de dispositivo
AT+ORGL	Restaurar configuración por defecto
AT+NAME	Conocer / Colocar nombre
AT+ROLE	Conocer / Colocar rol
AT+IAC	Conocer / Colocar código de acceso
AT+INQM	Conocer / Colocar modo de acceso
AT+PSWDAT+PIN	Conocer / Colocar contraseña de emparejamiento
AT+UART	Conocer / Colocar parámetro UART
AT+CMODE	Conocer / Colocar modo de conexión
AT+BIND	Conocer / Colocar dirección de vinculación *Bluetooth*
AT+POLAR	Conocer / Colocar polaridad de salida *LED*
AT+PIO	Colocar / Retirar pin I/O de un usuario
AT+ADDR	Conocer dirección de dispositivo *Bluetooth*

Su parte física se encuentra constituida por dos partes esenciales las cuales son:

- Dispositivo de radio: El cual se encarga de modular y transmitir la señal
- Controlador Digital: Se encuentra compuesto por un procesador de señales digitales o también llamadas DSP (*Digital Signal Processor*), el cual se encarga del procesamiento de la banda base y junto con esto el manejo de los protocolos ARQ y FEC.

La manera de identificar el módulo HC-05 es por sus 6 pines que se diferencian del HC- 06 con únicamente 4; de manera que sus pines quedan identificados como se muestra en la Figura 1.3.

Dentro de sus especificaciones técnicas se tiene [12]:

- Voltaje de operación: 4V– 6V
- Corriente de operación: 30mA
- Rango: menor a 100m
- Soporta Comunicación Serial (USART) y TTL
- Tasa de Baudios soportados: 9600, 19200, 38400, 57600, 115200, 230400, 460800
- Maneja FHSS (*Frecuency – Hopping Spread Spectrum*)

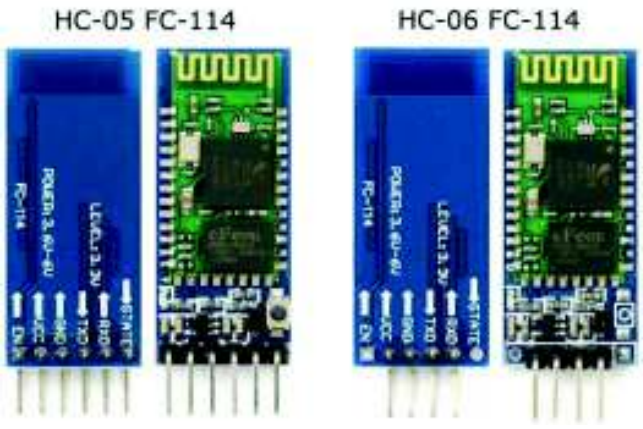


Figura 1.3 Módulos *Bluetooth* HC-05 y HC-06 [13].

La configuración de los parámetros (velocidad de comunicación, nombre, rol, pin) del Módulo HC-05, se debe realizar mediante la utilización de comandos AT (ver Anexo A-1). En la Tabla 1.2 se presentan los comandos más utilizados.

Sensor HC SR 04

El sensor HC SR04 es un sensor ultrasónico que mide distancia mediante el uso de ondas ultrasónicas las cuales emite a través de su cabezal y lo que recepta es la onda reflejada desde el objeto, teniendo así una medida basada en el tiempo que se da entre la emisión y la recepción de la onda, como se muestra en la Fórmula 1.

$$L = ½ * T * C \qquad (1)$$

Donde:

L: es la distancia
T: El tiempo entre la emisión y la recepción
C: Velocidad del sonido

En la Figura 1.4 se presenta el sensor HC SR04, el cual ha sido implementado para realizar el proyecto, y tiene las siguientes características:

- Voltaje de operación de 5V DC
- Corriente de operación de 15mA
- Frecuencia de trabajo de 40Hz
- 4m como rango máximo de medida
- 2cm como rango mínimo de medida
- Ángulo de medición establecido en 15 grados
- Señal de entrada de disparo 10us pulso TTL
- Dimensiones establecidas en 45x20x15 mm

Figura 1.4: Sensor ultrasónico HC-SR04 [14].

Para conocer más especificaciones técnicas sobre este sensor revisar el Anexo C-7.

Sensor DHT11

El sensor DHT11, ha sido implementado para aplicaciones de temperatura y humedad, ya que incluye una medición de humedad de carácter resistivo y una temperatura basada en un modelo NTC.

En la Figura 1.5 se muestra la asignación de pines del sensor DHT11

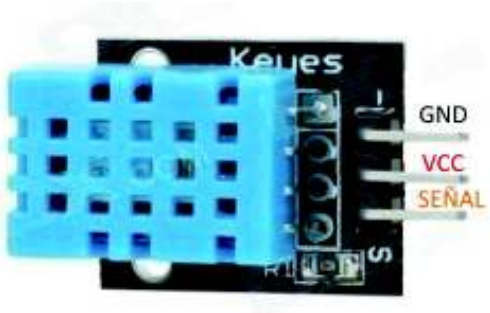


Figura 1.5: Sensor DHT11 [15].

Para más especificaciones técnicas referentes al sensor revisar el Anexo C-5.

LDR

Dentro del esquema, se ha considerado colocar LDRs como sensores de luz, los cuales se encuentran conectados de manera que formen un divisor de tensión, como se muestra en la Figura 1.6 [16].

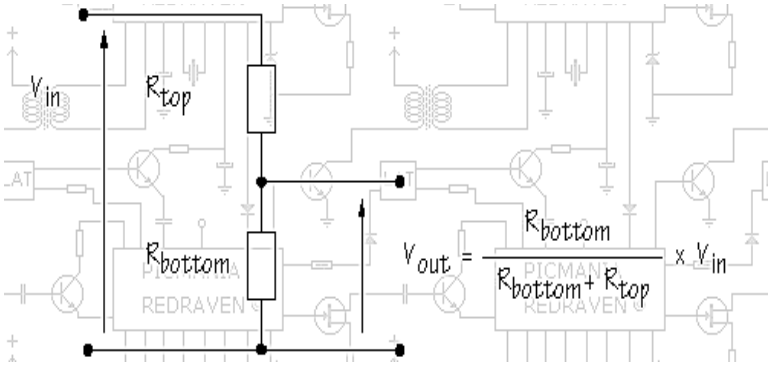


Figura 1.6: Divisor de Tensión LDR [16].

Donde:

V_{in}: Voltaje igual a 5V

V_{out}: Señal análoga de salida que será enviada al Arduino Mega

R_{top}: Fotocelda o LDR

R_{bottom}: Resistencia de 10 kΩ

Se tiene una resistencia de 10kΩ y un voltaje de 5V, por lo que el LDR quedará en la posición superior. Los datos ingresados han sido tratados según un código de ejemplo, disponible en foros de *Arduino*, como se podrá observar en el Anexo B-2.

Motor DC [17]

El motor DC, o también llamado motor de corriente continua, es de los más comunes y económicos que se encuentran en el mercado. Se constituye de dos imanes fijos a la carcasa y un bobinado de cobre que se encuentra en el eje del motor. Su funcionamiento radica en la conversión de energía eléctrica en mecánica, lo que conlleva a un movimiento rotatorio; al ser un elemento actuador, éste requiere de un controlador o *Driver*, que para esta aplicación se ha optado por el *Driver* L293D (ver Anexo C-2).

El motor seleccionado presenta la característica de caja reductora, la cual permite tener más fuerza de tracción, como se muestra en la Figura 1.7.

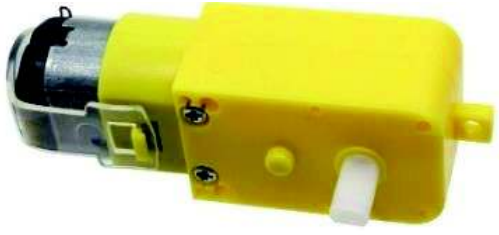


Figura 1.7: Motor DC con caja reductora [18].

Dentro de sus principales características se encuentran:

- Voltaje de Operación de 3-6V
- Velocidad Angular nominal: 125 RPM
- Reducción: 48:1
- Consumo máximo de corriente: 150mA
- Peso 50g

Adicional a estas características se adjunta las especificaciones técnicas de la caja reductora a distintos voltajes de alimentación [18] (ver Tabla 1.3).

Tabla 1.3: Especificaciones técnicas motor con caja reductora [18].

PARÁMETROS	DC 3V	DC 5V	DC 6V
Reducción	48:1		
Velocidad sin carga	125RPM	200RPM	230RPM
Velocidad con carga	95RPM	152RPM	175RPM
Torque de salida	0.8kgxcm	1.0kgxcm	1.1kgxcm
Velocidad del robot sin carga (metros/minuto)	25.9	41.4	47.7
Corriente	110-130mA	120-140mA	130-150mA
Diámetro máximo de la llanta	6.5cm		
Dimensiones	70mmx 22mmx 18mm		
Peso	50g		
Ruido	<65dB		

Motor a pasos

Un motor a pasos tiene cierto parecido a los motores DC; sin embargo, es fácil reconocerlo debido a que presenta un número de cables superior a un motor DC, los cuales pueden ser 5 o 6. Además, un motor a pasos permite tener un giro controlado, ya que posee un número mayor de bobinas para activar, lo cual requiere una secuencia establecida para su giro correcto. El motor a pasos a utilizar es un motor 28byj-48, como se muestra en la Figura 1.8, el cual podrá funcionar como bipolar o unipolar [19].

Figura 1.8: Motor a pasos 28bjy-48 [19].

Para el motor a pasos, se ha optado por la utilización de su *Driver* basado en el integrado ULN2003, el cual viene integrado con diodos *LED* y conectores *header* para su fácil conexión, como se muestra en la Figura 1.9.

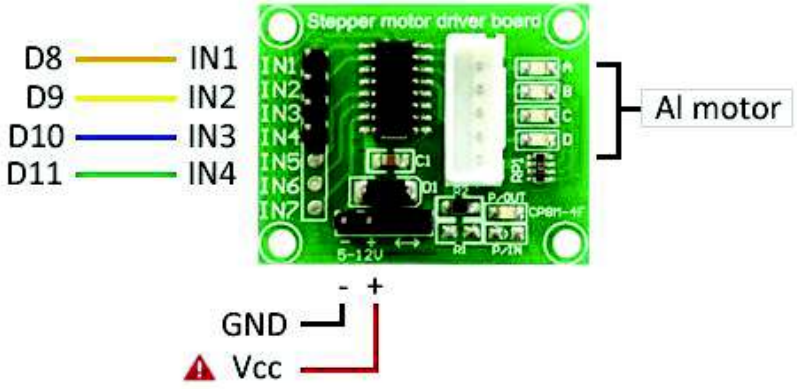


Figura 1.9: Driver ULN2003 [19].

Servomotor

Un servomotor es un tipo de motor especial que posee características de control de posición, basada en ángulos. Al decir que es un servomotor, se hace referencia a

que se encuentra compuesto por componentes electromecánicos y electrónicos, como se muestra en la Figura 1.10 [20].

Figura 1.10: Componentes de un servomotor [20].

Para la construcción del módulo se ha utilizado un servomotor SG90, el cual presenta dimensiones pequeñas con un torque de 1.8kgfxcm, con un ángulo de giro de 0 a 180 grados y con un peso de 9g. Además, su voltaje de operación es de aproximadamente 5V [21].

Para más información sobre especificaciones técnicas del servomotor ver Anexo - C-6.

Batería LiPo

Las baterías LiPo se caracterizan por ser una poderosa opción en almacenamiento y suministro de energía eléctrica para proyectos relacionados a la electrónica. Presenta la capacidad de almacenar altas densidades de energía ya que se encuentra compuesta por polímero de iones de litio, de ahí su denominación [22].

Dentro de las especificaciones que presenta una batería LiPo, se tienen las siguientes:

- Capacidad de la batería
- Velocidad de descarga
- Número de celdas y voltaje

El factor más importante es conocer la velocidad de descarga, ya que éste nos permite saber cuánta corriente producirá la batería LiPo. La Fórmula 2 para calcular la corriente máxima se expresa por:

$$C \times mAh \qquad (2)$$

C: velocidad de descarga

mAh: capacidad de la batería

La batería LiPo utilizada para el presente proyecto es una de 7.4v 2s de 300mAH 35-70c, por lo que se tiene una corriente máxima de:

$$70x300mAH = 2100mAH$$

Para conocer aspectos sobre el cuidado y correcto uso de las baterías LiPo, se puede visitar el portal web del fabricante, en este caso Hobby Kings.

2. METODOLOGÍA

2.1. Metodología Exploratoria

Se ha manejado metodología exploratoria debido a que actualmente en la Escuela de Formación de Tecnólogos no se brinda un curso específico para manejo de plataformas *Arduino* y lo que este abarca, tendiendo a ser un tema poco estudiado y desconocido para quienes realizaron este proyecto.

Se consideraron características que poseen las distintas plataformas *Arduino* que se encuentran dentro del mercado para así poderlas comparar con *Arduino* Mega con el fin de determinar las ventajas que esta presenta. Las características de la plataforma han sido las que han permitido establecer un diseño para la tarjeta principal, ya que, al poseer una cantidad alta de puertos para entradas y salidas digitales, permiten que el usuario no tenga complicación al momento de añadir distintas interfaces o indicadores a la plataforma *Arduino* Mega.

Dentro de las características de funcionamiento, se optó por la alimentación cableada para las aplicaciones que no requerían movilidad ya que, por los temas

revisados, *Arduino* permite alimentar y a su vez comunicarse con el computador mediante su cable USB, por lo que no se tendrá alimentación externa para aplicaciones que no representen un alto consumo de corriente.

2.2. Metodología Aplicada

Para la realización del presente proyecto, se realizó una previa investigación con respecto a la manera de construir el módulo didáctico sin que esto resulte ser complicado de manejar para el estudiante y las herramientas que debería contener para que sea un complemento al conocimiento teórico que se brindará en las clases. Se tomaron en cuenta aplicaciones realizadas previamente en materias relacionadas con microprocesadores, acoplándolas a esta nueva plataforma.

Dentro de las aplicaciones para el módulo, se poseen sensores, controladores para motores y un módulo de comunicación inalámbrica mediante tecnología *Bluetooth*; las cuales podrán ser combinadas de distintas maneras con el fin de obtener cualquier aplicación que el estudiante crea conveniente realizar.

3. RESULTADOS Y DISCUSIÓN

3.1. Descripción General del Proyecto

Según el objetivo planteado en este proyecto, se ha diseñado, construido y replicado un módulo que permite al estudiante aprender de forma didáctica sobre el manejo de actuadores, sensores y otros dispositivos electrónicos utilizando un microcontrolador.

El módulo utiliza una placa *Arduino* Mega como unidad de control cuyas características técnicas son suficientes para soportar y manejar el hardware y software que serán utilizados en el desarrollo de las prácticas de laboratorio.

Las prácticas de laboratorio que se proponen se basan en diferentes temas como son: control de motores, uso de sensores, comunicación inalámbrica y desarrollo de interfaces de usuario. Lo que permite al estudiante involucrarse con nuevos temas y reforzar los ya aprendidos.

3.2. Diseño del Módulo Didáctico

Diagrama esquemático del Módulo

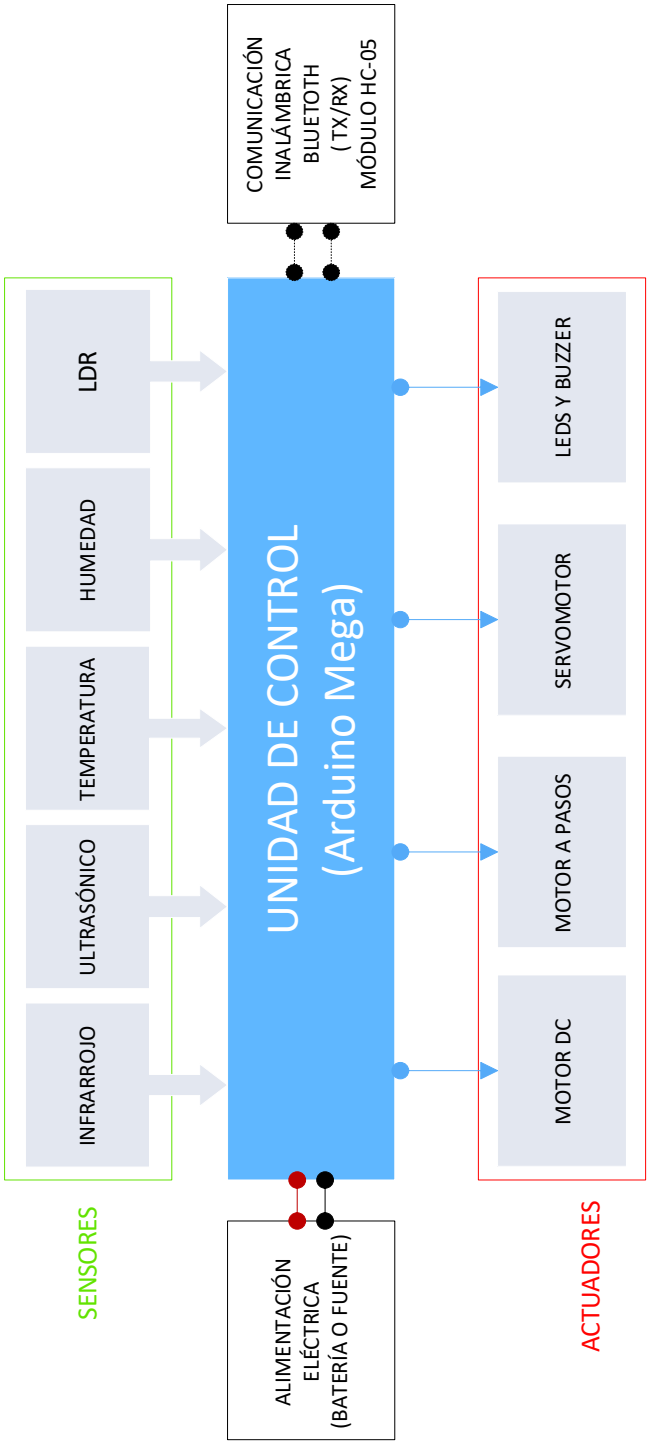


Figura 3.1: Diagrama Esquemático del Módulo

El módulo se encuentra constituido por cuatro partes importantes las cuales convergen en la unidad de control como se muestra en la Figura 3.1. Los sensores serán los encargados de enviar información hacia la unidad de control en base a su funcionamiento, es decir, dependiendo del tipo de sensor que se maneje, la información que llegue a la unidad de control varia. Dentro de los elementos actuadores o señaladores, serán quienes ejecuten tareas enviadas directamente desde la unidad de control.

La comunicación inalámbrica sea de una comunicación doble, ya que se enviarán y a su vez recibirán datos a través de esta etapa a cargo de un módulo de tecnología *Bluetooth*. Finalmente, la etapa de alimentación podrá ser cableada desde el computador, o a su vez una batería que permita el buen rendimiento de las aplicaciones dentro del módulo.

Diseño del circuito impreso del módulo

Para el diseño del circuito impreso del módulo se han considerado ciertos elementos que son necesarios para el desarrollo de las prácticas de laboratorio. Estos elementos, incluidos en el módulo, se denominan *Driver*s o controladores, y permiten el control de motores. Los *Driver*s incluidos en el módulo son los siguientes:

- *Driver* para motores DC (*Driver* L293D)
- *Driver* para motor a pasos bipolar (*Driver* L293D)
- *Driver* para motor a pasos unipolar (*Driver* ULN2003)

Además, el módulo didáctico posee otros elementos como: *LED*s, *Buzzer*, borneras y *header*. Los *header* serán utilizados para alimentación, conexión de sensores y conexión de otros dispositivos electrónicos.

Diagrama circuital *Driver* L293D

En la Figura 3.2 se muestra el diagrama circuital del *Driver* L293D que se utilizará para el control de motores DC y motores a pasos bipolar, los cuales irán conectados a sus salidas. El módulo didáctico tiene incluido dos *Driver*s en su placa, que reciben señal desde la unidad de control, cuya información puede interpretarse con

sentido de giro (IN1,2,3,4), y control de velocidad (EN1,2) en sus pines de entrada respectivamente.

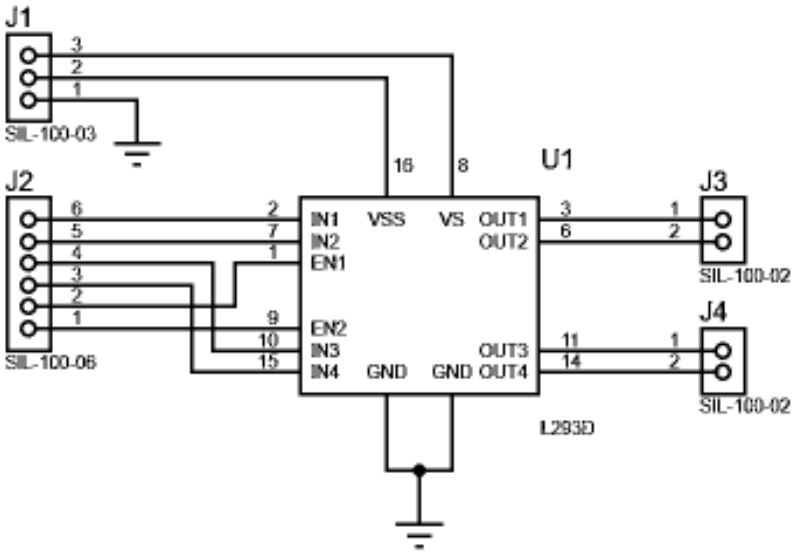


Figura 3.2: Diagrama circuital *Driver* L293D

Módulo *Driver* ULN2003

El *Driver* ULN2003 viene incluido con el motor 28BYJ-48, que es un motor a pasos unipolar. Por lo tanto, se utilizará dicho *Driver* para el control del motor. En la Figura 3.3 se puede observar el diseño de fábrica, que tiene dicho *Driver*. Este *Driver* recibirá en sus entradas las secuencias para poder realizar los giros correspondientes para motores a pasos unipolares, lo cual se ejecutará en sus salidas activando las bobinas del motor en cuestión.

Figura 3.3: *Driver* ULN2003 para controlar un motor a pasos unipolar [19].

Diagrama circuital de diodos *LED*

Como se muestra en la Figura 3.4, cada *LED* se conectó a una resistencia de 330 ohms para limitar la corriente que cruza por ellos. Además, el diagrama está compuesto por *header*s para poder ser cableados con la unidad de control o ser alimentados directamente.

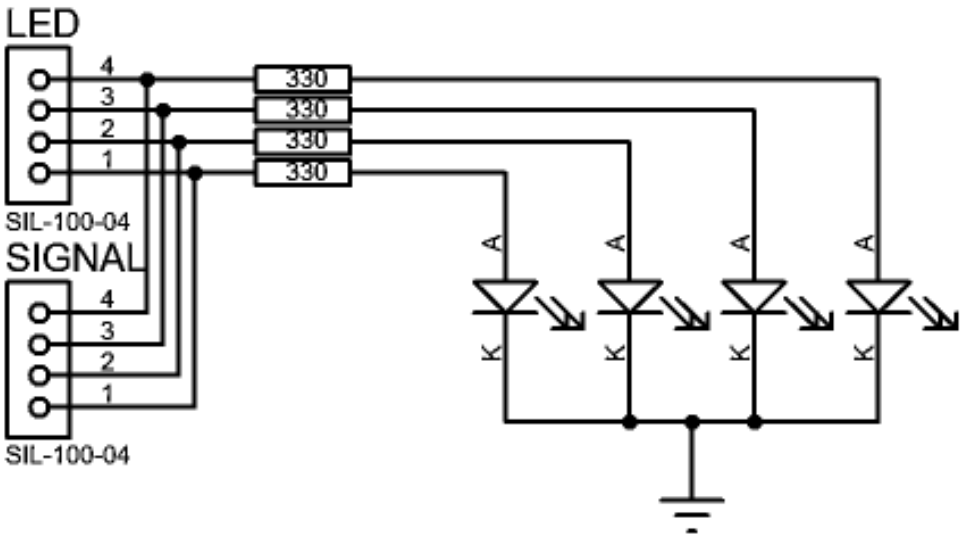


Figura 3.4: Diagrama de conexiones diodos *LED*

Diagrama circuital *Buzzer*

Para la conexión del *Buzzer*, se realizó un amplificador de corriente con un transistor 2N3904, el cual recibirá alimentación y señal por parte de la unidad de control en el conector *header*, como se muestra en la Figura 3.5.

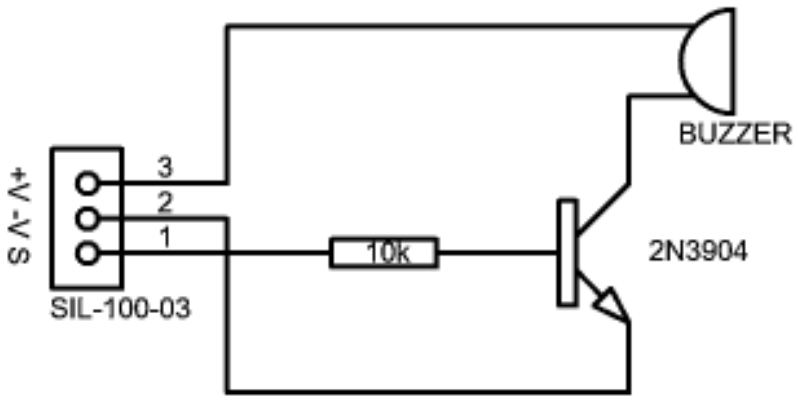


Figura 3.5: Diagrama circuital *Buzzer*

Diagrama circuital fotocelda o LDR

El diagrama circuital realizado con el LDR se basa en un divisor de tensión con una resistencia de 10 kΩ, lo cual permite tener las variaciones de voltaje conforme la resistencia de la LDR cambie sus valores ante la presencia o ausencia de luz, como se muestra en la Figura 3.6. Dichas variaciones serán enviadas hacia un conversor A/D de la unidad de control para poder tratar los datos dentro del código.

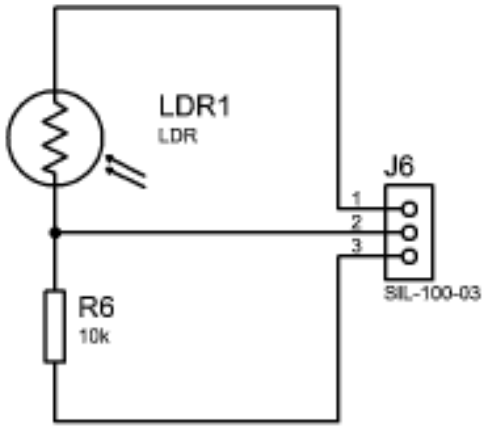


Figura 3.6: Diagrama circuital LDR

Diagrama circuital de sensores, módulo *Bluetooth* y servomotor.

Para la conexión de estos dispositivos se ha utilizado solamente *header*s ya que estos están distribuidos a lo largo del módulo, por lo tanto, tienen *header* para la conexión entre el dispositivo y la placa, y *header* para la conexión del dispositivo con el microcontrolador. En la Figura 3.7 se puede observar el diagrama realizado para estos dispositivos.

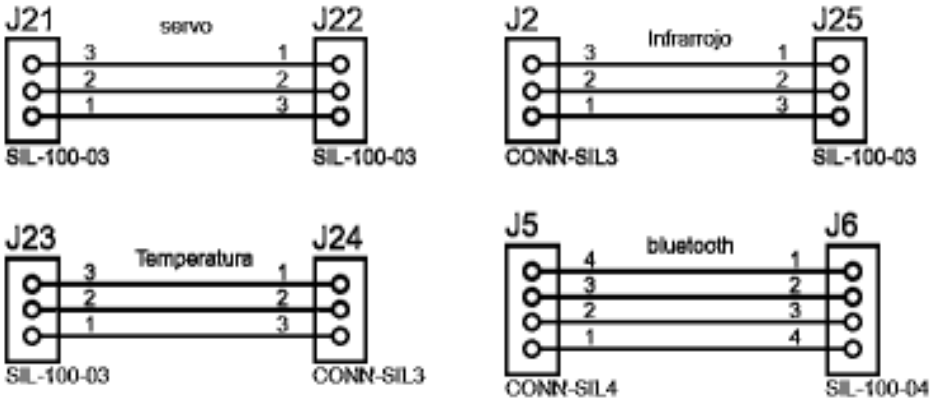


Figura 3.7: Diagramas circuitales para conexión de sensores y otros dispositivos

Diagrama circuital completo del Módulo

Para completar la tarjeta de control, se han agrupado todos los circuitos anteriormente mostrados, con el fin de integrarlos en una sola, en la cual se deberá incluir la unidad de control, es decir, el *Arduino* Mega 2560 y el *Driver* ULN2003 con el fin de complementar la tarjeta en los espacios marcados del diagrama, como se muestra en la Figura 3.8.

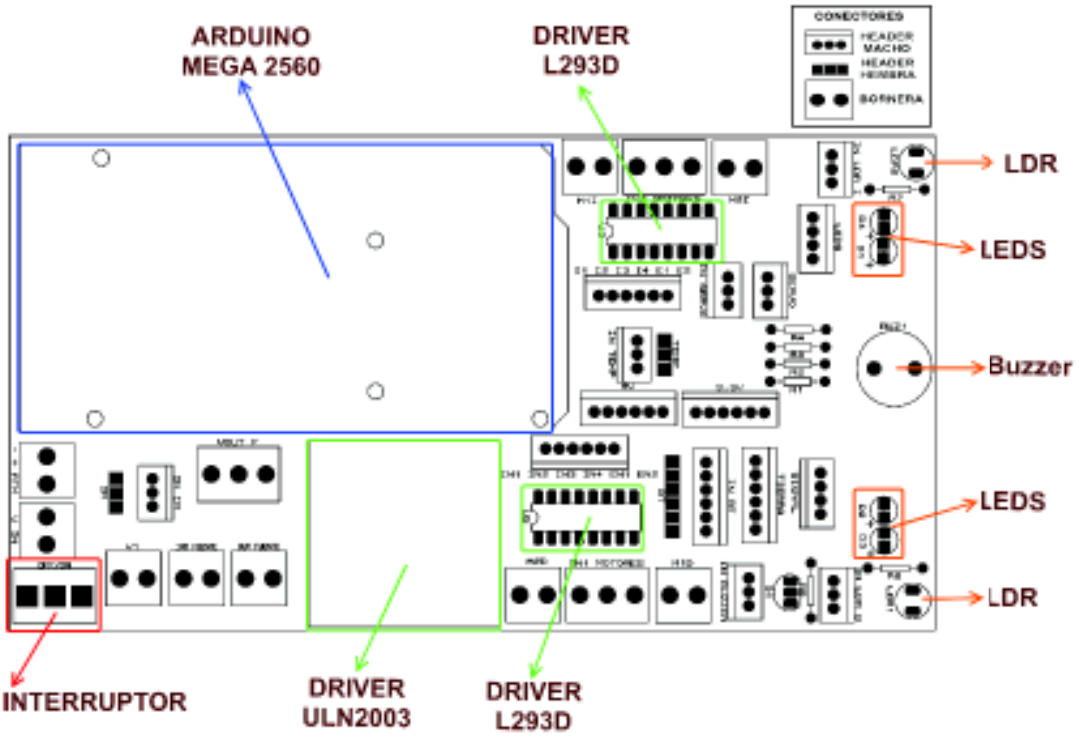
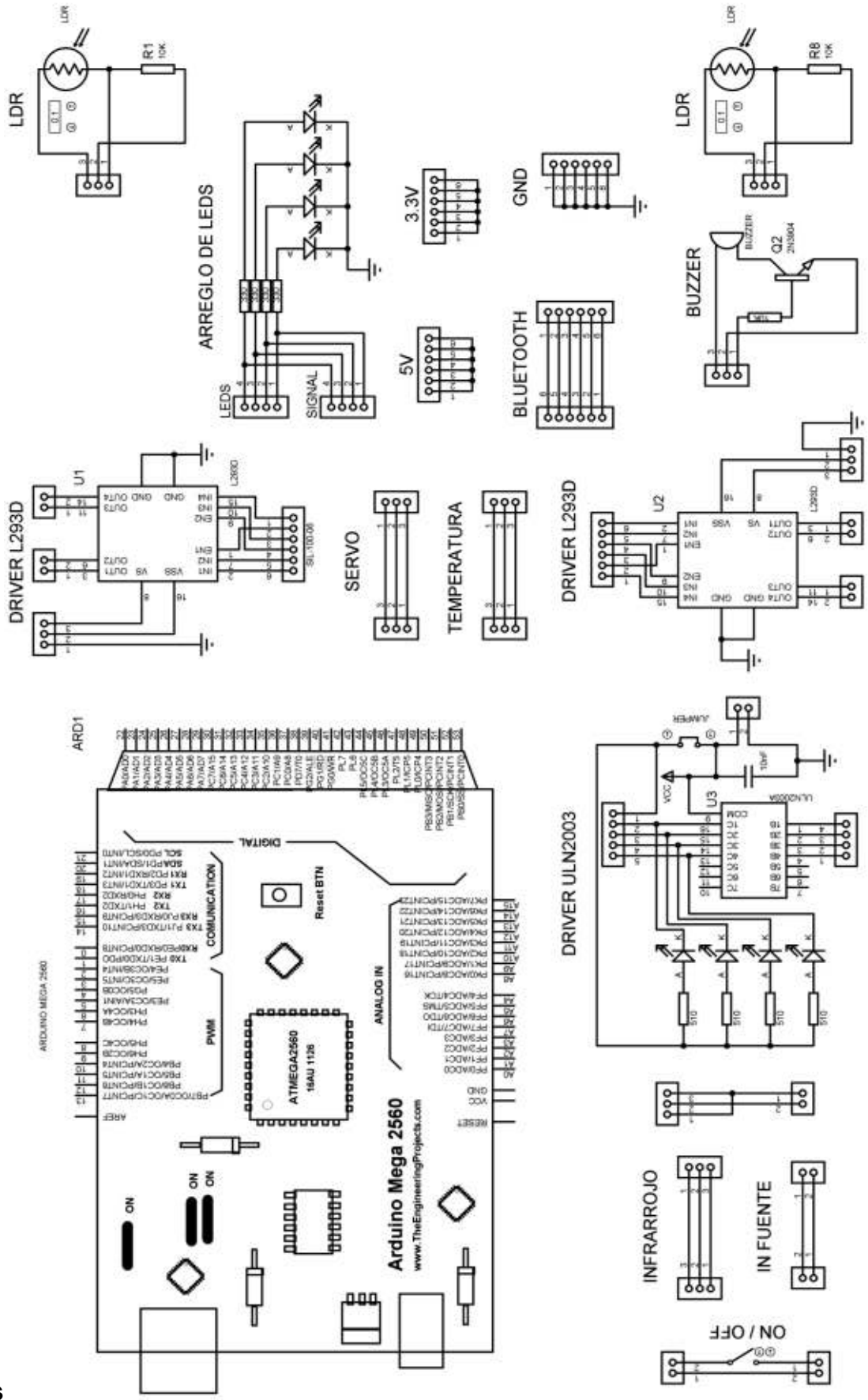


Figura 3.8: Distribución de *Driver*s y otros elementos en la tarjeta

Se puede observar que los conectores *header* destinados a la alimentación se encuentran puenteados con el fin de ser un solo punto del que puedan salir 5 cables para distintos componentes del módulo, sirviendo, así como una regleta de alimentación proveniente de la Unidad de Control.

También se agregó un interruptor para la activación de la alimentación externa; y a su vez, se colocaron borneras, como se muestra en la Figura 3.9, con el fin de aportar a posibles cambios de la tarjeta, ya sea la incorporación de una fuente step-up para el módulo o posible uso de otro elemento que requiera un voltaje y corriente superior al que proporciona la batería entregada.

Dejando finalmente una tarjeta, cuyos elementos se conectarán a la Unidad de Control conforme se lo requiera.


LDR
R1
10K
ARREGLO DE LEDS
LEDS
SIGNAL
3.3V
5V
GND
BLUETOOTH
LDR
R8
10K
BUZZER
Q2
2N3904
DRIVER L293D
U1
L293D
SERVO
TEMPERATURA
DRIVER L293D
U2
L293D
DRIVER ULN2003
U3
ULN2003A
INFRARROJO
IN FUENTE
ON / OFF
ARD1
ATMEGA2560
16AU 1126
Arduino Mega 2560
www.TheEngineeringProjects.com
DIGITAL
ANALOG IN
COMMUNICATION
PWM
Reset BTN
ON

s

Figura 3.9: Diagrama circuital completo

En la Figura 3.10, se observa el circuito impreso de la tarjeta, el diagrama fue realizado con el software de diseño Proteus.

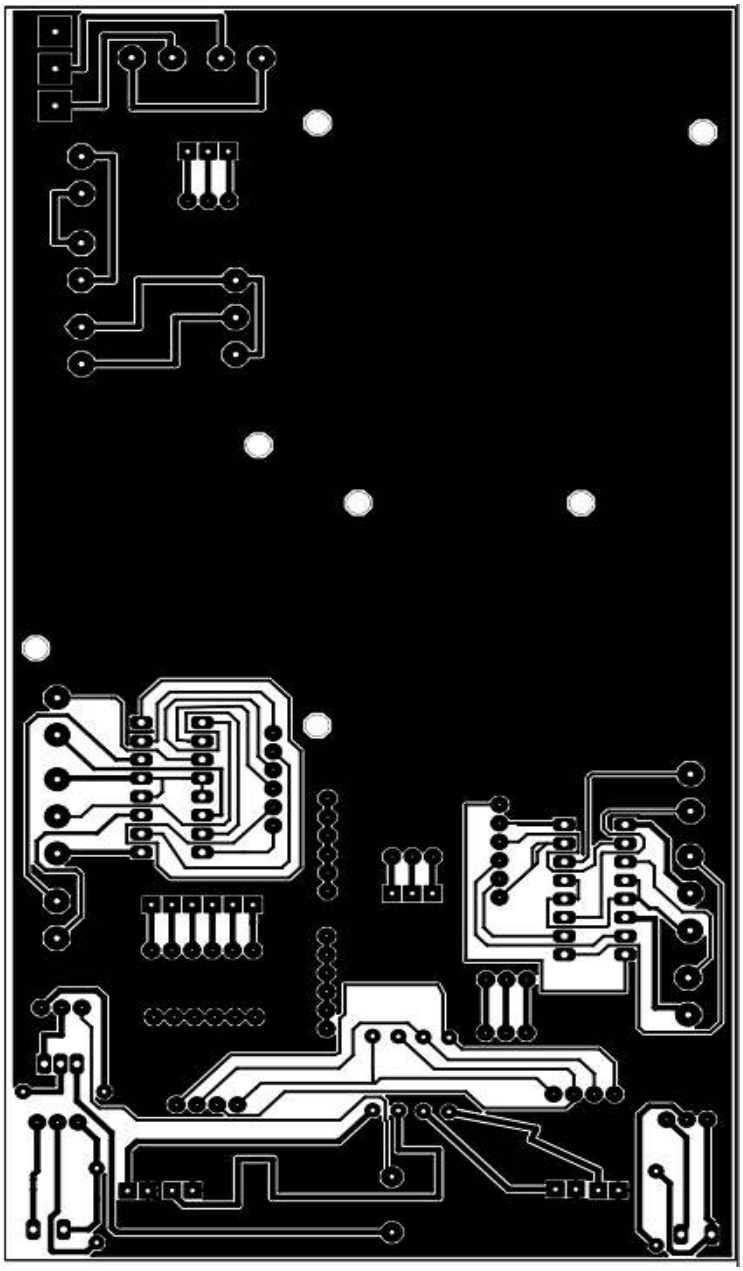


Figura 3.10: Circuito impreso realizado en Proteus

3.3. Construcción del Módulo Didáctico

Una vez terminado el diseño del circuito impreso a través de software, se procede con la fabricación de la tarjeta. A continuación, se describe el procedimiento para la fabricación de dicha placa electrónica.

Fabricación de placa electrónica

Para la elaboración de la placa del módulo, se utilizó papel termotransferible para estampar el circuito sobre el cobre de la baquelita; seguidamente, se utilizó cloruro férrico para el grabado químico. En la Figura 3.11, se muestra una fotografía con dos baquelitas; la baquelita superior tiene el circuito impreso estampado y la inferior una placa después de haber sido tratada con cloruro férrico.

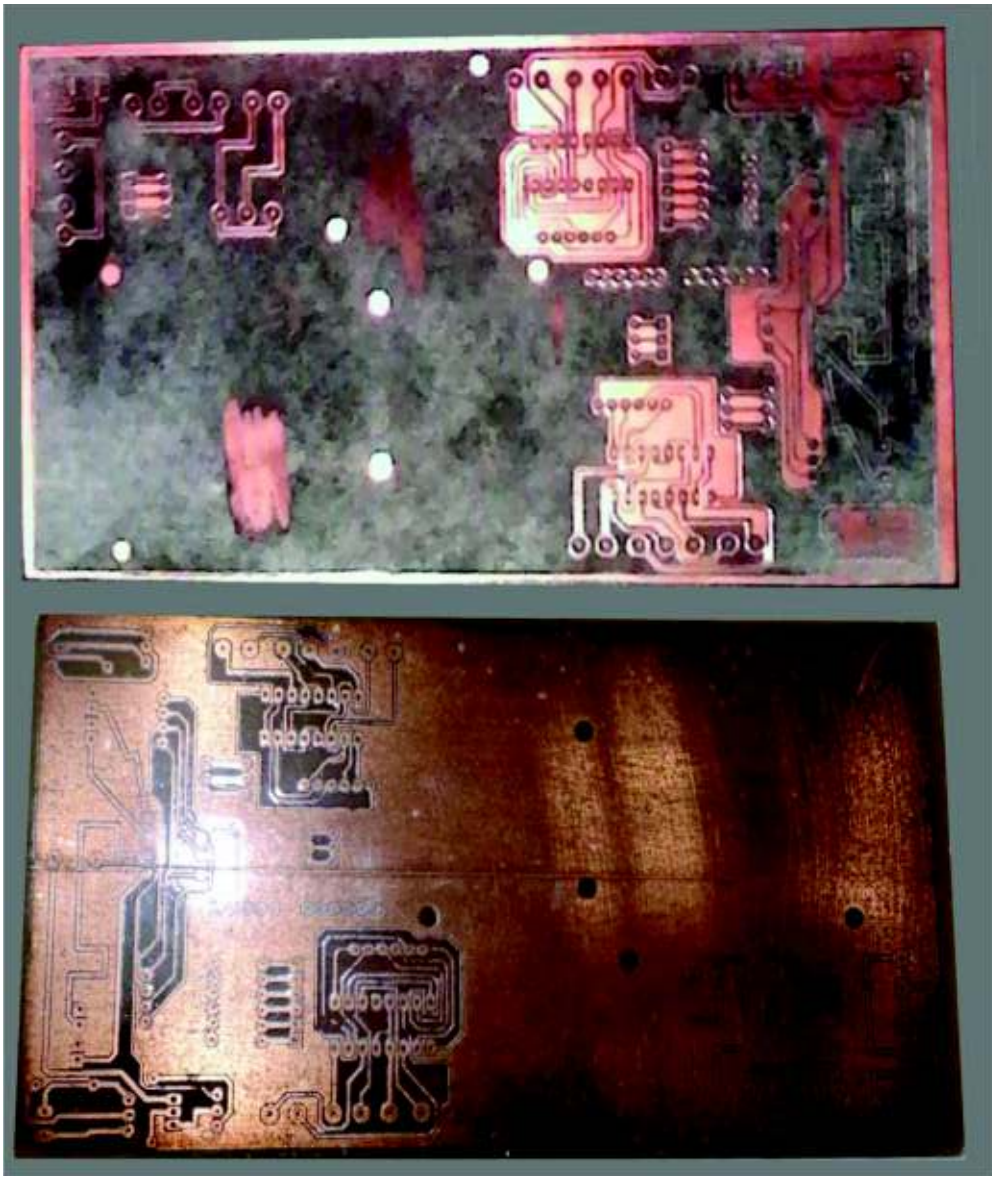


Figura 3.11 Circuito impreso antes y después de aplicar el químico

La transferencia de la máscara de elementos, en la parte superior de la tarjeta, se la realizó de igual manera utilizando papel termotransferible. El acabado final se puede observar en la Figura 3.12.

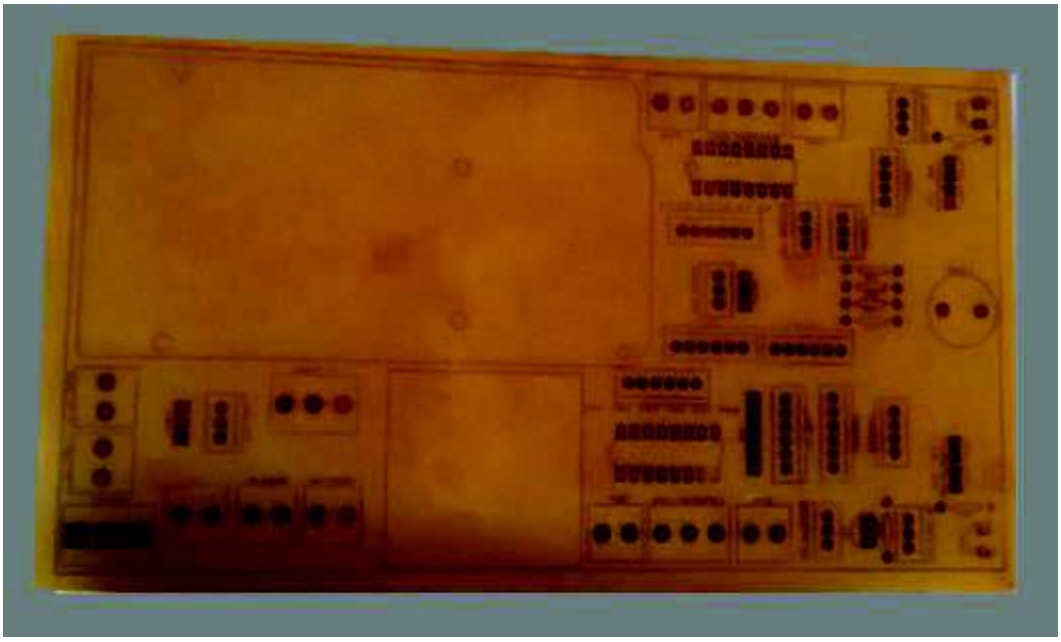


Figura 3.12: Baquelita con máscara de elementos

A continuación, se procede a soldar los elementos en su posición correspondiente siguiendo los diagramas realizados para el diseño del circuito impreso. La placa con sus elementos soldados se muestra en la Figura 3.13.

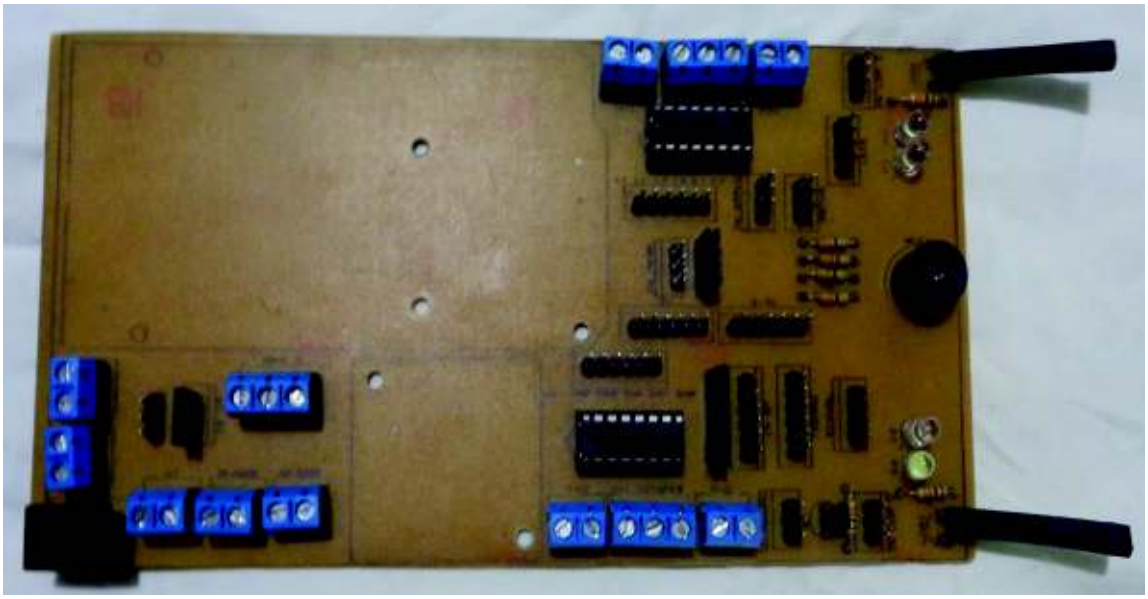


Figura 3.13: Tarjeta con elementos soldados

Finalmente, se procede a ensamblar el *Arduino* Mega y el *Driver* para motor a pasos unipolar utilizando pernos y tuercas, como se muestra en la Figura 3.14.

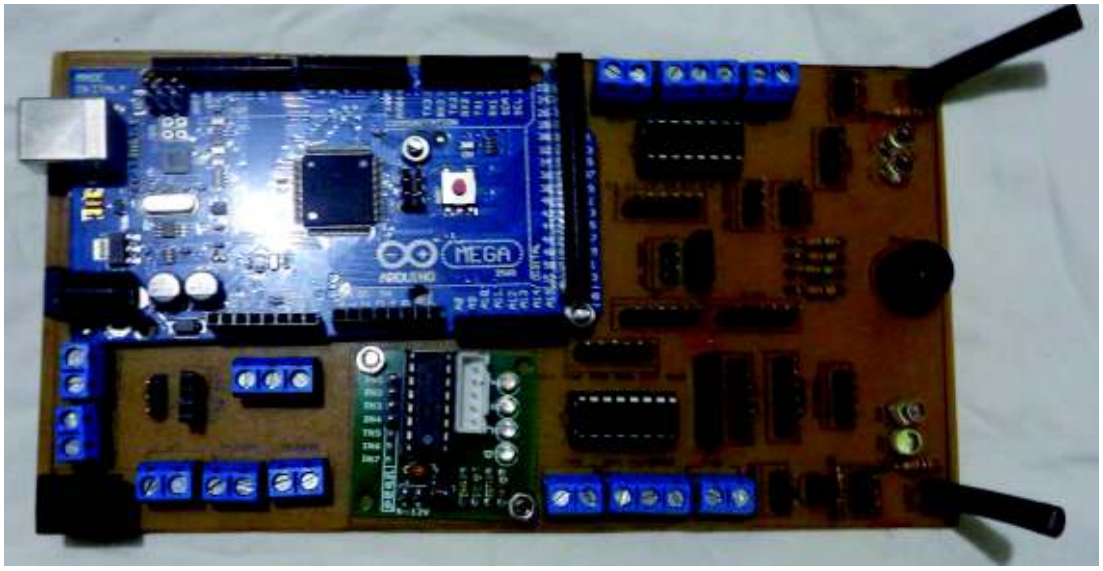


Figura 3.14: Tarjeta módulo terminada

Ensamblaje de estructura de acrílico con tarjeta

Para la estructura móvil del módulo, se ha utilizado un armazón de acrílico que se encuentra disponible en el mercado, el cual justamente es dedicado para aplicaciones de robótica. En la Figura 3.15 [23], se puede observar la estructura desarmada completamente.

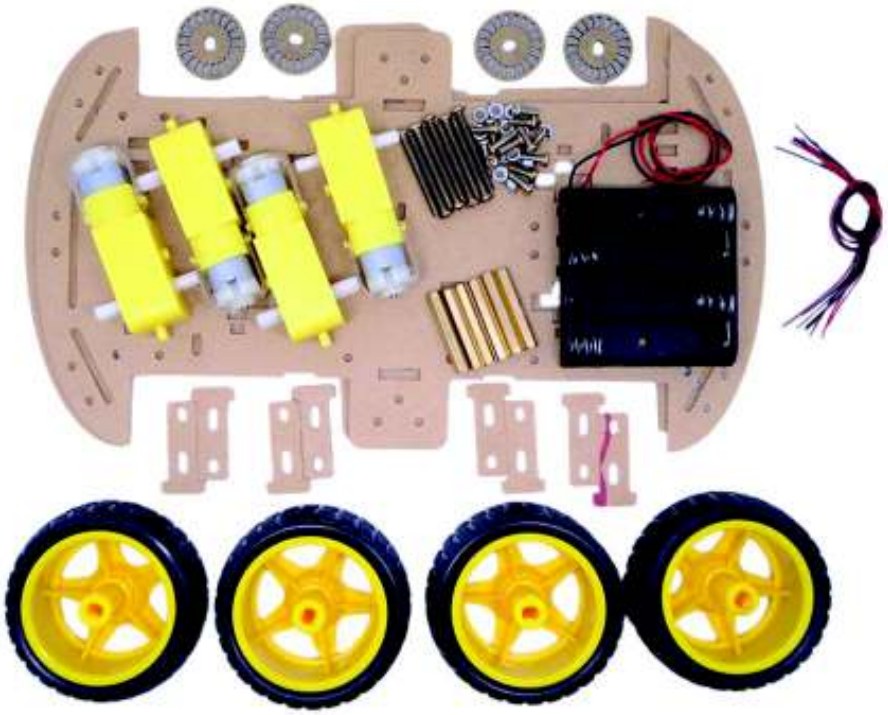


Figura 3.15: Estructura móvil desarmada

La estructura o chasis fue ensamblado fácilmente utilizando los materiales suministrados en la compra. Se puede observar la estructura armada completamente en la Figura 3.16.

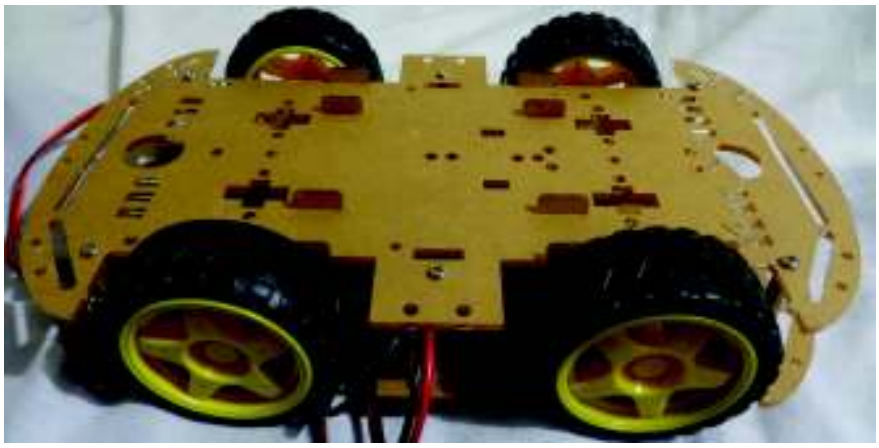


Figura 3.16: Estructura ensamblada

Para fijar la tarjeta al chasis, se utilizaron las perforaciones del *Arduino* Mega, de manera que estos coincidan con las perforaciones que presenta la estructura de acrílico. En la Figura 3.17, se muestran las perforaciones utilizadas en el *Arduino* Mega.

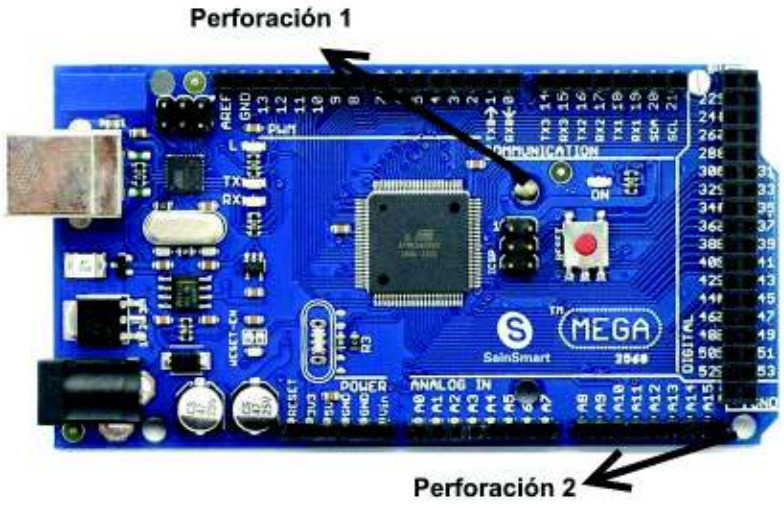


Figura 3.17: Perforaciones utilizadas del *Arduino* Mega para ensamblarla al chasis

Además, se colocó un servomotor en la parte frontal del chasis, que trabaje junto con el sensor ultrasónico y permita tener distintos ángulos de posicionamiento, como se muestra en la Figura 3.18.

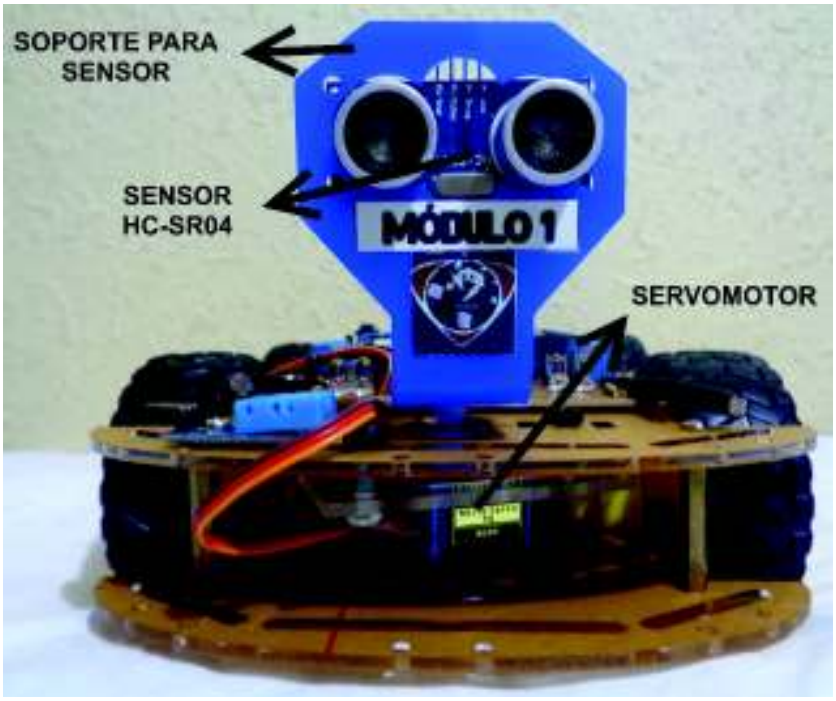


Figura 3.18: Vista frontal del módulo didáctico

Finalmente, se colocaron los sensores faltantes, con pernos y tuercas, en la estructura. La distribución de elementos ensamblados se lo puede ver en la Figura 3.19. Cabe recalcar que se ha realizado el mismo proceso para realizar los 9 módulos restantes.

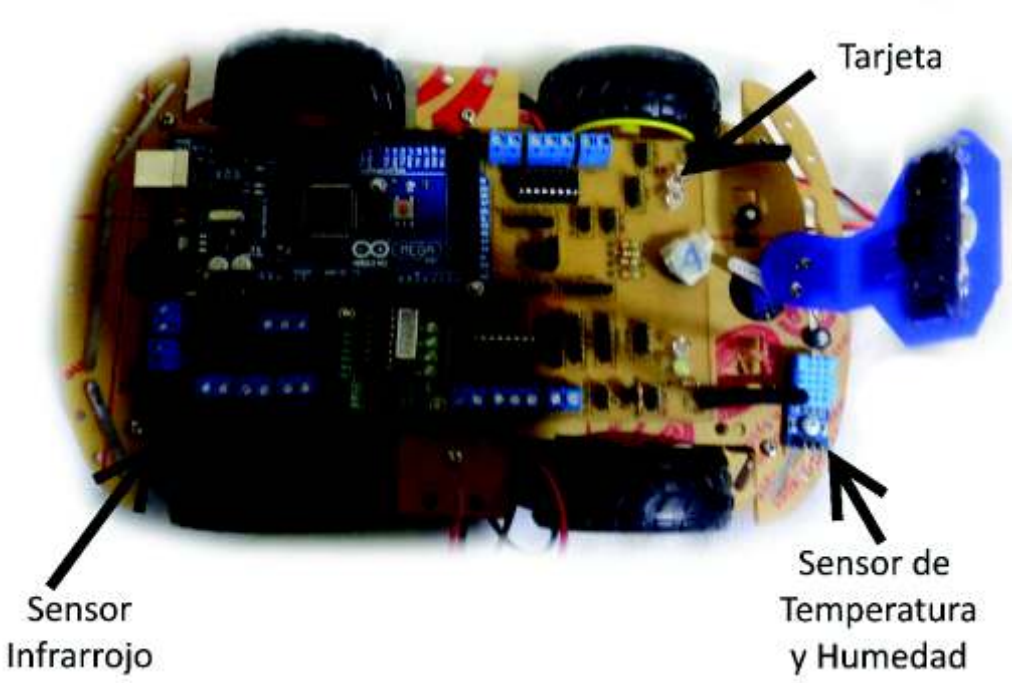


Figura 3.19: Distribución de elementos en un módulo terminado

3.4. Prueba de funcionamiento del Módulo

Con el fin de conocer el correcto funcionamiento del módulo y de su unidad de control, se han establecido las siguientes pruebas de funcionamiento:

Encendido de unidad de control

Al conectar el cable USB como se indica en la Figura 3.20, se deberán encender las luces indicadoras “ON” y “L”, las cuales nos permitirán comprobar que la unidad de control se encuentra en buen estado y no existen problemas de corriente dentro de ella, además que nos permitirá conocer el buen estado del cable utilizado.

Figura 3.20: Cable USB con terminales Tipo A y Tipo B

Verificación de elementos indicadores

Para esta verificación se debe realizar conexiones con el fin de conocer el buen estado de los elementos indicadores, siendo estos *Buzzer* y *LED*s. Para la comprobación del *Buzzer*, se procederá a realizar la conexión en base a la Figura 3.21, lo que deberá encender el *Buzzer* emitiendo sonido.

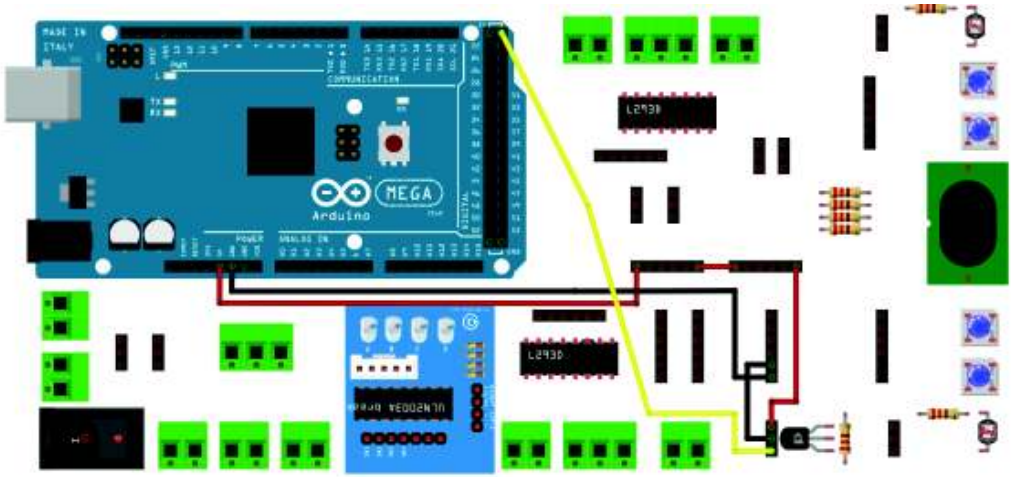


Figura 3.21: Prueba *Buzzer*

Para la verificación de *LED*s, se deberá realizar las conexiones indicadas en la Figura 3.22 las cuales salen de los pines de 5V que posee la unidad de control, de manera que se podrá verificar el encendido de los mismo, junto con la verificación de las salidas de voltaje y comprobar que los cables a utilizar estén en buen estado.

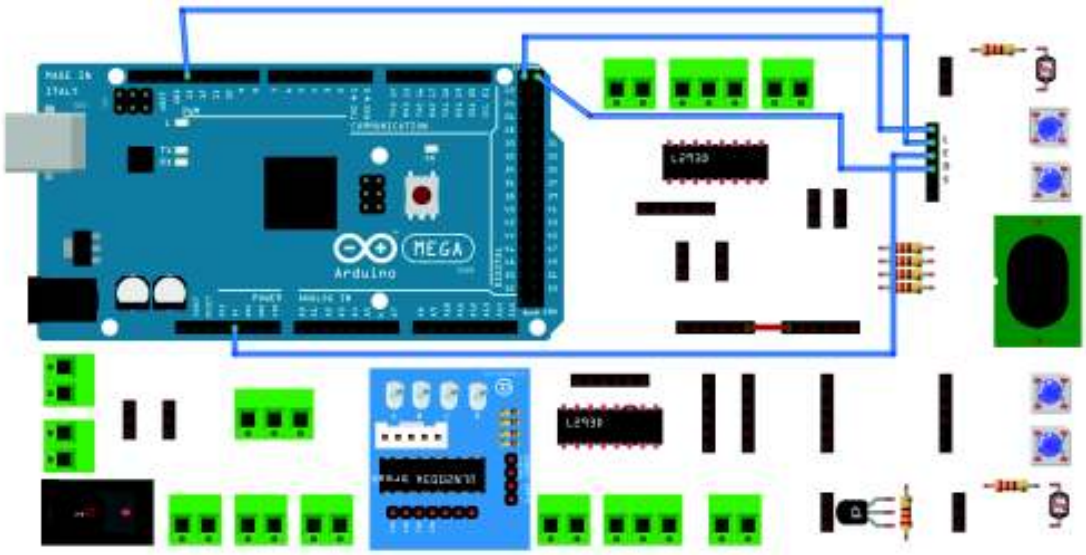


Figura 3.22: Prueba *LED*s y voltajes

Posibles errores en pruebas de funcionamiento

Luces indicadoras no encienden

Este fallo se puede ocasionar por el mal estado del cable USB, Jack USB de computador en mal estado o mala conexión en el Jack USB del *Arduino* al no introducir bien el conector.

Para corregir este posible fallo, se recomienda:

- Utilizar un puerto USB del computador distinto.
- Verificar que el cable USB a utilizar no tenga marcas de óxido ni deformaciones en los conectores, caso contrario cambiarlo.

***LED*s o *Buzzer* no encienden**

El no encendido de *LED*s y *Buzzer*, puede ocasionarse por cables en mal estado, salida de voltaje defectuosa de la unidad de control, conectores de cables que no hacen contacto con el conector *header* y posibles errores de conexión.

Para corregir estos posibles fallos, se recomienda:

- Verificar que los terminales de los cables a utilizar para la conexión se encuentren en buen estado, es decir que encajen con el *header* macho de los elementos a probar.

- Si nota que el cable utilizado para conexión es muy flexible, considere una posible rotura interna de cable, si es posible mida continuidad con un multímetro para descartarlo inmediatamente.

- Con el tiempo, el uso de los pines de la unidad de control se pueden ver desgastados, por lo que se deben verificar con un multímetro las salidas de voltaje, para revisar que los conectores hembra estén en buen estado.

- Verificar los diagramas de conexión para no cometer errores en la polaridad al momento de alimentar los elementos.

Para una mayor información con respecto al Software y Hardware del módulo, se recomienda revisar el Manual de usuario disponible en el Anexo D.

3.5. Prácticas de Laboratorio Desarrolladas

Como complemento de la implementación del módulo didáctico, han sido desarrolladas prácticas de laboratorio que incluyen temas relacionados con comunicación serial, sensores, control de motores y comunicación inalámbrica mediante el módulo *Bluetooth*.

Dichas prácticas, se encontrarán implementadas y previamente comprobadas su funcionalidad y cumplimiento de objetivos, luego de lo cual se ha creado un manual que permita tanto a estudiante como docente lograr la implementación de cada una de estas prácticas y obtener los resultados esperados.

Práctica 1: Control de sensores

Objetivos

- Desarrollar un programa en el IDE de *Arduino* que permita la transmisión de datos de varios sensores a través del puerto serie.
- Analizar los valores entregados por los sensores de temperatura, humedad, luz y distancia e interpretarlos mediante los dispositivos indicadores y actuadores del módulo a fin de lograr visualizar las variaciones de dichas magnitudes.

Componentes de Hardware y Software

En la presente práctica se emplean los siguientes componentes de hardware y software:

Hardware

- Sensor DHT-11
- LDR
- Sensor Ultrasónico
- Cables
- *Buzzer*
- *LED*s
- *Arduino* Mega 2560

Software

- IDE de *Arduino*

Descripción de la práctica

Esta práctica consiste en la elaboración de un sistema básico de sensores cuyo funcionamiento es monitorear los valores obtenidos y gestionar a su vez elementos como *LED*s y *Buzzer* dependiendo de dichos valores.

Los sensores que serán utilizados para esta práctica son los siguientes: LDR, HCSR-04, DHT11. Estos son sensores digitales a excepción del LDR, que es un sensor analógico y por lo tanto debe ser conectado a un pin análogo del *Arduino*, mientras que los dos restantes deben ser conectados a los pines digitales. Los

elementos como *LED*s y *Buzzer* también deben ser conectados a los pines digitales del *Arduino* como se indica en la Figura 3.23.

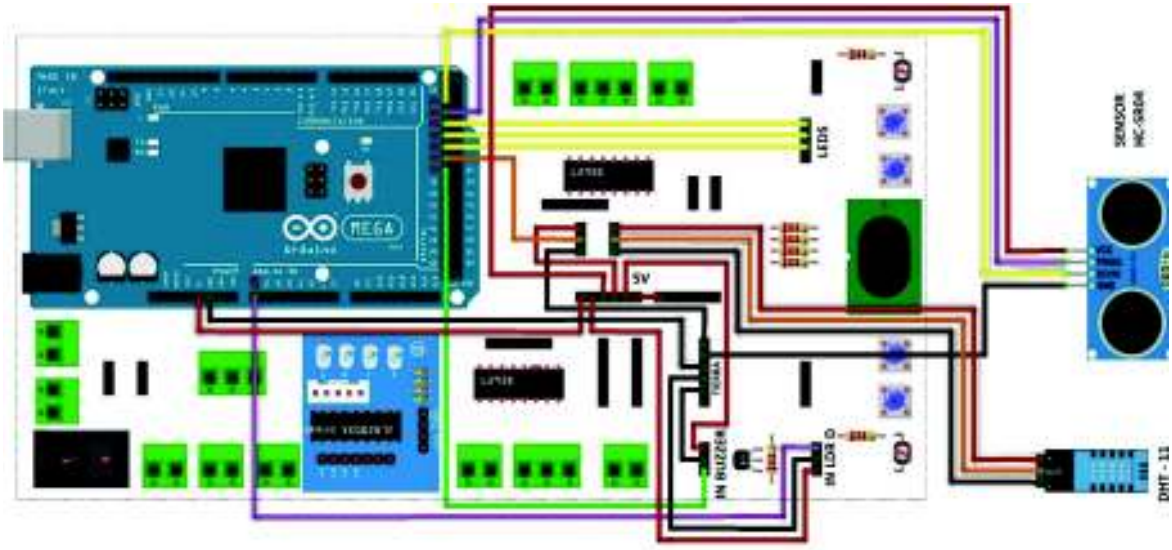


Figura 3.23: Diagrama de conexión sensores

El sensor LDR será utilizado para medir la cantidad de luz, el sensor HCSR-04 se lo utilizará para medir la distancia y finalmente el sensor DHT11 permitirá medir dos magnitudes a la vez como son; temperatura y humedad.

Una vez completada la conexión de los sensores, *Buzzer* y *LED*s se procede a desarrollar el software que los controla. Este software estará basado en la librería protothread que es fundamental para el manejo de sistemas similares al de esta práctica, ya que permite un flujo de control del software sin utilizar máquinas de estados complejas, es decir, facilita realizar aplicaciones que implican tareas múltiples. Por lo tanto, el software estará diseñado para realizar las siguientes tareas: lectura de los valores medidos por los sensores, enviar datos al computador a través de la comunicación serial del *Arduino* Mega y finalmente, gestionar los *LED*s y *Buzzer* dependiendo de los valores leídos anteriormente como se observa en la Figura 3.24. Estas 3 actividades serán realizadas por el *Arduino* Mega de manera simultánea gracias a la librería utilizada.

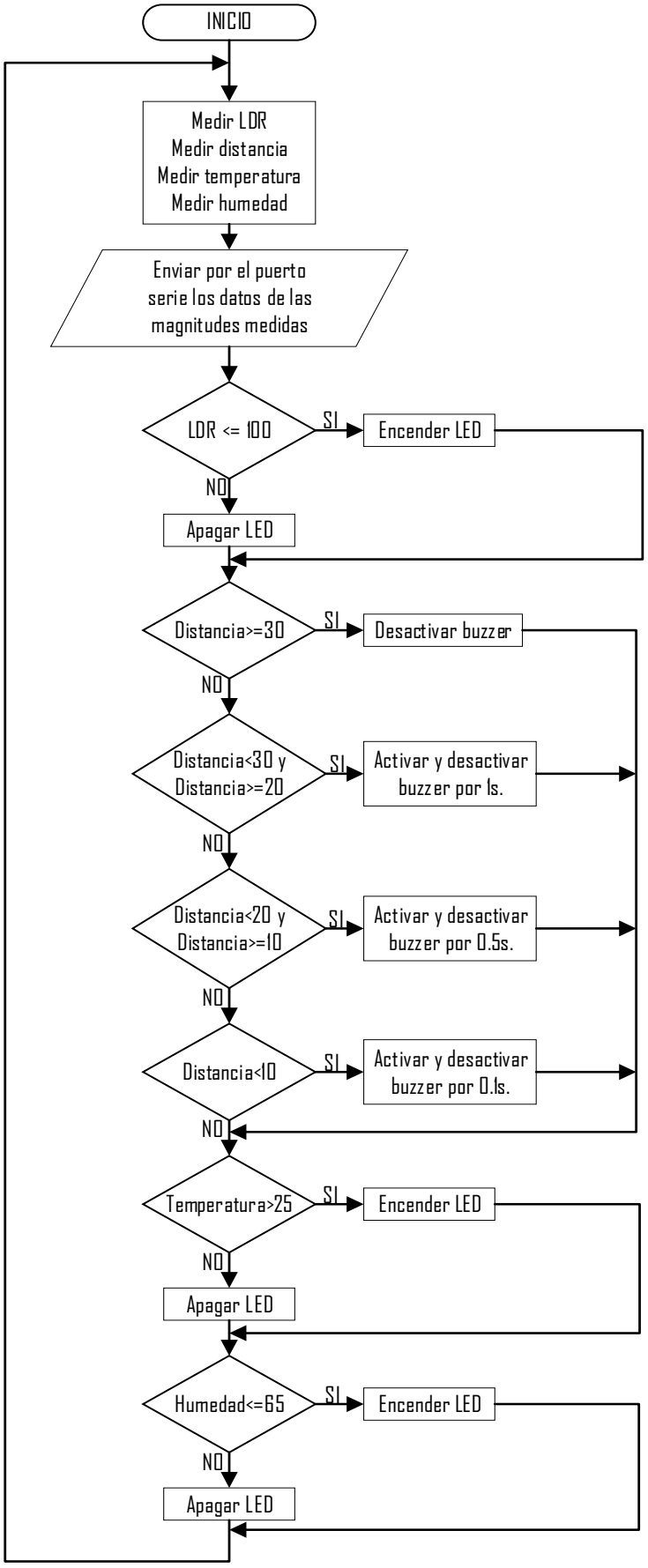


Figura 3.24: Diagrama de flujo de programación *Arduino* Mega de control de sensores

Las hojas guía para estudiante y docente se encuentran en el Anexo B, incluido el código de programación.

Práctica 2: Control de motores

Objetivos

- Utilizar comandos de nivel medio en el IDE *Arduino* y *Visual Studio* mediante el uso de librerías a fin de reducir y optimizar el código implementado en la práctica.
- Identificar una secuencia que permita el control de un motor a pasos unipolar y bipolar de manera que se logre ejecutar giros en sentido horario y antihorario.
- Desarrollar un esquema que permita controlar el giro de un motor unipolar y un servomotor mediante una interfaz en *Visual Studio* 2015.

Componentes de Hardware y Software

En esta práctica se emplean los siguientes componentes de hardware y software:

Hardware

- *Arduino* Mega 2560
- *Driver* ULN2003
- Servomotor SG-90
- Motor a pasos unipolar
- Cables

Software

- IDE de *Arduino*
- *Visual Studio*

Descripción de la práctica

La práctica de control de motores llevará al estudiante a un entorno más complejo de programación dentro de *Arduino* IDE; sin embargo, mantiene la sobriedad de los comandos de *Arduino*, los cuales permiten tener un código fácil de entender y

ejecutar, de manera que se logren obtener los resultados esperados. Se mantiene el entorno de comunicación serial entre computador y plataforma.

Esta práctica consiste en la elaboración de un sistema de control básico para motores a paso, ya sean estos unipolar o bipolares, de manera que se ejecute el giro de los grados ingresados en una interfaz por parte del usuario.

Para esta práctica, se ha optado por la utilización de un módulo *Driver* de control de motor unipolar ULN2003 y un servomotor los cuales será conectado a los pines de salida digital del *Arduino* MEGA, como se indica en la Figura 3.25. Además, la alimentación hacia estos dos elementos actuadores estará dada por la salida de voltaje del *Arduino*, ya que la corriente y voltaje suministrado por este, es suficiente para ejecutar el giro del motor.

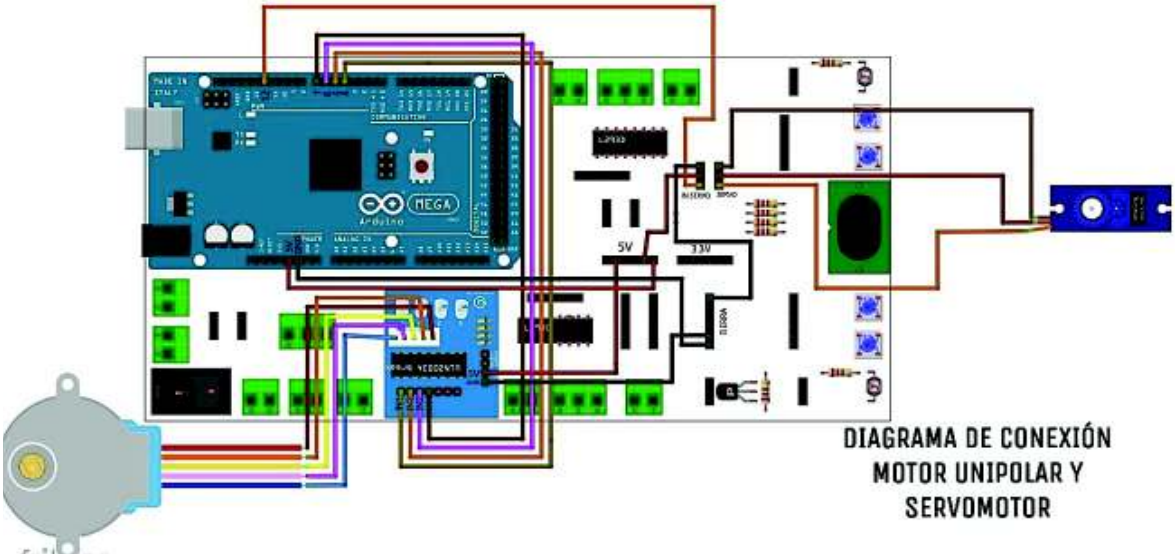


Figura 3.25: Diagrama de conexión motor unipolar y servomotor

Una vez completada la conexión de los elementos actuadores, se procede a desarrollar el software destinado al control. Este software estará divido en dos partes: software de *Arduino* Mega y software de *Visual Studio*.

El software destinado al *Arduino* Mega se encargará de receptar los datos a través del puerto serial, interpretarlos, almacenarlos y procesarlos conforme el ángulo de giro del motor que se esté utilizando como se indica en la Figura 3.26, es decir se realizará un cálculo para poder establecer el número de veces que se deberá repetir la secuencia de giro hasta alcanzar el valor del grado ingresado junto con el sentido

seleccionado de giro para el caso del motor unipolar. Por otro lado, para el servomotor, los datos a recibir únicamente será el valor en grados a posicionar en un rango de 0 a 180 grados.

El software destinado a *Visual Studio* será el que permita la conexión del computador con el *Arduino* mediante una interfaz intuitiva para el usuario como se observa en la Figura 3.28. Este programa se encargará de relacionar los datos ingresados por el usuario y convertirlos en cadena de caracteres que únicamente serán reconocidos por el *Arduino* Mega, de manera que se logren ejecutar las tareas asignadas para cada caso como se muestra en la Figura 3.27.

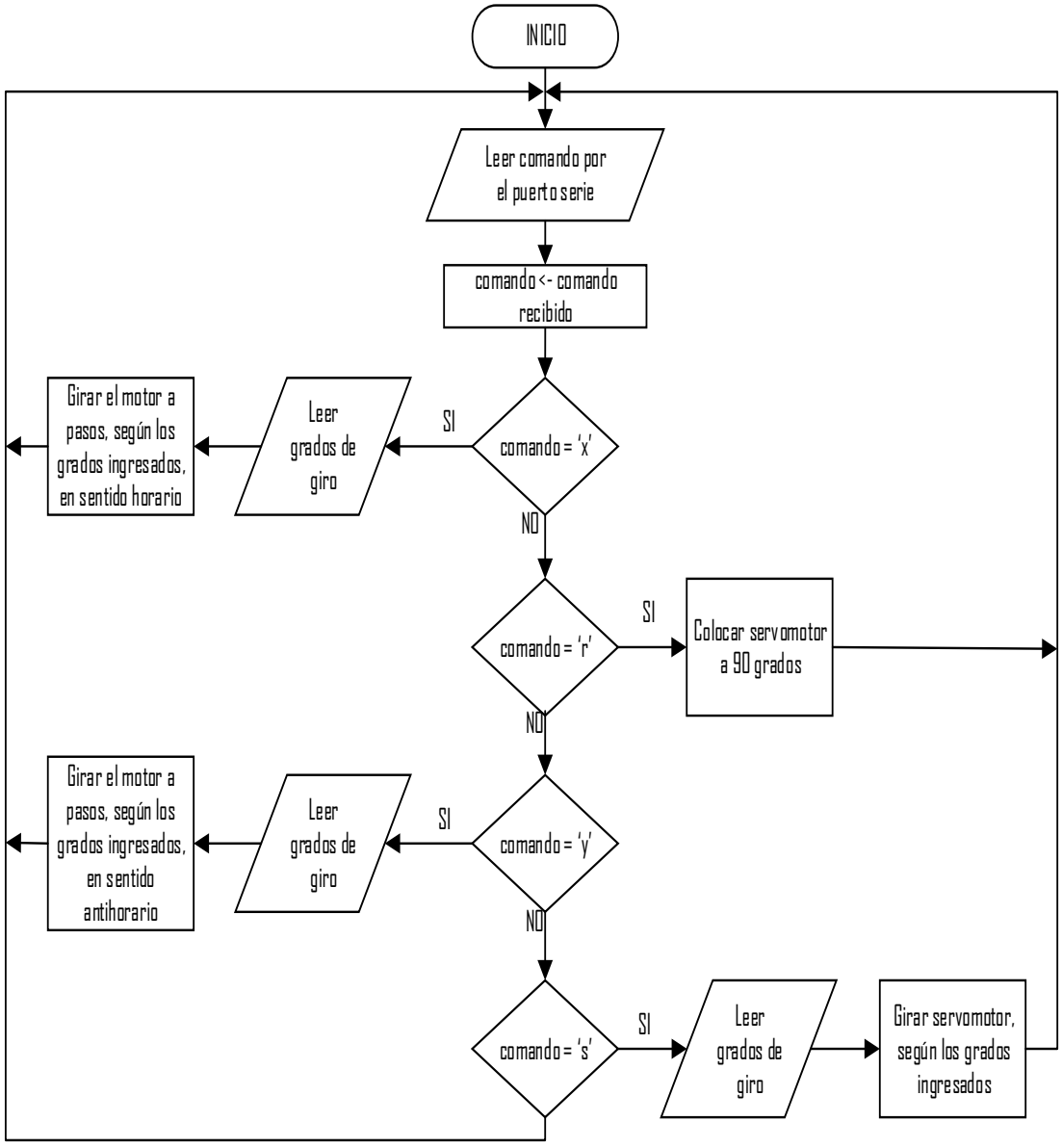


Figura 3.26: Diagrama de flujo programación en *Arduino* Mega de control de motores

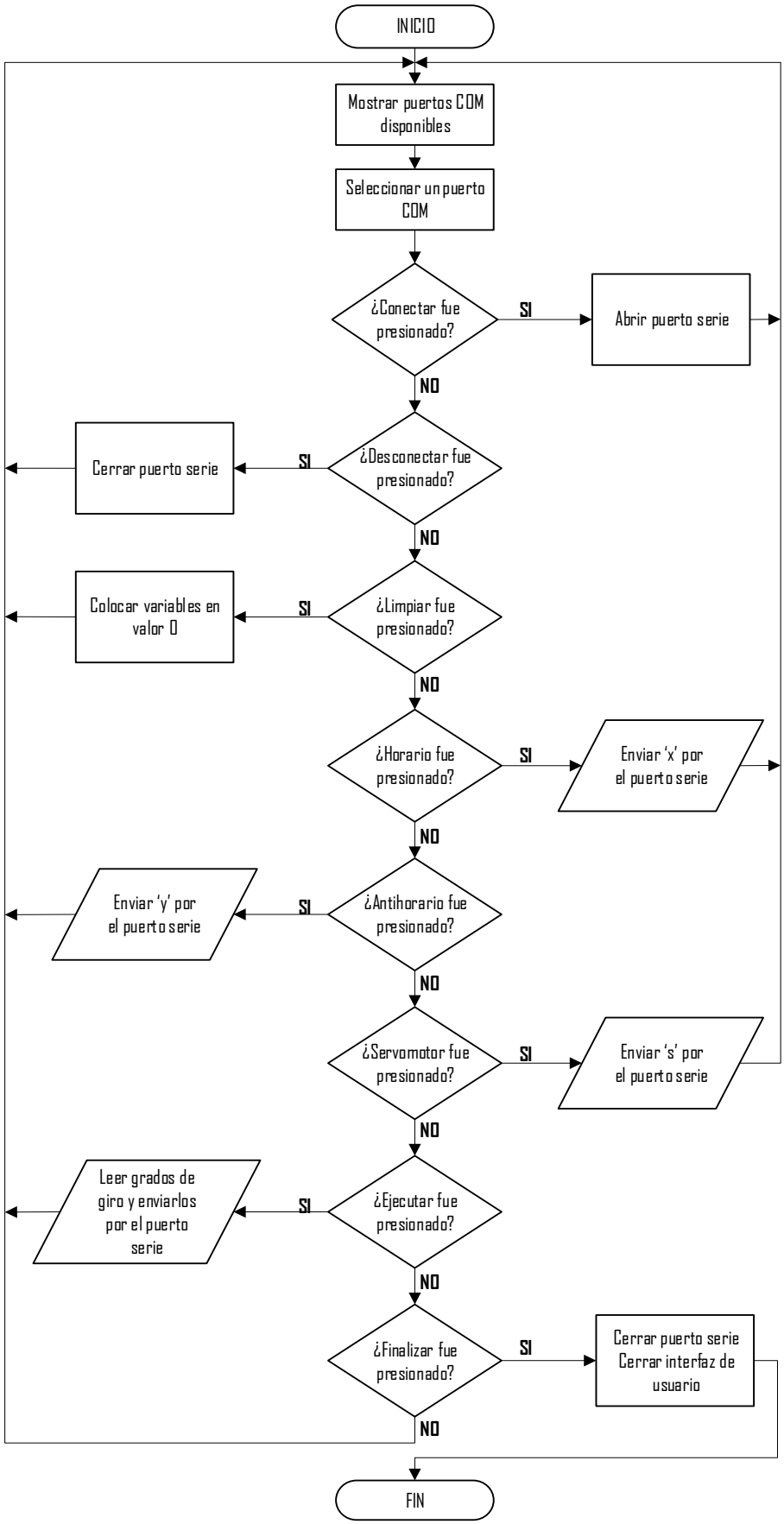


Figura 3.27: Diagrama de flujo de interfaz de usuario de control de motores

Figura 3.28: Interfaz de usuario realizado en Visual Studio.

Las hojas guía para estudiante y docente se encuentran en el Anexo B, incluido el código de programación correspondiente.

Práctica 3: Control de los motores de la estructura utilizando módulo *Bluetooth*

Objetivos

- Manejar el funcionamiento del módulo *Bluetooth* HC-05 mediante el uso de comandos de comunicación serial, para aplicaciones de comunicación inalámbrica.
- Controlar cuatro motores DC, utilizando el *Driver* L293D, para movilidad de la estructura.
- Desarrollar una App que permita controlar el módulo y sus interfaces desde el teléfono celular, utilizando *App Inventor*.

Componentes de Hardware y Software

En esta práctica se utilizaron los siguientes componentes de hardware y software.

Hardware

- *Arduino* Mega 2560
- Baterías
- Motores DC
- *Driver* L293D
- Cables
- Sensor ultrasónico HC-SR04
- Teléfono móvil con SO Android
- Servomotor SG9

Software

- *App Inventor*
- IDE de *Arduino*

Descripción de la práctica

Con la implementación de la comunicación inalámbrica mediante *Bluetooth*, el estudiante tendrá la oportunidad de verificar el funcionamiento de la plataforma móvil al implementar una aplicación básica realizada en App Inventor, lo que requerirá la utilización de sus teléfonos celulares con Android. Una aplicación que incluirá todos los elementos de la tarjeta del módulo didáctico, los cuales serán activados mediante una lista de comandos enviada desde el teléfono móvil hacia el *Arduino* Mega 2560.

La práctica contendrá el código fuente de control dentro del *Arduino* mega, el cual ejecutarán las tareas seleccionadas por parte del usuario a través de la lectura de su puerto serial denominado “Modo Manual”. La unidad de control, en este caso *Arduino* Mega, se encargará de receptar los datos ingresados en su puerto serie, con lo cual procederá a discriminar los distintos casos, en los que determinará la tarea a ejecutar, que en este caso son movimientos de la estructura en las direcciones: adelante, atrás, izquierda y derecha. Además, permitirá cambiar el modo de funcionamiento a “Modo Automático” y encender *LED*s y *Buzzer* como se indica en la Figura 3.29.

Dentro del modo de operación “Automático” se considera la autonomía del módulo en una aplicación de evasor de obstáculos con medición de distancia en tres puntos, siendo estos en las posiciones de 0, 90 y 180 grados, los cuales vienen dados por el servomotor. Al iniciar este modo de operación, el *Arduino* MEGA pasará a ejecutar el bucle, en el cual realiza el posicionamiento del servomotor para una posterior lectura de la distancia y su almacenamiento, una vez finalizadas las lecturas en las tres posiciones anteriormente dichas, realizará una comprobación de manera que la dirección a elegir será la que tenga el mayor valor de las tres lecturas como se indica en la Figura 3.30. Además, realizará una lectura del puerto serie en caso de que el usuario desee utilizar el modo de operación “Manual".

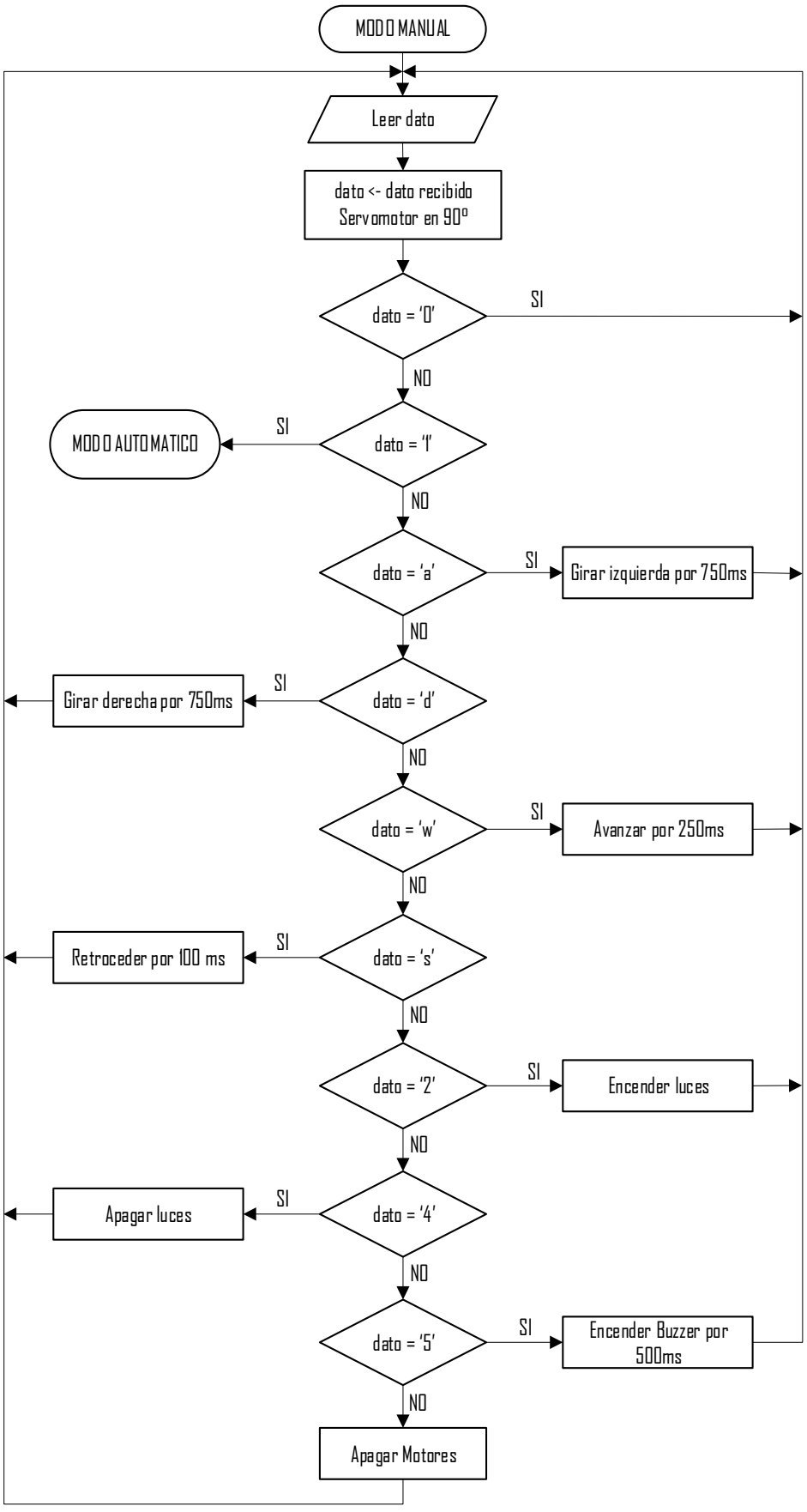


Figura 3.29: Diagrama de flujo del modo manual

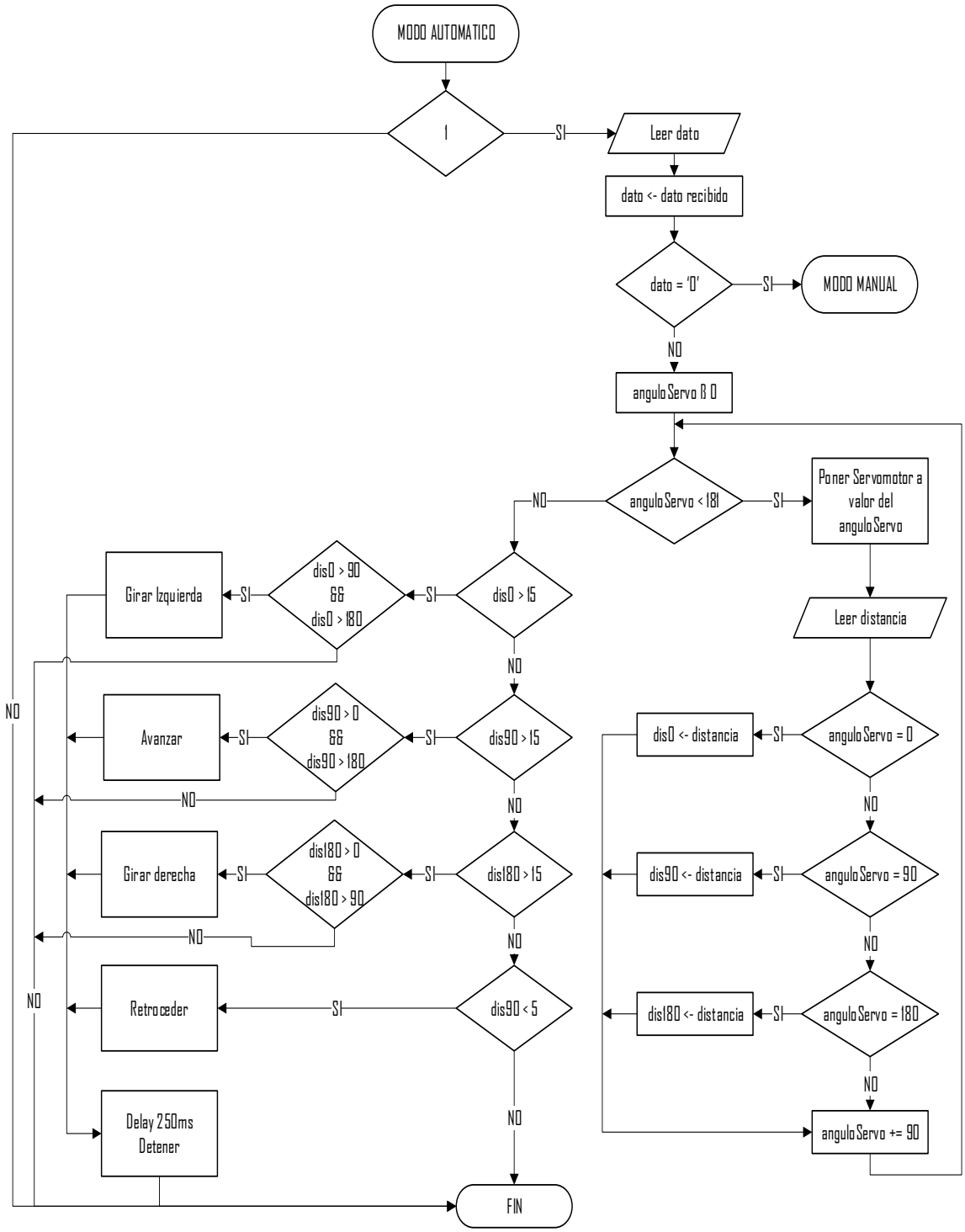


Figura 3.30: Diagrama de flujo del modo automático

Por otro lado, dentro de la aplicación que se encargará de enviar los caracteres a comparar mediante el teléfono móvil, se tendrá una interfaz como la que se indica en la Figura 3.31. Esta aplicación enlaza el *Arduino* Mega junto con el teléfono móvil, en el que se presentan botones, los cuales al ser pulsados enviarán el carácter asignado para cada tarea asignada como se indica en la Figura 3.32.

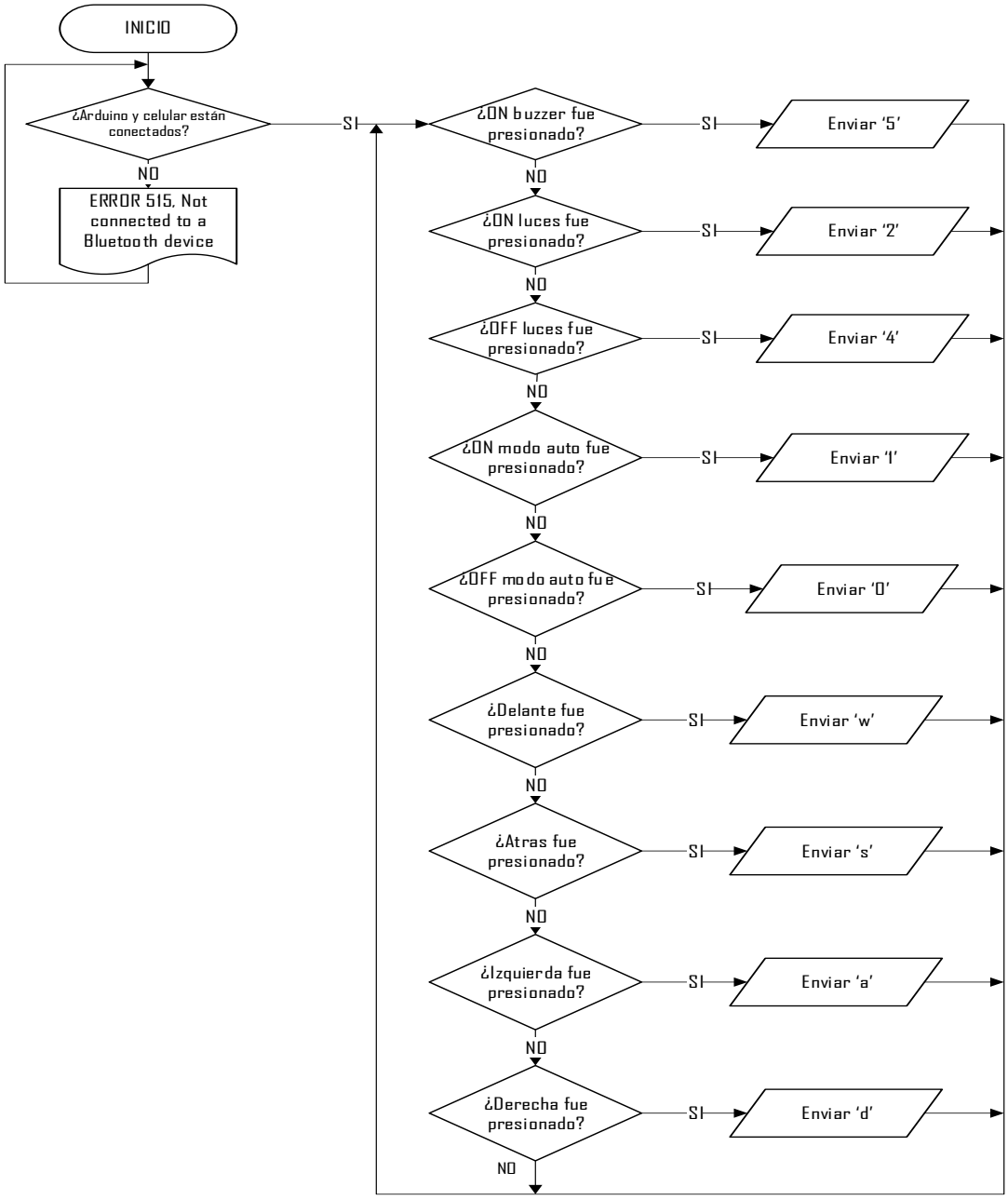


Figura 3.31: Diagrama de flujo aplicación

Figura 3.32: Aplicación para teléfonos con SO Android realizado en App Inventor

3.6. Costos de Implementación

Para la descripción de los costos asociados al presente proyecto, no se consideraron rubros por trabajo intelectual.

Tabla 3.1: Costo total de implementación de los 10 módulos didácticos

ELEMENTO	CANTIDAD	PRECIO UNITARIO	TOTAL (USD)
Broca 1,2mm	2	0,75	1,50
Broca 8mm	4	0,65	2,60
Cloruro Férrico	4	0,65	2,60
Estaño de soldar	1	13,90	13,90
Lija	4	0,70	2,80
Tinnher o Diluyente	1	1,50	1,50
Tubo termo retráctil 5mm - 1m	1	0,70	0,70
Cargador Baterías	1	26,00	26,00
40 cables 20 cm Macho-Hembra	10	4,00	40,00
40 cables 20 cm Macho-Macho	10	4,00	40,00
Arduino Mega	10	24,00	240,00
Baquelita 10x20	10	1,45	14,50
Batería LIPO	10	15,00	150,00
Bornera 2 pines	90	0,20	18,00
Bornera 3 pines	30	0,30	9,00
Buzzer	10	1,00	10,00
Cable USB tipo B	10	2,00	20,00
Chasis de acrílico, ruedas y motores DC	10	30,00	300,00
Conector *Header* Hembra 40 pines	10	0,55	5,50
Conector *Header* Macho 40 pines	20	0,55	11,00
Conector Plug DC	10	0,25	2,50
Diodo *LED* 5mm	40	0,15	6,00
L293D	20	3,15	63,00
LDR	20	0,35	7,00
Módulo *Bluetooth* HC-05	10	10,20	102,00
Motor Paso a Paso unipolar	10	6,50	65,00
Resistencias 1/4W	70	0,03	2,10
Sensor infrarrojo	10	5,00	50,00
Sensor Temperatura DHT-11	10	5,00	50,00
Sensor Ultrasónico HC-SR04	10	5,00	50,00
Servomotor	10	6,00	60,00
Soporte para ultrasónico	10	1,00	10,00
Interruptor	10	0,50	5,00
Transistor 2N3904	10	0,08	0,80
Zócalo	20	0,14	2,80
	TOTAL PROYECTO		1385,80

4. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

- La utilización de baterías LiPo está dedicada a aplicaciones donde se requiera el uso de los cuatro motores DC de la plataforma, por lo que estos representan un alto consumo de energía para poder ser sustanciados por el *Arduino* Mega.

- La tarjeta realizada está considerada con un diseño de tierra común para poder evitar problemas de corriente al momento de utilizar alimentación externa e individual, previniendo posibles daños a elementos y a la plataforma *Arduino*.

- Las interfaces de usuario para el manejo de datos en *Visual Studio*, se las realizaron con lenguaje *Basic*, considerando los conocimientos adquiridos en la materia de Control con Microprocesadores.

- Se comprobó que el funcionamiento de la estructura móvil con alimentación externa posee una autonomía aproximada de 1 hora en funcionamiento, tiempo suficiente para que se pueda demostrar la práctica dentro del laboratorio.

- Las condiciones del código implementado en cada práctica pueden cambiar según se lo requiera, ya que se lo puede optimizar de manera que se obtengan menos líneas de código y el resultado deseado.

- Se realizaron esquemas de cableado para cada circuito que forma parte de la tarjeta como lo son: *Drivers* de motores, *LEDs, Buzzer, LDRs con* el fin de que se facilite la conexión de los elementos y prevenir posibles errores al momento de identificar los conectores de cada circuito.

- El *Driver* para control de motores se lo escogió en base a sus características operacionales y principalmente por su tamaño, ya que representa un espacio pequeño dentro de la tarjeta base, permitiendo así tener más elementos dentro de la misma.

- Se consideró que es necesario entregar un esquemático donde se presentan las conexiones de la tarjeta principal ante posible cambio con respecto a las aplicaciones determinadas dentro del manual de prácticas.
- Los recursos utilizados de la plataforma *Arduino Mega*, en las prácticas de laboratorio, tanto en pines de conexión, así como en memoria no superan el diez por ciento, por lo tanto, la unidad de control del módulo puede ser utilizada para realizar aplicaciones complejas.

4.2. Recomendaciones

- Se recomienda que las baterías LiPo se encuentren cargadas en su totalidad previo funcionamiento de la práctica para obtener un mejor desempeño de los motores.
- Antes de alimentar los distintos elementos, se debe revisar que las conexiones se encuentren debidamente realizadas.
- Procurar que los cables se encuentren en el Pin correcto del *Arduino* Mega.
- El manejo de la estructura debe ser delicado, ya que se encuentra conformada por piezas de acrílico el cual puede romperse fácilmente ante un golpe fuerte.
- En caso de conectar una fuente amplificadora o módulo STEP UP, procurar que se respeten los parámetros de la batería LiPo, todo esto con el fin de evitar accidentes con estas baterías.
- En caso de presentar un sobrecalentamiento en las baterías, proceder a retirarla y colocarlas en un lugar alejado, ya que estas son inflamables y pueden emitir humo e incendiarse.
- En caso de tener que reemplazar algún elemento que se encuentra soldado a la tarjeta base, revisar los *Layouts*.
- La interfaz de usuario para la aplicación de comunicación con *Bluetooth* se encuentra realizada en App Inventor. Sin embargo, se la puede realizar en *Visual Studio* o cualquier programa que permita crear interfaces de usuario,

únicamente respetando que los parámetros interpretados por el *Arduino* y los enviados por la interfaz sean los mismos.

- En caso de tener conectado el módulo *Bluetooth* en los terminales dedicados a comunicación serial por defecto del *Arduino*, se debe proceder a la desconexión de este al momento de cargar el programa. Este problema se puede solucionar mediante el uso de los demás puertos Serie de *Arduino* para el módulo *Bluetooth*.
- Si se requiere añadir más interfaces de *Arduino*, localizarlas de mejor manera dentro del módulo, haciendo uso de las perforaciones que posee la estructura de acrílico.
- Evitar halar los cables que conectan los motores DC con caja reductora, ya que los pines que estos poseen son muy delicados y propensos a romperse.
- Es necesario tener un cargador dedicado para las baterías LiPo, el cual cumpla con las especificaciones de la batería aquí utilizada, razón por la cual se provee un cargador junto con los módulos.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] *Arduino*, «*Arduino* Mega 2560 REV3,» [En línea]. *Available*: https://store.arduino.cc/*Arduino*-mega-2560-rev3. [Último acceso: 26 Noviembre 2017].

[2] *Arduino*, «What is *Arduino*?,» [En línea]. *Available*: https://www.*Arduino*.cc/en/Guide/Introduction. [Último acceso: 26 Noviembre 2017].

[3] *Arduino*, «Language Reference,» [En línea]. *Available*: https://www.*Arduino*.cc/reference/en/. [Último acceso: 26 Noviembre 2017].

[4] *Arduino*, «*Arduino* IDE,» s.f. [En línea]. *Available*: https://www.*Arduino*.cc/en/Main/Software. [Último acceso: 2 Mayo 2018].

[5] *Arduino*, «*Arduino* Software IDE,» 7 Septiembre 2015. [En línea]. *Available*: https://www.*Arduino*.cc/en/Guide/Environment. [Último acceso: 2 Mayo 2018].

[6] *Arduino*, «*Arduino* Mega 2560,» [En línea]. *Available*: https://store.*Arduino*.cc/usa/*Arduino*-mega-2560-rev3. [Último acceso: 12 Febrero 2018].

[7] Embarcados, «*Arduino* MEGA 2560,» 28 April 2014. [En línea]. *Available*: http://www.embarcados.com.br/*Arduino*-mega-2560/. [Último acceso: 2 Mayo 2018].

[8] IBM, «Comunicación Serie,» s.f. [En línea]. *Available*: https://www.ibm.com/support/knowledgecenter/es/ssw_aix_72/com.ibm.aix.networkcomm/asynch_serialcomm.htm. [Último acceso: 2 Mayo 2018].

[9] A. *Arduino*, «Comunicación Serie *Arduino*,» s.f. [En línea]. *Available*: https://aprendiendo*Arduino*.wordpress.com/2016/07/02/comunicacion-serie-*Arduino*/#com ments. [Último acceso: 2 Mayo 2018].

[10] A. *Arduino*, «Bluetooth en *Arduino*,» s.f. [En línea]. *Available*: https://aprendiendo*Arduino*.wordpress.com/2016/11/13/bluetooth-en-*Arduino*/. [Último acceso: 2 Mayo 2018].

[11] *Prometec*, «EL módulo bluetooth HC 05,» s.f. [En línea]. *Available*: https://www.*Prometec*.net/bt-hc05/. [Último acceso: 2 Mayo 2018].

[12] *Components*, «*HC-05 Bluetooth Module,»* s.f. [En línea]. *Available*: https://components101.com/wireless/hc-05-bluetooth-module. [Último acceso: 2 Mayo 2018].

[13] Unprogramador, «Configuración del módulo Bluetooth HC 05,» 11 June 2018. [En línea]. *Available*: https://unprogramador.com/configuracion-del-modulo-blueto oh-hc-05/. [Último acceso: 2 Mayo 2018].

[14] Tindie, «*HC-SR04 Ultrasonic Sensor*,» s.f. [En línea]. *Available*: https://www.tindie.com/products/andygrove73/hc-sr04-ultrasonic-sensor-pack-of-8/. [Último acceso: 2 Mayo 2018].

[15] *Prometec*, «Sensores de Temperatura DHT11,» s.f. [En línea]. *Available*: https://www.*Prometec*.net/sensores-dht11/. [Último acceso: 2 Mayo 2018].

[16] PicMania, «Divisores de Tensión,» s.f. [En línea]. *Available*: http://picmania.garcia-cuervo.net/electronica_basica_divisor_tension.htm. [Último acceso: 2 Mayo 2018].

[17] A. G. Naranjo, «Motor de corriente continua,» s.f. [En línea]. *Available*: https://sites.google.com/site/alvarogarcianaranjo/mootor-de-corriente-continua. [Último acceso: 2 Mayo 2018].

[18] Naylamp, «Motor con caja reductora,» s.f. [En línea]. *Available*: https:// naylampmechatronics.com/motores-dc/20-motor-dc-caja-reductora-y-llanta-goma.html. [Último acceso: 2 Mayo 2018].

[19] L. Llamas, «Motor paso a paso 28BYJ-48 con *Arduino* y Driver,» 13 Agosto 2016. [En línea]. *Available*: https://www.luisllamas.es/motor-paso-paso-28byj-48-*Arduino*-driver-uln2003/. [Último acceso: 2 Mayo 2018].

[20] Panamahitek, «Qué es y cómo funciona un servomotor,» 2 Diciembre 2016. [En línea]. *Available*: http://panamahitek.com/que-es-y-como-funciona-un-servomotor/. [Último acceso: 2 Mayo 2018].

[21] akizukidenshi, «SG90 Micro Servo,» s.f. [En línea]. *Available*: http://akizukidenshi.com/download/ds/towerpro/SG90.pdf. [Último acceso: 2 Mayo 2018].

[22] Teslabem, «Baterias LiPo,» 3 May 2017. [En línea]. *Available*: http://blog. teslabem.com/como-usar-y-cuidar-las-baterias-lipo/. [Último acceso: 2 Mayo 2018].

[23] Ebay, «*Intelligent Robot Kit*,» s.f. [En línea]. *Available*: https://ebay.to/2sIl5Do. [Último acceso: 2 Mayo 2018].

[24] *BlueHack*, «Comandos AT,» s.f. [En línea]. *Available*: http://*BlueHack*.elhac ker.net/proyectos/comandosat/comandosat.html. [Último acceso: 2 Mayo 2018].

[25] Microsoft, «*Visual Studio IDE overview*,» s.f. [En línea]. *Available*: https://docs. microsoft.com/en-us/visualstudio/ide/visual-studio-ide. [Último acceso: 2 Mayo 2018].

[26] MIT App Inventor, «*About App Inventor*,» s.f. [En línea]. *Available*: http:// app Inventor.mit.edu/explore/about-us.html. [Último acceso: 2 Mayo 2018].

[27] Enacom, «Bandas no licenciadas,» s.f. [En línea]. *Available*: https://www.enaco m.gob.ar/bandas-no-licenciadas_p680. [Último acceso: 2018 Mayo 2018].

[28] Geek en formación, «Interfaces de comunicación SPI, I2C, UART,» 5 January 2013. [En línea]. *Available*: http://geekenformacion.blogspot.com/2013/01/ interfac es-de-comunicacion-spi-i2c-uart.html. [Último acceso: 2 Mayo 2018].

[29] GENBETA, «*Processing* un lenguaje para creadores audiovisuales,» 30 March 2013. [En línea]. *Available*: https://www.genbetadev.com/herramientas/ processing-un-lenguaje-para-creadores-audiovisuales. [Último acceso: 2 Mayo 2018].

[30] L. d. V. Hernández, «*Processing, Wiring and Arduino,»* s.f. [En línea]. *Available*: https://programarfacil.com/blog/*Arduino*-blog/processing-wiring-*Arduino*/. [Último ac ceso: 2 Mayo 2018].

[31] Microchip, «*Microchip products*,» [En línea]. *Available*: http://www.microchip. com/wwwproducts/en/PIC16F870. [Último acceso: 26 Noviembre 2017].

[32] MikroElektronika, «mikroC PRO for PIC,» [En línea]. *Available*: https://shop. mikroe.com/mikroc-pic. [Último acceso: 26 Noviembre 2017].

6. ANEXOS

Anexo A: Términos y definiciones

Anexo B: Prácticas de laboratorio

Anexo C: Hojas técnicas

Anexo D: Manual de usuario

Anexo A: Términos y definiciones

A-1. Comandos AT [24]

Los comandos AT son instrucciones codificadas que forman un lenguaje de comunicación entre el hombre y un terminal modem, de manera que se pueda configurar y proporcionar instrucciones hacia el modem. La terminología AT proviene de la abreviatura de *attention.*

Los comandos AT son muy conocidos dentro de la configuración y comunicación con módulos móviles GSM para realizar llamadas de datos o de voz; sin embargo, estos también son implementados para la configuración del módulo *bluetooth* HC-05.

A-2. *Visual Studio* [25]

Visual Studio es un entorno interactivo de desarrollo (IDE), el cual permite editar una gran cantidad de códigos y así depurarlos con el fin de crear aplicaciones para distintas plataformas como Android, iOS, Windows, Web o para la nube.

En la realización de las prácticas para el presente proyecto, se ha utilizado *Visual Studio* con el fin de crear pequeñas interfaces utilizando lenguaje Basic, permitiendo al usuario establecer una comunicación serial con la plataforma *Arduino* Mega.

A-3. MIT *App Inventor* [26]

MIT App Inventor es un entorno intuitivo y visual de programación que permite a las personas construir aplicaciones totalmente funcionales para smartphones y *tablets* creado por Google bajo licencia gratuita.

Su modo de programación se encuentra basada en código de bloques (como se muestra en la Figura 6.1) [26] con lo cual las personas poseen un conjunto de herramientas básicas las cuales deberán irse enlazando hasta lograr la creación de la aplicación deseada.

Figura 6.1: Bloques de código

A-4. Banda Libre [27]

Como su nombre lo dice, las bandas de frecuencias libres o no licenciadas son aquellas en las que los dispositivos de radiocomunicaciones pueden operar sin requerir alguna planificación centralizada por parte de la autoridad de comunicaciones. La banda libre se destina íntegramente a los dispositivos de radiocomunicaciones, sin dividir sus canales, sin dejar de lado parámetros importantes como los límites de potencia o densidad de potencia radiada, y su ancho de banda.

Dentro de las bandas de frecuencia libre, cada usuario es responsable de la coordinación; sin embargo, se apoya en la inmunidad contra interferencias, propias de la tecnología que se emplee y el modo de acceso múltiple a la banda.

A-5. UART [28]

Dentro de las funciones principales del chip UART (*Universal Asynchronous Receiver – Transmiter*) se encuentran el manejo de interrupción de dispositivos conectados al puerto serie y la conversión de datos provenientes del mismo en formato paralelo y a formato serie.

UART es el protocolo más utilizado, su principal diferencia entre SPI a I2C es que es asíncrono y estos dos últimos están sincronizados mediante una señal de reloj. La velocidad de datos UART está limitada a 2Mbps.

A-6. ICSP [28]

También llamada programación serial en circuito, es la manera en que algunos dispositivos lógicos, microcontroladores y otros circuitos electrónicos se pueden

programar mientras se encuentran instalados en un sistema completo, sin requerir programación previa a su instalación en el sistema.

Es un método de programar directamente los microcontroladores de AVR, PIC y otros.

A-7. I2C [28]

Este protocolo de comunicación es de tipo serie sincrónico, de manera que I2C implementa dos cables para su funcionamiento, siendo el primero utilizado para el reloj (SCL) y el otro para el dato (SDA). El maestro y el esclavo enviarán datos por el mismo cable, el cual se encuentra controlado por el maestro, quien implementa la señal de reloj.

A-8. Processing y Wiring [29] [30]

Processing es un lenguaje de programación orientada a diseñadores en donde no es necesario tener conocimientos de programación para poder usarlo. Su ventaja radica en su distribución bajo licencia GNU GPL, lo que lo convierte en una herramienta alternativa al software propietario dentro de la implementación de proyectos multimedia de diseñadores audiovisuales.

Wiring es un pequeño circuito, formado por un microcontrolador, siendo capaz de ser programado generando así prototipos de electrónica que se encuentran controlados por una multitud de dispositivos conectados al microcontrolador.

Su principal característica radica en el hecho de simplificar el lenguaje C/C++, haciendo que las operaciones de entrada y salida sean mucho más fáciles.

Anexo B: Prácticas de Laboratorio

B-1:

Hojas guía para estudiantes

ESCUELA POLITÉCNICA NACIONAL
ESCUELA DE FORMACIÓN DE TECNÓLOGOS
HOJA GUÍA
PRÁCTICA 1

1. TEMA: Control de sensores

2. OBJETIVOS

- Desarrollar un programa en el IDE de *Arduino* que permita la transmisión de datos de varios sensores a través del puerto serie.
- Analizar los valores entregados por los sensores de temperatura, humedad, luz y distancia e interpretarlos mediante los dispositivos indicadores y actuadores del módulo a fin de lograr visualizar las variaciones de dichas magnitudes.

3. TRABAJO PREPARATORIO

3.1 Cuestionario

- Revisar el diagrama de conexiones, incluido al final del documento.
- Realice una breve descripción de las principales características técnicas de *Arduino* Mega 2560.
- Investigue y describa las características principales de los siguientes sensores: LDR, DHT11 y HC-SR04.
- Consulte acerca de los siguientes comandos de *Arduino* IDE.
  - *Serial.begin()*
  - *Serial.print()*
  - *Serial.println()*
  - *pinMode()*
  - *analogRead()*
  - *digitalWrite()*
  - *pulseIn()*
  - *millis()*
- Consulte el funcionamiento de la librería *protothread* (pt) y escriba la o las ventajas de utilizarla.

4. DESCRIPCIÓN DE LAS ACTIVIDADES Y PROCEDIMIENTO DE LA PRÁCTICA

4.1. Realizar las conexiones necesarias de los sensores (DHT-11, LDR y HC-SR04), *Buzzer* y *LED*s con el módulo didáctico.

4.2. Elaborar un programa en el IDE de *Arduino* que permita leer las magnitudes físicas de los sensores mencionados en el paso anterior. La información obtenida debe ser enviada por el puerto serial con un formato similar al mostrado en el siguiente ejemplo:

```
**************************
LDR: 367.00
DISTANCIA: 0.00 (cm)
TEMPERATURA: 19.00 (°C)
HUMEDAD: 66.00 (%RH)
**************************
```

4.3. Compilar el programa creado y subirlo al *Arduino* Mega del módulo didáctico.

4.4. Verificar el programa utilizando el Monitor Serie del IDE de *Arduino*.

4.5. Mejorar el programa anterior, añadiendo nuevas funciones. Las mejoras consisten en activar o desactivar distintos dispositivos según los valores entregados por los sensores. Los dispositivos que se utilizarán son los disponibles en el módulo didáctico, es decir, *Buzzer* y *LED*s.

4.6. Verificar que la información sea transmitida y que los dispositivos realicen sus respectivas funciones según los valores mostrados en el monitor serie.

5. BIBLIOGRAFÍA

[1] *Arduino*, «*ARDUINO MEGA 2560 REV3*,» 2018. [En línea]. *Available*: https://store.Arduino.cc/usa/Arduino-mega-2560-rev3. [Último acceso: 10 Febrero 2018].

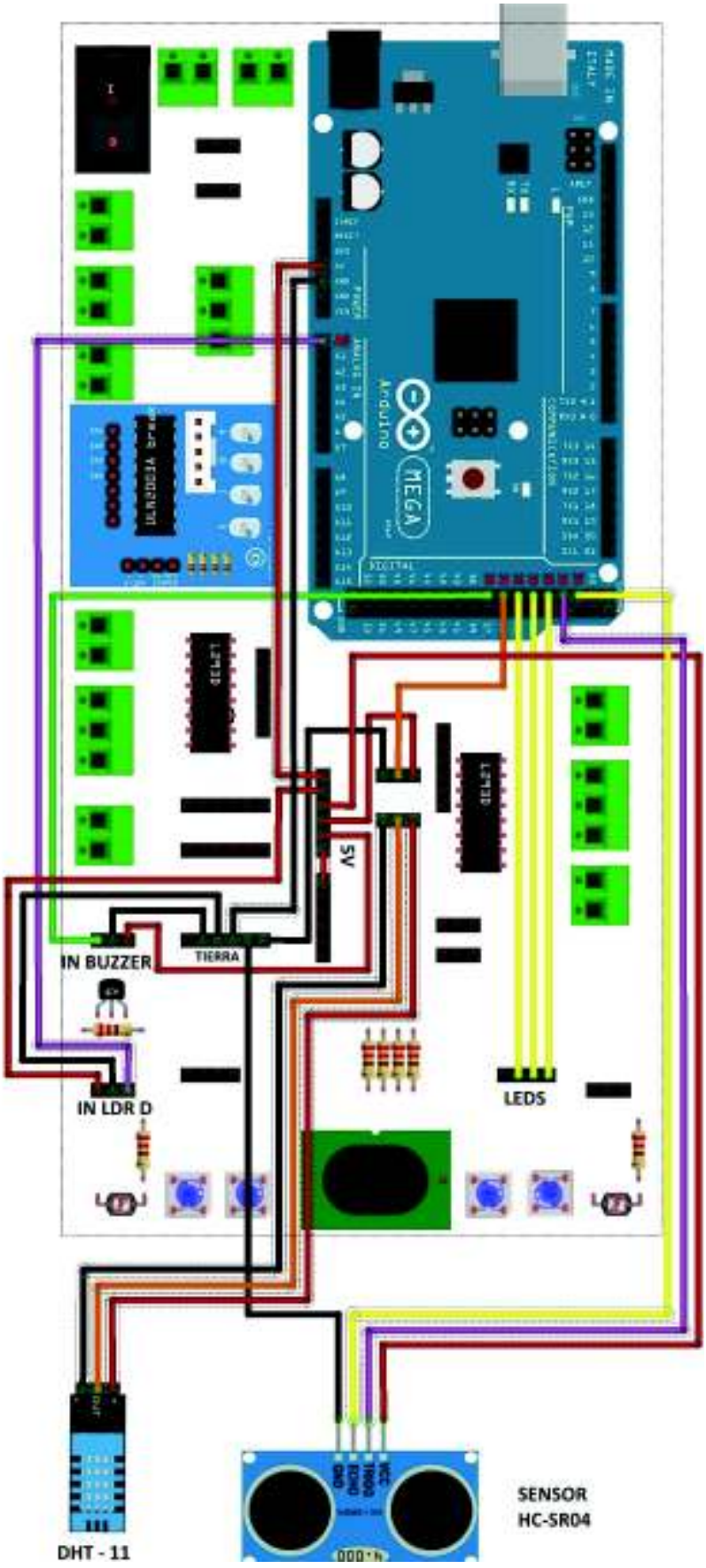
[2] P. Alcalde, Electrotecnia, España: Paraninfo S.A, 2003.

[3] AOSONG, «*Temperature and humidity module*,» 2018. [En línea]. *Available*: https://akizukidenshi.com/download/ds/aosong/DHT11.pdf. [Último acceso: 10 Febrero 2018].

[4] ELECFreaks, «*Ultrasonic Ranging Module HC-SR04*,» 2018. [En línea]. *Available*: http://www.micropik.com/PDF/HCSR04.pdf. [Último acceso: 10 Febrero 2018].

[5] *Arduino*, «*Language Reference - FUNCTIONS*,» 2018. [En línea]. *Available*: https://www.arduino.cc/reference/en/#functions. [Último acceso: 10 Febrero 2018].

DIAGRAMA DE CONEXIONES – PRÁCTICA 1



ESCUELA POLITÉCNICA NACIONAL
ESCUELA DE FORMACIÓN DE TECNÓLOGOS
HOJA GUÍA
PRÁCTICA 2

1. TEMA: Control de motores

2. OBJETIVOS

- Utilizar comandos de nivel medio en el IDE *Arduino* y *Visual Studio* con el fin de reducir y optimizar el código implementado en la práctica.
- Identificar una secuencia que permita el control de un motor a pasos unipolar y bipolar de manera que se logre ejecutar giros en sentido horario y antihorario.
- Desarrollar un esquema que permita controlar el giro de un motor unipolar y un servomotor mediante una interfaz en *Visual Studio* 2015.

3. TRABAJO PREPARATORIO

3.1 Cuestionario

- Revisar el diagrama de conexiones, incluido al final del documento.
- Consulte sobre los siguientes puntos para el IDE *Arduino*:
  - *Librería Servo.h*
  - *Serial.begin()*
  - *Serial.read()*
  - *Serial.println()*
  - *pinMode ()*
  - *digitalWrite()*
  - *switch*
  - *return*

  - Sentencia if
  - Sentencia for
  - Creación de funciones void.
  - Declaración de variables.
  - Conversión de variables String a Integer.
  - Lectura de un número de más de una cifra en el puerto serial.
  - Conexión del *Arduino* Mega a la PC y apertura del monitor serie
- Consulte sobre los siguientes comandos de Visual Studio
  - *ProgressBar(value, maximum, minimum)*
  - *Textbox. Text*
  - *ListBox.items.Add()*
  - *SerialPort()*
- Breve descripción de *Driver* ULN2003, servo motor SG90 y *Arduino* Mega.
- Consultar secuencia para motor unipolar.

- Analice los siguientes bloques de código, e identifique que tipo de aplicación podrían tener.

```
Private Sub Button8_Click(sender As Object, e As EventArgs) Handles Button8.Click
    SerialPort1.Close()
    TextBox2.Text = "DESCONECTADO"

End Sub
```

```
Private Sub Button7_Click(sender As Object, e As EventArgs) Handles Button7.Click
    Try
        With SerialPort1
            .BaudRate = 9600
            .DataBits = 8
            .Parity = IO.Ports.Parity.None
            .StopBits = 1
            .PortName = ComboBox1.Text
            .Open()
            If .IsOpen Then
                TextBox2.Text = "CONECTADO"
            Else
                MsgBox("SIN CONEXION", MsgBoxStyle.Critical)
            End If
        End With
    Catch ex As Exception
        MsgBox(ex.Message, MsgBoxStyle.Critical)
    End Try

End Sub
```

```
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Timer1.Enabled = True

        Try
            ComboBox1.Items.Clear()
            For Each puerto As String In My.Computer.Ports.SerialPortNames
                ComboBox1.Items.Add(puerto)
            Next
            If ComboBox1.Items.Count > 0 Then
                ComboBox1.SelectedIndex = 0
            Else
                MsgBox("No se encuentran puertos disponibles")

            End If
        Catch ex As Exception
            MsgBox(ex.Message, MsgBoxStyle.Critical)
        End Try
    End Sub
End Class
```

4. DESCRIPCIÓN DE LAS ACTIVIDADES Y PROCEDIMIENTO DE LA PRÁCTICA

4.1. Realizar las conexiones especificadas por el docente para el uso del motor unipolar con el módulo ULN 2003 en la placa didáctica.

4.2. Mediante el uso de la lectura del puerto serial como cadena de caracteres, elaborar un código que permita girar “n” grados en sentido horario, y luego girar los mismos “n” grados en sentido antihorario. Se sabe que el motor gira aproximadamente 0.65 grados por paso, es decir, por cada secuencia completa. Los grados estarán comprendidos entre 0 y 360. Realizar la prueba de funcionamiento con el monitor serie de *Arduino*

Numero de pasos = valor ingresado / 0.65

4.3 Elaborar un código que permita controlar el sentido de giro horario, antihorario del motor unipolar y a su vez controlar el ángulo de giro de un servomotor. El ángulo de giro será “n” (ingresado por el usuario) al igual que en el anterior literal. Además, incluir un caso que permita

hacer un *reset* para volver el valor de las variables a cero. Utilizar la sentencia *Switch* para los distintos casos.

Factores para considerar:

- r = *reset*
- s = servo
- x = horario
- y= antihorario
- para los sentidos de giro y servo motor utilizar funciones *void*.

4.4 Pruebas de funcionamiento:

- Mediante el monitor serie de *Arduino* comprobar los grados y sentido de giro del motor unipolar para los siguientes grados: 45, 90, 180, 360
- Para el servo motor comprobar las siguientes posiciones: 0, 45, 90, 135, 180

4.5. Abrir *Visual Studio* y crear un nuevo proyecto .vb, el cual contendrá un puerto serial, un *Timer* y la siguiente forma:

4.6. Configurar los botones:

- Dentro del botón de CONECTAR deberán establecerse los parámetros para la comunicación Serial. Además, se deberá mostrar que se encuentra conectado.
- EL botón DESCONECTAR deberá cerrar el puerto serial y mostrar que se encuentra desconectado.
- Los botones HORARIO, ANTIHORARIO y SERVOMOTOR, enviarán las letras previamente asignadas para cada uno de estos modos, es decir, HORARIO deberá enviar al puerto serial una “x”, ANTIHORARIO una “y” y SERVOMOTOR una “s”.
- El botón EJECUTAR, será el encargado de enviar el valor en grados que se encuentren en un *TextBox* asignado a GRADOS DE GIRO; a su vez, mostrar el valor en la *ProgressBar.* Además, pasará el valor enviado a la *ListBox*, la cual quedará como registro de los grados ejecutados
- EL botón LIMPIAR, simplemente dejará vacíos todos los *TextBox* y pondrá la *ProgressBar* en cero.
- El botón FINALIZAR, mostrará un mensaje el cual indicará si el usuario desea o no terminar la simulación, de ser “SI” la ventana se cerrará y el programa terminará.

- Dentro del *TIMER,* se deberá establecer las condiciones para el buffer de salida, y en caso de necesitarlo se puede optar por un buffer de entrar, en caso de que se quieran leer los datos que envía el *Arduino*.

4.7. Verificar resultados con los datos del punto 4.4, conectando la interfaz al puerto COM asignado al *Arduino* Mega.

5. BIBLIOGRAFÍA

[1] *Arduino*, «*ARDUINO MEGA 2560 REV3*,» 2018. [En línea]. *Available*: https://store.*Arduino*.cc/usa/*Arduino*-mega-2560-rev3. [Último acceso: 10 Febrero 2018].

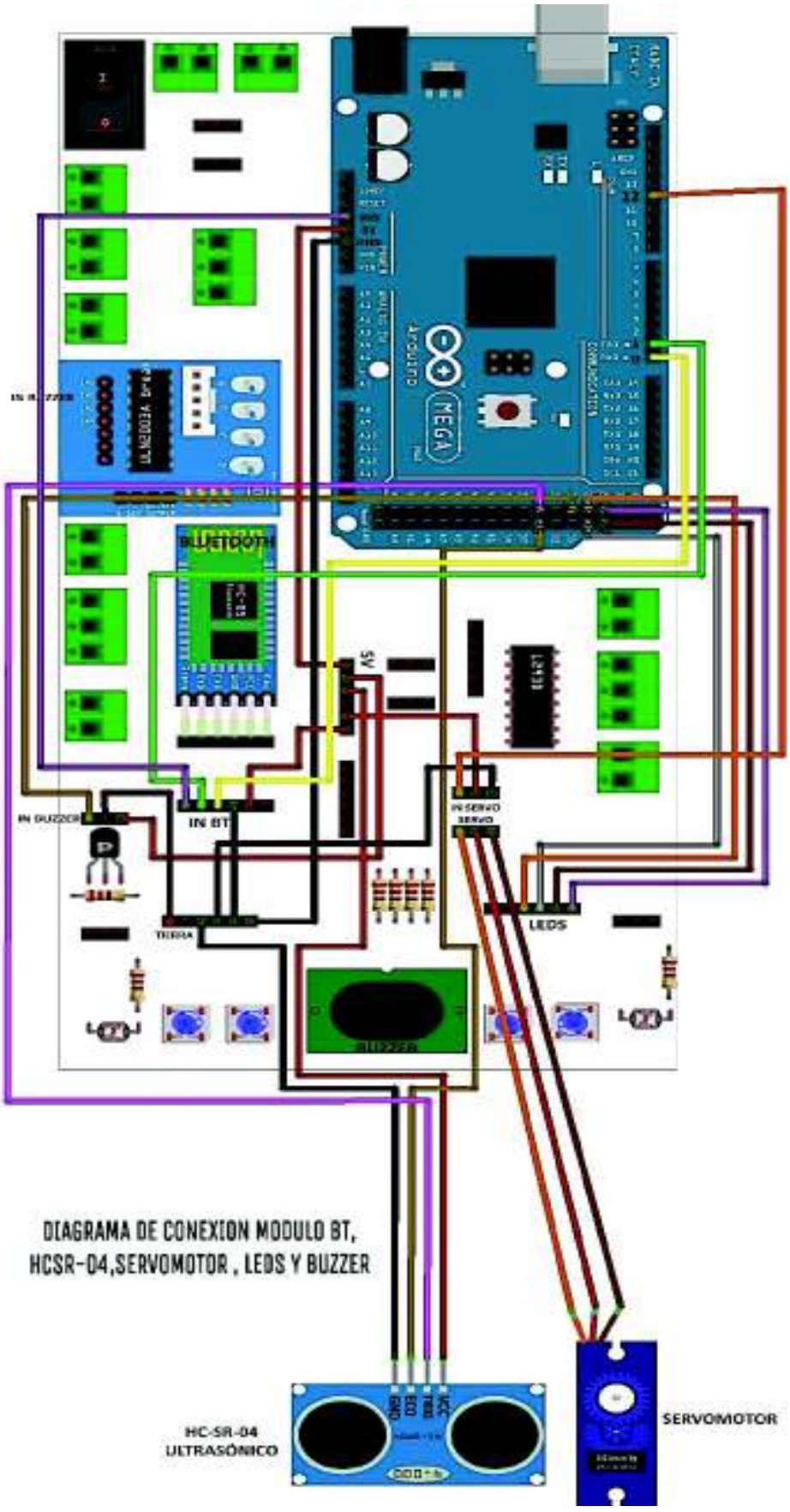
[2] P. Alcalde, Electrotecnia, España: Paraninfo S.A, 2003.

[3] AOSONG, «*Temperature and humidity module*,» 2018. [En línea]. *Available*: https://akizukidenshi.com/download/ds/aosong/DHT11.pdf. [Último acceso: 10 Febrero 2018].

[4] ELECFreaks, «*Ultrasonic Ranging Module HC-SR04*,» 2018. [En línea]. *Available*: http://www.micropik.com/PDF/HCSR04.pdf. [Último acceso: 10 Febrero 2018].

[5] *Arduino*, «*Language Reference - FUNCTIONS*,» 2018. [En línea]. *Available*: https://www.*arduino*.cc/reference/en/#functions. [Último acceso: 10 Febrero 2018].

DIAGRAMA DE CONEXIONES – PRÁCTICA 2



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

HOJA GUÍA

PRÁCTICA 3

1. TEMA: Control de los motores de la estructura utilizando Módulo *Bluetooth*.

2. OBJETIVOS

- Manejar el funcionamiento del módulo *Bluetooth* HC-05 mediante el manejo de comandos de comunicación serial, para aplicaciones de comunicación inalambrica.
- Controlar cuatro motores DC, utilizando el *Driver* L293D, para movilidad de la estructura.
- Desarrollar una App que permita controlar la módulo y sus interfaces desde el teléfono celular, utilizando App Inventor.

3. TRABAJO PREPARATORIO

3.1 Cuestionario

- Revisar los diagramas de conexiones, incluidos al final del documento.
- Consulte sobre los siguientes puntos para el IDE *Arduino*:
  - *Librería Servo.h*
  - *Serial.begin()*
  - *Serial.read()*
  - *Serial.println()*
  - *Pinmode ()*
  - *digitalWrite()*
  - *Switch*
  - Conexión del *Arduino* Mega a la PC y apertura del Monitor Serie
  - Configuración módulo HC-05 mediante comandos AT
- Breve descripción de *Driver* L293D, Servo motor SG90, módulo Hc-05 y *Arduino* Mega.
- Breve descripción de cómo realizar una aplicación para SO Android utilizando App inventor.

4. DESCRIPCIÓN DE LAS ACTIVIDADES Y PROCEDIMIENTO DE LA PRÁCTICA

4.1. Realizar las conexiones, del *Driver* L293D, módulo *Bluetooth* Hc-05, HC-SR04, *Buzzer*, y *LED*s, como especifica el docente.

4.2. Elaborar un programa que permita recibir datos por el puerto serial, los cuales serán interpretados por el *Arduino* para realizar las siguientes funciones:

- w = avanzar
- s = retroceder
- d = girar derecha
- a = avanzar hacia la izquierda
- 2 = encender *Buzzer* por 100ms
- 4 = encender *LED*s
- 5 = apagar *LED*s

4.3. Elaborar un código que permita un funcionamiento autónomo de la estructura, en el cual se deben considerar los siguientes aspectos:

- El servomotor debe ubicarse en tres ángulos (0°, 90°, 180°), siempre y cuando se encuentre en modo automático.
- El sensor HC-SR04 tiene que medir la distancia en los tres ángulos que el servomotor permite.
- El programa debe analizar las distancias medidas para que la estructura elija la dirección con mayor distancia.
- A este modo autónomo se ingresará cuando el monitor serie detecte un 1 y saldrá cuando detecte un 0.

4.4. Crear una aplicación en Android utilizando App Inventor que permita el control del módulo inalámbricamente en forma automática y manual. La aplicación debera enviar los caracteres mencionados en el punto 4.2.

5. BIBLIOGRAFÍA

[1] *Arduino*, «*Functions*,» [En línea]. *Available*: https://www.a*rduino*.cc/en/Reference/FunctionDeclaration. [Último acceso: 12 Febrero 2018].

[2] PanamaHitek, «Comunicación Serial con Arduino,» [En línea]. *Available*: http://panamahitek.com/comunicacion-serial-con-*Arduino*/. [Último acceso: 12 Febrero 2018].

[3] Naylamp, «Configuración del módulo *Bluetooth* Hc-05,» [En línea]. *Available*: http://www.naylampmechatronics.com/blog/24_configuracion-del-modulo-bluetooth-hc-05-usa.html. [Último acceso: 12 Febrero 2018].

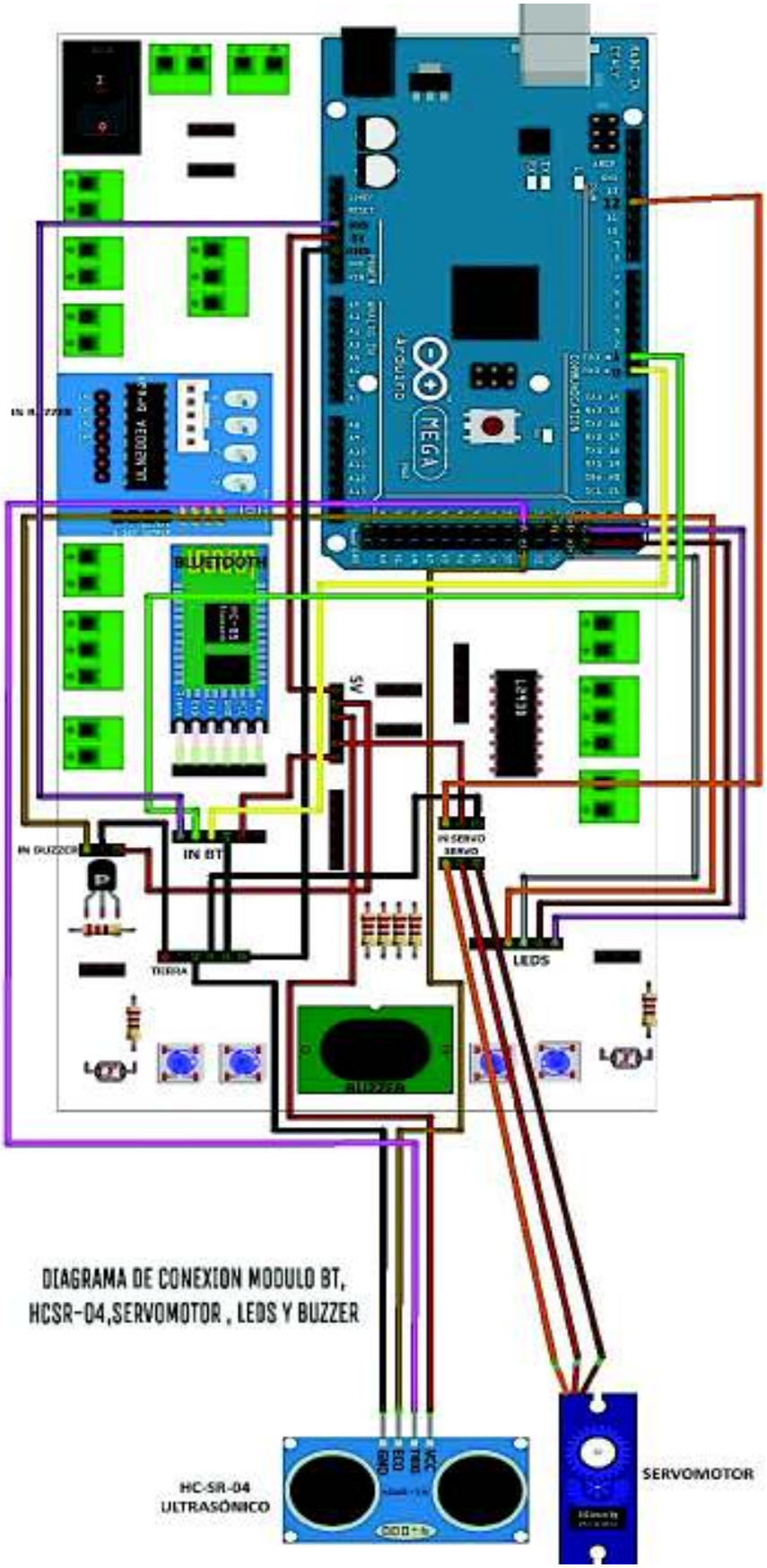
[4] Texas Instruments, «L293x,» [En línea]. *Available*: http://www.ti.com/lit/ds/symlink/l293.pdf. [Último acceso: 12 Febrero 2018].

[5] Aprendiendo *Arduino*, «Bluetooth en *Arduino*,» [En línea]. *Available*: https://aprendiendo a*rduino*.wordpress.com/tag/hc-05/. [Último acceso: 12 Febrero 2018].

[6] Akizukidenshi, «SG90,» [En línea]. *Available*: http://akizukidenshi.com/download/ds/towerpro/SG90.pdf. [Último acceso: 12 Febrero 2018].

[7] *Arduino*, «*Arduino* Mega 2560 REV3,» [En línea]. *Available*: https://store.*Arduino*.cc/*Arduino*-mega-2560-rev3. [Último acceso: 26 Noviembre 2017].

DIAGRAMAS DE CONEXIÓN – PRÁCTICA 3



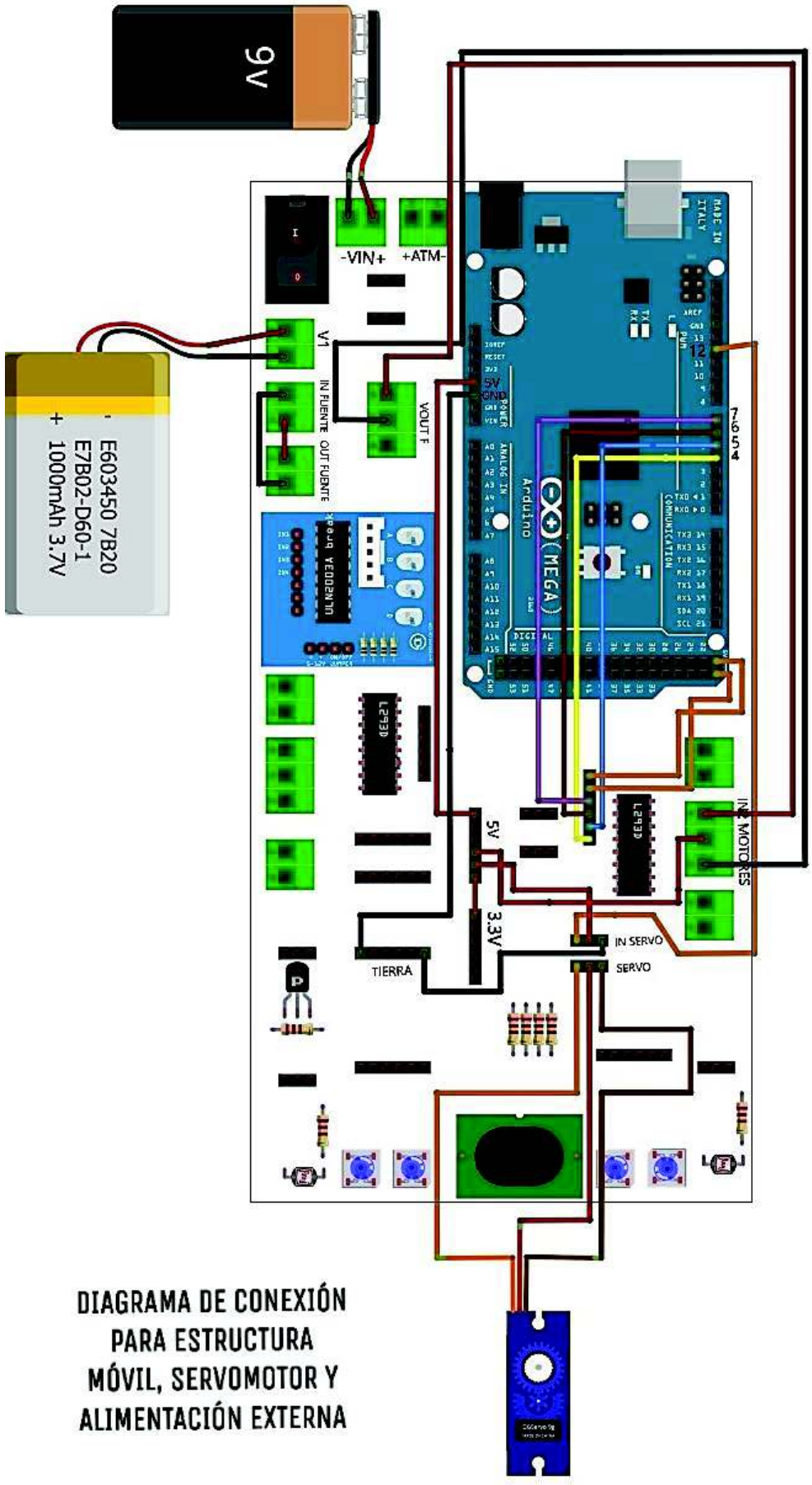


DIAGRAMA DE CONEXIÓN PARA ESTRUCTURA MÓVIL, SERVOMOTOR Y ALIMENTACIÓN EXTERNA

B-2:

Hojas guía para el instructor

ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

HOJA GUÍA

PRÁCTICA 1

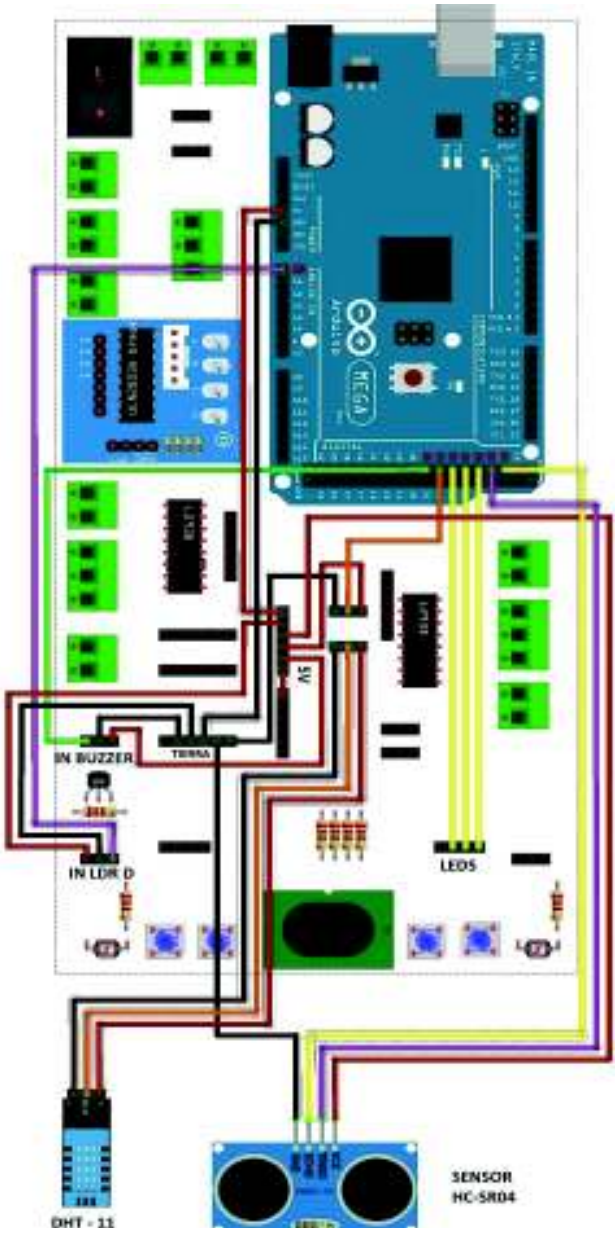
1. TEMA: Control de sensores

2. DESARROLLO DE LA PRÁCTICA

NOTA: Tiempo estimado para realizar la presente práctica de laboratorio tanto software como hardware es de 1 hora con 30 minutos. Se recomienda además que un alumno se encargue de las conexiones mientras el otro inicia con la programación del *Arduino*.

DIAGRAMA DE CONEXIONES

Realizar las conexiones de los sensores, *Buzzer* y *LED*s como se muestra a continuación.



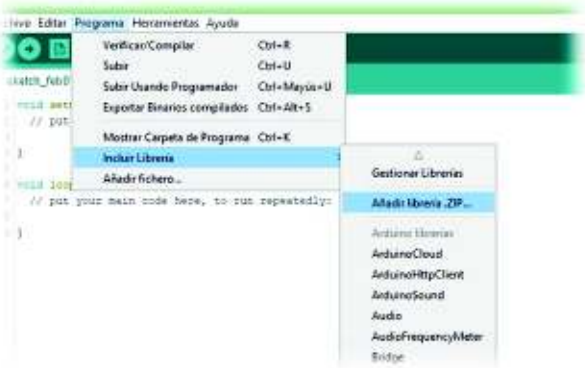
PROGRAMA *ARDUINO*

Ejecutar el IDE de *Arduino*.

Una vez ejecutado se mostrará la siguiente ventana en la cual se digitará el código.

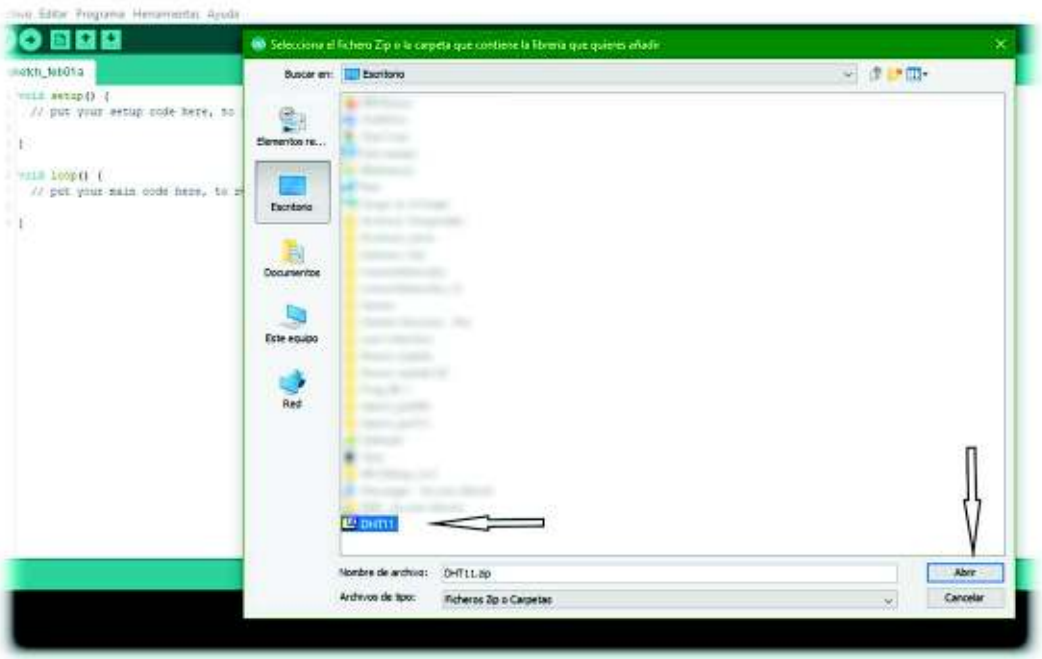
Previo a la digitación del programa se debe añadir dos librerías; una para el manejo del sensor DHT-11 y otra para realizar un programa que ejecute tareas múltiples. Para ello se proporcionarán los archivos .zip correspondientes a las librerías y se procede con la instalación como se muestra en los siguientes pasos:

- Paso 1: Ir a la Barra de menú del IDE de *Arduino*/Incluir Librería/Añadir librería .ZIP..



- Paso 2: Después se mostrará un explorador de archivos en el que se debe localizar el archivo DHT11.zip. Una vez encontrado el archivo se procede a hacer clic en Abrir. Esperar un momento y dejar que el IDE realice la instalación.

*NOTAS: *De preferencia guardar el archivo DHT11.zip y pt.zip en el escritorio para una rápida localización.*
*** Los pasos son útiles para la instalación de cualquier otra librería.*



Proceder con la digitación del código en el IDE de *Arduino* según los siguientes pasos:

- **Paso 1:** Importar librerías y declarar las variables necesarias, antes del *void setup().*

```
#include <pt.h>
#include <DHT11.h>

#define luzPin      A0
#define echoPin     24
#define triggerPin  26
#define ledPin_1    28
#define ledPin_2    30
#define ledPin_3    32
#define buzzerPin   36
const int dht11Pin = 34;

DHT11 dht11(dht11Pin);

struct pt secuencia_1;
struct pt secuencia_2;
struct pt secuencia_3;

int     varError;
float   luzMedida,
        disMedida,
        temMedida,
        humMedida;
long    tiempo;

void setup()
{
```

- **Paso 2:** Dentro de la función *setup(),* designar puertos de entrada, puertos de salida, activar el puerto serie a 9600 baudios e inicializar los protothreads (PT_INIT).

```
26  void setup()
27  {
28      pinMode(echoPin,    INPUT);
29      pinMode(triggerPin, OUTPUT);
30      pinMode(dht11Pin,   INPUT);
31      pinMode(ledPin_1,   OUTPUT);
32      pinMode(ledPin_2,   OUTPUT);
33      pinMode(ledPin_3,   OUTPUT);
34      pinMode(buzzerPin,  OUTPUT);
35
36      Serial.begin(9600);
37
38      PT_INIT(&secuencia_1);
39      PT_INIT(&secuencia_2);
40      PT_INIT(&secuencia_3);
41  }
```

- **Paso 3:** En la función principal *loop()*, llamar a las funciones protothreads *leerSensor(&secuencia_1), enviarDatosSerial(&secuencia_2) y controlDispositivos(&secuencia_3).*

```
43  void loop()
44  {
45      leerSensor(&secuencia_1);
46      enviarDatosSerial(&secuencia_2);
47      controlDispositivos(&secuencia_3);
48  }
```

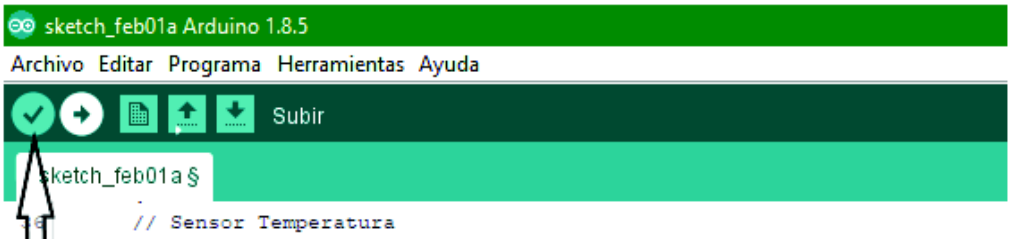
- **Paso 4:** Crear las funciones protothreads:

```
50  void leerSensor(struct pt *pt)
51  {
52      PT_BEGIN(pt);
53
54      static long t = 0;
55
56      do
57      {
58          /** LDR */
59          luzMedida = analogRead(luzPin);
60
61          /**HC-SR04 **/
62
63          digitalWrite(triggerPin, LOW);
64          delayMicroseconds(2);
65          digitalWrite(triggerPin, HIGH);
66          delayMicroseconds(10);
67          digitalWrite(triggerPin, LOW);
68          tiempo = pulseIn(echoPin, HIGH);
69          disMedida = tiempo / 58;
70
71          /** DHT-11 */
72          varError = dht11.read(humMedida, temMedida);
73          t = millis();
74          PT_WAIT_WHILE (pt, (millis()-t)<1000);
75
76      }while(true);
77
78      PT_END(pt);
79  }
```

```
void enviarDatosSerial(struct pt *pt)
{
    PT_BEGIN(pt);
    static long t = 0;

    do{
        Serial.print("LDR: ");
        Serial.println(luzMedida);

        Serial.print("DISTANCIA: ");
        Serial.print(disMedida);
        Serial.println(" (cm)");

        if (varError == 0)
        {
            Serial.print("TEMPERATURA: ");
            Serial.print(temMedida);
            Serial.println(" (°C)");

            Serial.print("HUMEDAD: ");
            Serial.print(humMedida);
            Serial.println(" (%RH)");
            Serial.println("***************************");
        }
        else
        {
            Serial.println();
            Serial.print("Error No: ");
            Serial.println(varError);
            Serial.println("***************************");
        }
        t = millis();
        PT_WAIT_WHILE (pt, (millis()-t)<1000);

    } while(true);
    PT_END(pt);
}

void controlDispositivos(struct pt *pt)
{
    PT_BEGIN(pt);

    static long t = 0;

    do
    {

        /** >>>> LDR <<<< */
        if (luzMedida <= 100)
        {
            digitalWrite(ledPin_1, HIGH);
        }
        else
        {
            digitalWrite(ledPin_1, LOW);
        }

        /** >>>> HC-SR04 <<<< */
        if (disMedida >= 30)
        {
            digitalWrite(buzzerPin, LOW);
        }
```

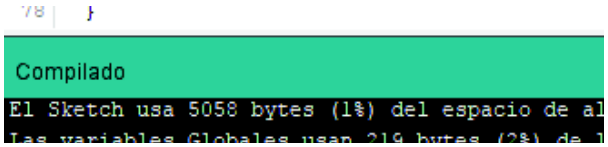
```
        else if(disMedida >= 20 && disMedida < 30)
        {

            digitalWrite(buzzerPin, HIGH);
            t = millis();
            PT_WAIT_WHILE (pt, (millis()-t)<1000);

            //noTone(buzzerPin);
            digitalWrite(buzzerPin, LOW);
            t = millis();
            PT_WAIT_WHILE (pt, (millis()-t) < 1000);

        }
        else if(disMedida >= 10 && disMedida < 20)
        {

            digitalWrite(buzzerPin, HIGH);
            t = millis();
            PT_WAIT_WHILE (pt, (millis()-t) < 500);

            digitalWrite(buzzerPin, LOW);
            t = millis();
            PT_WAIT_WHILE (pt, (millis()-t) < 500);

        }
        else if(disMedida < 10)
        {

            digitalWrite(buzzerPin, HIGH);
            t = millis();
            PT_WAIT_WHILE (pt, (millis()-t) <100);

            digitalWrite(buzzerPin, LOW);
            t = millis();
            PT_WAIT_WHILE (pt, (millis()-t) <100);

        }

        /** >>>> DHT-11 <<<< */
        if (temMedida >= 25)
        {
            digitalWrite(ledPin_2, HIGH);
        }
        else
        {
            digitalWrite(ledPin_2, LOW);
        }
        if (humMedida <= 65)
        {
            digitalWrite(ledPin_3, HIGH);
        }
        else
        {
            digitalWrite(ledPin_3, LOW);
        }

        t = millis();
        PT_WAIT_WHILE (pt, (millis()-t) < 1);

    } while (true);
    PT_END(pt);
}
```

Una vez terminado el programa es necesario verificarlo. Para ello se debe hacer clic sobre el botón verificar que está ubicado en la parte superior izquierda de la ventana principal del IDE de *Arduino*.



NOTA: **Si el programa no ha sido guardado en ningún momento del proceso anterior, el IDE solicitará que se guarde el programa antes de verificarlo. Para ello asigne un nombre al programa, después clic en guardar y continuará con la verificación.*

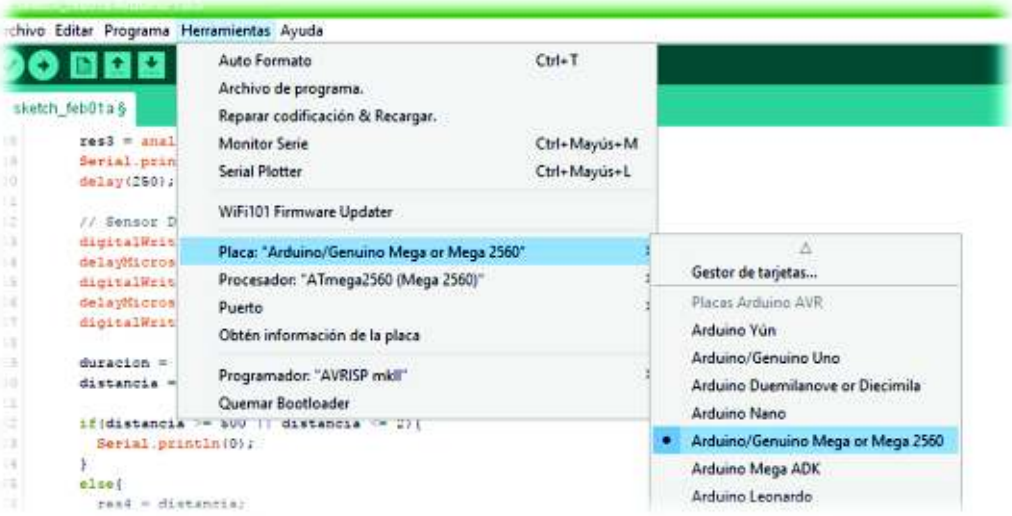
Si el programa está escrito correctamente se mostrará un mensaje de compilado en la parte inferior izquierda de la ventana principal del IDE de *Arduino*



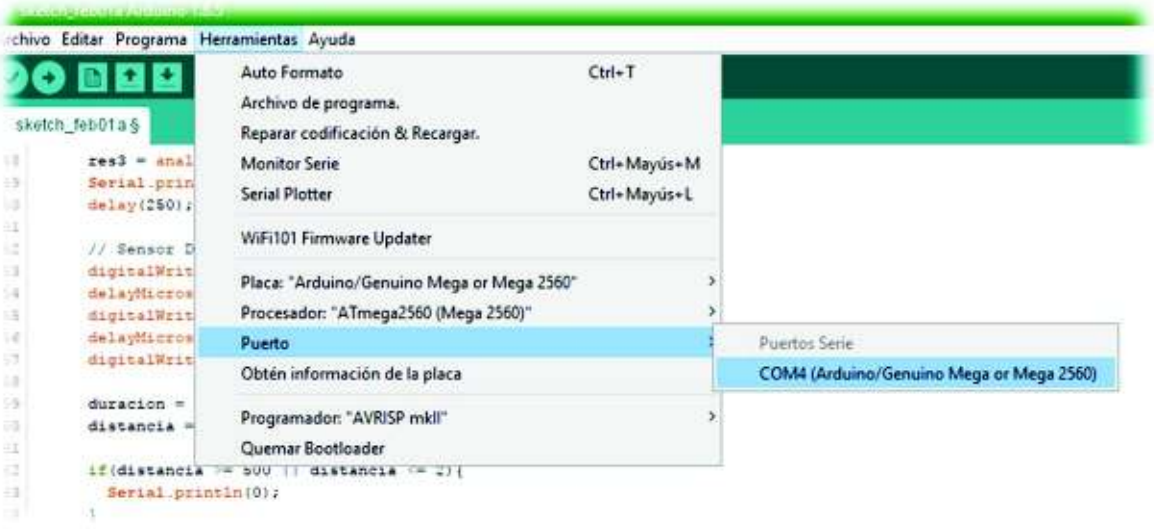
Una vez verificado el código se procederá a subirlo al *Arduino* Mega del módulo didáctico. Seguir los siguientes pasos para poder hacerlo.

- **Paso 1:** Conectar el *Arduino* al computador, utilizando el cable USB proporcionado.
  - Conector USB tipo A ----- PC
  - Conector USB tipo B ----- *Arduino*

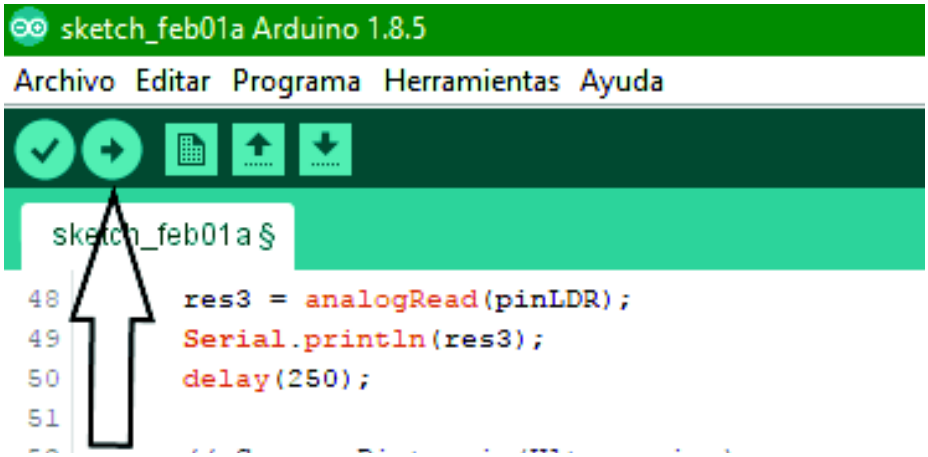
- **Paso 2:** Seleccionar el tipo de *Arduino*.



- **Paso 3**: Configurar el puerto.
  *NOTA: *El puerto no necesariamente debe ser el mismo que el mostrado en la imagen.*

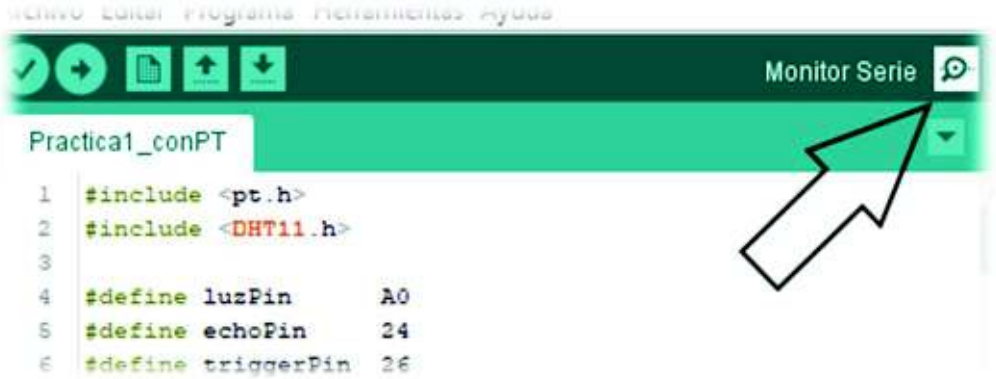


- **Paso 4:** Subir al *Arduino*. Clic en el botón subir y esperar que el IDE finalice.

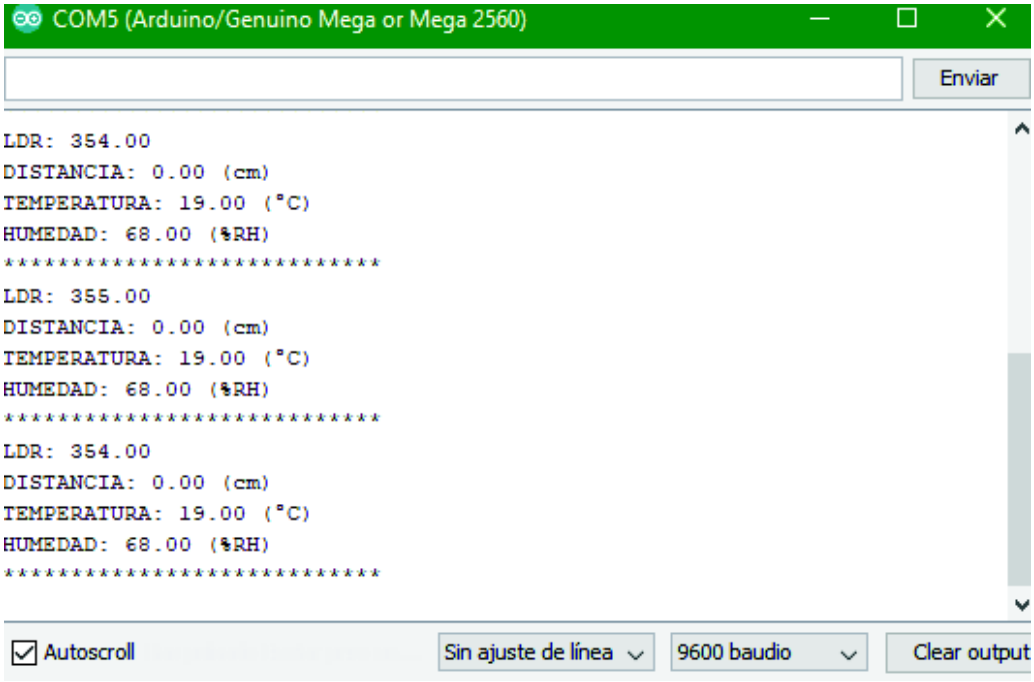


Una vez programado el *Arduino* Mega, mantenerlo conectado y abrir el monitor serie del IDE de *Arduino* para verificar que el microcontrolador está enviando datos al computador. De la siguiente manera:

Paso 1: Para abrir el monitor serie, hacer clic sobre la siguiente opción.



Paso 2: Se abrirá una ventana similar a la siguiente y se mostrará la información de los sensores.



3. RECOMENDACIONES

- Se recomienda utilizar la líbreria protothread para realizar multitareas, ya que facilita que las funciones se realicen con la "frecuencia suficiente", es decir, evita perdida de datos de los sensores y retardos en el tiempo de reacción de los dispositivos que dependen de los sensores.
- Es importante verificar que el puerto COM designado para el *Arduino* Mega sea seleccionado en el menú del IDE de *Arduino*, para evitar errores en la programación del *Arduino* y la transmisión y recepción de datos a través del puerto serie.

4. BIBLIOGRAFÍA

[1] *Arduino, «ARDUINO MEGA 2560 REV3,» 2018. [En línea]. Available: https://store.Arduino.cc/ usa/Arduino-mega-2560-rev3. [Último acceso: 10 Febrero 2018].*

[2] *P. Alcalde, Electrotecnia, España: Paraninfo S.A, 2003.*

[3] *AOSONG, «Temperature and humidity module,» 2018. [En línea]. Available: https://akizukidenshi.com/download/ds/aosong/DHT11.pdf. [Último acceso: 10 Febrero 2018].*

[4] *ELECFreaks, «Ultrasonic Ranging Module HC-SR04,» 2018. [En línea]. Available: http://www. micropik.com/PDF/HCSR04.pdf. [Último acceso: 10 Febrero 2018].*

[5] *Arduino, «Language Reference - FUNCTIONS,» 2018. [En línea]. Available: https://www.arduino.cc/reference/en/#functions. [Último acceso: 10 Febrero 2018].*

ESCUELA POLITÉCNICA NACIONAL
ESCUELA DE FORMACIÓN DE TECNÓLOGOS
HOJA GUÍA
PRÁCTICA 2

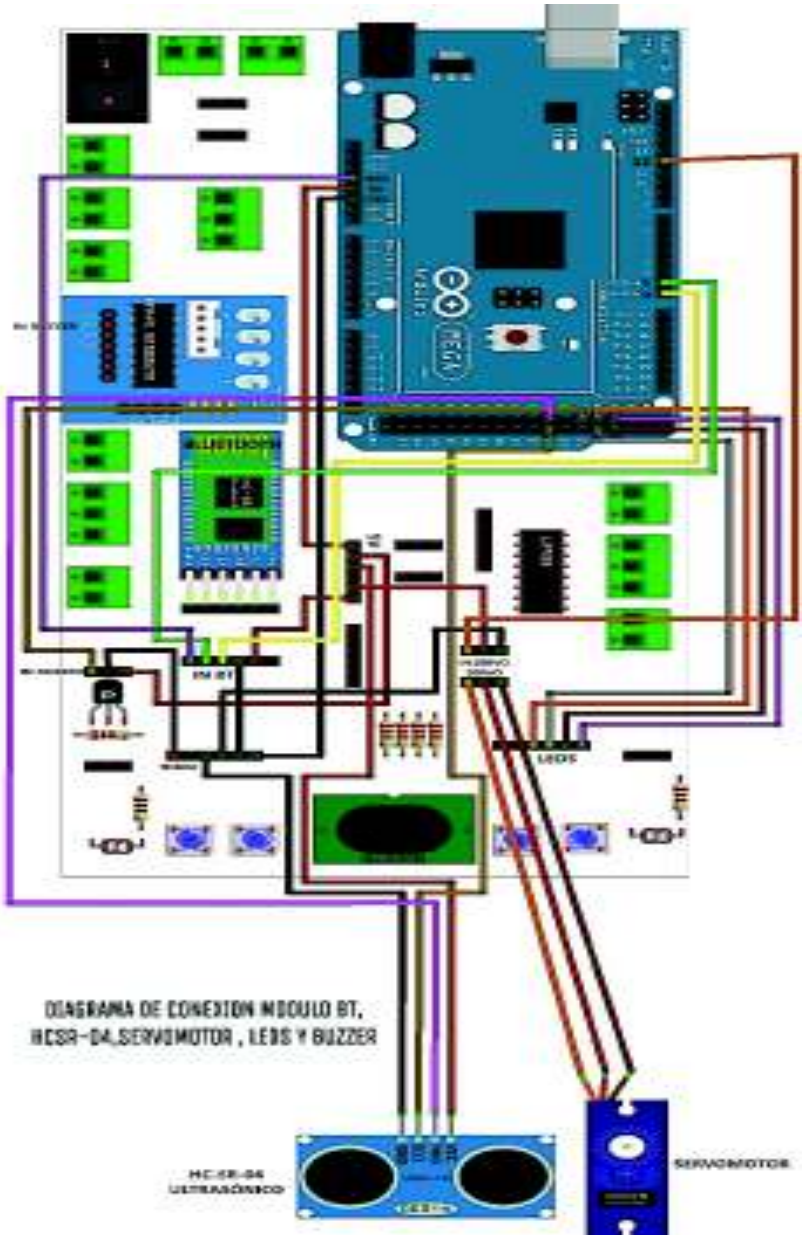
1. TEMA: Control de motores

2. DESARROLLO DE LA PRÁCTICA

NOTA: Tiempo estimado para realizar la presente práctica de laboratorio tanto software como hardware es de 2 horas con 30 minutos. Se recomienda además que un alumno se encargue de las conexiones mientras el otro inicia con la programación del *Arduino* o de la interfaz.

DIAGRAMA DE CONEXIONES

Realizar las conexiones del servomotor y motor a pasos unipolar como se muestra en el siguiente diagrama.



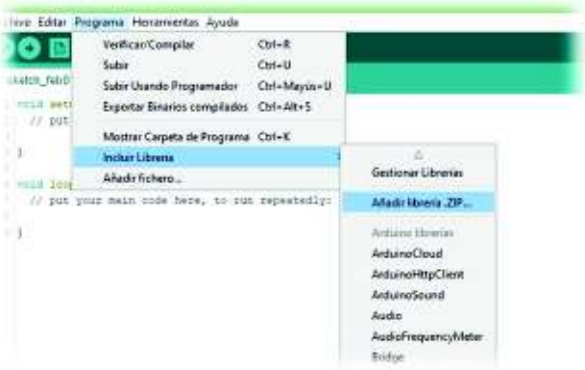
PROGRAMA *ARDUINO*

Ejecutar el IDE de *Arduino*.

Una vez ejecutado se mostrará la siguiente ventana en la cual se digitará el código.

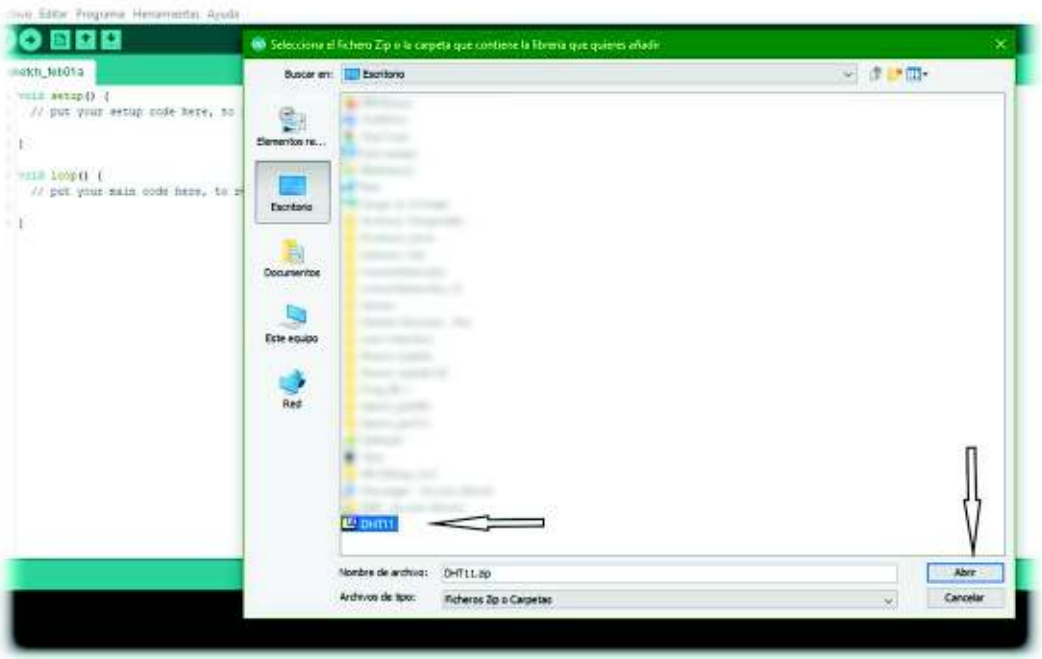
En caso de que el IDE de *Arduino* no posea la Librería *Servo.h*, se realizan los siguientes pasos para incluir una librería previamente descargada de la red y se procede con la instalación como se muestra en los siguientes pasos:

- **Paso 1:** Ir a la Barra de menú del IDE de *Arduino*/Incluir Librería/Añadir librería .ZIP..



- **Paso 2:** Después se mostrará un explorador de archivos en el que se debe localizar el archivo DHT11.zip. Una vez encontrado el archivo se procede a hacer clic en Abrir. Esperar un momento y dejar que el IDE realice la instalación.

*NOTAS: *De preferencia guardar el archivo DHT11.zip y pt.zip en el escritorio para una rápida localización.*
*** Los pasos son útiles para la instalación de cualquier otra librería.*



Proceder con la digitación del código en el IDE de *Arduino* para satisfacer el literal 4.2 de la hoja guía del estudiante.

- **Paso 1:** Importar librerías y declarar las variables necesarias, antes del void setup().

```
int m1=4; // declaracion de pines para motor
int m2=5;
int m3=6;
int m4=7;
String bufferString = ""; // variable string para cadena de puerto serial
float n;                  //variable que va almacenar el numero de pasos a realizar

int tiempo;               //variable para delay
```

- **Paso 2:** Se establecen el modo de trabajo de los pines y otras características.

```
void setup() {
  Serial.begin(9600);     //se considera la velocidad para la comunicacion serial en 9600
 tiempo = 5;              // el delay sera de 5 milisegundos
  pinMode(m1, OUTPUT);    //configuracion de pines de motor como salidas
  pinMode(m2, OUTPUT);
  pinMode(m3, OUTPUT);
  pinMode(m4, OUTPUT);


}
```

- **Paso 3:** En el bucle principal se procede a realizar la lectura del puerto serial, y a su vez almacena el valor ingresado en una variable entera. Luego de lo cual, se llama a las funciones que contienen los giros.

```
void loop() {
  bufferString = "";                    // la cadena empieza vacia
  if (Serial.available() > 0) {         //secuencia de lectura del puerto serie para cadena de caracteres
    delay(20);
    while (Serial.available() > 0) {
    bufferString += (char)Serial.read();
    }
    Serial.println(bufferString);
    n = bufferString.toInt();           //convierte el bufferString  en un valor entero y se lo asigna a n
    n= n /0.65;                         //realiza el calculo de pasos a realizar
    if(n>0){horario();}                 //compara si el valor es mayor que cero y llama a horario
    delay(100);
    antihorario();                      //llama a antihorario con el mismo valor de n
    }

  }}
```

*Las siguientes funciones se deben incluir después del *"void loop()".*

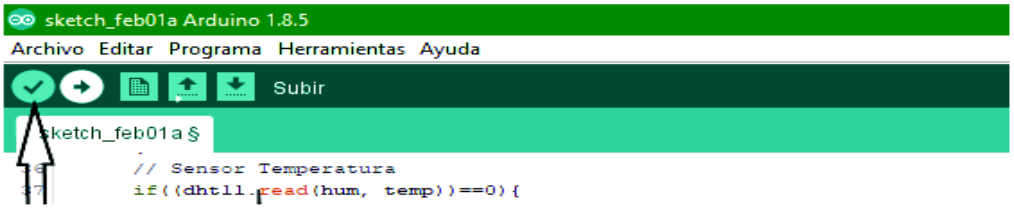
Función sentido horario que contiene el encendido simple, el cual funciona para unipolar como para bipolar.

```
 void horario(){
for (int i=0; i<n; i++){                //bulce for para el numero de pasos que debe dar hasta alcanzar los "n" grados

  digitalWrite(m1,HIGH);                //secuencia de movimiento horario
  digitalWrite(m2,LOW);
  digitalWrite(m3,LOW);
  digitalWrite(m4,LOW);
  delay(tiempo);
  digitalWrite(m1,LOW);
  digitalWrite(m2,HIGH);
  digitalWrite(m3,LOW);
  digitalWrite(m4,LOW);
  delay(tiempo);
  digitalWrite(m1,LOW);
  digitalWrite(m2,LOW);
  digitalWrite(m3,HIGH);
  digitalWrite(m4,LOW);
  delay(tiempo);
  digitalWrite(m1,LOW);
  digitalWrite(m2,LOW);
  digitalWrite(m3,LOW);
  digitalWrite(m4,HIGH);
  delay(tiempo);
  digitalWrite(m1,LOW);
  digitalWrite(m2,LOW);
  digitalWrite(m3,LOW);
  digitalWrite(m4,LOW );
  delay(tiempo);
}}
```

Función sentido antihorario que contiene el encendido simple, el cual funciona para unipolar como para bipolar.

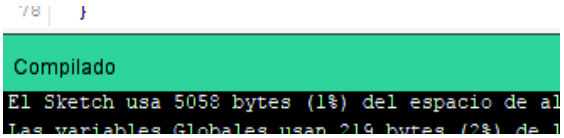
```
 void antihorario(){
for (int i=0; i<n; i++){                //bulce for para el numero de pasos que debe dar hasta alcanzar los "n" grados

  digitalWrite(m4,HIGH);                //secuencia de movimiento antihorario
  digitalWrite(m3,LOW);
  digitalWrite(m2,LOW);
  digitalWrite(m1,LOW);
  delay(tiempo);
  digitalWrite(m4,LOW);
  digitalWrite(m3,HIGH);
  digitalWrite(m2,LOW);
  digitalWrite(m1,LOW  );
  delay(tiempo);
  digitalWrite(m4,LOW);
  digitalWrite(m3,LOW);
  digitalWrite(m2,HIGH);
  digitalWrite(m1,LOW);
  delay(tiempo);
  digitalWrite(m4,LOW);
  digitalWrite(m3,LOW);
  digitalWrite(m2,LOW);
  digitalWrite(m1,HIGH);
  delay(tiempo);
  digitalWrite(m4,LOW);
  digitalWrite(m3,LOW);
  digitalWrite(m2,LOW);
  digitalWrite(m1,LOW);
   delay(tiempo);

}}
```

Una vez terminado el programa es necesario verificarlo. Para ello se debe hacer clic sobre el botón verificar que está ubicado en la parte superior izquierda de la ventana principal del IDE de *Arduino*.



NOTA: **Si el programa no ha sido guardado en ningún momento del proceso anterior, el IDE solicitará que se guarde el programa antes de verificarlo. Para ello asigne un nombre al programa, después clic en guardar y continuará con la verificación.*

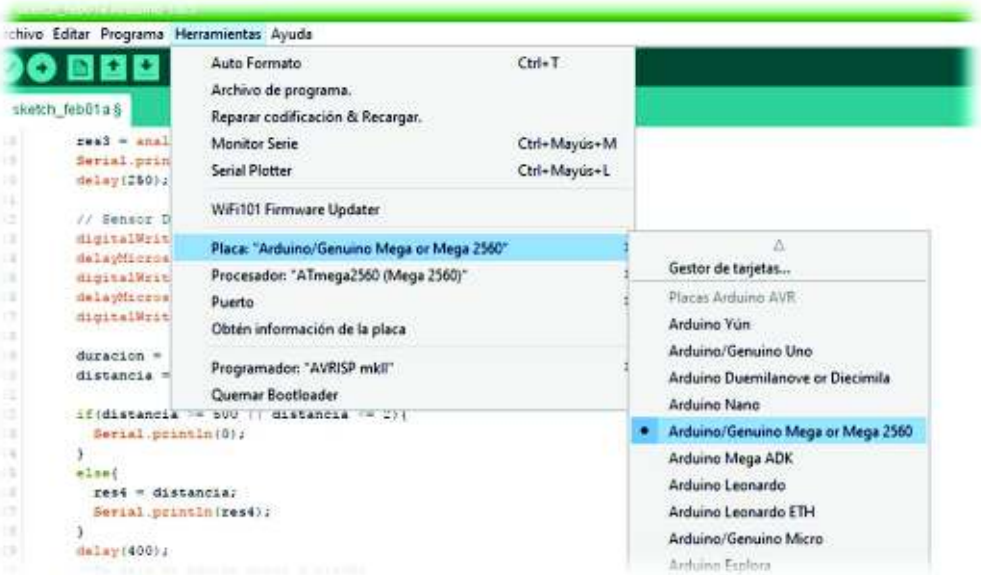
Si el programa está escrito correctamente se mostrará un mensaje de compilado en la parte inferior izquierda de la ventana principal del IDE de *Arduino*



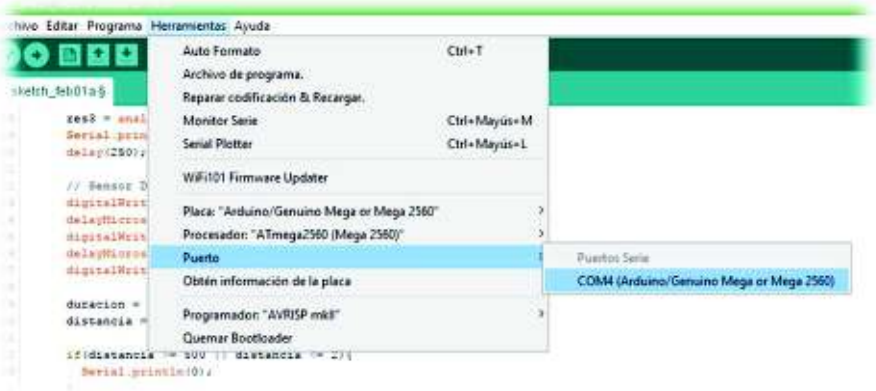
Una vez verificado el código se procederá a subirlo al *Arduino* Mega del módulo didáctico. Seguir los siguientes pasos para poder hacerlo.

- **Paso 1:** Conectar el *Arduino* al computador, utilizando el cable USB proporcionado.
  - Conector USB tipo A ----- PC
  - Conector USB tipo B ----- *Arduino*

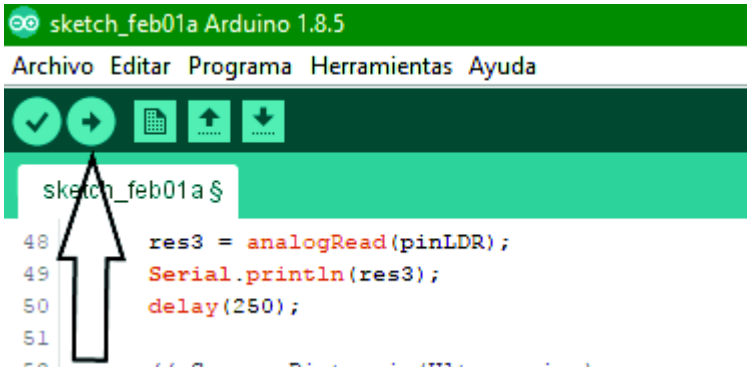
- **Paso 2:** Seleccionar el tipo de *Arduino*.



- **Paso 3:** Configurar el puerto.
  *NOTA: *El puerto no necesariamente debe ser el mismo que el mostrado en la imagen.*



- **Paso 4:** Subir al *Arduino*. Clic en el botón subir y esperar que el IDE finalice.



Para la prueba en el monitor serie, tan solo se debe abrir el monitor serie de la manera presentada en el preparatorio, y escribir el valor de grados que se desean girar.

Proceder con la digitación del código en el IDE de *Arduino* para el literal 4.3 de la hoja guía del estudiante, la siguiente manera:

Es recomendable iniciar un nuevo proyecto en el IDE de *Arduino*, ya que el siguiente código incluye varios cambios con respecto al realizado anteriormente.

Se declaran las variables, incluyen librerías y asignan pines a utilizar, antes del *"void setup()".*

```
#include <Servo.h>                  //libreria para servomotor parte del IDE
Servo servoMotor;                   //asignacion de un nombre al Servo
String cadena;                      //variable string para almacenar los valores
int i;                             //variable para ciclo for
char comando;                      // variable para comando switch
int sumas;                         //variable para conteo de pasos
int m1=4;                          //
int m2=5;                             // asignacion de pines para motor
int m3=6;
int m4=7;                          //
int tiempo;                        //variable para delay
```

Se establece el modo de operación de los pines, y otras características.

```
void setup() {
  Serial.begin(9600);               // inicializacion del puerto serial con velocidad 9600
  servoMotor.attach(12);            // asignacion del pin para el servo
  pinMode(m1, OUTPUT);              //declaracion de pines como salidas
  pinMode(m2, OUTPUT);
  pinMode(m3, OUTPUT);
  pinMode(m4, OUTPUT);
  tiempo = 3;                       //se establece el delay en 3 milisegundos
  }
```

El *void loop()* será el que contenga la sentencia *Switch* con todos los posibles casos, los cuales serán: horario, antihorario, servo motor y *reset.*

```
void loop() {
   cadena= " ";                     // se vacia la cadena decaracteres
   delay(100);
   comando = Serial.read();         //lee el contenido del puerto serial y lo almacena en comando
   delay(1000);                     //delay

switch(comando){                    //sentencia Switch

case 'x':                           //caso para sentido horario
   while(1){                        //bucle para poder leer la cadena completa del puerto serial
if(Serial.available()){
  while(Serial.available())
  {
  char c= Serial.read();
  if((c!='\r')&&(c!='\n'))
  {
    cadena+=c;
    }
delay(20);
  }
  sumas=cadena.toInt();             // convierte la cadena tipo string a entera
  sumas=sumas/0.65;                 // se divie el valor ingresado en grados, para el grado que da el motor
  if(sumas>0){horario();return;}    //se verifica que el valor sea diferente de cero y llama a la funcion horario
}}


break;

case 'r':                           //caso para reset
    cadena= " ";                    // limpia la cadena de caracteres
    servoMotor.write(90);           //coloca el servo en 90 grados
break;

case 'y':                          //caso para sentido antihorario


while(1){                          //bucle para lectura de cadena completa del puerto serial
if(Serial.available()){
  while(Serial.available())
  {
  char c= Serial.read();
  if((c!='\r')&&(c!='\n'))
  {
    cadena+=c;
    }
delay(20);
  }
  sumas=cadena.toInt();            //convierte la cadena tipo string a entera
  sumas=sumas/0.65;                //se divide el valor ingresado en grados, para el grado que da el motor
  if(sumas>0){antihorario();return;}  // se verifica que el valor sea diferente de cero y llama a la funcion antihorario
}}

break;
```

```
case 's':                               // caso para servo motor

 while(1){                              //bulce para lectura de cadena completa del puerto serial
if(Serial.available()){
  while(Serial.available()>0)
  {
  char c= Serial.read();
  if((c!='\r')&&(c!='\n'))
  {
    cadena+=c;
    }
delay(20);
  }
  sumas=cadena.toInt();
  if((sumas<181)&&(sumas>-1)){servo(); return;} // verifica que el valor ingresado este entre 0 y  180
}}                                              // llama a la funcion servo

break;
}
  }
```

Dentro de cada caso, se procede a la lectura de los datos del puerto serial, para luego de esto llamar a las siguientes funciones según correspondan. Es importante recordar que las siguientes funciones se escribirán luego del "*void loop ()".*

Función que permite el giro antihorario del motor unipolar mediante el encendido simple:

```
void antihorario(){

for (int i=0; i<sumas; i++){
  //Secuencia para giro antihorario
  digitalWrite(m4,HIGH);
  digitalWrite(m3,LOW);
  digitalWrite(m2,LOW);
  digitalWrite(m1,LOW);
  delay(tiempo);
  digitalWrite(m4,LOW);
  digitalWrite(m3,HIGH);
  digitalWrite(m2,LOW);
  digitalWrite(m1,LOW);
  delay(tiempo);
  digitalWrite(m4,LOW);
  digitalWrite(m3,LOW);
  digitalWrite(m2,HIGH);
  digitalWrite(m1,LOW);
  delay(tiempo);
  digitalWrite(m4,LOW);
  digitalWrite(m3,LOW);
  digitalWrite(m2,LOW);
  digitalWrite(m1,HIGH);
  delay(tiempo);
  digitalWrite(m4,LOW);
  digitalWrite(m3,LOW);
  digitalWrite(m2,LOW);
  digitalWrite(m1,LOW);
  delay(tiempo);      }

}
```

Función que permite el giro horario del motor unipolar mediante el encendido simple:

```
void horario(){

for (int i=0; i<sumas; i++){
  //Secuencia para giro horario
  digitalWrite(m1,HIGH);
  digitalWrite(m2,LOW);
  digitalWrite(m3,LOW);
  digitalWrite(m4,LOW);
  delay(tiempo);
  digitalWrite(m1,LOW);
  digitalWrite(m2,HIGH);
  digitalWrite(m3,LOW);
  digitalWrite(m4,LOW);
  delay(tiempo);
  digitalWrite(m1,LOW);
  digitalWrite(m2,LOW);
  digitalWrite(m3,HIGH);
  digitalWrite(m4,LOW);
  delay(tiempo);
  digitalWrite(m1,LOW);
  digitalWrite(m2,LOW);
  digitalWrite(m3,LOW);
  digitalWrite(m4,HIGH);
  delay(tiempo);
  digitalWrite(m1,LOW);
  digitalWrite(m2,LOW);
  digitalWrite(m3,LOW);
  digitalWrite(m4,LOW);
  delay(tiempo);       }

  }
```

Finalmente se tiene la función que permite el control de giro del servomotor.

```
void servo( ){

  sumas=cadena.toInt();
   servoMotor.write(sumas);
  delay(1000);

    }
```

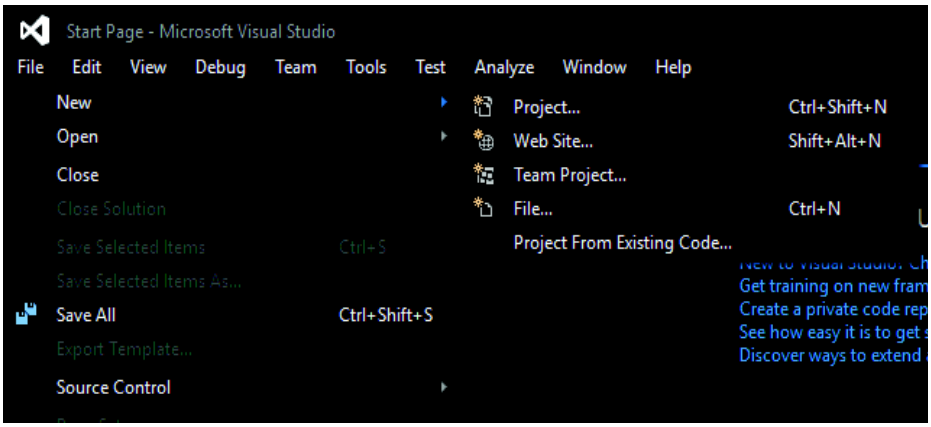
Se procede a cargar el programa como se lo realizó en el código anterior; posteriormente se realizan las pruebas de funcionamiento al igual que en el anterior literal, se las realiza con el monitor serie del *Arduino*.

Visual Studio

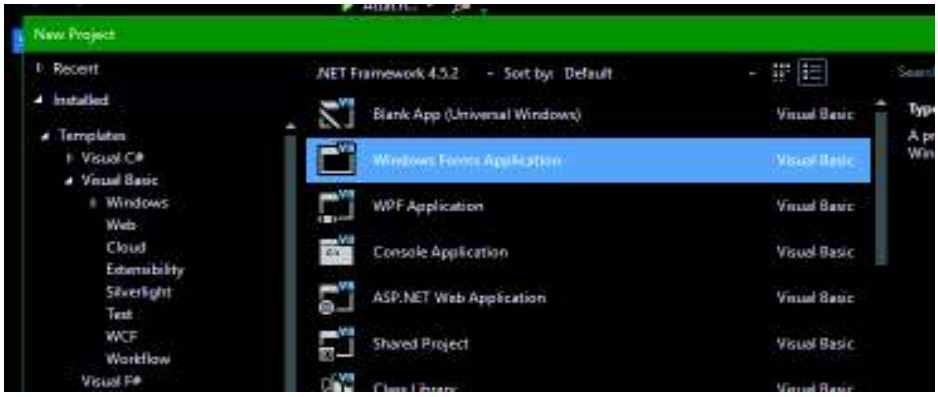
Abrir *Visual Studio.*

Crear un nuevo proyecto en lenguaje *Visual Basic*.

- **Paso 1:** Clic en File/Clic en New/Clic en Project…

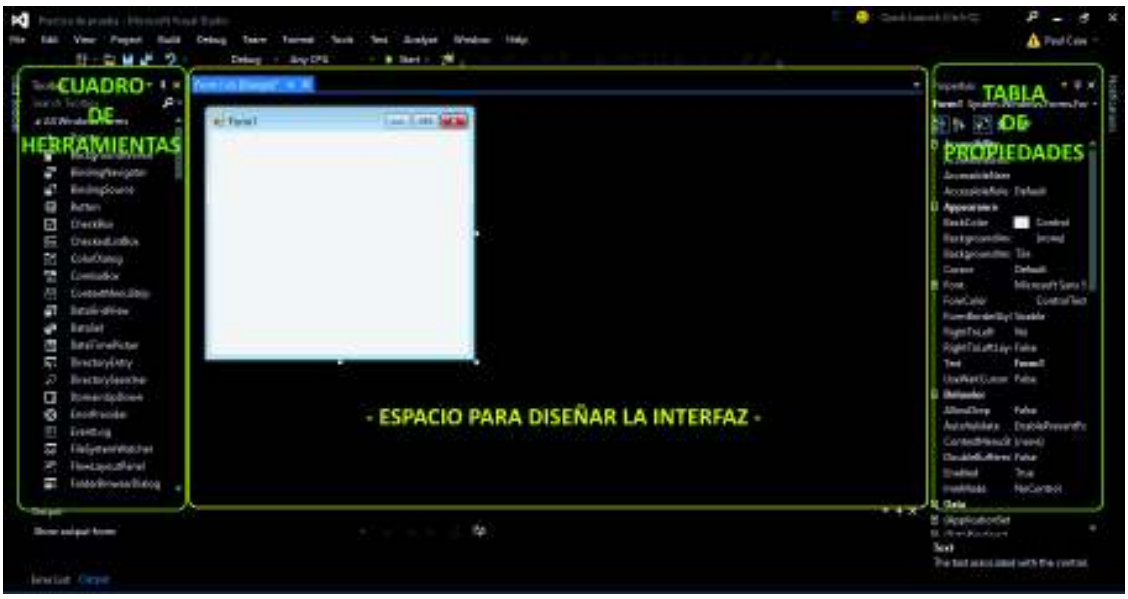


- **Paso2:** Especificar el lenguaje y el tipo de aplicación que se desea realizar.
  Clic en *Visual Basic*/Clic en Windows Forms Application/

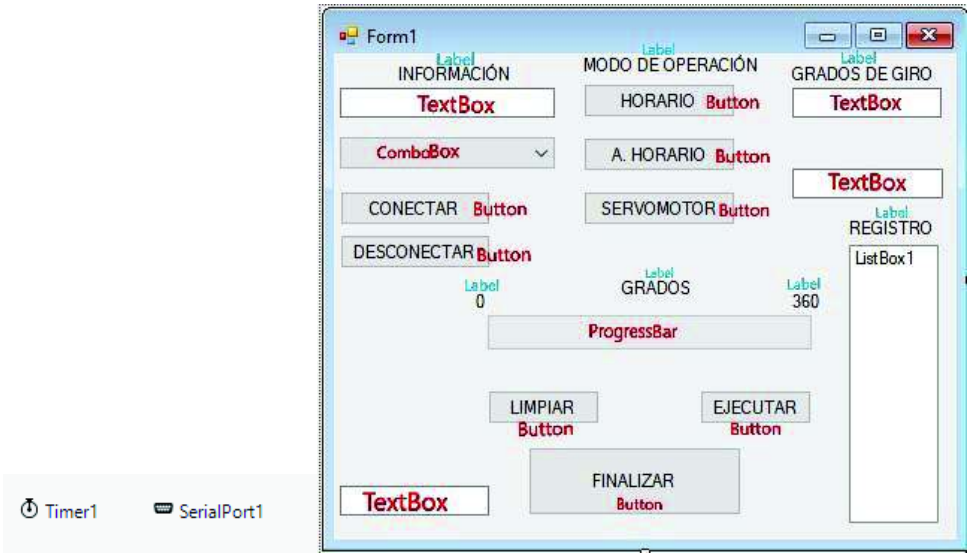


- **Paso 3:** Si se desea se puede configurar al proyecto una ubicación y un nombre deseado. Finalmente, clic en OK para crear el proyecto.

- **Paso 4:** El proyecto generará una pestaña llamada Form1.vb en el cual se agregarán los elementos deseados.

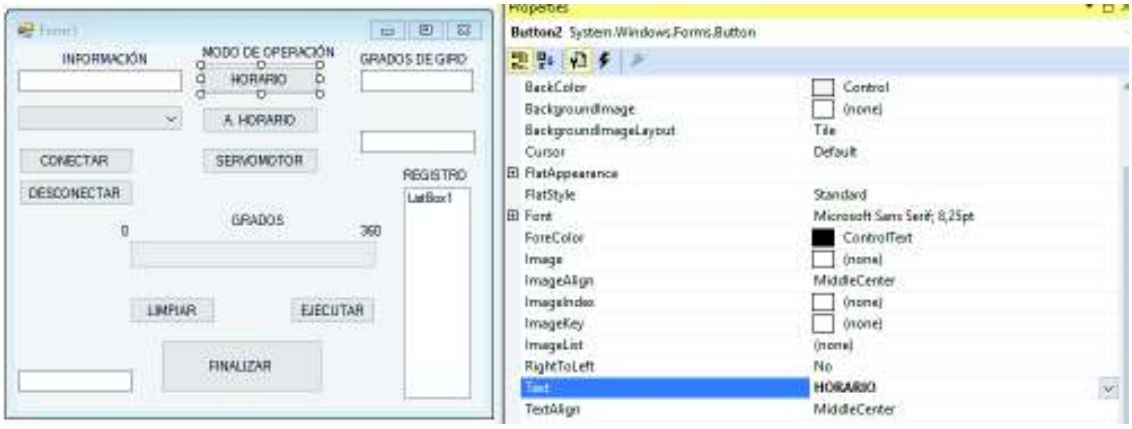


En el espacio dedicado para el diseño de la interfaz agregar los elementos necesarios hasta conseguir el siguiente diseño establecido en el literal 4.5 de la hoja guía. Para agregar un elemento basta con buscarlo en el cuadro de herramientas y arrastrarlo hasta la ventana de la interfaz. Añadir dos elementos extras del cuadro de herramientas llamados: *SerialPort y Timer.*



Las palabras INFORMACIÓN, MODO DE OPERACIÓN, GRADOS DE GIRO, REGISTRO, GRADOS 0, 360, son *Labels*, las cuales se las coloca desde el cuadro de herramientas al igual que los otros elementos.

Para cambiar el nombre por defecto de los elementos agregados (Botones, *Labels*, etc.), basta con seleccionar el elemento; en este caso un botón y en la tabla de Propiedades – *Text*, y se procede a colocar los nombres según la siguiente asignación:



Los elementos vienen con un nombre por defecto, por ejemplo: los *TextBox*, se nombran automáticamente como *TextBox1, TexBox2, TextBox3,* etc.

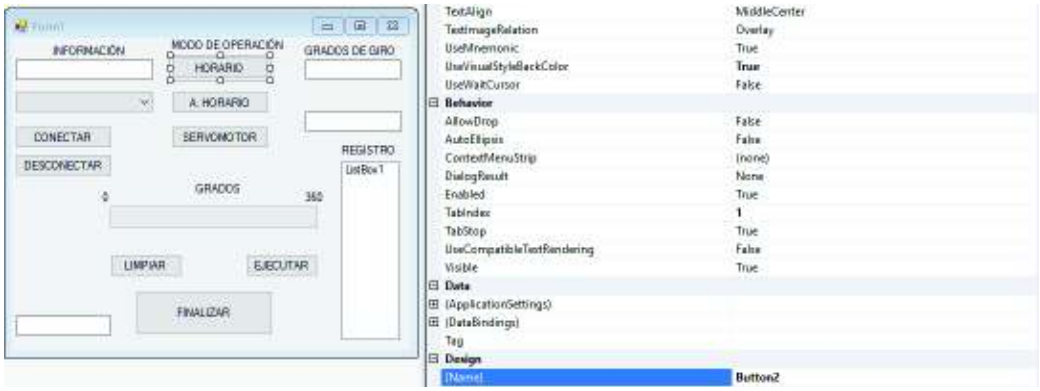
Para evitar confusiones en el momento de programar se recomienda cambiar los nombres por defecto, de ciertos elementos, según convenga.

Para cambiar el nombre de cualquier elemento se debe seguir los siguientes pasos:

- **Paso 1:** Seleccionar el elemento que se desea cambiar el nombre.

- **Paso 2:** Dirigirse a la tabla de propiedades y buscar la propiedad "*Name*" y ahí cambiar el nombre que viene por defecto. En este caso se colocó el nombre *"Button2".*

*Nota: * Este proceso es útil para cambiar el nombre a cualquier otro elemento.*



Seguir los pasos para cambiar el nombre de los elementos y colocar los siguientes, para que coincidan con el código de ejemplo de este manual.

- Conectar = *Button 7*
- Desconectar = *Button 8*
- Horario = *Button 2*
- AHorario = *Button 3*
- Servomotor = *Button 6*
- Limpiar = *Button 4*
- Ejecutar = *Button 5*
- Finalizar = *Button 1*

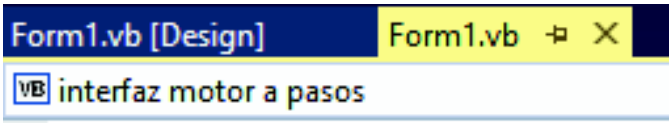
Escribir el código para los elementos necesarios. Para ello realizar los siguientes pasos:

- Para programar cada elemento basta con hacer doble clic sobre el elemento deseado para que se cree una función por defecto, pero suele haber funciones que son necesarias escribirlas ya que cumplen una tarea en específico.
- Inicialmente hacer clic sobre el form1, es decir, la ventana que contiene al resto de elementos.

- En esta sección del código se importarán librerías y se declaran variables. Además, en esta sección se colocará el código que permita leer los puertos COM disponibles en el computador.

```
Public Class Form1
    Dim Integ As Integer

    Dim strBufferIn, strBufferOut, texto As String
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        TextBox1.Text = ""
        strBufferIn = ""
        strBufferOut = ""

        Timer1.Enabled = True

        Try
            ComboBox1.Items.Clear()
            For Each puerto As String In My.Computer.Ports.SerialPortNames
                ComboBox1.Items.Add(puerto)
            Next
            If ComboBox1.Items.Count > 0 Then
                ComboBox1.SelectedIndex = 0
            Else
                MsgBox("No se encuentran puertos disponibles")

            End If
        Catch ex As Exception
            MsgBox(ex.Message, MsgBoxStyle.Critical)
        End Try
    End Sub
End Class
```

Es importante recordar que, para generar la función de cada elemento del diseño, se deberá volver a la pestaña de “Diseño”, y hacer doble clic sobre el elemento según corresponda. Para volver al diseño, basta con dar clic sobre la pestaña *Form1.v[Design].*



Para configurar el *Timer* se procederá a dar clic sobre el ícono Timer1, y se escribirá el siguiente código. Este permitirá leer datos entrantes por el monitor serie y a su vez enviar datos a través de este.

```
  Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick

      If strBufferIn <> "" Then
          strBufferIn = ""
          SerialPort1.DiscardInBuffer()
      End If
      If strBufferOut <> "" Then
          SerialPort1.Write(strBufferOut)
          strBufferOut = ""
      End If
      TextBox1.Text = "Seleccione Modo"
  End Sub
```

En lo que concierne a los botones se tendrá los siguientes códigos:

CONECTAR:

```
Private Sub Button7_Click(sender As Object, e As EventArgs) Handles Button7.Click
    Try
        With SerialPort1
            .BaudRate = 9600
            .DataBits = 8
            .Parity = IO.Ports.Parity.None
            .StopBits = 1
            .PortName = ComboBox1.Text
            .Open()
            If .IsOpen Then
                TextBox2.Text = "CONECTADO"
            Else
                MsgBox("SIN CONEXION", MsgBoxStyle.Critical)
            End If
        End With
    Catch ex As Exception
        MsgBox(ex.Message, MsgBoxStyle.Critical)
    End Try

End Sub
```

DESCONECTAR:

```
Private Sub Button8_Click(sender As Object, e As EventArgs) Handles Button8.Click
    SerialPort1.Close()
    TextBox2.Text = "DESCONECTADO"

End Sub
```

HORARIO:

El momento que se asigna un valor a *strBufferOut,* este sale a través del puerto serial.

```
Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
    strBufferOut = "x"
    TextBox4.Text = "HORARIO"
    TextBox1.Text = "Asigne Grados"
    ProgressBar1.Minimum = 0
    ProgressBar1.Maximum = 360
End Sub
```

AHORARIO:

```
Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click
    strBufferOut = "y"
    TextBox4.Text = "ANTIHORARIO"
    TextBox1.Text = "Asigne Grados"
    ProgressBar1.Minimum = 0
    ProgressBar1.Maximum = 360

End Sub
```

SERVO MOTOR:

```
Private Sub Button6_Click(sender As Object, e As EventArgs) Handles Button6.Click
    strBufferOut = "s"
    TextBox4.Text = "SERVOMOTOR"
    TextBox1.Text = "Asigne Grados"
    ProgressBar1.Minimum = 0
    ProgressBar1.Maximum = 180
End Sub
```

LIMPIAR:

```
Private Sub Button4_Click(sender As Object, e As EventArgs) Handles Button4.Click
    strBufferOut = "r"
    ProgressBar1.Value = 0
    TextBox1.Text = ""
    TextBox4.Text = ""
    TextBox5.Text = ""
    ListBox1.Items.Clear()
End Sub
```

EJECUTAR:

Envía el valor que se encuentre en el *TextBox5*, el cual será el valor en grados que deberá girar el motor acorde al sentido previamente elegido. Además, colocar el valor enviado en la barra de progreso**.**

```
Private Sub Button5_Click(sender As Object, e As EventArgs) Handles Button5.Click

    strBufferOut = TextBox5.Text
    texto = TextBox5.Text
    Int32.TryParse(texto, Integ)
    ListBox1.Items.Add(Integ)
    Timer1.Start()
    ProgressBar1.Value = Integ
    TextBox1.Text = ""
    TextBox4.Text = ""
    TextBox5.Text = ""

End Sub
```

FINALIZAR:

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    Dim Termino As Integer
    Termino = MessageBox.Show("Desea terminar la conexión?", "Atención", MessageBoxButtons.YesNo, MessageBoxIcon.Information)
    If (Termino = vbYes) Then
        SerialPort1.Close()
        Me.Close()
    End If
End Sub
```

Modo de uso de la interfaz:

1. Ejecutar la interfaz (START).
2. Seleccionar en Combo Box el puerto COM perteneciente al *Arduino* Mega.
3. Presionar Conectar, se mostrará en el *TextBox2* la palabra “conectado”.
4. Seleccionar el modo de operación entre: HORARIO, AHORARIO Y SERVO.
5. Colocar el valor de grados que se desea ejecutar en el TextBox5.
6. Presionar Ejecutar
7. Volver a seleccionar un modo de operación y repetir el punto 5.
8. Para vaciar los registros y el valor de la barra de progreso, clic sobre el botón “LIMPIAR”.
9. Para cerrar, clic en finalizar y clic en “Si”.

3. RECOMENDACIONES

- En caso de que se utilice la conexión con el *Driver* L293D, cambiar el tiempo de *delay* a 5 milisegundos, ya que en 3 milisegundos no logran activarse las bobinas.
- Tener en cuenta que el valor que permite el servomotor se encuentra entre 0 y 180 grados.

- Verificar que las conexiones hacia las entradas del módulo ULN2003 se encuentren bien realizadas, ya que de lo contrario el motor podría no girar como se espera.
- El circuito no necesita de una alimentación externa para su funcionamiento por lo que las conexiones de alimentación se realizan directo desde los pines del *Arduino*.
- Hay que recordar que la secuencia de encendido simple puede ser utilizada de igual manera con motores bipolares; por lo que la conexión del *Driver* L293D, es una manera de comprobar el funcionamiento de ésta en caso de que se lo requiera.
- La alimentación al *Arduino* se la realiza mediante el cable USB tipo B, el cual ha sido utilizado para cargar el programa.
- Es importante tener en cuenta que, si se está ejecutando la interfaz en *Visual Studio*, y ésta se encuentra comunicándose con el *Arduino*; el puerto COM se encontrará ocupado, por lo que, cualquier otra aplicación que quiera comunicarse con el *Arduino* mediante comunicación serial no lo podrá hacer.

4. BIBLIOGRAFÍA

[1] *Arduino*, «*Functions*,» [En línea]. *Available*: https://www.a*rduino*.cc/en/Reference/Function Declaration. [Último acceso: 12 Febrero 2018].

[2] Microsoft, «*Featured API*,» [En línea]. *Available*: https://msdn.microsoft.com/library. [Último acceso: 12 Febrero 2018].

[3] *Arduino*, «*String to Int Function*,» [En línea]. *Available*: https://www.*Arduino*.cc/en/Tutorial/StringToIntExample. [Último acceso: 12 Febrero 2018].

[4] PanamaHitek, «Enviar numeros de mas de un digito en *Arduino*,» [En línea]. *Available*: http://panamahitek.com/enviar-numeros-de-mas-de-un-digito-a-*Arduino*/. [Último acceso: 12 Febrero 2018].

[5] Electrónica teórica y práctica, «Circuito ULN 2003,» [En línea]. *Available*: http://electronica-teoriaypractica.com/circuito-uln2003/. [Último acceso: 12 Febrero 2018].

[6] Akizukidenshi, «SG90,» [En línea]. *Available*: http://akizukidenshi.com/download/ds/towerpro/SG90.pdf. [Último acceso: 12 Febrero 2018].

[7] *Arduino*, «*Arduino Mega 2560*,» [En línea]. *Available*: https://store.*Arduino*.cc/usa/*Arduino*-mega-2560-rev3. [Último acceso: 12 Febrero 2018].

[8] server-die, «Secuencia para manejar motores paso a paso unipolar,» [En línea]. *Available*: http://server-die.alc.upv.es/asignaturas/lsed/2002-03/MotoresPasoaPaso/ftomotpap.htm. [Último acceso: 12 Febrero 2018].

ESCUELA POLITÉCNICA NACIONAL
ESCUELA DE FORMACIÓN DE TECNÓLOGOS
HOJA GUÍA
PRÁCTICA 3

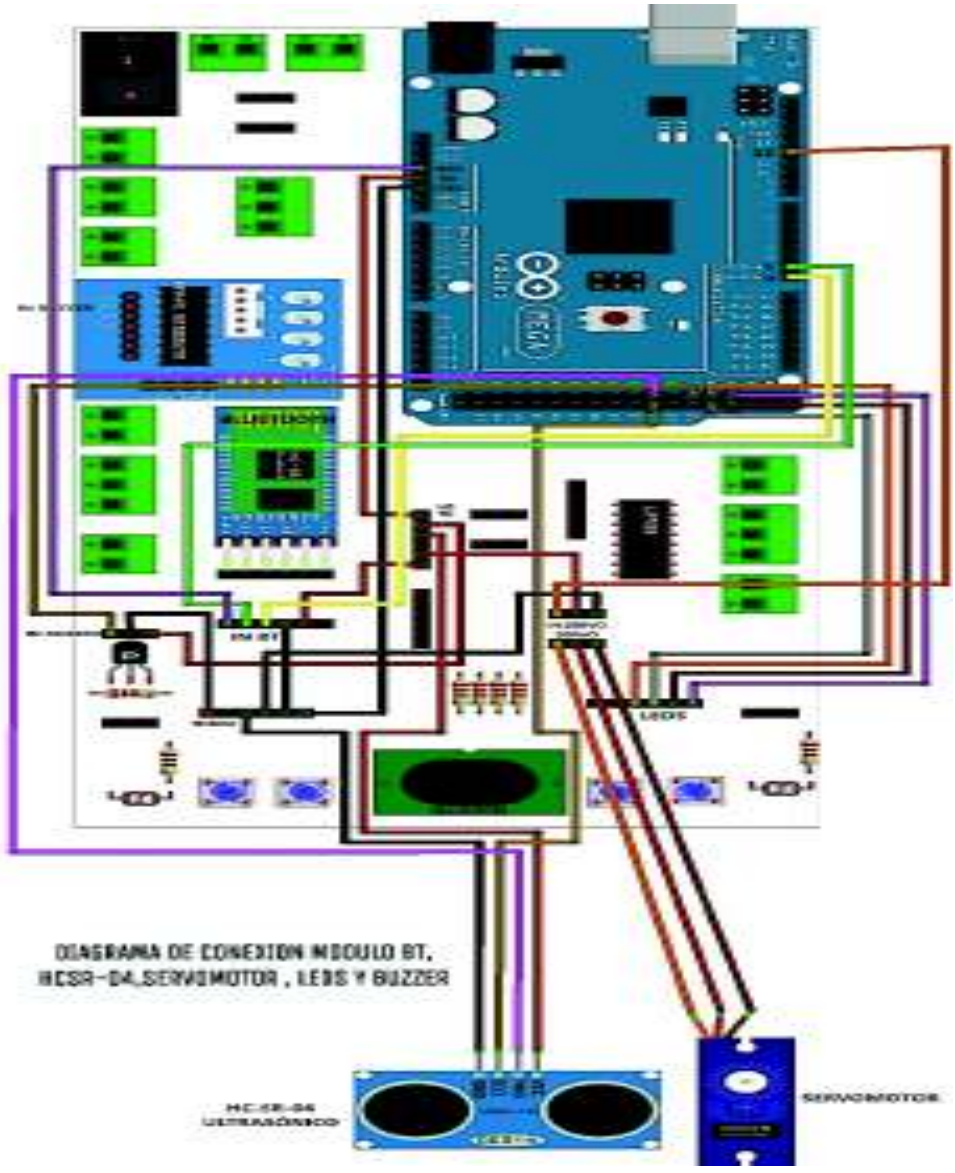
1. TEMA: Control de los motores de la estructura utilizando Módulo *Bluetooth*.

2. DESARROLLO DE LA PRÁCTICA

NOTA: Tiempo estimado para realizar la presente práctica de laboratorio tanto software como hardware es de 2 horas con 30 minutos. Se recomienda además que un alumno se encargue de las conexiones mientras el otro inicia con la programación del *Arduino* o de la interfaz.

DIAGRAMA DE CONEXIONES

Realizar las conexiones del servomotor, motores DC, *Buzzer* y *LED*s como se muestra en los siguientes diagramas.



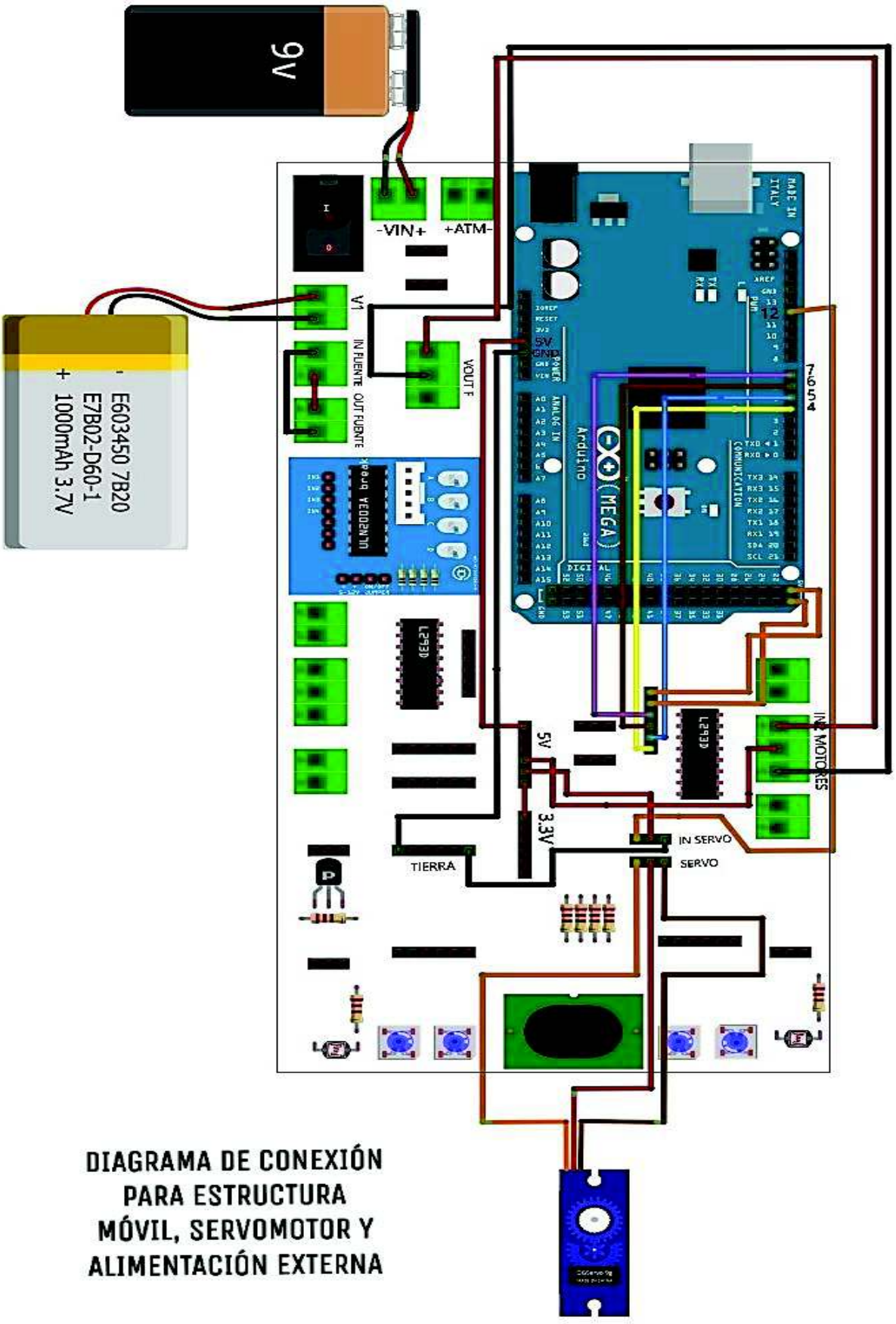


DIAGRAMA DE CONEXIÓN PARA ESTRUCTURA MÓVIL, SERVOMOTOR Y ALIMENTACIÓN EXTERNA

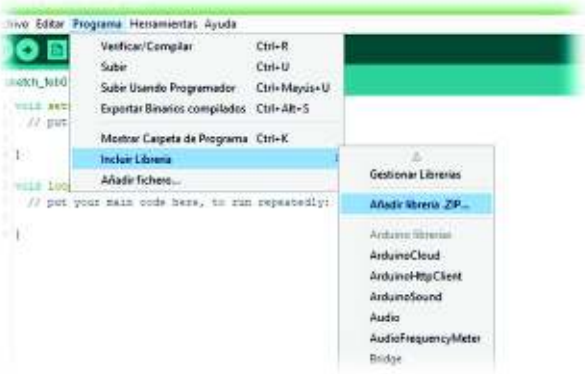
PROGRAMA *ARDUINO*

Ejecutar el IDE de *Arduino*.

Una vez ejecutado se mostrará la siguiente ventana en la cual se digitará el código.

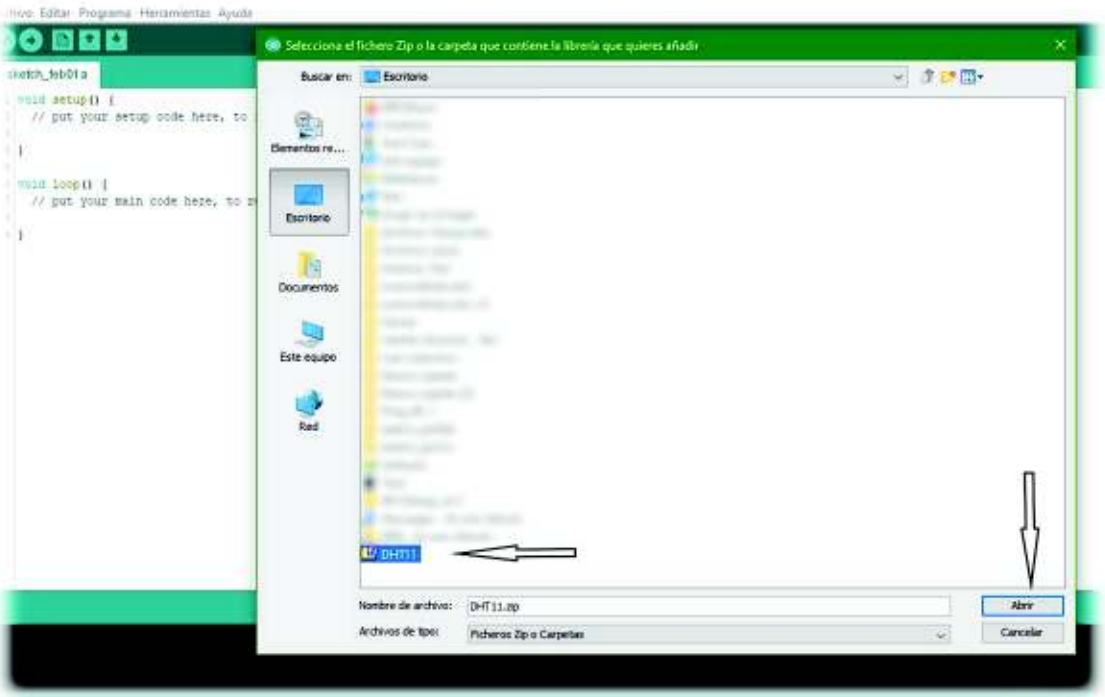
En caso de que el IDE de *Arduino* no posea la Librería Servo.h, se realizan los siguientes pasos para incluir una librería previamente descargada de la red y se procede con la instalación como se muestra en los siguientes pasos:

- **Paso 1:** Ir a la Barra de menú del IDE de *Arduino*/Incluir Librería/Añadir librería .ZIP..



- **Paso 2:** Después se mostrará un explorador de archivos en el que se debe localizar el archivo DHT11.zip. Una vez encontrado el archivo se procede a hacer clic en Abrir. Esperar un momento y dejar que el IDE realice la instalación.

*NOTAS: *De preferencia guardar el archivo DHT11.zip y pt.zip en el escritorio para una rápida localización.*
*** Los pasos son útiles para la instalación de cualquier otra librería.*



Proceder con la digitación del código en el IDE de *Arduino* para satisfacer el literal 4.2 de la hoja guía del estudiante.

- **Paso 1:** Importar librerías y declarar las variables necesarias, antes del *void setup().*

```
#include <Servo.h>
#include <SoftwareSerial.h>
#define TxD 1
#define RxD 0
#define EchoPin     35
#define TriggerPin 48

SoftwareSerial Blue(TxD, RxD);


int direccion = 0;

const int distanciaMin = 15;
const int t_mov_manual = 150;

int IN1 = 4;
int IN2 = 5;
int IN3 = 6;
int IN4 = 7;

int LED1 = 16;
int LED2 = 17;
int LED3 = 18;
int LED4 = 19;

int buzzer = 38;

Servo pinServoMotor;

void SerialDataReceived();
void ManualMode();
void AutoMode();
void MoveTo();
```

- **Paso 2:** Se establecen el modo de trabajo de los pines y otras características.

```
void setup()
{
    Serial.begin(9600);
    Blue.begin(9600);
    pinServoMotor.attach(12);

    pinMode(IN1, OUTPUT);
    pinMode(IN2, OUTPUT);
    pinMode(IN3, OUTPUT);
    pinMode(IN4, OUTPUT);

    pinMode(LED1, OUTPUT);
    pinMode(LED2, OUTPUT);
    pinMode(LED3, OUTPUT);
    pinMode(LED4, OUTPUT);
    pinMode(EchoPin, INPUT);
    pinMode(TriggerPin, OUTPUT);
    pinMode(buzzer, OUTPUT);
}
```

- **Paso 3:** En la función principal *void loop(),* se hace un llamado de las funciones o subrutinas que conforman el programa.

```
void loop()
{
    /**@descripcion: lee los datos recibidos a traves del puerto serial*/
    SerialDataReceived();
    /**@descripcion: activa los motores para ir a up, down, right, left*/
    MoveTo();
    /**@descripcion: activa el modo manual por defecto */
    ManualMode();
}
```

- **Paso 4:** Crear las funciones *void* que fueron llamadas en el paso anterior.

```
void SerialDataReceived()
{
    char datoRecibido = Serial.read();
    switch(datoRecibido)
    {
        case '0':
        ManualMode();
        break;

        case '1':
        AutoMode();
        break;

        case 'a':
        direccion = 1;
        break;

        case 'd':
        direccion = 2;
        break;

        case 'w':
        direccion = 3;
        break;

        case 's':
        direccion = 4;
        break;
```

```
            case '2':
            direccion = 5;
            break;

            case '4':
            direccion = 6;
            break;

            case '5':
            direccion = 7;
            break;

            default:
            direccion = 0;
            break;
        }
}
```

```
void MoveTo()
{

    switch(direccion)
    {
        case 0: //Apagado
        digitalWrite(IN1, LOW);
        digitalWrite(IN2, LOW);
        digitalWrite(IN3, LOW);
        digitalWrite(IN4, LOW);
        break;

        case 1: //Izquierda
        digitalWrite(IN1, LOW);
        digitalWrite(IN2, HIGH);
        digitalWrite(IN3, HIGH);
        digitalWrite(IN4, LOW);
        direccion = 0;
        break;

        case 2: //Derecha
        digitalWrite(IN1, HIGH);
        digitalWrite(IN2, LOW);
        digitalWrite(IN3, LOW);
        digitalWrite(IN4, HIGH);
        direccion = 0;
        break;

        case 3: //Delante
        digitalWrite(IN1, LOW);
        digitalWrite(IN2, HIGH);
        digitalWrite(IN3, LOW);
        digitalWrite(IN4, HIGH);
        direccion = 0;
        break;

        case 4: //Atras
        digitalWrite(IN1, HIGH);
        digitalWrite(IN2, LOW);
        digitalWrite(IN3, HIGH);
        digitalWrite(IN4, LOW);
        direccion = 0;
        break;
```

```
            case 5: //Buzzer
            tone(buzzer,440);
            delay(100);
            noTone(buzzer);
            delay(200);
            tone(buzzer,550,300);
            delay(200);
            digitalWrite(buzzer,LOW);
            break;

            case 6: //LEDS ON
            digitalWrite(LED1,HIGH);
            digitalWrite(LED2,HIGH);
            digitalWrite(LED3,HIGH);
            digitalWrite(LED4,HIGH);
            break;

            case 7: //LEDS OFF
            digitalWrite(LED1,LOW);
            digitalWrite(LED2,LOW);
            digitalWrite(LED3,LOW);
            digitalWrite(LED4,LOW);
            break;
        }
    }

void AutoMode()
{
    long tiempo;
    float distanciaMedida;
    int delay_giro, delay_avance;
    delay_giro = 750;
    delay_avance = 250;

    while(1)
    {
        int anguloServo;
        float distancia_0, distancia_90, distancia_180;
        char datoRecibido = Serial.read();

        if(datoRecibido == '0')
        {
            break;
        }
        anguloServo = 0;

        // *** Lectura Distancia en 0, 90, 180  *** //
        while(anguloServo < 181)
        {
            pinServoMotor.write(anguloServo);
            delay(1000);
            digitalWrite(TriggerPin, LOW);
            delayMicroseconds(2);
            digitalWrite(TriggerPin, HIGH);
            delayMicroseconds(10);
            digitalWrite(TriggerPin, LOW);

            tiempo = pulseIn(EchoPin, HIGH);
            distanciaMedida = float((tiempo/2)/29);
            Serial.print("0: ");
            Serial.println(distanciaMedida);

            if(anguloServo == 0)
            {
                distancia_0 = distanciaMedida;
            }
            else if (anguloServo == 90)
            {
                distancia_90 = distanciaMedida;
            }
            else if (anguloServo == 180)
            {
                distancia_180 = distanciaMedida;
            }
            anguloServo += 90;
        }
```

```
        // *** Comparacion Distancias *** //

        if (distancia_0 > distancia_180 && distancia_0 > distancia_90 && distancia_0 > distanciaMin)
        {

            direccion = 2;
            MoveTo();
            delay(delay_giro);

            MoveTo();

            direccion = 3;
            MoveTo();
            delay(delay_avance);

            MoveTo();

        }
        else if (distancia_180 > distancia_0 && distancia_180 > distancia_90 && distancia_180 > distanciaMin)
        {
            direccion = 1;
            MoveTo();
            delay(delay_giro);

            MoveTo();

            direccion = 3;
            MoveTo();
            delay(delay_avance);

            MoveTo();
        }
         else if (distancia_90 > distancia_0 && distancia_90 > distancia_180 && distancia_90 > distanciaMin)
         {
             if(distancia_90 < 5)
             {
                 direccion = 4;
                 MoveTo();
                 delay(100);

                 MoveTo();
             }
             else
             {
                 direccion = 3;
                 MoveTo();
                 delay(delay_avance);

                 MoveTo();
             }

         }

    }

}

void ManualMode()
{
    delay(t_mov_manual);
    pinServoMotor.write(90);
}
```

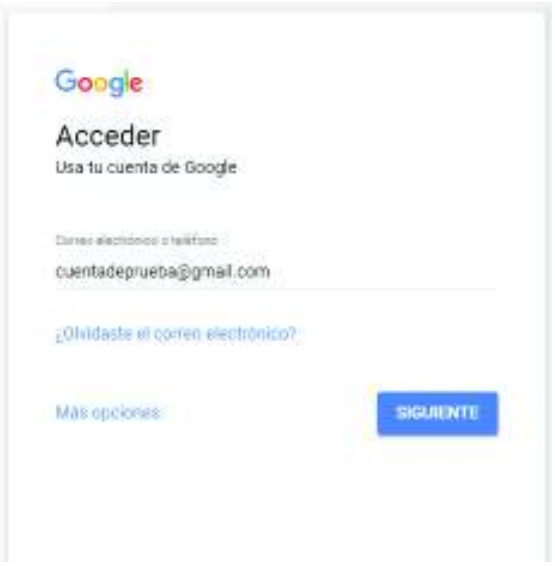
ELABORAR UNA APP QUE PERMITA ENVIAR CARACTERES DE CONTROL AL *ARDUINO* QUE CUMPLA CON LAS SIGUIENTES CONDICIONES:

- La aplicación debe permitir dos modos de funcionamiento uno automático y otro manual.
- En modo manual la aplicación debe tener botones que permitan el direccionamiento del módulo didáctico.
- La aplicación debe tener botones para activar y desactivar el modo automático.
- Desde la aplicación se debe activar y desactivar los *LED*s. Además, debe permitir activar el *Buzzer* por un instante de tiempo.
- Finalmente, debe permitir enlazar el celular con el *Arduino*.

CREACIÓN DE LA APP

Ingresar a la página principal de *App Inventor* (appinventor.mit.edu/) y hacer clic sobre *Create apps!* que está ubicado en la parte superior derecha.

Para poder ingresar es necesario tener una cuenta de Google. Ingresar su usuario y contraseña.



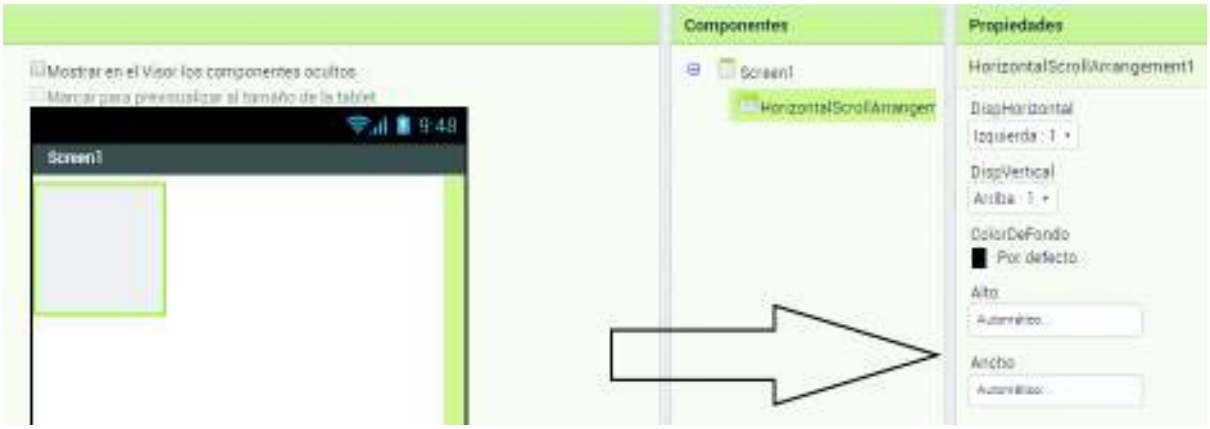
Para crear un nuevo proyecto hacer clic sobre Proyectos y clic en Comenzar un proyecto nuevo…

Se le solicitará ingresar un nombre para el proyecto. Se recomienda escribir un nombre relacionado a la aplicación.

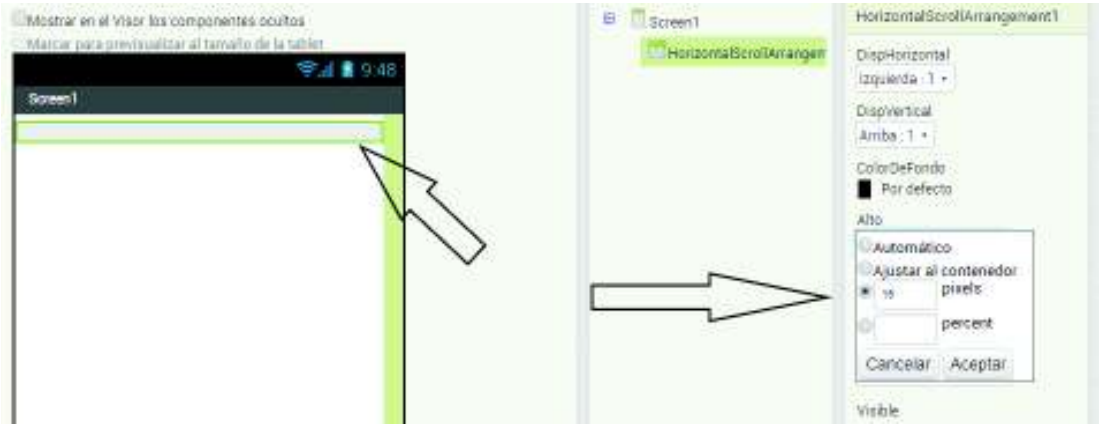
A continuación, se diseñará la aplicación utilizando las herramientas ubicadas en la parte izquierda y se modificará ciertos parámetros de dichas herramientas en la sección de propiedades.

Primero se realizará una distribución de pixeles ya sea para botones o *labels* utilizando la herramienta *HorizontalArrangement* ubicada en la tabla de herramientas, sección *"Layout*". Para ello arrastrar la herramienta hacia la sección de diseño.

*Nota: * Por defecto el ancho y alto de HorizontalArrangement vienen en automático.*

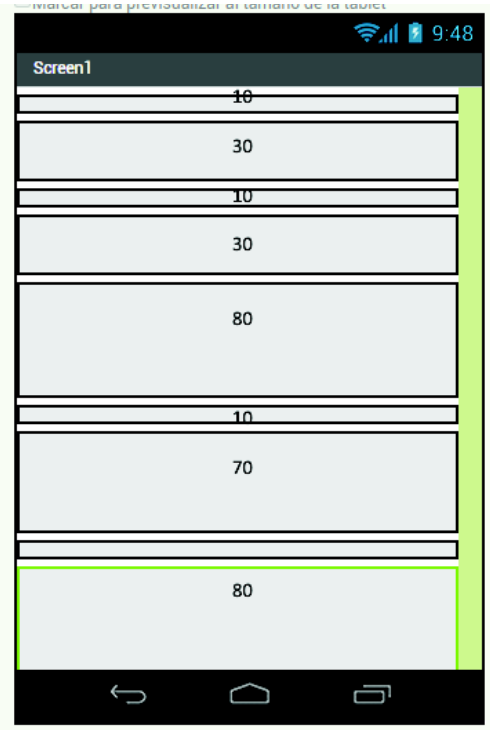


Colocar un alto de 15 pixeles y un ancho de “Ajustar al contenedor”.

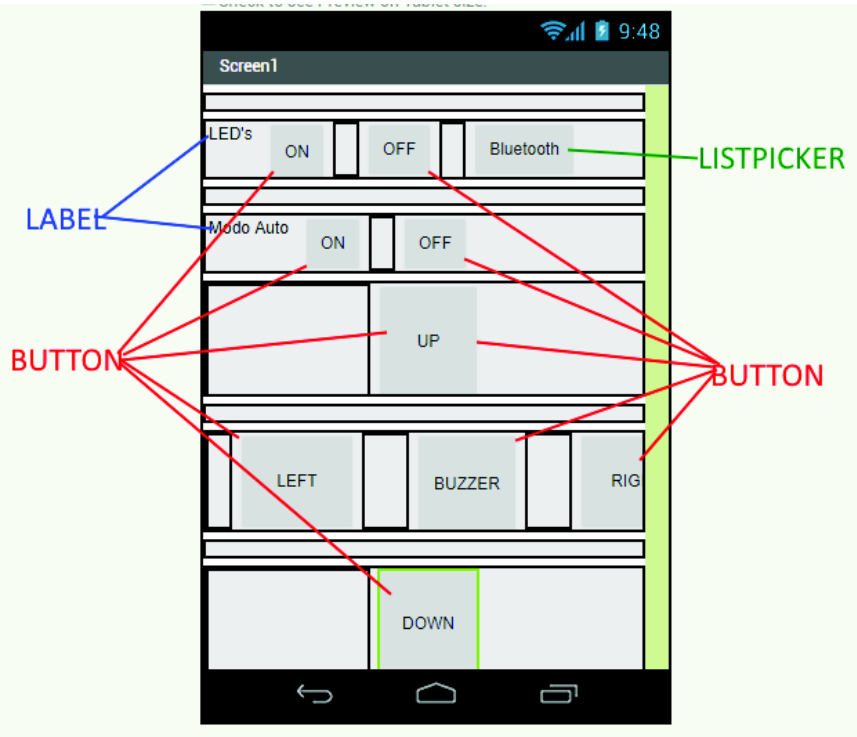


Con la herramienta anterior realizar un diseño similar al siguiente, teniendo en cuenta que el ancho esté configurado como “Ajustar al contenedor” y el alto con los valores mostrados en la siguiente imagen.

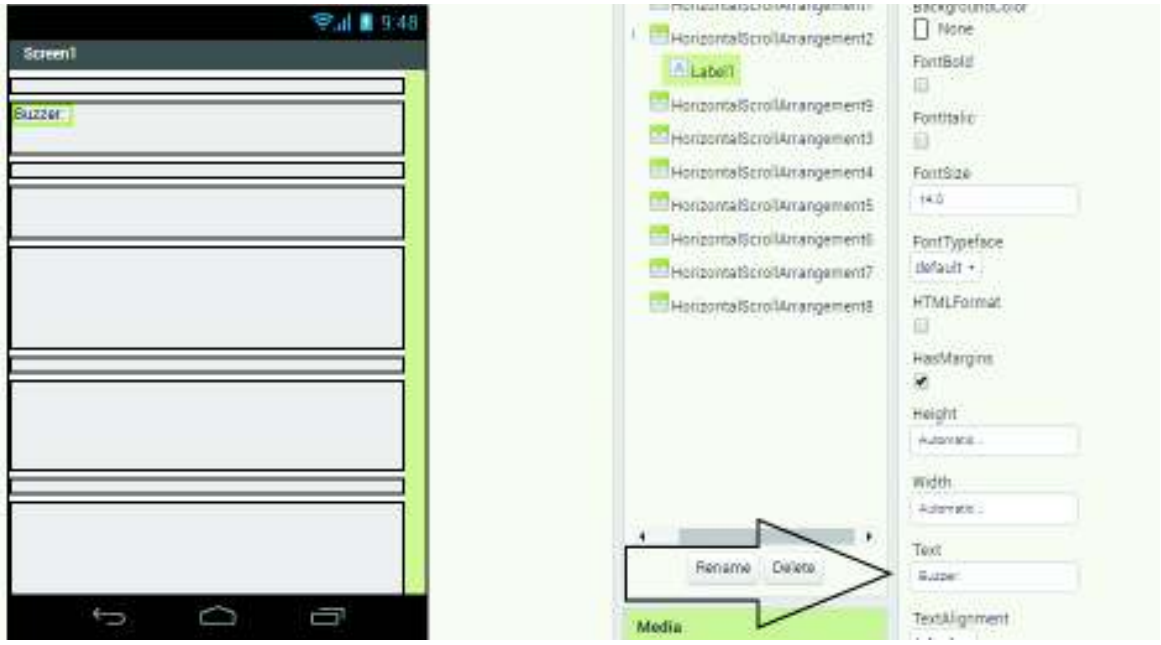
*Nota: *Los valores mostrados en la imagen son solo una recomendación, cada usuario puede crear la app con el tamaño deseado.*



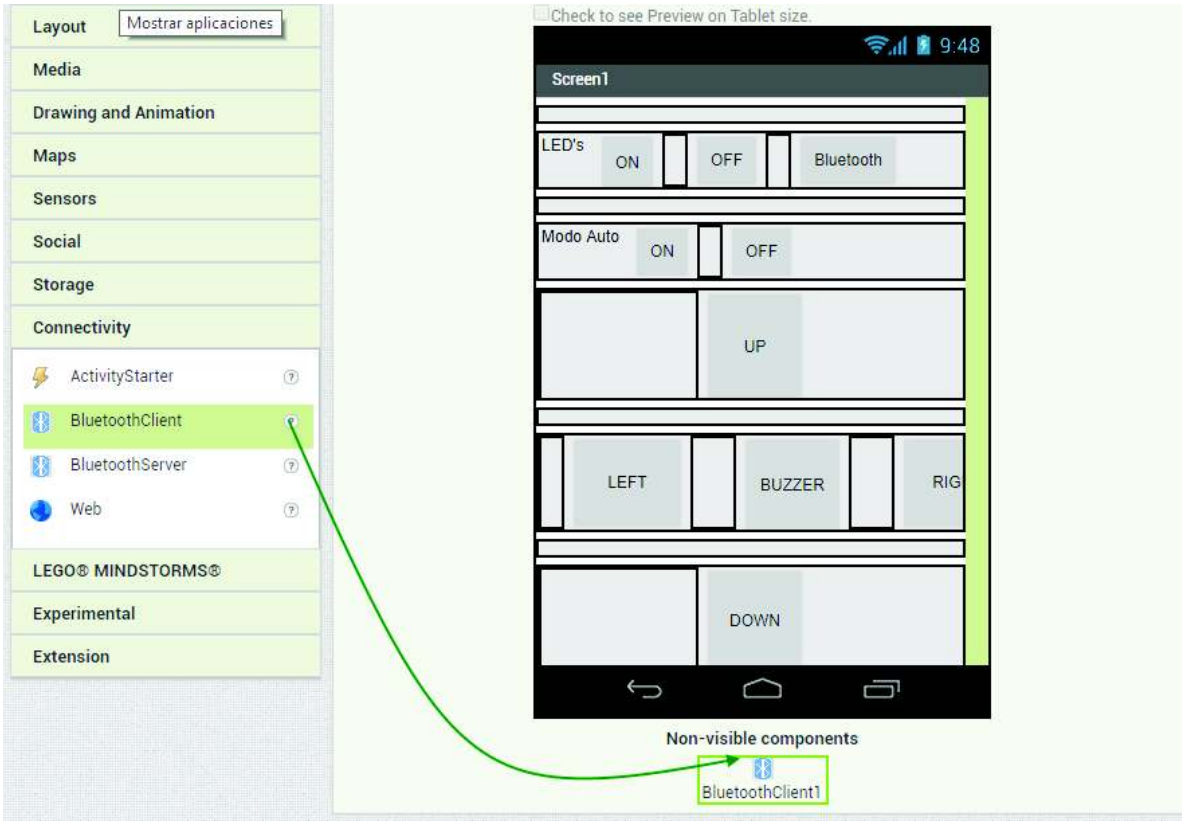
A continuación, utilizar las herramientas *Label, Button, ListPicker y HorizontalArrangement* para realizar el siguiente diseño. Las dimensiones dependerán del gusto del usuario, se recomienda separar los botones con la herramienta *HorizontalArrangement* con las siguientes dimensiones alto (Ajustar al contenedor) y ancho 15 pixeles.



Para cambiar el texto de los *Label, Button y ListPicker* se necesita seleccionar el elemento y dirigirse a la sección *"Text".* El proceso sirve para cualquier otro tipo de elemento.



Agregar el elemento *BluetoothClient,* basta con arrastrarlo como cualquier otro elemento.

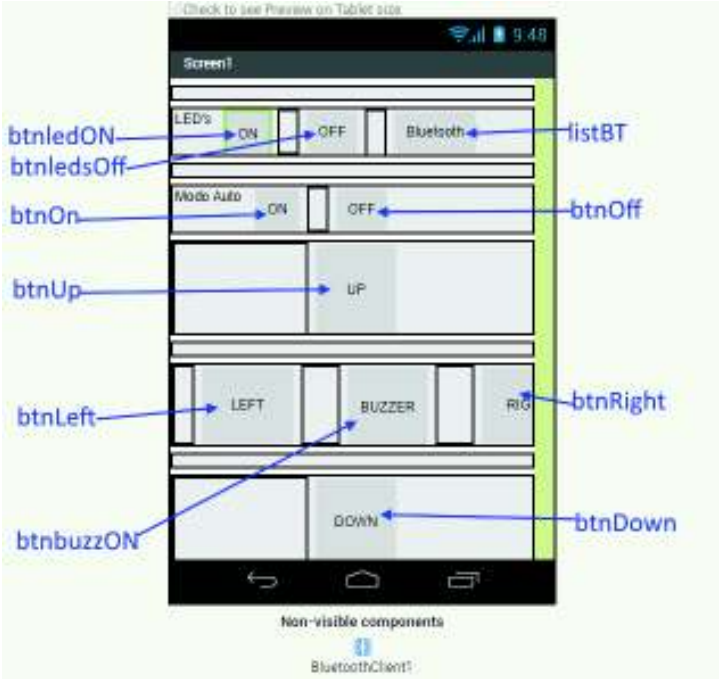


A continuación, se renombrará a cada elemento para que se haga más sencilla la programación de la aplicación.

Para renombrar un elemento primero hay que seleccionarlo.

Después en la parte derecha del diseño hay una sección llamada "*Components",* ahí se debe ubicar el botón *rename*. Hacer clic sobre él y se solicitará el nuevo nombre.

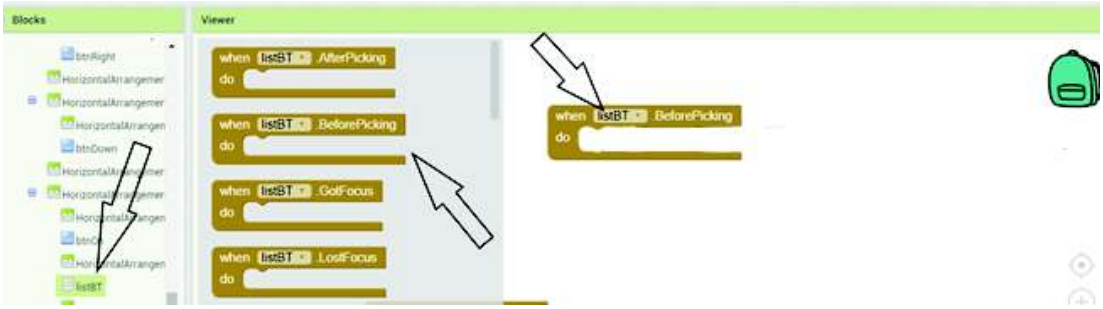
Renombrar los elementos con los siguientes nombres:

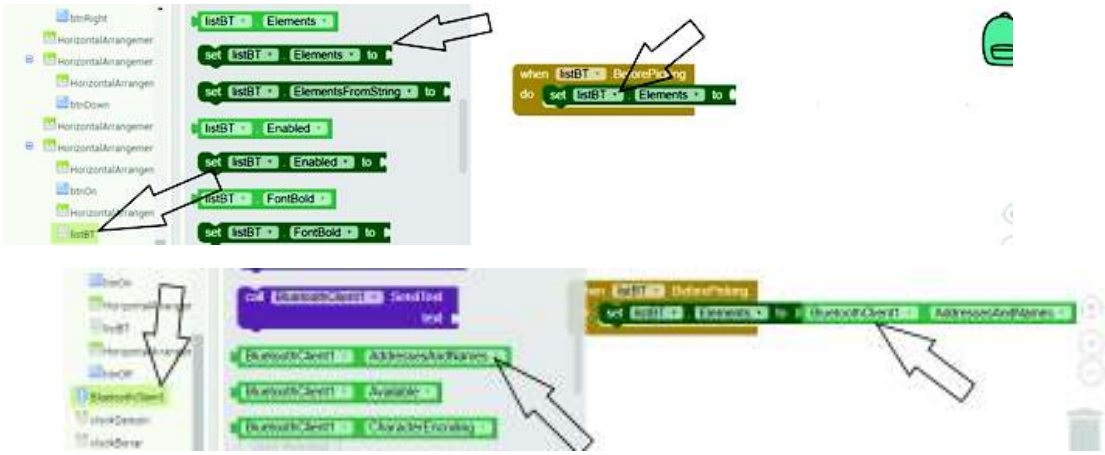


El siguiente paso es programar cada elemento utilizando bloques, para ello hacer clic en Blocks ubicado en la parte superior derecha. En donde se abrirá una ventana similar a la siguiente.

La programación se realizará utilizando los bloques disponibles en la parte izquierda, para utilizarlos basta con arrastrarlos.

Para encontrar el bloque indicado, éste debe estar relacionado; es decir, si en el bloque se menciona por ejemplo al *listBT*, se deberá dirigir a la sección *listBT* y buscar el bloque. Como se muestra en las siguientes imágenes.





Con el proceso anterior, realizar el siguiente programa.

Esta sección del programa enlista los dispositivos *Bluetooth* previamente acoplados al dispositivo móvil.

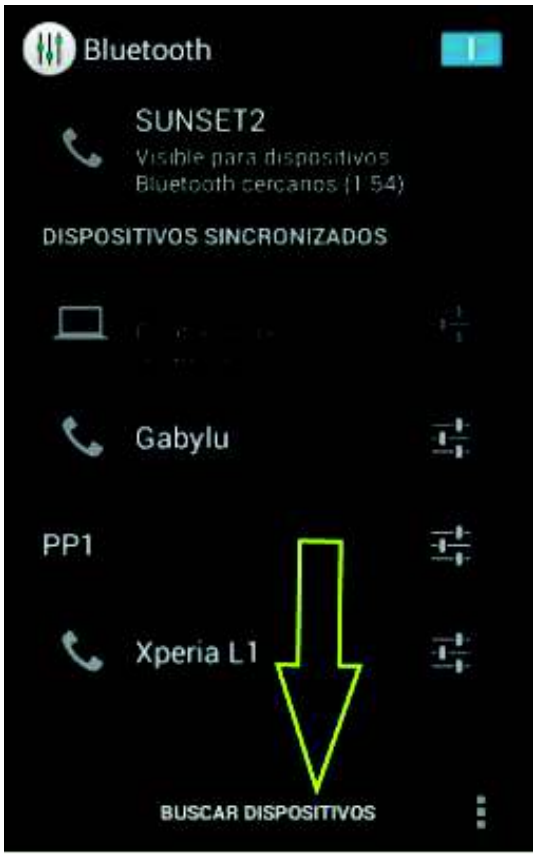
En esta sección del programa los botones cuando sean presionados enviarán sus respectivos caracteres. Botones mostrados en color rojo son bloques tipo Texto.

Finalmente compilara el programa para crear el archivo .apk o generar un código QR. Para ello, clic en *Build.*

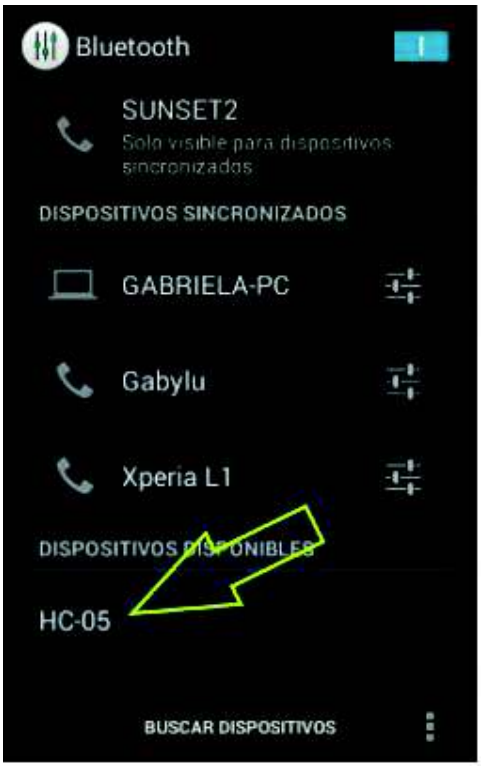
PASOS ACOPLAR EL MÓDULO CON LA APLICACIÓN EN ANDROID

En el dispositivo Android encender el BT, previamente el módulo HC-05 debe estar activo.

Buscar nuevos dispositivos para detectar el módulo *Bluetooth* HC-05.

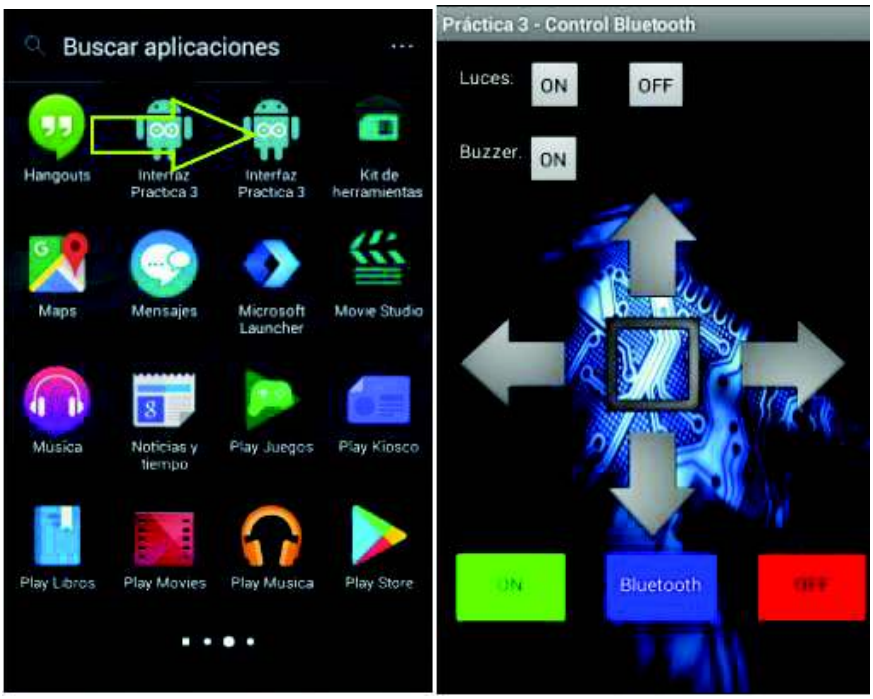


Esperar a que el dispositivo móvil detecte el módulo.



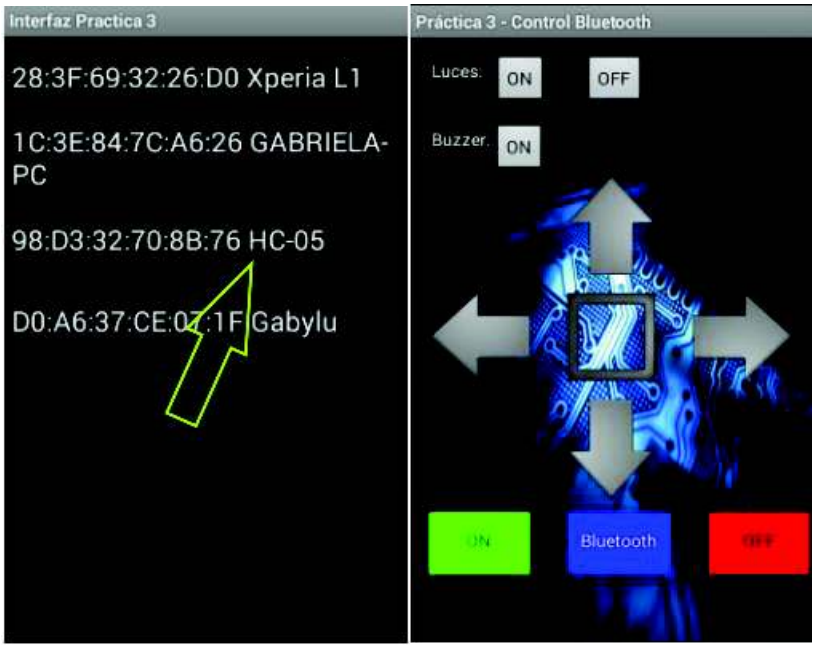
Seleccionar el módulo *Bluetooth* y sincronizar con el dispositivo móvil. Ingresar el PIN solicitado por el módulo que por defecto es 1234. Aceptar y esperar hasta que se vinculen.

El siguiente paso es abrir la aplicación diseñada anteriormente.



Presionar sobre la opción *Bluetooth*, y esperar a que se despliegue una lista en la cual debe constar el módulo HC-05 que se vinculó anteriormente. Presionar sobre la dirección del módulo y esperar.

*Notas: * Si el dispositivo no se acopla a la primera vez, se debe intentar nuevamente hasta que se enlace.*
*** Para realizar las pruebas y verificar de forma rápida que la conexión se ha establecido se puede encender o apagar los LEDs o tocar la bocina.*
**** Para el funcionamiento de la estructura móvil, se deben conectar las 2 baterías, desconectar el cable USB y conectar el cable de alimentación externa del Arduino que se encuentra en el módulo didáctico.*
***** Finalmente, presionar el botón de encendido del módulo didáctico.*



3. RECOMENDACIONES

- Al momento de carga el programa al *Arduino* Mega, se deben desconectar los cables que van al módulo hc05 correspondientes a TX y RX, ya que de lo contrario no se podrá cargar el programa y afectará al módulo.
- Se debe considerar el uso de una batería de 9V para alimentar el *Arduino*, ya que la provista con la estructura únicamente permite la alimentación de los motores, como se mostros en el diagrama "L293D"
- En caso de tener puntos ciegos de medida con la aplicación del modo autónomo, se puede optar por medir distancias en más ángulos, es decir cada 10 o cada 20 grados, para tener una visión más real de los objetos que rodean la estructura.
- Para un óptimo rendimiento, procurar que las baterías LiPo se encuentren cargadas en su voltaje de 7.4 voltios aproximadamente.
- Si se produce una rápida descarga de la batería LiPo, esta podría calentarse e incluso causar incidentes por lo que se recomienda tener cuidado al manipularlas.

4. BIBLIOGRAFÍA

[1] *Arduino*, «*Functions*,» [En línea]. *Available*: https://www.a*rduino*.cc/en/Reference/FunctionDeclaration. [Último acceso: 12 Febrero 2018].

[2] PanamaHitek, «Comunicación Serial con Arduino,» [En línea]. *Available*: http://panamahitek.com/comunicacion-serial-con-*Arduino*/. [Último acceso: 12 Febrero 2018].

[3] Naylamp, «Configuración del módulo *Bluetooth* Hc-05,» [En línea]. *Available*: http://www.naylampmechatronics.com/blog/24_configuracion-del-modulo-bluetooth-hc-05-usa.html. [Último acceso: 12 Febrero 2018].

[4] Texas Instruments, «L293x,» [En línea]. *Available*: http://www.ti.com/lit/ds/symlink/l293 .pdf. [Último acceso: 12 Febrero 2018].

[5] Aprendiendo *Arduino*, «Bluetooth en *Arduino*,» [En línea]. *Available*: https://aprendiendo a*rduino*.wordpress.com/tag/hc-05/. [Último acceso: 12 Febrero 2018].

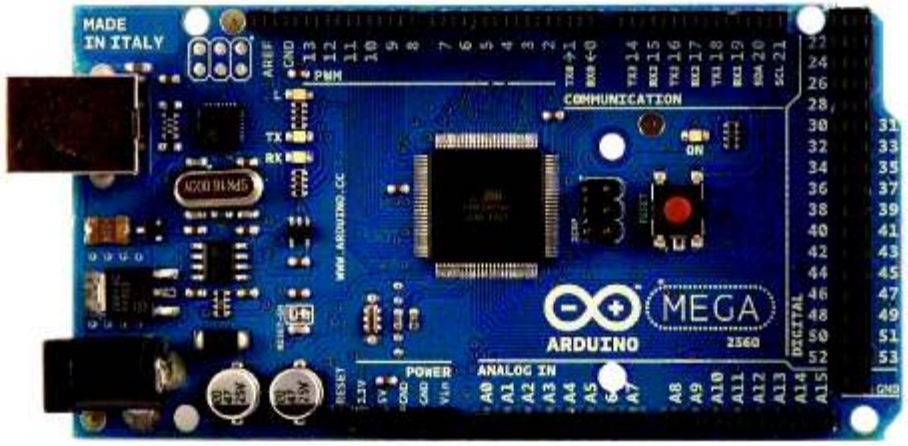
[6] Akizukidenshi, «SG90,» [En línea]. *Available*: http://akizukidenshi.com/download/ds/towerpro/SG90.pdf. [Último acceso: 12 Febrero 2018].

[7] *Arduino*, «*Arduino* Mega 2560 REV3,» [En línea]. *Available*: https://store.*Arduino*.cc/ *Arduino*-mega-2560-rev3. [Último acceso: 26 Noviembre 2017].

Anexo C: Hojas técnicas

C-1: *Arduino* MEGA 2560

Arduino MEGA 2560



CE

Product Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Index

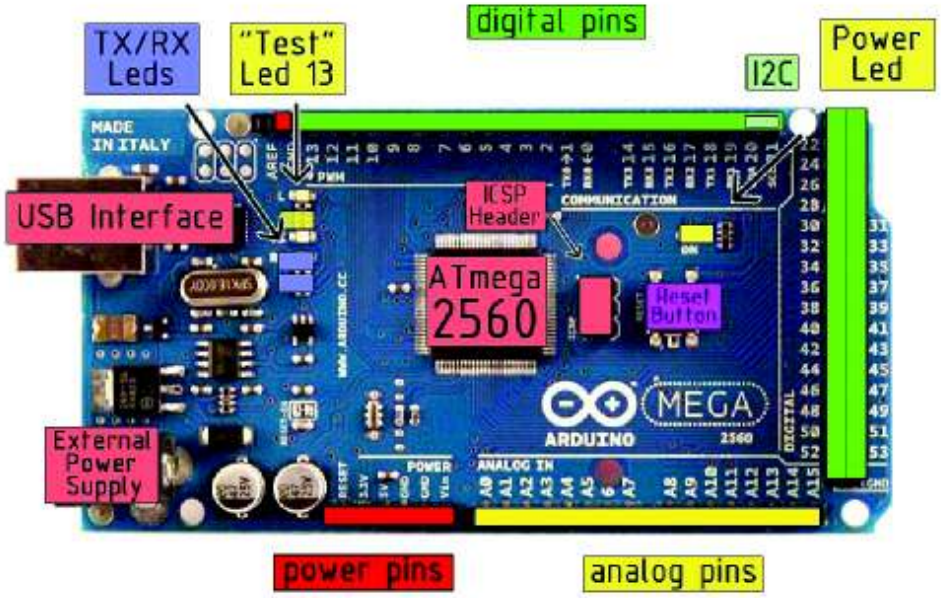
Technical Specification

EAGLE files: arduino-mega2560-reference-design.zip Schematic: arduino-mega2560-schematic.pdf

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

the board



Power

The Arduino Mega2560 can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 0 to 13.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Duemilanove and Diecimila.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- **I^2C: 20 (SDA) and 21 (SCL).** Support I^2C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I^2C pins on the Duemilanove.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Mega's digital pins.

The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. To use the SPI communication, please see the ATmega2560 datasheet.

Programming

The Arduino Mega2560 can be programmed with the Arduino software (download). For details, see the reference and tutorials.

The Atmega2560 on the Arduino Mega comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Mega2560 is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega2560 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Mega2560 is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Mega2560. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Mega contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

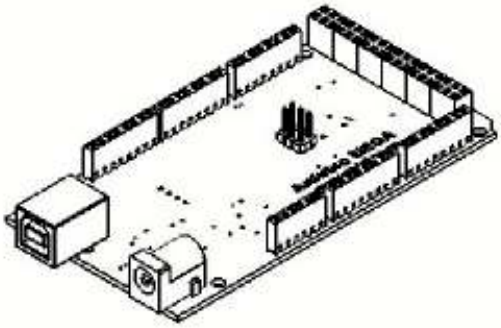
The Arduino Mega has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

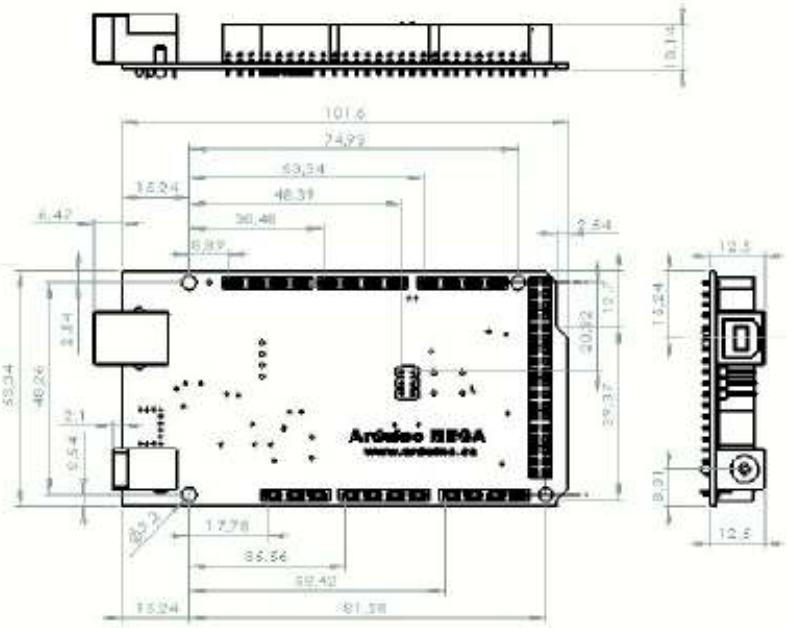
Physical Characteristics and Shield Compatibility

The maximum length and width of the Mega PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The Mega is designed to be compatible with most shields designed for the Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega and Duemilanove / Diecimila. **Please note that I^2C is not located on the same pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).**

Dimensioned Drawing





C-2: Driver L293D

Product Folder | Sample & Buy | Technical Documents | Tools & Software | Support & Community

TEXAS INSTRUMENTS

L293, L293D

SLRS009D – SEPTEMBER 1986 – REVISED JANUARY 2016

L293x Quadruple Half-H Drivers

1 Features

- Wide Supply-Voltage Range: 4.5 V to 36 V
- Separate Input-Logic Supply
- Internal ESD Protection
- High-Noise-Immunity Inputs
- Output Current 1 A Per Channel (600 mA for L293D)
- Peak Output Current 2 A Per Channel (1.2 A for L293D)
- Output Clamp Diodes for Inductive Transient Suppression (L293D)

2 Applications

- Stepper Motor Drivers
- DC Motor Drivers
- Latching Relay Drivers

3 Description

The L293 and L293D devices are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, DC and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN.

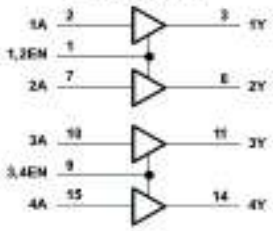
The L293 and L293D are characterized for operation from 0°C to 70°C.

Device Information(1)

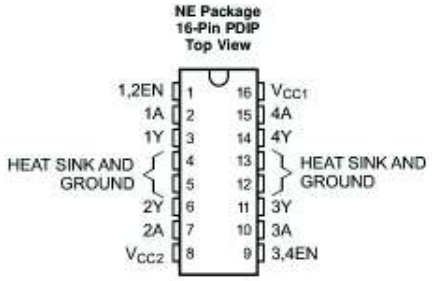
PART NUMBER	PACKAGE	BODY SIZE (NOM)
L293NE	PDIP (16)	19.80 mm × 6.35 mm
L293DNE	PDIP (16)	19.80 mm × 6.35 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

Logic Diagram



5 Pin Configuration and Functions



Pin Functions

PIN		TYPE	DESCRIPTION
NAME	NO.		
1,2EN	1	I	Enable driver channels 1 and 2 (active high input)
<1:4>A	2, 7, 10, 15	I	Driver inputs, noninverting
<1:4>Y	3, 6, 11, 14	O	Driver outputs
3,4EN	9	I	Enable driver channels 3 and 4 (active high input)
GROUND	4, 5, 12, 13	—	Device ground and heat sink pin. Connect to printed-circuit-board ground plane with multiple solid vias
V_{CC1}	16	—	5-V supply for internal logic translation
V_{CC2}	8	—	Power VCC for drivers 4.5 V to 36 V

C-3: ULN2003

ULN2002A/ ULN2003A/ ULN2004A

HIGH VOLTAGE, HIGH CURRENT DARLINGTON TRANSISTOR ARRAYS

Description

The ULN2002A, ULN2003A and ULN2004A are high voltage, high current Darlington arrays each containing seven open collector common emitter pairs. Each pair is rated at 500mA. Suppression diodes are included for inductive load driving, the inputs and outputs are pinned in opposition to simplify board layout.

Device options are designed to be compatible with common logic families:

ULN2002A (14-25V PMOS)
ULN2003A (5V TTL, CMOS)
ULN2004A (6-15V CMOS, PMOS)

These devices are capable of driving a wide range of loads including solenoids, relays, DC motors, LED displays, filament lamps, thermal print-heads and high-power buffers.

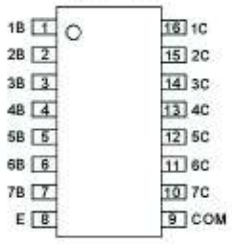
The ULN2002A, ULN2003A and ULN2004A are available in both a small outline 16-pin package (SO-16) and PDIP-16 package.

Features

- 500mA Rated Collector Current (Single Output)
- High Voltage Outputs: 50V
- Output Clamp Diodes
- Inputs Compatible with Popular Logic Types
- Relay Driver Applications
- "Green" Molding Compound (No Br, Sb)
- **Totally Lead-Free & Fully RoHS Compliant (Notes 1 & 2)**
- **Halogen and Antimony Free. "Green" Device (Note 3)**

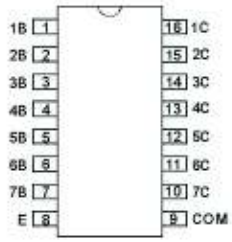
Pin Assignments

(Top View)



SO-16

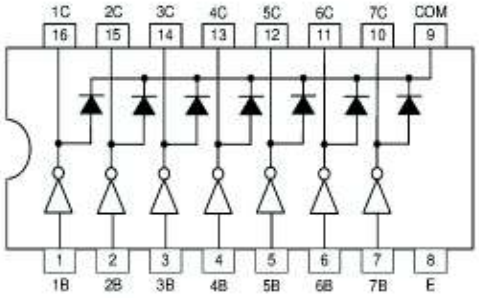
(Top View)



PDIP-16

Notes:
1. No purposely added lead. Fully EU Directive 2002/95/EC (RoHS) & 2011/65/EU (RoHS 2) compliant.
2. See http://www.diodes.com/quality/lead_free.html for more information about Diodes Incorporated's definitions of Halogen- and Antimony-free, "Green" and Lead-free.
3. Halogen- and Antimony-free "Green" products are defined as those which contain <900ppm bromine, <900ppm chlorine (<1500ppm total Br + Cl) and <1000ppm antimony compounds.

Connection Diagram



C-4: MÓDULO HC-05

ITead Studio
Make Innovation easier

Tech Support: info@iteadstudio.com

HC-05

-Bluetooth to Serial Port Module

Overview

HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup.
Serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. It uses CSR Bluecore 04-External single chip Bluetooth system with CMOS technology and with AFH(Adaptive Frequency Hopping Feature). It has the footprint as small as 12.7mmx27mm. Hope it will simplify your overall design/development cycle.

Specifications

Hardware features

- Typical -80dBm sensitivity
- Up to +4dBm RF transmit power
- Low Power 1.8V Operation ,1.8 to 3.6V I/O
- PIO control
- UART interface with programmable baud rate
- With integrated antenna
- With edge connector

PIN Name	PIN #	Pad type	Description	Note
GND	13 21 22	VSS	Ground pot	
3.3 VCC	12	3.3V	Integrated 3.3V (+) supply with On-chip linear regulator output within 3.15-3.3V	
AIO0	9	Bi-Directional	Programmable input/output line	
AIO1	10	Bi-Directional	Programmable input/output line	
PIO0	23	Bi-Directional RX EN	Programmable input/output line, control output for LNA(if fitted)	
PIO1	24	Bi-Directional TX EN	Programmable input/output line, control output for PA(if fitted)	
PIO2	25	Bi-Directional	Programmable input/output line	
PIO3	26	Bi-Directional	Programmable input/output line	
PIO4	27	Bi-Directional	Programmable input/output line	
PIO5	28	Bi-Directional	Programmable input/output line	
PIO6	29	Bi-Directional	Programmable input/output line	
PIO7	30	Bi-Directional	Programmable input/output line	
PIO8	31	Bi-Directional	Programmable input/output line	
PIO9	32	Bi-Directional	Programmable input/output line	
PIO10	33	Bi-Directional	Programmable input/output line	
PIO11	34	Bi-Directional	Programmable input/output line	

RESETB	**11**	CMOS input with weak internal pull-up	Reset if low.input debouncde so must be low for >5MS to cause a reset	
UART_RTS	**4**	**CMOS output, tri-stable with weak internal pull-up**	**UART request to send, active low**	
UART_CTS	**3**	**CMOS input with weak internal pull-down**	**UART clear to send, active low**	
UART_RX	**2**	**CMOS input with weak internal pull-down**	**UART Data input**	
UART_TX	**1**	**CMOS output, Tri-stable with weak internal pull-up**	**UART Data output**	
SPI_MOSI	**17**	**CMOS input with weak internal pull-down**	**Serial peripheral interface data input**	
SPI_CSB	**16**	**CMOS input with weak internal pull-up**	**Chip select for serial peripheral interface, active low**	
SPI_CLK	**19**	**CMOS input with weak internal pull-down**	**Serial peripheral interface clock**	
SPI_MISO	**18**	**CMOS input with weak internal pull-down**	**Serial peripheral interface data Output**	
USB_-	**15**	**Bi-Directional**		

USB_+	20	Bi-Directional		
NC	14			
PCM_CLK	5	Bi-Directional	Synchronous PCM data clock	
PCM_OUT	6	CMOS output	Synchronous PCM data output	
PCM_IN	7	CMOS Input	Synchronous PCM data input	
PCM_SYNC	8	Bi-Directional	Synchronous PCM data strobe	

AT command Default:

How to set the mode to server (master):

1. Connect PIO11 to high level.
2. Power on, module into command state.
3. Using baud rate 38400, sent the "AT+ROLE=1\r\n" to module, with "OK\r\n" means setting successes.
4. Connect the PIO11 to low level, repower the module, the module work as server (master).

AT commands: (all end with \r\n)

1. Test command:

Command	Respond	Parameter
AT	OK	-

2. Reset

Command	Respond	Parameter
AT+RESET	OK	-

3. Get firmware version

Command	Respond	Parameter
AT+VERSION?	+VERSION:<Param> OK	Param : firmware version

Example:

AT+VERSION?\r\n

+VERSION:2.0-20100601

OK

C-5: DHT11

Name: **DHT11 Humidity and Temperature Digital Sensor**
Code: **MR003-005.1**

This board is a breakout board for the DHT11 sensor and gives a digital output that is proportional to temperature and humidity measured by the sensor. Technology used to produce the DHT11 sensor grants high reliability, excellent long-term stability and very fast response time.

Each DHT11 element is accurately calibrated in the laboratory. Calibration coefficient is stored in the internal OTP memory and this value is used by the sensor's internal signal detecting process.

The single-wire serial interface makes the integration of this sensor in digital system quick and easy.

Sensor physical interfacing is realized through a 0.1" pitch 3-pin connector: +5V, GND and DATA. First two pins are power supply and ground and they are used to power the sensor, the third one is the sensor digital output signal.

Its small physical size (1.05"x0.7") and its very light weight (just 0.1oz) make this board an ideal choice to implementing small robots and ambient monitoring systems.

CONNECTIONS

DATA	Serial data output
GND	Ground
+5V	Power supply (+5V)

Tab.1 – Connections

MICROBOT di Prosseda Mirko – Strada Chiesuola 117, 04010 Borgo Carso (LT), Italy
web: http://www.microbot.it email: info@microbot.it tel: +39-347-2159275

Fig. 1 - Signals

CHARACTERISTICS

Supply voltage	+5V
Supply current (running)	0.5mA typ. (2.5mA max.)
Supply current (stand-by)	100uA typ. (150uA max.)
Temperature range	0 / +50°C ±2°C
Humidity range	20-90%RH ±5%RH
Interface	Digital
Dimensions	1.05" x 0.7" (connectors excluded)
Weight	0.1 oz (2.7g)

Tab.2 - Characteristics

SENSOR UTILIZATION

The single-wire bus needs a 5Kohm pull-up resistor and the connection with the system is realized as showed in Fig.2.

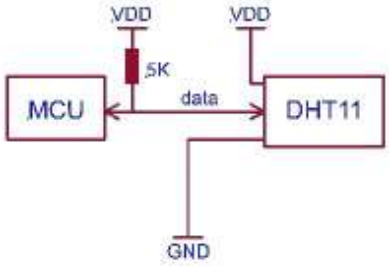


Fig. 2 - System

MICROBOT di Prosseda Mirko – Strada Chiesuola 117, 04010 Borgo Carso (LT), Italy
web: http://www.microbot.it email: info@microbot.it tel: +39-347-2159275

When power is supplied to the sensor, you haven't to send any instruction to the sensor in within one second in order to pass the start-up status.

After DHT11 is powered up, it goes in low power standby mode and it waits to recognize a Start Signal on the DATA line. Start Signal consists in a low voltage level on DATA line for a minimum of 18mS to ensure the DHT11 detects it, then followed by a pull-up voltage for about 40us.

Now the microcontroller has to wait the DHT11 transmission.

Once DHT11 detects the Start Signal, it will send out a low voltage level response signal, which lasts 80us. Then it sets the voltage level from low to high and keeps it for 80us.

Now data transmission will start. Every bit of data begins with the 50us low voltage level and then switch to high voltage level; high voltage level duration depends on the bit value that have to be transmitted: a 1 bit has an high voltage level duration of 27uS, a 0 bit has an high voltage level duration of 70uS.

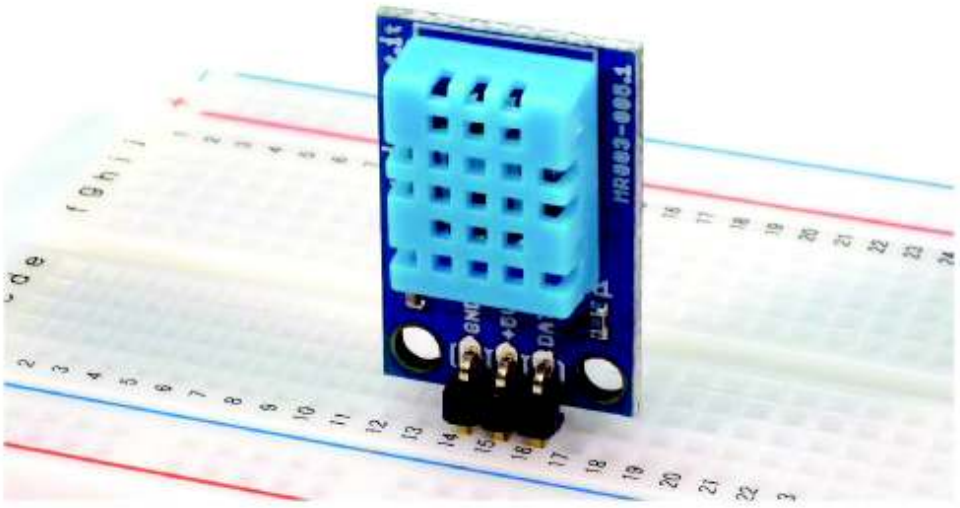
When the last bit data is transmitted, DHT11 pulls down the voltage level and keeps it for 50us, then it leaves the line pulled-up and goes back in the stand-by mode.

To make another sensor read it needs to repeat this cycle, sending again the Start Signal after a minimum of one second from the previous cycle.

A complete data transmission is 40bit, so a communication process is about 4mS. DHT11 sensor sends higher data bit first (MSB) in the following format:

Data = 8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data + 8bit check-sum.

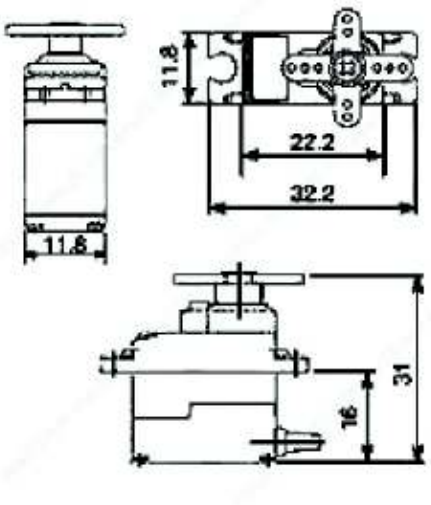
If the data transmission is right, the check-sum will be equal to the last 8bit of the sum of the four byte transmitted.



MICROBOT di Prosseda Mirko – Strada Chiesuola 117, 04010 Borgo Carso (LT), Italy
web: http://www.microbot.it email: info@microbot.it tel: +39-347-2159275

C-6: SG90

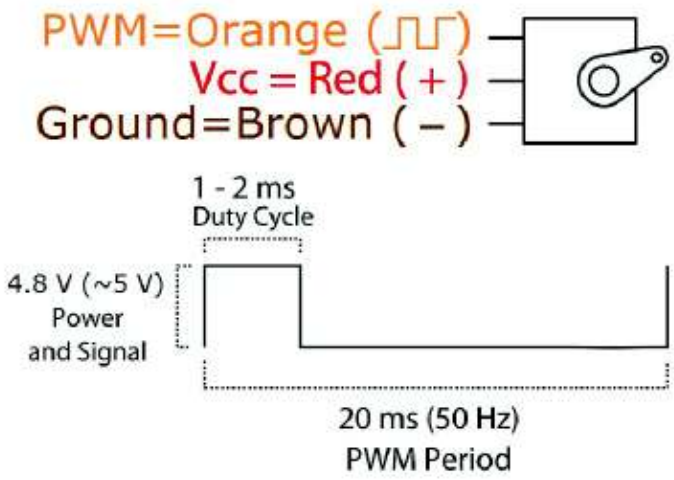
SG90 9 g Micro Servo



Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but *smaller*. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.

Specifications

- Weight: 9 g
- Dimension: 22.2 x 11.8 x 31 mm approx.
- Stall torque: 1.8 kgf·cm
- Operating speed: 0.1 s/60 degree
- Operating voltage: 4.8 V (~5V)
- Dead band width: 10 µs
- Temperature range: 0 ℃ – 55 ℃



Position "0" (1.5 ms pulse) is middle, "90" (~2 ms pulse) is all the way to the right, "-90" (~1 ms pulse) is all the way to the left.

C-7 HC-SR04

Ultrasonic Ranging Module HC - SR04

Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:
(1) Using IO trigger for at least 10us high level signal,
(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.
Test distance = (high level time×velocity of sound (340M/S) / 2,

Wire connecting direct as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

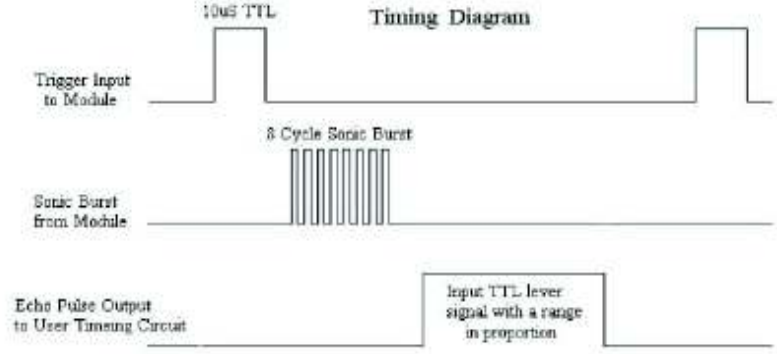
Electric Parameter

Working Voltage	DC 5 V
Working Current	15mA
Working Frequency	40Hz
Max Range	4m
Min Range	2cm
MeasuringAngle	15 degree
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion
Dimension	45*20*15mm

Vcc Trig Echo GND

Timing diagram

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



Attention:

- The module is not suggested to connect directly to electric, if connected electric, the GND terminal should be connected the module first, otherwise, it will affect the normal work of the module.
- When tested objects, the range of area is not less than 0.5 square meters and the plane requests as smooth as possible, otherwise ,it will affect the results of measuring.

Anexo D: Manual de Usuario

Descripción del Equipo

El equipo es un módulo didáctico basado en la plataforma *Arduino* Mega que posee los elementos necesarios para realizar aplicaciones con sensores, motores y comunicación inalámbrica. Este módulo beneficiará a los estudiantes de la ESFOT, que cursan la materia de Laboratorio de Microprocesadores, ya que lo utilizarán como herramienta para desarrollar sus prácticas de laboratorio.

Objetivos del Equipo

- Introducir al estudiante en temas como domótica y robótica utilizando la plataforma *Arduino* Mega como unidad de control.
- Optimizar el tiempo invertido en prácticas de laboratorio.

Componentes del Equipo

Debido a que el módulo es una herramienta para el desarrollo de prácticas de laboratorio se incluyen para su funcionamiento, los siguientes elementos:

- Tarjeta del módulo didáctico
- Chasis con 4 motores DC y ruedas
- Batería LiPo 2S 7.4V 300mAh
- Servomotor SG90
- Motor a pasos 28bjy-48
- Módulo de sensor de distancia HC-SR04, con soporte
- Módulo de sensor Infrarrojo
- Módulo de sensor de temperatura y humedad DHT-11
- Módulo Bluetooth HC-05
- Cable USB con terminales tipo A y tipo B
- Conector DC macho

A continuación, en la Figura 1 se muestra el módulo y los componentes anteriormente mencionados.

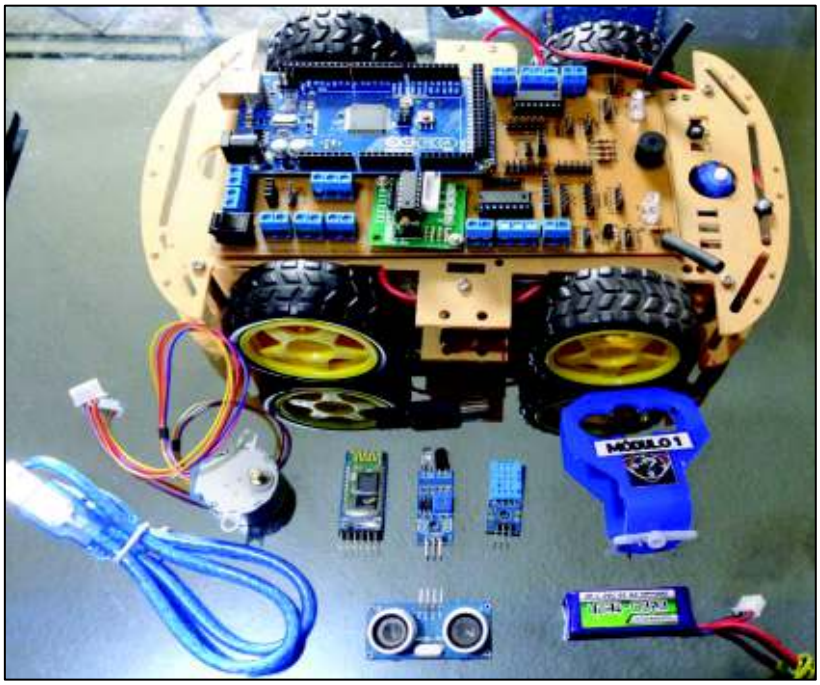


Figura 1: Componentes del módulo didáctico basado en Arduino Mega

Guía de Uso del Módulo Didáctico

Esta sección del manual tiene como objetivo guiar al usuario en el uso correcto del módulo didáctico. Los pasos que debe realizar el usuario están descritos e ilustrados.

Para familiarizar al usuario con el módulo, específicamente la tarjeta, se muestra en la siguiente Figura 2 la distribución de los elementos que la conforman.

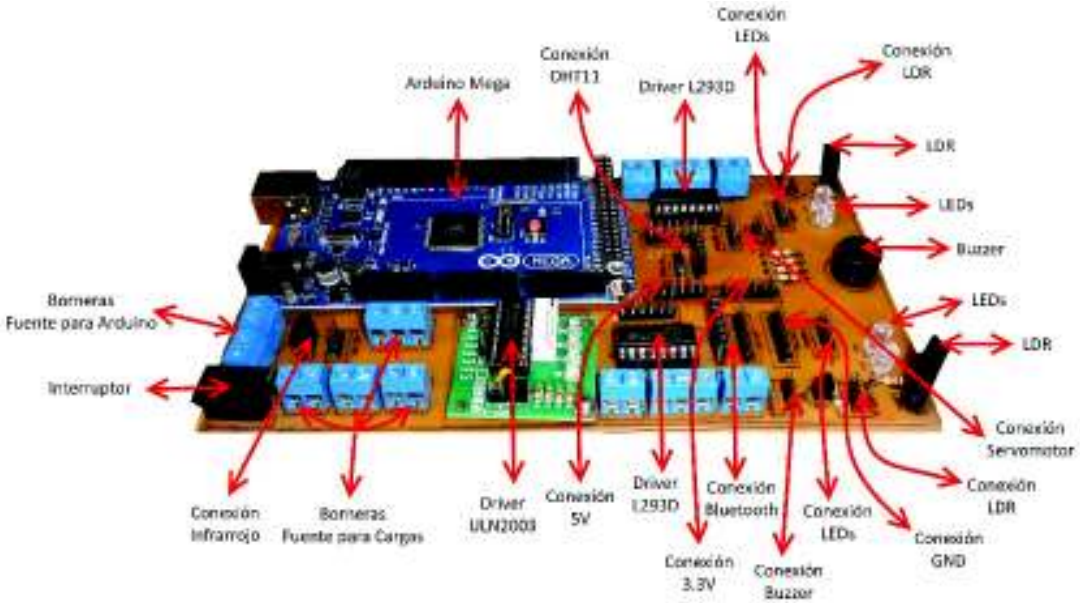


Figura 2: Componentes de la tarjeta del módulo didáctico

Instalación de *Arduino* IDE

La plataforma *Arduino* Mega, integrado en el módulo didáctico, requiere de un entorno de desarrollo para escribir un código fuente, compilarlo y posteriormente subir dicho código al microcontrolador de esta plataforma. Este entorno de desarrollo, llamado *Arduino* IDE, es distribuido gratuitamente por sus creadores y se lo puede obtener en la página oficial de *Arduino*.

Para obtener el *Arduino* IDE, visitar su página oficial (http://www.*Arduino*.cc/en/Main/Software) y descargar la última versión disponible, como se muestra en la siguiente Figura 3.

Figura 3: Página oficial de descargas de Arduino

Nota: El usuario debe descargar el software de acuerdo con su sistema operativo, el ejemplo está realizado en el sistema operativo más común, es decir, Windows.

Una vez descargado el software, ejecutar el instalador y proceder con la instalación. Aparecerá una ventana similar a la siguiente Figura 4. En la cual se debe aceptar los acuerdos de licencia.

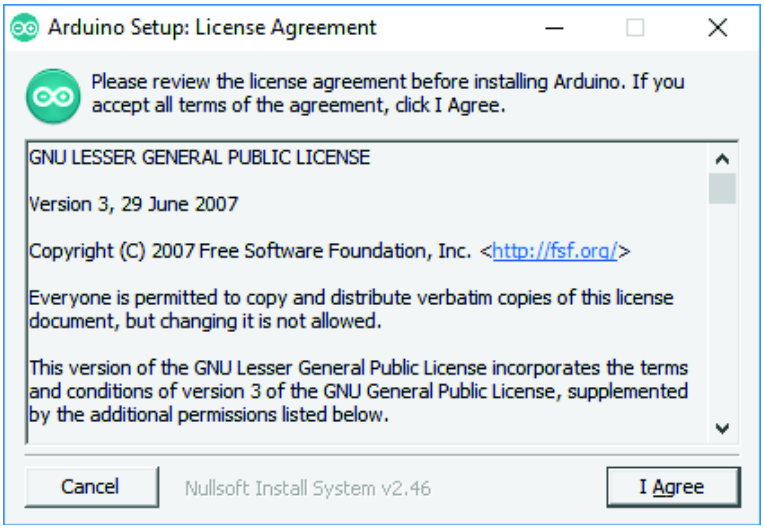


Figura 4: Acuerdo de licencia de Arduino IDE

Después, marcar todas las opciones y hacer clic en siguiente, como se muestra en la siguiente Figura 5.

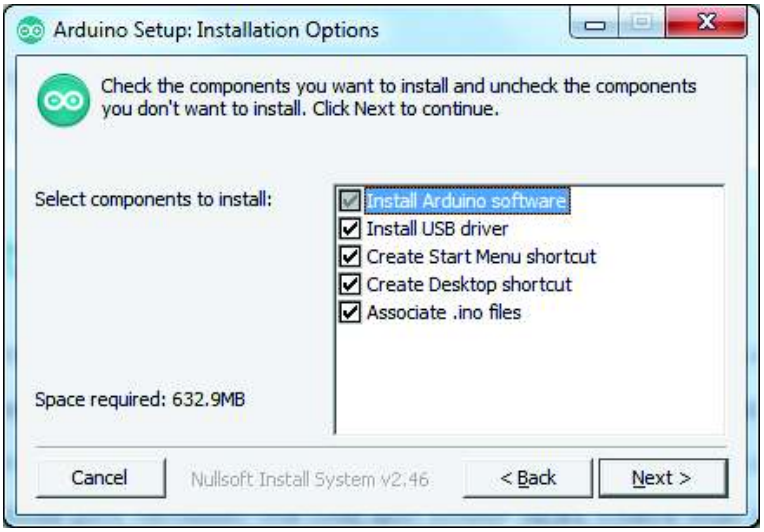


Figura 5: Ventana de opciones de configuración para la instalación de Arduino IDE

En el proceso de instalación se solicitará instalar los drivers del IDE, clic en Instalar y esperar, como se observa en la Figura 6.

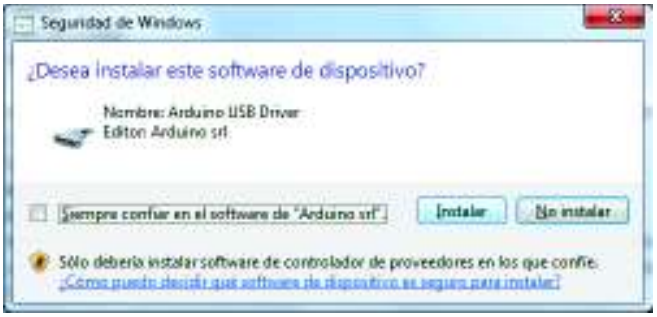


Figura 6: Ventana de instalación de drivers para Arduino IDE

Finalmente, clic en cerrar para culminar con el proceso de instalación, como se muestra en la Figura 7.

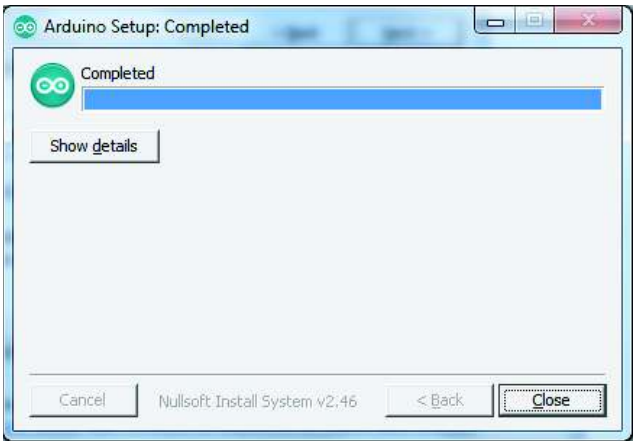


Figura 7: Instalación de Arduino IDE completada

Creación de un Nuevo Proyecto

Ejecutar el IDE de *Arduino*.

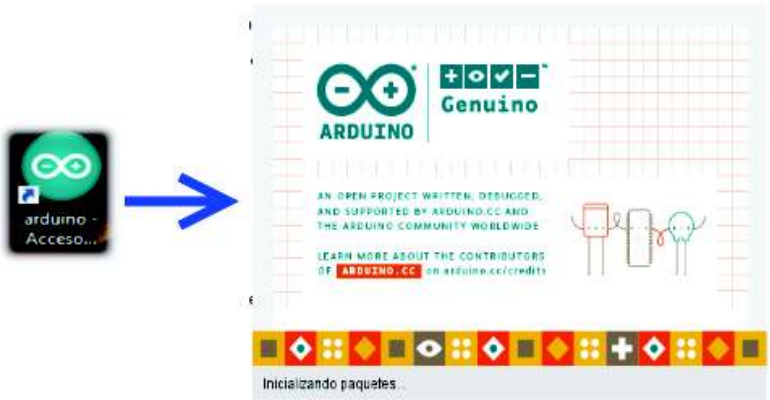


Figura 8: Ícono del Arduino IDE y proceso de ejecución

Una vez ejecutado se mostrará la siguiente ventana. Esta ventana es el editor de texto del IDE de *Arduino*, en el cual, se puede escribir el código fuente deseado.

Nota: Para crear una nueva ventana, para un nuevo programa, presionar las teclas Ctrl+N o dirigirse al menú Archivo y elegir la opción Nuevo.

Figura 9: Editor de texto de Arduino IDE

En el editor de texto del IDE, escribir el programa deseado, como se muestra en la siguiente Figura 10. El programa mostrado en el ejemplo es una aplicación de *LED* intermitente a 1 segundo.

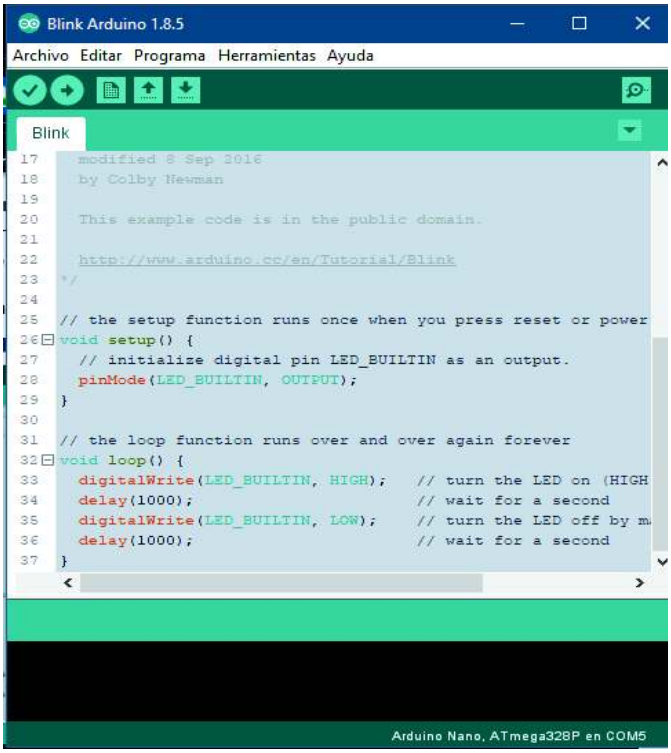


Figura 10: Código fuente escrito en Arduino IDE

Una vez escrito y guardado el programa es necesario verificarlo. Para ello se debe hacer clic sobre el botón verificar que está ubicado en la parte superior izquierda de la ventana principal del IDE de *Arduino*.

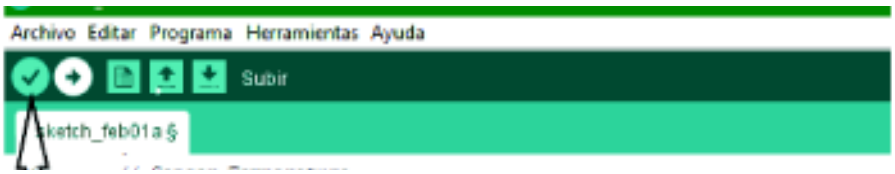


Figura 11: Verificación de sintaxis del código fuente

NOTA: **Si el programa no ha sido guardado en ningún momento del proceso anterior, el IDE solicitará que se guarde el programa antes de verificarlo. Para ello asigne un nombre al programa y un directorio, después clic en guardar y continuará con la verificación.*

Si el programa está escrito correctamente se mostrará un mensaje de compilado en la parte inferior izquierda de la ventana principal del IDE de *Arduino*. Pero si el programa tiene errores el usuario debe hacer las respectivas correcciones ya que normalmente son errores de sintaxis del código.

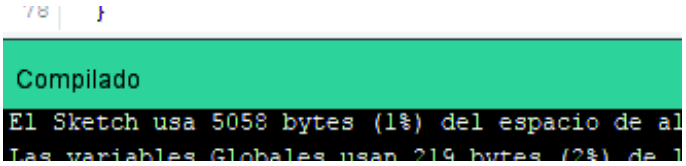


Figura 12: Código fuente compilado exitosamente

Una vez verificado el código se procederá a subirlo al *Arduino* Mega del módulo didáctico. Seguir los siguientes pasos para poder hacerlo.

- **Paso 1:** Conectar el *Arduino* al computador, utilizando el cable USB proporcionado.

  - Conector USB tipo A ----- PC
  - Conector USB tipo B ----- *Arduino*

Figura 13: Cable USB con terminales Tipo A y Tipo B

- **Paso 2:** Seleccionar el tipo de *Arduino*.

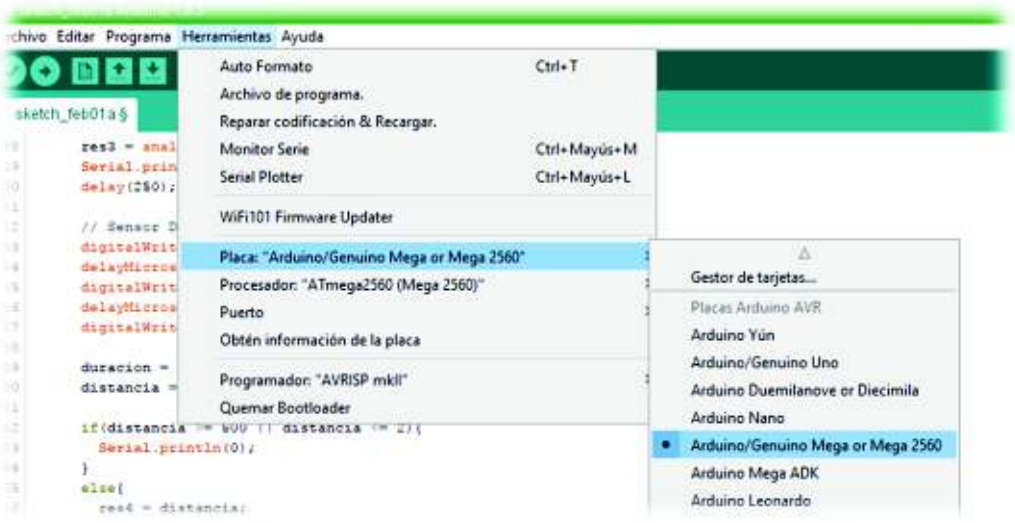


Figura 14: Selección del tipo de plataforma utilizada

- **Paso 3**: Configurar el puerto.
  *NOTA: *El puerto no necesariamente debe ser el mismo que el mostrado en la imagen.*

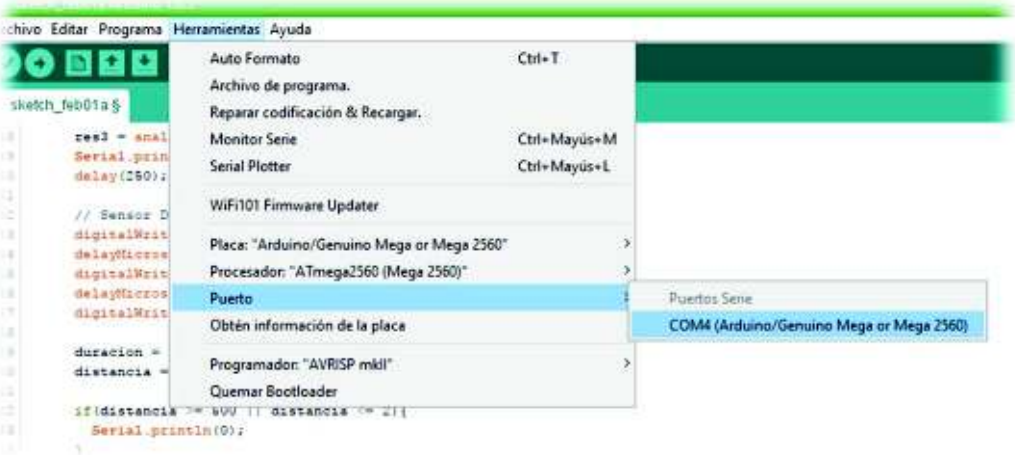


Figura 15: Configuración de puerto

- **Paso 4:** Subir al *Arduino*. Clic en el botón subir y esperar que el IDE finalice.

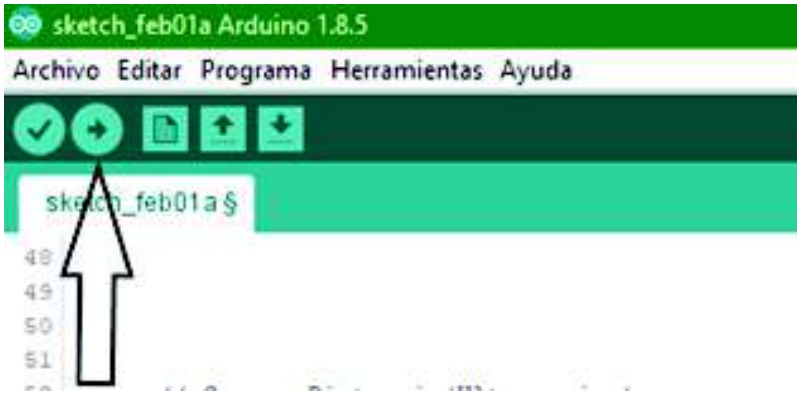


Figura 16: Subir programa a la plataforma Arduino Mega

Monitor Serial del IDE de *Arduino*

Si el programa realizado requiere utilizar un monitor serial, el IDE de *Arduino* facilita el uso de esta herramienta ya que tiene incluido un monitor, y para utilizarlo se debe previamente haber configurado el puerto, el tipo de *Arduino* mantener y mantener el *Arduino* conectado con el cable USB al computador. Después abrir el monitor serie del IDE de *Arduino* para verificar que el microcontrolador está enviando datos al computador. De la siguiente manera:

Paso 1: Para abrir el monitor serie, hacer clic sobre la siguiente opción.

Figura 17: Proceso para abrir monitor serie

Paso 2: Se abrirá una ventana similar a la siguiente y se mostrará lo que el *Arduino* Mega está enviando al computador. En este caso el *Arduino* envía al computador datos de sensores.

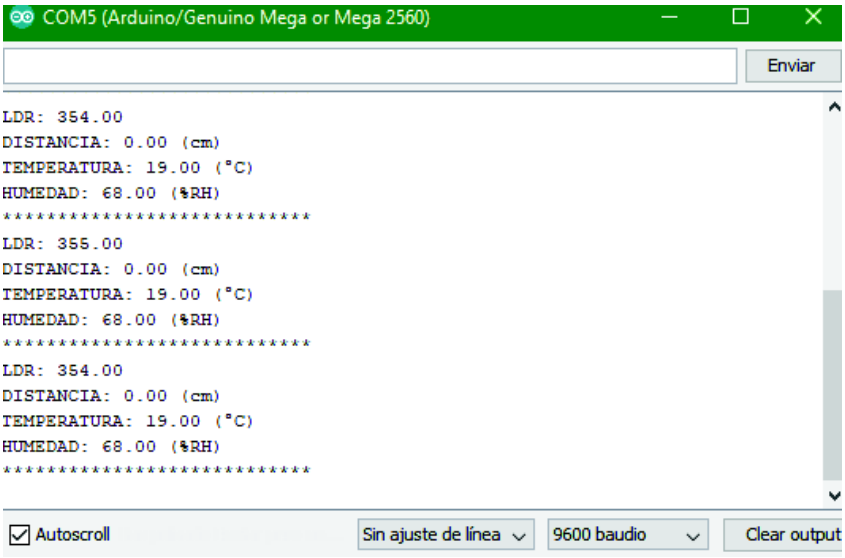


Figura 18: Monitor serie del IDE de Arduino

Instalar Librerías para IDE de *Arduino*

En caso de que el proyecto requiera de una librería y el IDE de *Arduino* no la posea, a continuación, se muestra los pasos a seguir para incluir una librería previamente descargada de la red.

- **Paso 1:** Ir a la Barra de menú del IDE de *Arduino*/Incluir Librería/Añadir librería.ZIP.

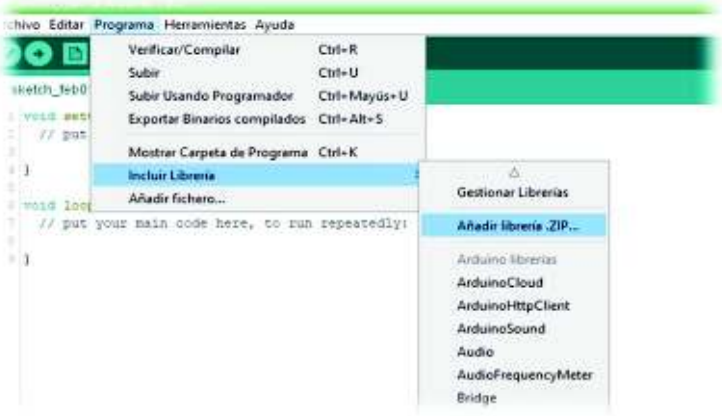


Figura 19. Proceso de instalación de librería

- **Paso 2:** Después se mostrará un explorador de archivos en el que se debe localizar el archivo con el nombre de la librería deseada, en este caso es DHT11.zip. Una vez encontrado el archivo se procede a hacer clic en Abrir. Esperar un momento y dejar que el IDE realice la instalación.

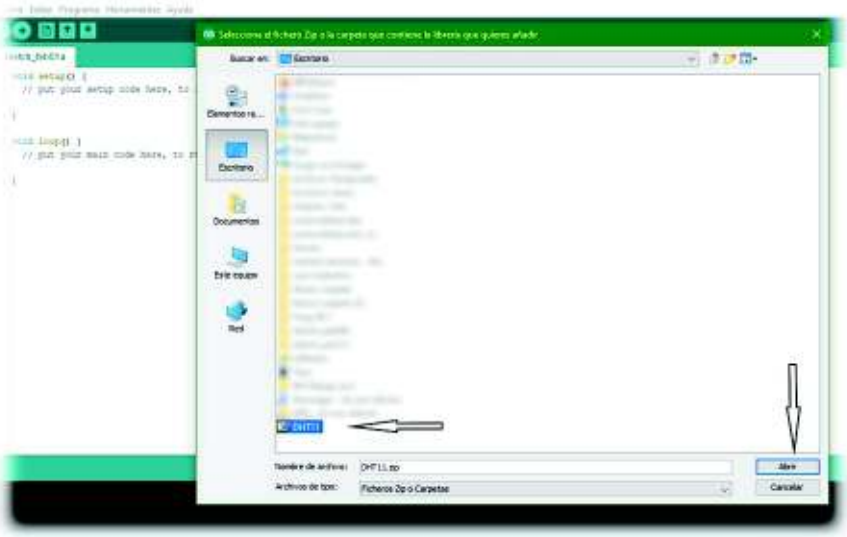


Figura 20: Explorador de archivos de Windows

Alimentación Eléctrica del Módulo

Para alimentar eléctricamente el módulo existen varias formas que se mencionan a continuación:

Alimentación a través del puerto USB de un computador

Para alimentar el módulo desde el computador se utilizará el cable USB, incluido con el módulo, y se procederá a la conexión como se muestra en la siguiente Figura 21 El *LED* verde es un indicador de que el *Arduino* está encendido y que la alimentación es correcta.

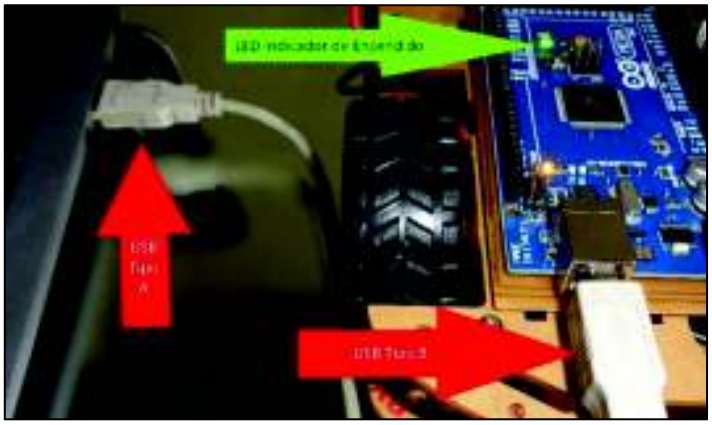


Figura 21: Alimentación a través de puerto USB de un computador

Para utilizar este puerto como fuente de energía, la aplicación debe cumplir las siguientes condiciones:

- El circuito debe operar con un voltaje menor o igual a 5 Vcc.
- Las cargas conectadas en cada pin del *Arduino* Mega deben consumir una corriente menor o igual a 40 mA.
- La corriente total del circuito debe ser menor a 500 mA.

Alimentación con batería o adaptador

La plataforma *Arduino* Mega tiene un regulador de voltaje de 5V, el cual, será utilizado para regular el voltaje entregado por una batería LiPo de 7.2V. Para ello se verifica que el interruptor de la tarjeta este apagado y después proceder a realizar las conexiones (Cables rojos V+ y Cables negros GND), como se muestra en la Figura 2:

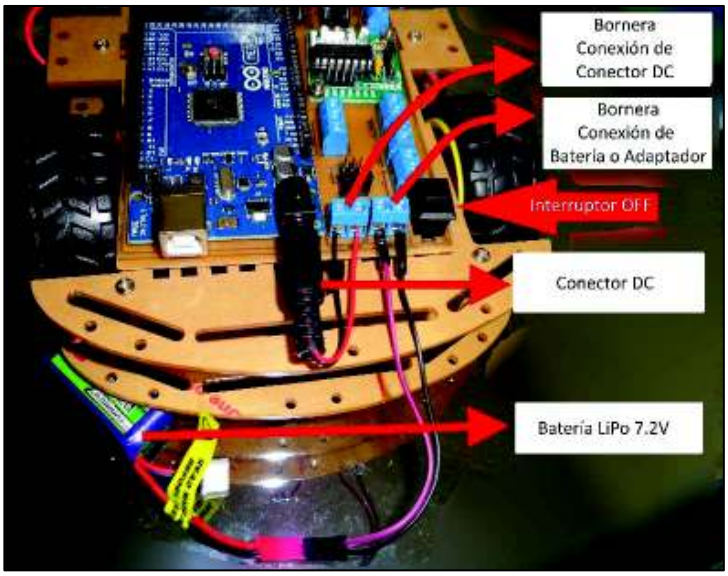


Figura 22: Alimentación eléctrica con batería LiPo

Después de haber realizado la conexión se procede a activar el interruptor y verificar si el *LED* verde se enciende, lo que indica que el *Arduino* esta energizado correctamente, como se observa en la Figura 22.

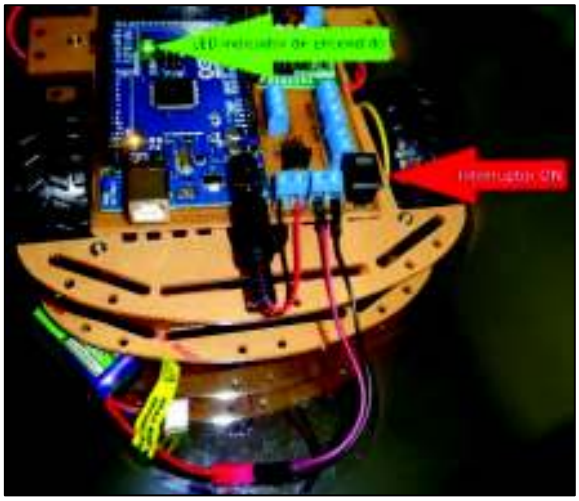


Figura 23. Verificación de encendido a través de LED indicador

IMPORTANTE: Si el usuario está usando este tipo de alimentación y desea subir un programa al *Arduino Mega* primero se debe apagar la plataforma utilizando el interruptor y después se procede normalmente a programar el Arduino como se mencionó en pasos anteriores. Esto se debe tener en cuenta para evitar daños en la plataforma, en la fuente externa o en el puerto USB del computador.

Para utilizar este conector la aplicación debe cumplir las siguientes condiciones:

- El circuito debe operar con un voltaje menor o igual a 5Vcc.
- Las cargas conectadas en cada pin del *Arduino* Mega deben consumir una corriente menor o igual a 40mA.
- La corriente total del circuito debe ser menor a 1000mA.
- El voltaje de entrada de la batería o adaptador se recomienda que sea entre 7V y 12V, pero también se puede utilizar voltajes entre 6V y 20V.

Alimentación para motores y otras cargas

Para alimentar cargas como motores u otros dispositivos que requieran de un voltaje mayor a 5V y/o una intensidad de corriente alta, se debe agregar otra fuente de energía, como se muestra en la Figura 24 y Figura 25. Tener en cuenta que los cables rojos son V+ y los cables negros son GND.

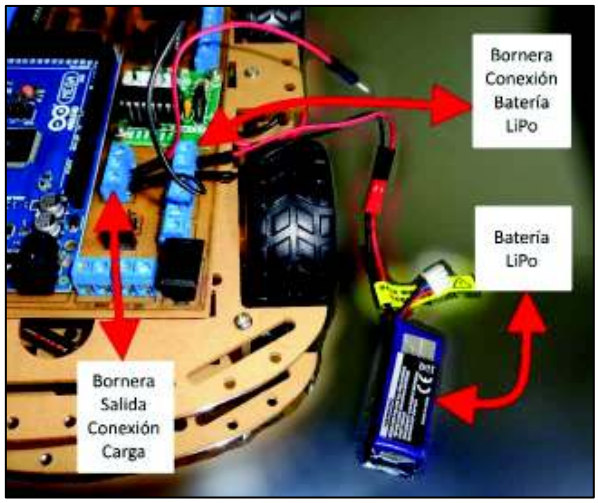


Figura 24: Primera forma de conexión para fuente externa para cargas

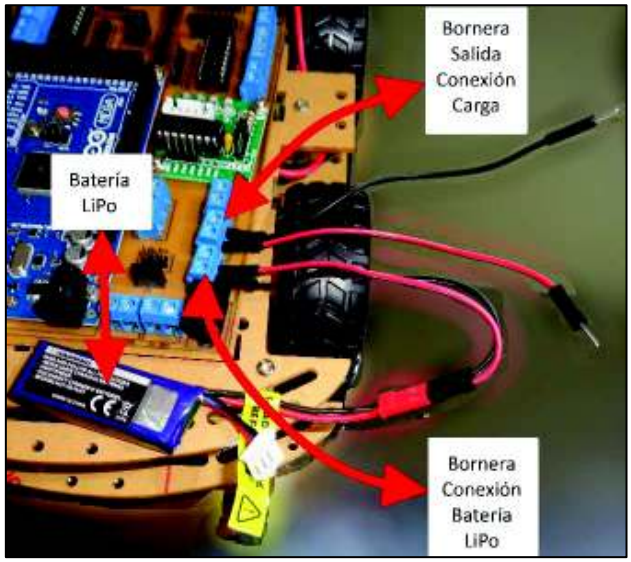


Figura 25: Segunda forma de conexión para fuente externa para cargas

La fuente de energía utilizada para alimentar a la carga ya sea esta; una batería o un adaptador, queda bajo el criterio del usuario ya que los valores de la fuente dependen de la carga utilizada, por lo que se recomienda elegir la fuente cuidadosamente.

Interfaces del Módulo

En esta sección se presenta detalladamente la función y conexión de los pines de cada interfaz disponible en el módulo. Las interfaces que posee el módulo son las siguientes:

- Interfaces de alimentación
- Interfaces para dos LDR
- Sensor ultrasónico HC-SR04 y soporte
- Interfaz para sensor DHT-11
- Interfaz para sensor infrarrojo
- Interfaz para motor a pasos unipolar
- Interfaz para motor DC
- Interfaz para servomotor
- Interfaz para buzzer
- Interfaces para LEDs
- Interfaz para Bluetooth

Para una mejor explicación se utilizará el diagrama eléctrico que se muestra en la siguiente Figura 26.

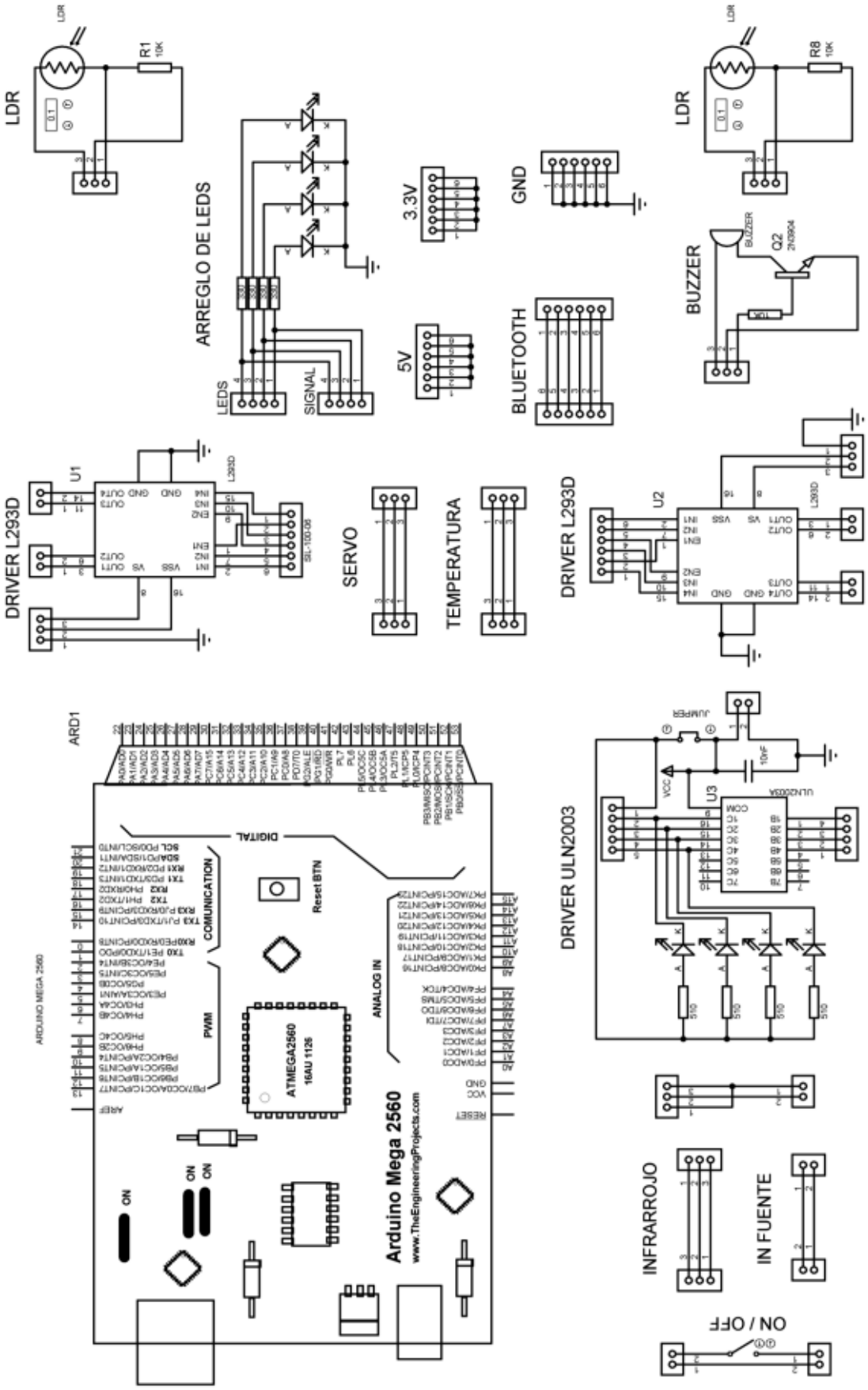


Figura 26: Circuito eléctrico de la placa del módulo

Además, para que el usuario pueda identificar y ubicar las interfaces en la placa del módulo se recomienda revisar la Figura 2 presentada al principio de este manual.

Interfaces de alimentación para 5V, 3.3V y GND

El módulo tiene pines para conexión de 5V, 3.3V y GND, cada uno cuenta con 6 pines respectivamente, los cuales servirán para conectar las interfaces que lo requieran.

Los pines por si solos no están energizados y para que puedan suministrar su voltaje respectivo, estos requieren de un puente (cable) desde el Arduino Mega a uno de los pines según el tipo de voltaje, como se describe a continuación.

- Pines 5V: Se cuenta con 6 pines, uno de estos debe ser conectado al pin de 5V o Vcc del Arduino.
- Pines 3.3V: Se cuenta con 6 pines, uno de estos debe ser conectado al pin de 3.3V del Arduino.
- Pines GND o Tierra: Se cuenta con 6 pines, uno de estos debe ser conectado al pin GND del Arduino.

Observación: *Los pines de GND están conectados en placa con los LEDs, es decir, una vez conectado el pin de tierra al Arduino, los LEDs también se conectarán a GND.*

Interfaces para LEDS

La placa cuenta con cuatro LEDs que el usuario puede usar para proyectos como secuencia de encendido, control de brillo, indicadores de algún evento, etc. Para su funcionamiento los LEDs requieren de conexión a tierra y un voltaje positivo, como se describe a continuación.

- Pines GND: Se cuenta con 6 pines, uno de estos debe ser conectado al pin GND del Arduino.
- Pines LEDs: Estos pines pueden ser conectados a los pines digitales del Arduino para control de encendido o apagado. También pueden ser conectados a pines de salidas PWM del Arduino para control de intensidad de brillo de los LEDs.

Interfaces para LDR

El módulo está compuesto por dos sensores LDR, utilizados para sensar los niveles de iluminación que inciden sobre este.Cada interfaz tiene 3 pines de conexión que son: Vcc, GND y Señal, el circuito eléctrico puede observarse en la Figura 26.

- Pin Vcc: Este pin debe ser conectado a 5V de la placa, que servira como voltaje de entrada para el divisor de tensión entre el LDR y la resistencia fija de 10kΩ.
- Pin GND: Este pin debe ser conectado a cualquiera de los 5 pines de tierra de la placa.
- Pin Señal: Este es un pin de salida de una señal análoga que varía entre 0V - 5V y debe ser conectado a uno de los pines análogos del Arduino.

Observación: Para la lectura de la señal análoga se utilizará la función *analogRead(pin)* del pin al que se haya conectado. La resolución del conversor análogo a digital del Arduino es de 10 bits.

Interfaces para DHT-11

El módulo posee un sensor DHT-11 que mide dos magnitudes físicas como son: temperatura y humedad. El módulo tiene 3 pines de conexión Vcc, GND y Señal, estos 3 pines deben ser

conectados del módulo a la placa y de la placa al Arduino. El circuito eléctrico de los pines de esta interfaz se puede observar en la Figura 26.

- Pin Vcc: Según el fabricante del módulo este sensor requiere de 5V para su funcionamiento, por lo tanto, se debe conectar a 5V disponible en los pines de la placa.
- Pin GND: El pin de tierra del módulo DHT-11 debe ser conectado a los pines disponibles de tierra de la placa.
- Pin Señal: Este pin tiene una señal digital de salida por lo tanto se debe conectar a un pin digital del Arduino. El pin del Arduino debe ser declarado como entrada.

Observación: Para leer la señal del módulo DHT-11, se debe importar una librería llamada "DHT11" y utilizar la función *dht11.Read(humedad, temperatura)* del pin al que esté conectado. Los rangos de temperatura son de 0 a 50°C /± 2°C y los rangos de humedad son de 20 a 90%RH/±5%RH. Para mayores especificaciones consultar en internet el datasheet del módulo.

Interfaces para sensor Infrarrojo

El módulo posee un sensor Infrarrojo que detecta obstáculos y emite una señal digital, es decir; envía al Arduino 0L si detecta un obstáculo y 1L si no los hay. El módulo tiene 3 pines de conexión Vcc, GND y Señal, estos 3 pines deben ser conectados del módulo a la placa y de la placa al Arduino. El circuito eléctrico de los pines de esta interfaz se puede observar en la Figura 26.

- Pin Vcc: Según el fabricante del módulo este sensor requiere de 5V para su funcionamiento, por lo tanto, se debe conectar a 5V disponible en los pines de la placa.
- Pin GND: El pin de tierra del módulo Infrarrojo debe ser conectado a los pines disponibles de tierra de la placa.
- Pin Señal: Este pin tiene una señal digital de salida por lo tanto se debe conectar a un pin digital del Arduino. El pin del Arduino debe ser declarado como entrada.

Observación: Para leer la señal del módulo Infrarrojo, se debe utilizar la función *digital.Read(pin)* del pin al que esté conectado. Los rangos de distancia a los que puede detectar obstáculos son de 0 a 40 cm y es regulable a través de su potenciómetro. Para mayores especificaciones consultar en internet el datasheet del módulo.

Interfaces para sensor ultrasónico HC-SR04

El módulo didáctico posee un sensor ultrasónico HC-SR04 que mide distancia utilizando ondas de sonido. El módulo tiene 4 pines de conexión Vcc, GND, Echo y Trigger, los pines de alimentación deben ser conectados en placa y los restantes deben ser conectados directamente a pines digitales del Arduino.

- Pin Vcc: Según el fabricante este sensor requiere de 5V para su funcionamiento, por lo tanto, se debe conectar a 5V disponible en los pines de la placa.
- Pin GND: El pin de tierra del módulo Infrarrojo debe ser conectado a los pines disponibles de tierra de la placa.
- Pin Echo: Este pin envía una señal digital a la unidad de control, por lo tanto, se debe conectar a un pin digital del Arduino. El pin del Arduino debe ser declarado como entrada.

- Pin Trigger: Este pin recibe una señal digital, por lo tanto, se debe conectar a un pin digital del Arduino. El pin del Arduino debe ser declarado como salida.

Observación: Para enviar una señal al pin Trigger del sensor se debe utilizar la función *digital.Write(pin)* del pin al que esté conectado y para detectar el tiempo en que demora en pasar el pin Echo de LOW a HIGH se debe utilizar la función *pulseIn(pin , HIGH)* . Los rangos de distancia a los que puede detectar obstáculos son de 2 cm a 40 m. Para mayores especificaciones consultar en internet el datasheet del módulo.

Interfaz para Buzzer

La placa cuenta con un buzzer que el usuario puede usar para proyectos como emitir sonidos a diferentes frecuencias, indicador de algún evento, etc. Para su funcionamiento el buzzer requiere de conexión a tierra, un voltaje positivo de 5V y una señal de control, como se describe a continuación.

- Pin GND: Este pin debe ser conectado al pin GND de la placa.
- Pines Vcc: Este pin tiene que ser conectado a 5V.
- Pin Señal: Este es un pin de entrada que debe ser conectado a un pin digital del Arduino para activar o desactivar el buzzer. El pin del Arduino debe ser declarado como salida.

Observación: Para enviar una señal de activación del buzzer desde el Arduino se debe utilizar la función *digital.Write(pin)* del pin al que esté conectado. Además, se puede generar sonidos a distintas frecuencias utilizando pines digitales y la función *tone(pin, frecuencia, duración).* Los rangos de frecuencia que soporta el buzzer son de 0 a 2300Hz / ±300Hz.

Interfaz para Bluetooth HC-05

El módulo didáctico posee una interfaz para conectar un módulo HC-05, que se lo puede utilizar para enviar o recibir datos de otro dispositivo de forma inalámbrica. El módulo tiene 6 pines de los cuales solamente 4 deben conectarse a la interfaz, los pines de conexión son: Vcc, GND, TX y RX, estos pines deben ser conectados del módulo a la placa y de la placa al Arduino. El circuito eléctrico de los pines de esta interfaz se puede observar en la Figura 26.

- Pin Vcc: Según el fabricante del módulo Bluetooth se requiere de 5V para su funcionamiento, por lo tanto, se debe conectar a 5V disponible en los pines de la placa.
- Pin GND: El pin de tierra del módulo debe ser conectado a los pines disponibles de tierra de la placa.
- Pin RX: Este pin recibe los datos desde el Arduino, por lo tanto, debe ser conectado a un pin TX del Arduino que transmita dichos datos.
- Pin TX: Este pin transmite los datos hacia el Arduino por lo tanto debe ser conectado a un pin RX del Arduino que reciba dichos datos.

Observación: Para trabajar con este módulo Bluetooth se requiere de una librería llamada “SoftwareSerial”.

Interfaces para Servomotor

El módulo didáctico posee un servomotor sg90. El módulo tiene 3 pines de conexión Vcc, GND y Señal, estos 3 pines deben ser conectados del módulo a la placa y de la placa al módulo. El circuito eléctrico de los pines de esta interfaz se puede observar en la Figura 26.

- Pin Vcc: Según el fabricante del módulo este sensor requiere de 5V para su funcionamiento, por lo tanto, se debe conectar a 5V disponible en los pines de la placa.
- Pin GND: El pin de tierra del servomotor debe ser conectado a los pines disponibles de tierra de la placa.
- Pin Señal: Este pin recibirá una señal digital desde el Arduino por lo tanto se debe conectar a un pin digital del Arduino. El pin debe ser declarado como salida.

Observación: Para trabajar con el servomotor se utilizará una librería llamada "servo". Los ángulos a los que puede rotar este servomotor son de 0° a 180°.

Interfaces para motor a pasos

El módulo didáctico posee un driver ULN2003D exclusivo para el motor a pasos 28byj-48. Consta de 4 pines de control denominados IN1, IN2, IN3 e IN4, 5 pines para activar las bobinas del motor y 2 pines para alimentación (Vcc y GND). El diagrama circuital de esta interfaz se lo puede observar en la Figura 26.

- Pin Vcc: Según el fabricante del módulo este driver puede ser alimentado entre 5V a 12V para su funcionamiento, por lo tanto, al poseer 5V disponible en los pines de la placa se procede a conectar en dichos pines.
- Pin GND: El pin de tierra del servomotor debe ser conectado a los pines disponibles de tierra de la placa.
- Pines IN1 IN2 IN3 IN4: Estos pines deben ir conectados a pines digitales del Arduino y deben ser activados en cierta secuencia para que el motor funcione ya sea en giro horario o antihorario.
- Pines del motor: El motor debe ser conectado en estos pines, y se activaran según la secuencia realizada en los pines de control mencionados anteriormente.

Interfaces para motores DC

El módulo didáctico posee un driver L293D que permite conectar hasta dos motores, pero el módulo tiene los motores de cada lado conectados en paralelo por lo tanto el driver interpretará esos dos motores como uno solo. Consta de 4 pines de control denominados IN1, IN2, IN3 e IN4, 2 pines Enables 4 pines para activar los motores y 3 pines para alimentación (Vss Vs y GND). El diagrama circuital de esta interfaz se lo puede observar en la Figura 26.

- Pin Vss: Según el fabricante este driver puede ser alimentado con 5V para la sección de control, por lo tanto, se debe conectar a un pin de 5V de la placa.
- Pin Vs: Este pin es para alimentar los motores soporta voltaje de hasta 36V.
- Pin GND: El pin de tierra del servomotor debe ser conectado a los pines disponibles de tierra de la placa.
- Pines IN1 IN2 IN3 IN4: Estos pines deben ir conectados a pines digitales del Arduino y deben ser activados en cierta forma para que el motor funcione ya sea en giro horario o antihorario.

- Pines Enable: Posee 2 pines Enable utilizado para activar los motores. También recibe señal PWM para el control de velocidad del motor.
- Pines del motor: El motor debe ser conectado en estos pines, y se activaran según la secuencia realizada en los pines de control mencionados anteriormente.

Prueba de Funcionamiento del *Arduino* Mega

Para verificar que el *Arduino* Mega esté funcionando correctamente se debe realizar una aplicación sencilla que es un *LED* intermitente integrado en la misma plataforma. Para ello, se debe utilizar el programa de ejemplo que nos ofrece el *Arduino* IDE.

Como primer paso se debe conectar el *Arduino* Mega al computador utilizando el cable USB, como se mostró en pasos anteriores. Después abrir el *Arduino* IDE en el computador y dirigirse al menú Archivo y seleccionar las siguientes opciones; Ejemplos, *Basics* y *Blink*. como se muestra en la Figura 26.

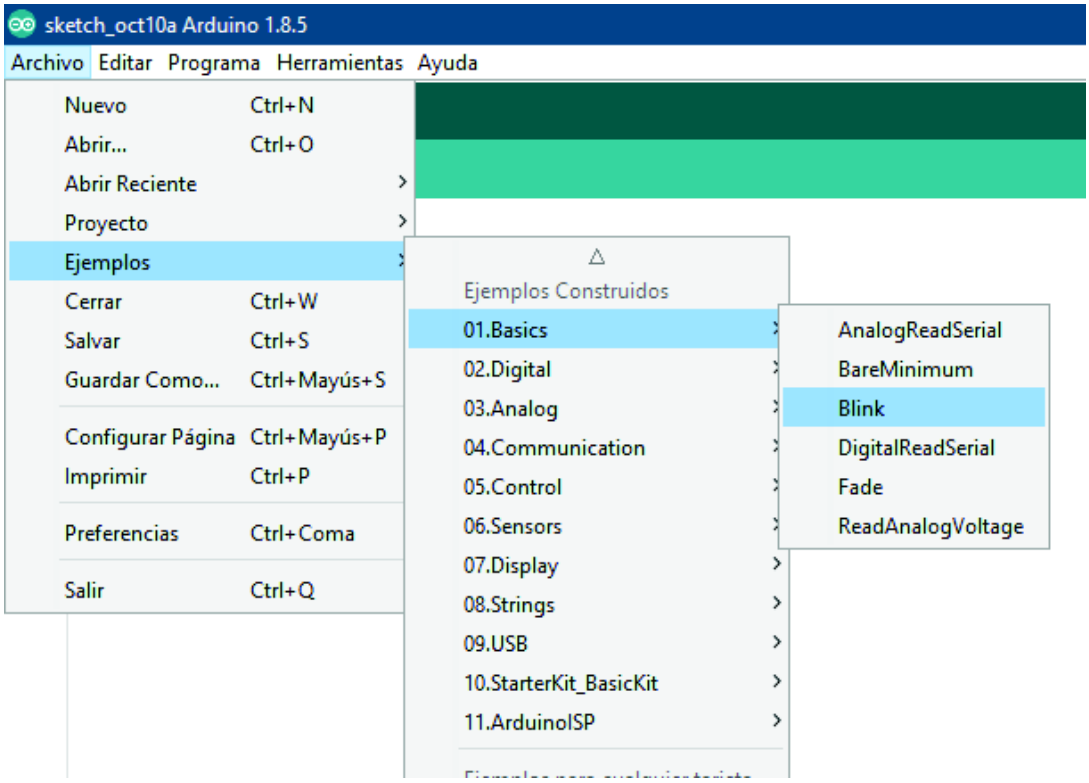


Figura 26: Programa de ejemplo para LED intermitente del IDE de Arduino

Después de haber realizado el paso anterior se desplegará una nueva ventana, como se muestra en la Figura 27, con el programa del *LED* intermitente. Verificar que el puerto y el *Arduino* sean los correctos y subir el programa a la plataforma, para hacerlo revisar los pasos de la sección “Creación de un Nuevo Proyecto” de la guía de usuario.

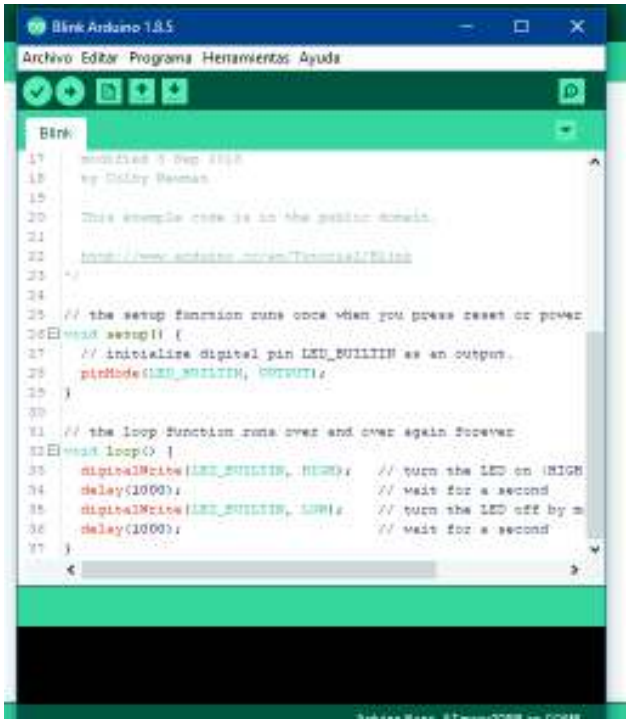


Figura 27: Programa de ejemplo del Arduino IDE

Una vez subido el programa a la plataforma, el *LED* comenzar a encenderse y a apagarse una y otra vez, como se muestra en la Figura 28, lo que indica que el *Arduino* Mega funciona correctamente.

Figura 28: Prueba de funcionamiento con código de ejemplo

Problemas comunes que se pueden dar con el uso del módulo

El computador no identifica y no asigna un puerto al *Arduino* Mega

Si el computador no realiza estas tareas implica daños en el puerto USB del computador, en el cable USB, o en el hardware del sistema de comunicación serial del *Arduino*.

Para solucionar el problema pruebe realizando lo siguiente:

- Compruebe si el cable está conectado correctamente.
- Utilice otro cable USB.
- Conecte el cable en otro puerto USB del computador.
- Conecte el módulo a otro computador y verifique.

Si ninguna de las opciones anteriores funciona implicaría daños en el hardware del *Arduino* lo que requiere un cambio inmediato de la plataforma.

El programa no puede subirse al *Arduino*

Si al intentar subir el programa al *Arduino* Mega presenta errores, realice lo siguiente:

- Compruebe si el cable está conectado correctamente.
- Revise si su código fuente tiene errores de sintaxis y corríjalas.
- Verifique si el puerto COM, que utiliza el *Arduino* Mega, está seleccionado.
- Compruebe si la placa seleccionada es *Arduino/Genuino Mega or Mega 2560*.
- Si está utilizando los pines de transmisión y recepción del *Arduino* Mega, desconecte y deje libre estos pines e intente nuevamente.

Si el error persiste implican daños en el hardware del *Arduino*, por lo que se recomienda el cambio inmediato de la plataforma.

Utiliza una batería u otra fuente de alimentación para el *Arduino*, pero no enciende.

Si está utilizando una batería o fuente para alimentar el *Arduino*, pero este no enciende verifique lo siguiente:

- Revise si las conexiones están realizadas correctamente.
- Compruebe con un multímetro si la fuente de energía conectada está suministrando voltaje.
- Verifique que el interruptor esté en modo encendido.
- Ajuste las borneras de la placa y verifique que los cables de la fuente y del conector tengan un buen contacto.
- Mida con un multímetro, si existe voltaje en el conector.
- Compruebe si las pistas electrónicas de las borneras están levantadas debido a la presión al momento de ajustar.

Si el error persiste implican daños en el hardware del *Arduino*, por lo que se recomienda el cambio inmediato de la plataforma.

Mantenimiento

Para el mantenimiento de estos módulos se debe seguir el procedimiento mostrado a continuación hasta detectar el o los elementos que ocasionaban un mal funcionamiento. El procedimiento es el siguiente:

- Extraer la placa electrónica de la estructura móvil.
- Limpieza de la placa con limpia contactos y cepillo.
- Inspección visual de posibles sueldas frías.
- Inspección visual de posibles daños en elementos electrónicos como circuitos integrados, transistores, resistencias, módulos, etc.
- Revisión de elementos electrónicos utilizando multímetro y circuitos de prueba como probador de transistores.
- Comprobación de continuidad en pistas conductoras de la placa electrónica.

Si se detecta alguna falla con el procedimiento anterior, se procede a realizar las respectivas reparaciones.

Recomendaciones Generales

Se recomienda que el rango de voltaje de entrada al regulador del *Arduino* Mega debe estar entre 7Vcc y 12Vcc, pero puede tener voltajes de entrada mínimo de 6Vcc y máximo de 20Vcc.

Cuando se alimente eléctricamente al *Arduino* Mega o a las cargas a través de baterías o adaptadores, se debe verificar que las conexiones estén polarizadas correctamente, tal y como se mostró en la guía de usuario, esto para evitar daños en el módulo.