TÍTULO DEL TRABAJO DE GRADO
MÓDULO DE ENTRENAMIENTO PARA EL DESARROLLO DE APLICACIONES ELECTRÓNICAS ORIENTADAS A LA INDUSTRIA BASADO EN MICROCONTROLADOR ARDUINO.

AUTORES
BRAYAN RICARDO SÁNCHEZ BARRANCO 1096253868

UNIDADES TECNOLÓGICAS DE SANTANDER
FACULTAD DE CIENCIAS NATURALES E INGENIERÍAS
TECNOLOGÍA EN OPERACÍON Y MANTENIMIENTO ELECTROMECANICO
BARRANCABERMEJA
FECHA DE PRESENTACIÓN:01-06-2020

TÍTULO DEL TRABAJO DE GRADO
MÓDULO DE ENTRENAMIENTO PARA EL DESARROLLO DE APLICACIONES ELECTRÓNICAS ORIENTADAS A LA INDUSTRIA BASADO EN MICROCONTROLADOR ARDUINO.

AUTORES
BRAYAN RICARDO SÁNCHEZ BARRANCO 1096253868

Trabajo de Grado para optar al título de
TECNOLOGO EN OPERACÍON Y MANTENIMIENTO ELECTROMECANICO

DIRECTOR
FREDY ALBERTO ROJAS ESPINOZA

CODIRECTOR
LUIS OMAR SARMIENTO

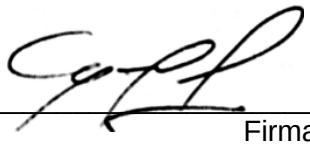
GRUPO DE INVESTIGACIÓN EN INGENIERIAS Y CIENCIAS SOCIALES – DIANOIA

UNIDADES TECNOLÓGICAS DE SANTANDER
FACULTAD DE CIENCIAS NATURALES E INGENIERÍAS
TECNOLOGÍA EN OPERACÍON Y MANTENIMIENTO ELECTROMECANICO
BARRANCABERMEJA
FECHA DE PRESENTACIÓN:01-06-2020

Nota de Aceptación

APROBADO

Firma del jurado



Firma del Jurado

DEDICATORIA

Primeramente quiero dar a Dios por darme la constancia y paciencia que necesité durante todo este trayecto transcurrido durante la culminación del programa académico, también debo dar gracias a mis padres, que, como siempre fueron y serán un apoyo incondicional para mi vida, animándome siempre y dándome los mejores consejos para formarme como persona, para que cuando llegue el día de laborar no solo sea un buen profesional si no también una buena persona al servicio de quienes me necesiten, también quiero dedicar esto a todas las personas que me quieren, que me apoyan, a todos mis familiares y amigos.

Brayan Ricardo Sanchez Barranco

AGRADECIMIENTOS

El autor quiere expresar agradecimientos a:

Las **Unidades Tecnológicas de Santander** por brindarme la oportunidad de formarme como profesional dándome los conocimientos para poder afrontar el mundo laborar y tener una mejor calidad de vida.

Un humilde agradecimiento al director de proyecto de grado **Ingeniero Fredy Alberto Rojas Espinoza** por tener la disposición y la paciencia para lograr este objetivo, guiándome en cada duda resultante a través de la construcción de este proyecto.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

RESUMEN EJECUTIVO

Arduino es una herramienta útil que nos sirve para programar y automatizar procesos haciéndolos más exactos y eficientes, lo cual, para el área de Ciencias e Ingenierías, Arduino es un elemento importante que compete conocerlo y desarrollarlo.

En consecuencia, con ello, se plantea el propósito de este trabajo de grado, que consiste en crear un Módulo de entrenamiento para el desarrollo de aplicaciones electrónicas orientadas a la industria basado en microcontrolador Arduino.

Su metodología consiste en hacer un paso a paso que parte de la programación, explicando de manera sustancial el por qué y para qué de la inclusión de ciertos comandos, el montaje de los diferentes elementos que se acoplarán al Arduino y Protoboard, sus conexiones, prácticas y demás; haciendo de este manual lo más comprensivo e interactivo posible, ya que será orientado para estudiantes con escaso conocimiento de la herramienta. Además, su contenido cuenta con el desarrollo de prácticas con el fin de optimizar el aprendizaje del mismo alcanzando así el desarrollo de competencias y habilidades en los estudiantes que les permitirán desenvolverse abiertamente en el manejo de esta herramienta.

En cohesión a lo anterior, se hará entrega del Módulo electrónico de entrenamiento dentro maletín para que este sea más compacto e interactivo y en él se podrá visualizar, realizar y comprender el desarrollo de las prácticas indicadas en este proyecto con siempre el fin de mejorar y agilizar el proceso de aprendizaje del estudiante.

PALABRAS CLAVE. Arduino, Microcontrolador, Industria, Aprendizaje

INTRODUCCIÓN

En este proyecto se presenta la explicación detallada de este y él porque es importante un módulo de entrenamiento para el desarrollo de aplicaciones electrónicas orientadas a la industria basado en microcontrolador Arduino el cual consiste en entregar un manual en el cual existe un cierto número de prácticas en las cuales se especifica detalladamente el por qué y para que, de las programaciones, también se presenta un módulo didáctico donde se podrá realizar dichas prácticas. Uno de los métodos de enseñanza más eficaces es el de aprender haciendo por lo cual muchas instituciones educativas, páginas web se han dado la tarea de crear manuales con sus respectivas guías de usuario como (Herrador, 2009), o prácticas como (Baeza, 2009) para que con la resolución de estos por parte de los estudiantes, los estudiantes formen bases más sólidaspara la realización de aplicaciones orientadas a la industria por lo cual es importante que las **Unidades Tecnológicas de Santander** cuente con su propio manual y Modulo didáctico para fortalecer las competencias en el área de Microcontroladores.

1. DESCRIPCIÓN DEL TRABAJO DE INVESTIGACIÓN

1.1. PLANTEAMIENTO DEL PROBLEMA

Debido a que hay diversidad de microcontroladores en el mercado, cada uno con sus respectivas especificaciones, es muy difícil abarcar la mayoría de estos en un plan de estudios.

Las Unidades Tecnológicas de Santander en el programa de Tecnología Operación y Mantenimiento Electromecánico, en su VI semestre de Plan de Estudios cuenta con una asignatura llamada **Microcontroladores,** basada mayormente en Arduino, sus componentes y pequeños dispositivos que se pueden acoplar a Arduino; sin embargo, la intensidad horaria destinada para la asignatura, no se distribuye para la exploración de otras gamas de microcontroladores, lo cual permitiría una vista más amplia al estudiante, potenciando significativamente sus competencias.

Los estilos de aprendizaje de los estudiantes varían, para lo cual, el docente debe implementar estrategias de enseñanza y aprendizaje, que le permitan al estudiante, desarrollar capacidades y habilidades en el área. Es por ello que, al no efectuar estrategias adecuadas, que amplíen la posibilidad de conocer otros tipos de microcontroladores, generaría dificultades a futuro en el campo profesional. En consecuencia, se genera dificultad para dar una vista más amplia sobre los diferentes dispositivos que se pueden emplear para automatizar procesos, hacerlos más eficientes, exactos y duraderos.

Si se implementan este tipo de manuales interactivos, se podrá lograr un conocimiento firme y solidez al manejo del microcontrolador Arduino, puesto que, este manual dará al estudiante un conocimiento acentuado que le permitirá investigar y poseer dominio del tema.

Los beneficios para la universidad podrían nombrarse en formar profesionales más capacitados, con un conocimiento más amplio que podrán aplicar en el campo laboral, favoreciendo la imagen de la Institución, lo que, a su vez, conllevaría a aumentar el número estudiantes matriculados en los diferentes programas que ofrece y más oportunidades de empleo para los egresados.

El manual de programaciones y montajes para el microcontrolador Arduino explicará en cada una de sus acciones el ¿Por qué?, ¿Para qué? De los diferentes montajes, programaciones y conexiones a través de paso a paso, siendo útil para estudiantes con poca información en estos, al proveer un conocimiento sustentado y de manera más ágil dando cabida a otros dispositivos que ayuden a dar y formar profesionales más capacitados, y crear beneficios para ellos y a la institución lo cual acarrea todos los efectos venideros ya mencionados.

¿Cómo se puede dinamizar la construcción de conocimiento por medio de la implementación de un módulo portátil de entrenamiento para el desarrollo de aplicaciones electrónicas orientadas a la industria basado en microcontrolador?

1.2. JUSTIFICACIÓN

Con la modernización de nuestra sociedad los procesos que eran mecánicos y operados por trabajadores que en ocasiones llegaban a ser inexactos ahora son automatizados, exactos, eficientes y más confiables, llegando así a que planes de estudio como las ingenierías electrónicas, eléctricas, mecatrónicas, hayan alcanzado papeles más importantes, ya que estos incurrieron en diseñar procesos y nuevas herramientas que permiten un desarrollo eficiente en las tareas que se tengan que cumplir. Estas ingenierías, se han tornado más importantes, es mayor el número de personas deseosas de acceder al conocimiento que estas imparten y uno de estos conocimientos importantes es la rama de la programación, dado que, a través de la modernización y los procesos ser cada vez más electrónicos y automáticos surge especialmente la necesidad de que los profesionales en estas áreas conozcan de programación y control.

Cabe notar que el aprendizaje de distintos lenguajes de programación como el C, C++, C#, Java, JavaScript, Perl, PHP, Phyton, Objetive-C y demás encierran conceptos que pueden llegar a ser tediosos y poco motivadores para los estudiantes, debido a que su contenido consiste en términos abstractos o que no se entienden por simple lógica, incluso, sin tener realmente claro su aplicación en su vida diaria o en situaciones que se presenten en su futuro empleo. La introducción de todo nuevo recurso didáctico ha de realizarse de formar organizada y sistematizada, la clase es un ambiente de alta incertidumbre y es imprescindible que el docente se asegure de que la introducción en el currículo de un determinado recurso se realiza de forma efectiva, previendo las posibles dificultades y oportunidades que este proceso pueda encerrar. (Correz, 2016)

La implementación de estas estrategias de aprendizaje didáctico, optimizará el tiempo y calidad de aprendizaje y dinamizará la compresión de temas de programación de Arduino para así dar cabida a demás temas, conocimientos, dispositivos de esta área que se basa en programaciones para automatizar procesos, volverlos exactos, eficientes y confiables.

A nivel mundial, los sistemas de control autónomos, en su infraestructura de hardware, los microcontroladores son el núcleo principal en el proceso y ejecución de tareas (Ayala & Yupa, 2013). Por eso es importante agilizar los procesos de aprendizaje para dar cabida a demás temas para que el estudiante desarrolle más competencias que le ayudarán en su futuro.

1.3. OBJETIVOS

1.3.1. Objetivo general

Implementar un módulo portátil de entrenamiento para el desarrollo de aplicaciones electrónicas orientadas a la industria basada en microcontrolador Arduino.

1.3.2. Objetivos específicos

- Diseñar un módulo didáctico de entrenamiento con microcontrolador y otros componentes para el desarrollo de aplicaciones industriales mediante una herramienta de software de diseño electrónico.

- Implementar el módulo de desarrollo con sus respectivos componentes acoplados Bluetooth, Wifi, Ethernet, USB, Sensor de temperatura, Humedad y Servomotora una plataforma portátil para entrenamiento de aplicaciones industriales basadas en microcontrolador Arduino.

- Realizar un manual de prácticas de laboratorio basado en el módulo de entrenamiento basado en Arduino con el fin de adquirir habilidades y destrezas en programación de aplicaciones electrónicas.

- Desarrollar y demostrar de modo practico-experimental, cada una de las prácticas propuestas en el manual con el fin de realizar las respectivas pruebas y posterior análisis de funcionamiento.

1.4. ESTADO DEL ARTE / ANTECEDENTES

1.4.1. Antecedentes nacionales

- ✓ **Diseño e implementación de guías enfocadas a un proyecto de robótica con Arduino.**

Consiste en la elaboración de guías desarrolladas para estudiar robótica escolar, dando uso a los diferentes materiales con los que la Escuela cuente. Las guías realizadas cuentas con animaciones, diseños y gráficos e información, estructurada por medio del programa Adobe Flash.

La autora destaca la importancia del uso de la herramienta Arduino en los diferentes espacios educativos, el material didáctico que se puede utilizar para adquirir una mayor comprensión de las actividades partiendo de los conceptos trabajados y el material general diseñado (Guías). Además, de los aportes y avances que se evidencian en relación al Pensamiento Lógico de los estudiantes. (Gil, 2015)

- ✓ **Implementando las metodologías Steam y aprendizaje basado en proyectos (abp) en la enseñanza de la física mediante Arduino.**

Se efectúa una idea a partir del Aprendizaje Basado en Proyectos (ABP), en la cual se crearon réplicas de prototipos creados con herramientas software y hardware libre Arduino, mediante sensores que establecen gráficas de estudio para una mejor comprensión de cada uno de las circunstancias o condiciones que se presentan durante el proceso de aprendizaje y enseñanza Steam, adquiriendo aprendizaje significativos y dominio en el área, aplicando los conocimientos en su práctica, permitiendo establecer posibles soluciones a situaciones futuras.(Sierra, Rojas, & García, 2019)

- ✓ **Aprendizaje basado en problemas (abp) para la enseñanza y el desarrollo de proyectos tecnológicos interdisciplinares basados en Arduino.**

Muestra la forma de abordar la enseñanza por medio del Aprendizaje Basado en Proyectos (ABP), con miras a proyectos tecnológicos, su importancia e incidencia

de la didáctica en el proceso formativo e integral de los estudiantes, en áreas de programación y robótica basadas en hardware y software open source, para un aprendizaje didáctico. Todo lo anterior, encaminado a un trabajo colaborativo, tomando planteamiento de problemas a partir de situaciones en un contexto social, para la unificación de aspectos y posturas para decisiones bajo consenso.

Este proyecto comprende tareas y actividades alrededor de competencias que prioricen el aprendizaje y permita a los estudiantes ser autodidactas, fortalecer capacidades y habilidades, que compaginen sus propias competencias en el uso de herramientas tecnológicas. (Rivera & Turizo, 2015)

1.4.2. Antecedentes internacionales

- ✓ **Estudio sobre la implementación de la herramienta Arduino en centro de formación profesional**

Se elabora con el fin de desarrollar competencias conceptuales y procedimentales en una muestra de estudiantes de Formación Profesional, todo esto, en relación a un estudio realizado sobre la implementación de la herramienta Arduino.

Para ello, se llevaron a cabo una serie de estudios no experimentales, tal como recolección de información que les permitió concluir la importancia de la plataforma en beneficio a la adquisición de conceptos y sus labores prácticas, siendo así, un eje y un recurso motivador en el desarrollo de habilidades en los escenarios en los que se desenvuelva el estudiante en relación al uso de la herramienta.(Correz, 2016)

- ✓ **Sistema de control demótico basado en Arduino, aplicación móvil y voz**

Se basa en la creación de una maqueta pequeña, manipulada a través de una aplicación móvil en Android conectada vía Bluetooth, procesado con Arduino. Este con el objetivo de controlar diferentes actividades de rutina en casa (luces, ventiladores, puertas, etc.) creado por medio de herramientas de software y hardware para su implementación en los espacios, todo esto realizado por comandos de voz. Así mismo, se emplea un manual al usuario que conlleve a la autosuficiencia, evitando que este se vea obligado a realizar ciertos esfuerzos o incomodidades para realizar las rutinas anteriormente nombradas. (Paucara, 2016)

- ✓ **Uso de Arduino en programación electrónica con metodología de aprendizaje basado en problemas**

Se dirige a la comunidad educativa de Secundaria para brindar una orientación electrónica, por medio del manejo, planteamiento y resolución de problemas cotidianos en los cuales ellos podrían verse inmersos, fomentando al mismo el trabajo colaborativo.

Se señala la importancia de la programación electrónica como asignatura básica en carreras universitarias relacionadas con las Ciencias de la Computación e Ingeniería Electrónica. Incorporando elementos y metodologías específicas para despertar la motivación en los estudiantes.

El autor incorporó la Metodología Basada en Problemas y la Placa de Arduino como material didáctico, teniendo en cuenta que los estudiantes tenían conocimientos conceptuales previos, lo cual facilitó su implementación. (Riveros, 2017)

- ✓ **Arduino, una herramienta accesible para el aprendizaje de programación**

Radica en la aplicación de la herramienta Arduino en el área de Robótica, en el que se propone enseñar la robótica de una manera atractiva para motivar a los estudiantes. Se tomó como punto de partida para su ejecución la realización de un taller con el fin de informar, capacitar y orientar a los estudiantes en el área; además, de actividades prácticas como control motriz y evasión de obstáculos mediante Bluetooth.

Se tuvo en cuenta los diferentes distractores tecnológicos que intervienen en el proceso de adquisición del conocimiento en los estudiantes (televisión, video juegos, redes sociales, etc.), para lo cual se propuso la plataforma Arduino, como herramienta perceptible que les permitiera comprender y acercarse a la lógica de la programación. (Vargas, Castillo & Brambila, 2015)

- ✓ **Uso de Arduino en programación electrónica con metodología de aprendizaje basado en problemas**

Se diseña con base a las enseñanzas de técnicas de aprendizaje, en estudiantes de Secundaria, en lo que respecta a conocimientos de programación electrónica, incorporando el Aprendizaje Basado en Problemas. Lo cual permitiría un

aprendizaje colaborativo y significativos por medio de actividades prácticas enmarcadas a situaciones de su cotidianidad y dar respuesta a las mismas.

Señala, además, la importancia de efectuar las TICS en escenarios educativos y el impacto que este genera en las actividades académicas en las Instituciones. (Riveros, 2017)

- ✓ **Una mirada al mundo Arduino**

Es un artículo descriptivo que facilita información global o de aspectos generales respecto a Arduino, específicamente en el Modelo UNO R3, señalando aspectos tales como placas lenguaje, gráficos para su respectiva programación, sensores y controladores.
También expone diferentes plataformas educativas y espacios interactivos en los cuales se emplean microcontroladores de Arduino para optimizar el conocimiento y dominio de la herramienta.(Herrero & Sanchez, 2015)

- ✓ **Ambientes de aprendizaje para la ciencia usando tecnología Arduino**

Permite a los participantes aplicar y complementar sus conocimientos en Electrónica, Programación y Matemáticas, por medio del uso de ambientes de programación de Arduino.

Se brindan talleres y capacitaciones con el objetivo de ampliar en estudiantes de Educación Superior sus conocimientos previos, para posteriormente definir proyectos de desarrollo en áreas como: domótica, robótica, seguridad y agricultura; seguido se realizó un riguroso seguimiento a cada uno y finalmente,

compartir los resultados obtenidos con la comunidad educativa. Aguilar & Márquez (s.f.)

- ✓ **Revista digital Arduino Bolivia**

Se crea esta revista con el fin de brindar bases conceptuales a manera de introducción, como punto de partida para el uso e implementación del microcontrolador Arduino, su importancia, características y usos que se pueden dar en la Industria y cotidianidad. Justifica a su vez, su importancia y beneficios, explicando de forma detallada, por medio de ilustraciones y seguimiento en relación a la programación de esta herramienta. (Condori, Ordoñez & Rodas, 2018)

- ✓ **Aprendiendo Arduino**

Se compone por un curso para el manejo de Arduino en una plataforma, compuesto de actividades teórico-prácticas para el óptimo afianzamiento de los conceptos facilitados en ella.

Esta plataforma cuenta con 7 niveles de aprendizaje, partiendo de conceptos generales de Arduino, manejo, programación y su respectiva elaboración en los espacios que se requiera.

Su fácil acceso y uso, permite lectura de manuales y prácticas con el material requerido para un mejor dominio en cuestiones de aplicación, abriendo paso a una socialización tecnológica. Quienes acceden a esta plataforma, adquieren conocimientos de varios tipos de microcontroladores, sensores y programación de los mismos. (Crespo, 2016)

- ✓ **Evaluación de la plataforma Arduino e implementación de un sistema de control de posición horizontal**

Expone a nivel de Hardware y Software la herramienta Arduino, dando a conocer su manejo y utilidad de los sensores usando la Tecnología Efecto Hall. Además, de implementar una programación para la adquisición de datos generalmente usada por estudiantes de Ingeniería Eléctrica y Electrónica, como lo es Labview.
Su objetivo principal es diseñar e implementar la plataforma Arduino en un sistema de control y posición horizontal. Realizar pruebas y desarrollar tareas como verificación y análisis del uso de la herramienta. (Tapia & Manzano, 2013)

- ✓ **Seguimiento de objetos empleando Afroge. Net y Arduino**

Se emplea para el desarrollo de este proyecto, una cámara de video con el fin de detectar señales de movimiento y velocidad en diferentes elementos, percibiendo su posición en un espacio. Se creó un lenguaje de programación y el dispositivo Arduino para el envío de señales que se empleen. (Esqueda, López, Espino, & Salazar, 2014)

- ✓ **Buenas prácticas para evitar vulnerabilidad en el diseño con plataformas de hardware abierto tipo Arduino**

Realiza un análisis de vulnerabilidad de dispositivos electrónicos de hardware abierto con placas de Arduino. De los cuales, los dispositivos considerados vulnerables, fueron los relacionados con manipulación de hardware, Firmware y los sistemas de comunicación Arduino. Finalmente, con base a los resultados

obtenidos de dicho análisis, plantean posibles soluciones o recomendaciones prácticas para el diseño y manipulación de estos dispositivos.(Bonilla, Gutierrez, Orueta, & Gil, 2017)

2. MARCOS REFERENCIALES

2.1.1. Marco teórico

Arduino es una plataforma de hardware de código abierto, basada en una placa de circuito impreso que contiene un microcontrolador de marca ATMEL que cuenta con entradas y salidas, analógicas y digitales, en un entorno de desarrollo que está basado en el lenguaje de Programación Procesing. El dispositivo conecta el mundo físico con el mundo virtual, o el mundo analógico con el digital controlando, sensores, alarmas, sistemas de luces, motores, y actuadores. (Tapia & Manzano, 2013a)

Hay muchas otros microcontroladores y plataformas disponibles para la computación física donde las funcionalidades y herramientas son muy complicadas de programar, Arduino simplifica el proceso de trabajar con microcontroladores, ofrece algunas ventajas y características respecto a otros sistemas. (Tapia & Manzano, 2013b)

- ✓ **Ventajas del uso de Arduino**

Según (Enríquez, 2009), las siguientes son las ventajas de usar una placa Arduino:

Barato: Las placas Arduino son relativamente baratas comparadas con otras plataformas microcontroladoras. La versión menos cara del módulo Arduino puede

ser ensamblada a mano, e incluso los módulos de Arduino pre ensamblados cuestan menos de 50$.

Multiplataforma: El software de Arduino se ejecuta en sistemas operativos Windows, Macintosh OSX y GNU/Linux. La mayoría de los sistemas microcontroladores están limitados a Windows.

Entorno de programación simple y claro: El entorno de programación de Arduino es fácil de usar para principiantes, pero suficientemente flexible para que usuarios avanzados puedan aprovecharlo también. Para profesores, está convenientemente basado en el entorno de programación Processing, de manera que estudiantes aprendiendo a programar en ese entorno estarán familiarizados con el aspecto y la imagen de Arduino.

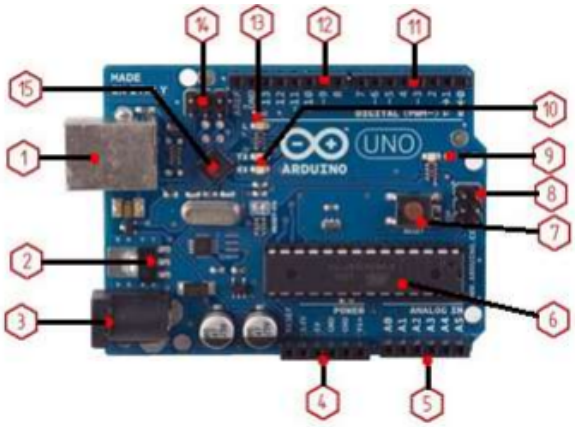
Código abierto y software extensible: El software Arduino está publicado como herramientas de código abierto, disponible para extensión por programadores experimentados. El lenguaje puede ser expandido mediante librerías C++, y la gente que quiera entender los detalles técnicos pueden hacer el salto desde Arduino a la programación en lenguaje AVR C en el cual está basado. De forma similar, puedes añadir código AVR-C directamente en tus programas Arduino si quieres.

Código abierto y hardware extensible: El Arduino está basado en microcontroladores ATMEGA8 y ATMEGA168 de Atmel. Los planos para los módulos están publicados bajo licencia CreativeCommons, por lo que diseñadores experimentados de circuitos pueden hacer su propia versión del módulo, extendiéndolo y mejorándolo. Incluso usuarios relativamente inexpertos pueden

construir la versión de la placa del módulo para entender cómo funciona y ahorrar dinero.

Descripción de la Placa Arduino.

Figura1: Placa Arduino



En la figura 1 se describen los componentes de la placa Arduino 1.

Fuente: (Tapia & Manzano, 2013)

La siguiente descripción de los componentes de la Placa Arduino es presentada según (Tapia & Manzano, 2013):

- Conector USB: proporciona la comunicación para la programación y la toma de datos, también provee una fuente de 5VDC para alimentar al Arduino, pero de baja corriente por lo que no sirve para alimentar motores de gran potencia.

- Regulador de voltaje de 5V: se encarga de convertir el voltaje ingresado por el plug 3, en un voltaje de 5V regulado necesario para el funcionamiento de la placa y para alimentar circuitos externos.

- Plug de conexión para fuente de alimentación externa: Es el voltaje que se suministra que debe ser directo y estar entre 6V y 18V o hasta 20V, generalmente se debe tener cuidado de que el terminal del centro del plug quede conectado a positivo ya que algunos adaptadores traen la opción de intercambiar la polaridad de los cables.

- Puerto de conexiones: Es constituido por 6 pines de conexión con las funciones de RESET que permite resetear el microcontrolador al enviarle un cero lógico. Pin 3.3V provee una fuente de 3.3VDC para conectar dispositivos externos como en la protoboard, por ejemplo. Pin 5V es una fuente de 5VDC para conectar dispositivos externos. Dos pines GND que permite la salida de cero voltios para dispositivos externos. Pin Vin, este pin está conectado con el dispositivo del plug 3 por lo que se usa para conectar la alimentación de la placa con una fuente externa de entre 6 y 12VDC en lugar del plug 3 o la alimentación por el puerto USB.

- Puertos de entradas análogas: lugar donde se conectan las salidas de los sensores análogos. Estos pines solo funcionan como entradas recibiendo voltajes entre cero y cinco voltios directos.

- Microcontrolador ATMega 328: Implementado con los Arduino uno en la versión SMD del Arduino UNO R2 se usa el mismo microcontrolador, pero

en montaje superficial, en este caso las únicas ventajas son la reducción del peso y ganar un poco de espacio.

- Botón Reset: permite resetear el microcontrolador haciendo que reinicie el programa.

- Pines de programación ICSP: Son usados para programar microcontroladores en protoboard o sobre circuitos impresos sin tener que retirarlos de su sitio.

- Led ON: Enciende cuando el Arduino está encendido.

- Leds de Recepción y Transmisión: Se encienden cuando la tarjeta se comunica con el PC. El Tx indica transmisión de datos y el Rx recepción.

- Puertos de conexiones de pines de entradas o salidas digitales: La configuración como entrada o salida debe ser incluida en el programa. Cuando se usa el terminal serial es conveniente no utilizar los pines como cero (Rx) y uno (Tx). Los pines 3, 5 y 6 están precedidos por el símbolo ~, lo que indica que permiten su uso como salidas controladas por ancho de pulso PWM.

- Puerto de conexiones 5 entradas o salidas adicionales: Las salidas 9, 10 y 11 permiten control por ancho de pulso; la salida 13 es un poco diferente pues tiene conectada una resistencia en serie lo que permite conectar un led directamente entre ella y tierra. Finalmente hay una salida a tierra GND

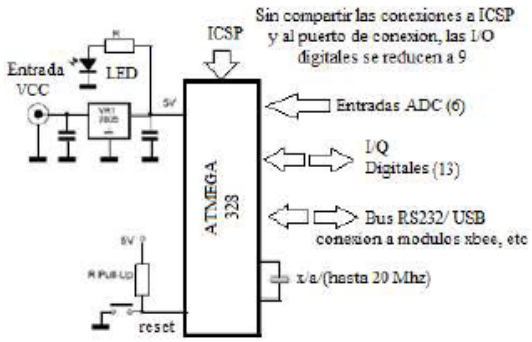
y un pin AREF que permite ser empleado como referencia para las entradas análogas.

- Led pin 13: Indica el estado en que se encuentra.

- Pines de programación ISCP: Son usados para programar microcontroladores en protoboard o sobre circuitos impresos sin tener que retirarlos de su sitio.

- Chip de comunicación: Permite la conversión de serial a USB.

Figura2: Diagrama de bloques sencillo para una placa Arduino



Representación de las entradas, salidas , USB, etc.

Fuente: (Tapia & Manzano, 2013)

2.1.2. Marco conceptual

✓ **Microcontrolador**

Un microcontrolador es un circuito integrado digital que es utilizado en diversas aplicaciones ya que este es programable y está compuesto por una unidad central de proceso (CPU), memorias (ROM y RAM) y líneas de entrada y salida (periféricos).

- ✓ **Efecto hall**

Es la aparición de un campo eléctrico por separación de cargas en el interior de un conductor por los cuales circula una corriente en presencia de un campo magnético con componentes perpendiculares al movimiento de las cargas.

- ✓ **LabVIEW**

Es una plataforma y entorno de desarrollo para diseñar sistemas, con un lenguaje de programación visual gráfico pensado para sistemas hardware y software de pruebas, control y diseño, simulado o real y embebido.

2.1.3. Marco legal

Se realizó una búsqueda y se llegó a concluir que no hay restricciones legales sobre el microcontrolador Arduino y su uso, por lo tanto, se orientara en camino a los fundamentos adquiridos en la carrera. Se pudo encontrar información legal respecto a importaciones de la herramienta, fomento de proyectos de investigación basados en tecnología y demás.

Decreto 591 del 26 de febrero de 1991 por el cual se regulan las modalidades específicas de contratos de fomento de actividades científicas y tecnológicas.

Ley 1286 de 2009, por la cual se modifica la Ley 29 de 1990, se transforma a Colciencias en Departamento Administrativo, se fortalece el Sistema Nacional de Ciencia, Tecnológica- e Innovación en Colombia y se dictan otras disposiciones.

Decreto 393 del 08 de febrero de 1991 por el cual se dictan normas sobre asociación para actividades científicas y tecnológicas, proyectos de investigación y creación de tecnologías.

Resolución No. 62 del 24 de febrero de 2014 por la cual se reglamenta y se establece la forma, contenido y términos para el Registro de los Contratos de Importación de Tecnología ante la Dirección de Impuestos y Aduanas Nacionales.

2.1.4. Marco histórico

Se estudiará la documentación histórica de los sistemas embebidos y del microprocesador Arduino para así obtener fundamentos para la construcción de este proyecto.

- ✓ **Microprocesador Arduino**

De forma estricta, el proyecto «Arduino» se inició en el año 2005 como un proyecto enfocado a estudiantes en el Instituto IVREA (IDII), en Ivrea (Italia). En aquellos años, los estudiantes usaban el microcontrolador BASIC Stamp, cuyo

costo era de $100USD, un costo considerablemente alto para un estudiante promedio. Antes del año 2005, específicamente durante el año 2003, Hernando Barragán había creado la plataforma de desarrollo Wiring como resultado de su proyecto de tesis en la maestría en el IDII, bajo la supervisión de Massimo Banzi y Casey Reas, quienes eran conocidos por haber trabajado en el lenguaje Processing y daban clases en el IDII El objetivo del proyecto era crear herramientas simples y de bajo costo para la creación de proyectos digitales por parte de personas sin altos conocimientos técnicos o sin un perfil de ingeniería. El proyecto Wiring era una placa de desarrollo de hardware que constaba de una placa de circuito impreso (PCB) con un microcontrolador ATmega168, un Ambiente de Desarrollo Integrado (IDE) basado en funciones de procesamiento y una biblioteca de funciones para programar fácilmente el microcontrolador. Regresando al año 2005, Massimo Banzi junto con David Mellis (otro estudiante del IDII) y David Cuartielles, agregaron soporte a Wiring para el microcontrolador ATmega8, más económico que el inicial (Atmega168). Pero en lugar de continuar el desarrollo en Wiring, se separaron del proyecto y lo renombraron Arduino.

El nombre *Arduino* viene de un bar en Ivrea, Italia; en donde algunos de los fundadores del proyecto Arduino solían reunirse. El bar tiene el nombre de " Bar di Re Arduino", y fue nombrado en honor a Arduino de Ivrea, quien fue el margrave de la Marcha de Ivrea y Rey de Italia desde el año 1002 hasta el año 1014.

El equipo inicial de Arduino estaba conformado por Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino y David Mellis. Hernando Barragán no fue invitado a participar.

Posteriormente el proyecto Wiring siguió adelante con placas menos densas y costosas que se distribuyeron a través de la comunidad de código abierto.

Adafruit Industries, un proveedor de componentes electrónicos y fabricante de placas de circuito impreso, entre ellas placas Arduino, ubicado en la ciudad de Nueva York, estimó a mediados del año 2011 que se habían producido comercialmente más de 300,000 placas Arduino oficiales. En el año 2013, estimó que se encontraban en manos de usuarios 700,000 placas Arduino oficiales.

En octubre del año 2016, Federico Musto (actualmente ex CEO de Arduino), adquirió el 50% de la compañía tras haber adquirido las acciones de uno de los miembros fundadores del equipo. En abril del año 2017, la revista Wired informó que Musto había "fabricado su propio expediente académico", habiéndolo publicado en el sitio web de Arduino, cuenta personal de LinkedIn, e incluso en documentos comerciales oficiales italianos. Musto afirmaba tener un PhD en ciencias de la computación por el Instituto Tecnológico de Massachusetts (MIT), y un MBA de la Universidad de Nueva York. La revista Wired reportó que ninguna de las universidades donde él afirmaba haber estudiado tenía registro alguno de la asistencia de Musto. Musto afirmó más tarde, en una entrevista a Wired, que realmente nunca había obtenido los grados académicos.

En el año 2017, Massimo Banzi anunció la creación de la «Fundación Arduino», declarando que sería «un nuevo comienzo para Arduino». Dicha fundación, según palabras del mismo Banzi, «permitirá defender los valores fundamentales de la Comunidad Arduino dentro del ecosistema de código abierto y hacer que nuestro compromiso (haciendo referencia a la empresa Arduino) con el código abierto sea más sólido que nunca". Sin embargo, ha existido cierta incertidumbre del desarrollo actual de dicha iniciativa.

La controversia en torno a Federico Musto continuó en julio del año 2017, según los informes, por haber retirado licencias de código abierto, esquemas y códigos del sitio web de Arduino, lo que provocó escrutinio y protesta por parte de la comunidad de makers.

En octubre del año 2017, Arduino anunció su asociación con la multinacional ARM Holdings (ARM). El anuncio decía, en parte, que "ARM reconoce la independencia como un valor central de Arduino... sin ningún acuerdo de uso exclusivo con la arquitectura ARM". Arduino tiene la intención de seguir trabajando con todos los proveedores y arquitecturas de tecnología.

Para la producción en serie de la primera versión se tomó en cuenta que el coste no fuera mayor de 30 euros, que fuera ensamblado en una placa de color azul, debía ser Plug and Play y que trabajara con todas las plataformas informáticas tales como MacOSX, Windows y GNU/Linux. Las primeras 300 unidades se las dieron a los alumnos del Instituto IVREA, con el fin de que las probaran y empezaran a diseñar sus primeros prototipos. (Wikipedia, 2019)

- ✓ **Disputa por la marca Arduino**

A principios de 2008, los cinco cofundadores del proyecto Arduino crearon la empresa Arduino LLC, cuyo propósito era englobar las marcas comerciales asociadas a las placas Arduino[15]. La fabricación y venta de las placas Arduino debía ser hecha por compañías externas, y Arduino LLC obtendría un royalty (comisión), de ellos. Los estatutos bajo los cuales se creó Arduino LLC especificaban que cada uno de los cinco fundadores originales transferirían la propiedad de la marca Arduino a la empresa recién formada (Arduino LLC).

A finales de 2008, la empresa de Gianluca Martino (Smart Projects), registró la marca Arduino en Italia y mantuvo esto en secreto de los otros cofundadores durante un periodo aproximado de dos años. Esto fue descubierto cuando la compañía Arduino LLC intentó registrar la marca en otras partes del mundo (originalmente ellos se habían registrado solo en EE. UU.), encontrando que está ya estaba registrada en Italia. Las negociaciones con Gianluca y su firma para poner la marca bajo control de la compañía Arduino LLC fallaron. En el año 2014, Smart Projects comenzó a negarse a pagar regalías. Luego nombraron a un nuevo CEO, Federico Musto, que renombró a la empresa Arduino SRL y creó el sitio web arduino.org, copiando los gráficos y el diseño del arduino.cc original. Esto resultó en una fractura en el equipo de desarrollo de Arduino.

En enero de 2015, Arduino LLC entabló una demanda contra Arduino SRL

En mayo de 2015, Arduino LLC creó la marca mundial Genuino, utilizada como marca fuera de los Estados Unidos.

En julio de 2017, la nueva compañía BCMI LABS LLC fundada por Massimo Banzi, David Cuartielles, David Mellis y Tom Igoe, adquirió Arduino AG y todas las marcas registradas de Arduino. Fabio Violante se convirtió en el nuevo CEO que reemplazaría a Federico Musto, quien ya no trabajaría para Arduino AG.

Durante la "WorldMaker Faire" en Nueva York del 1 de octubre de 2016, el cofundador y CEO de Arduino LLC (Massimo Banzi) y el CEO de Arduino SRL (Federico Musto), anunciaron la fusión de ambas compañías. (Wikipedia, 2019)

3. DESARROLLO DEL TRABAJO DE GRADO

3.1. DISEÑO DEL MÓDULO DE ENTRENAMIENTO BASADO EN MICROCONTROLADOR ARDUINO

Se diseña el módulo de entrenamiento con microcontrolador Arduino en la herramienta SolidWorks 2018, donde diseñamos los componentes que irán acoplados a él y el recipiente donde estarán estos.

El uso en esta aplicación fue realmente sencillo, ya que para la construcción del móduloy sus artefactos se basó en tareas repetitivasque a continuación se mostrarán para evidenciar el desarrollo del diseño dando una idea de cómo este se diseñó. Se utilizaron las siguientes herramientas ybasadas en ellas se logró el objetivo final que es el diseño, no se utilizaron más que estas herramientas mencionadas a continuación que de manera simple y breve ayudaron a que fuera fácil la construcción del módulo en esta herramienta.

En SolidWorks se utilizó varias herramientas para la construcción de las diferentes piezas que conforman el módulo entre las cuales están: insertar pieza, croquis, ensamblaje,extrusión, extruir corte, redondeo, redondeo de croquis, cota inteligente, líneas, spline, arco centro extremos, cuadrados, círculos, editar apariencias, la función insertar piezas y relaciones de posición. Con estas funciones disponibles en SolidWorks se diseña y se da una apariencia 3D.

La forma más sencilla de construir piezas es empezar en un plano 2D, dibujar lo que se quiere y extruirlo, en base a eso se hace lo que se quiere, agregándole redondeos, haciéndole cortes y demás.

Todas las medidas del módulo ilustradas en este documento fueron tomadas de cada una de las partes que conforman el módulo posterior a su construcción.

Figura3: Modulo didáctico de entrenamiento basado en microcontrolador Arduino.

Representación isométrica del Módulo didáctico de entrenamiento.

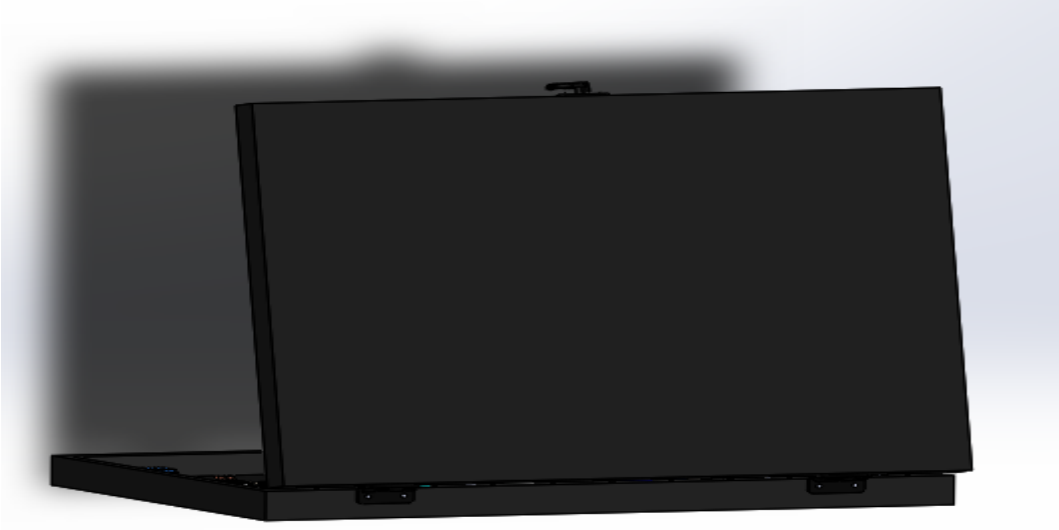
Fuente: (Autor, 2020)

Figura4: Modulo didáctico de entrenamiento basado en microcontrolador Arduino.

Vista superior del Módulo didáctico de entrenamiento.

Fuente: (Autor, 2020)

Figura5: Modulo didáctico de entrenamiento basado en microcontrolador Arduino.



Vista posterior del Módulo didáctico de entrenamiento.

Fuente: (Autor, 2020)

Figura6: Modulo didáctico de entrenamiento basado en microcontrolador Arduino.

Vista isométrica del Módulo didáctico de entrenamiento.

Fuente: (Autor, 2020)

Figura 7: Modulo didáctico.

Vista real del módulo didáctico.

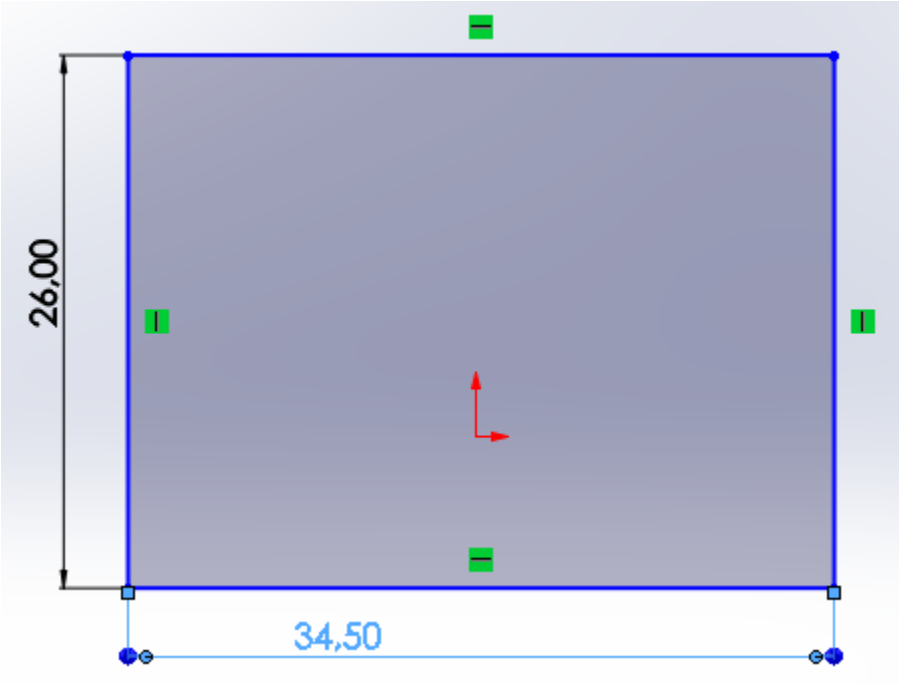
Fuente: (Autor, 2020)

3.1.1. Parte inferior del módulo.

Se empezó por diseñar la parte inferior del módulo, ya que como se puede observar en la figura 3, este módulo se abre de esa forma, por lo tanto, se divide en dos.

Se empezó por diseñar en croquis, un cuadrado de las mismas medidas del módulo a construir las cuales son:

Figura8: Medidas de la parte inferior del módulo.



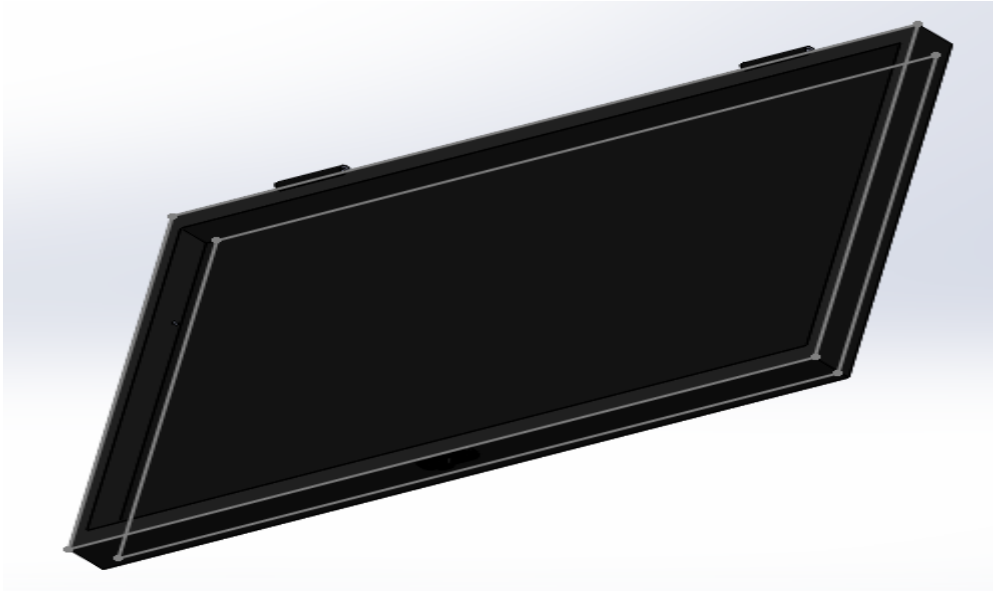
De acuerdo a la figura 7 se representan las medidas de la base del Módulo.

Fuente: (Autor, 2020)

Después de esto lo que se hace es extruir para que este cuadrado se forme un cubo, se extruye a 2 cm ya que como la altura total del módulo son 4 centímetros su mitad seria dos.

Ya teniendo el cubo se desea hacer un vaciado dentro de ese cubo ya creado, entonces procedemos a dibujar un cuadrado en una de las caras de ese cubo y con la función cota inteligente definimos el espesor de las paredes del módulo, acotando este segundo cuadrado a una distancia de0.6 cm que es el espesor del módulo. Después de esto se le aplicó un extruir corte de 1.5 cm creando un efecto de vaciado en esta parte inferior del módulo.

Figura9: Parte inferior ya extruida.

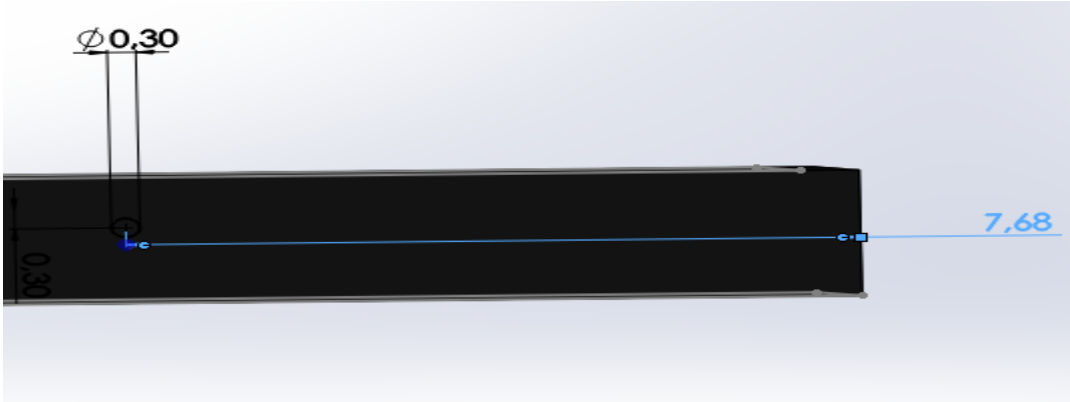


De acuerdo a la figura 8 se representa la parte inferior del modulo ya extruida y con un extruir corte para dar el aspecto de vaciado.

Fuente: (Autor, 2020)

A esta parte del módulose le hizo un pequeño orificio por el cual se encajará el eslabón que se puede observar en la ilustración 3, se hará utilizando las funciones de croquis, circulo y extruir corte con las siguientes medidas aproximadas:

Figura10: Medidas de orificio para el eslabón.

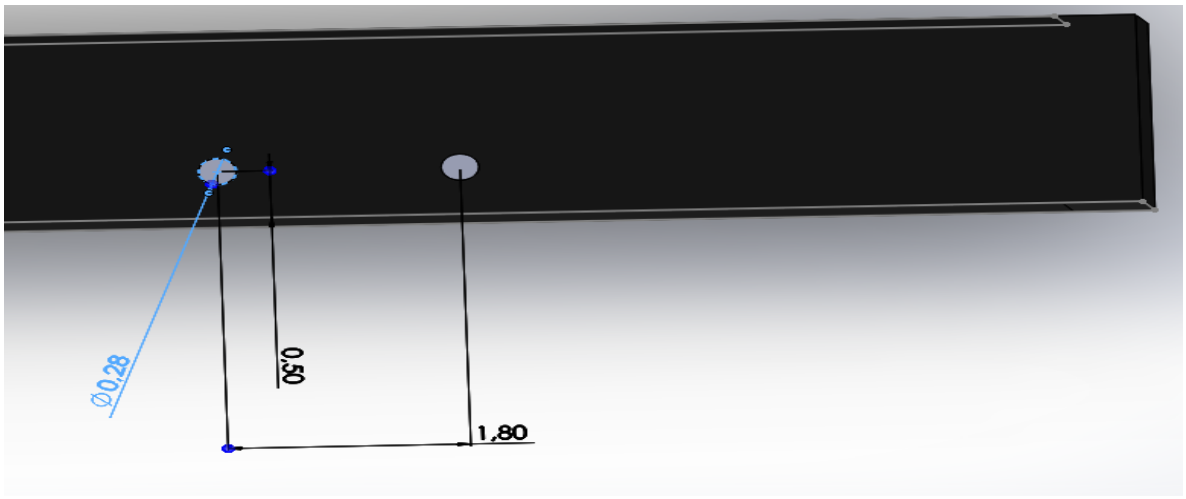


Medidas del orificio donde se acoplarán los eslabones para sostener la parte superior del módulo.

Fuente: (Autor, 2020)

Lo siguiente será hacer unos pequeños cilindros que representarán los tornillos de las bisagras, utilizando de nuevo la función, circulo, extruir y cota inteligente utilizando las siguientes medidas:

Figura11: Medidas de tornillos de bisagras.



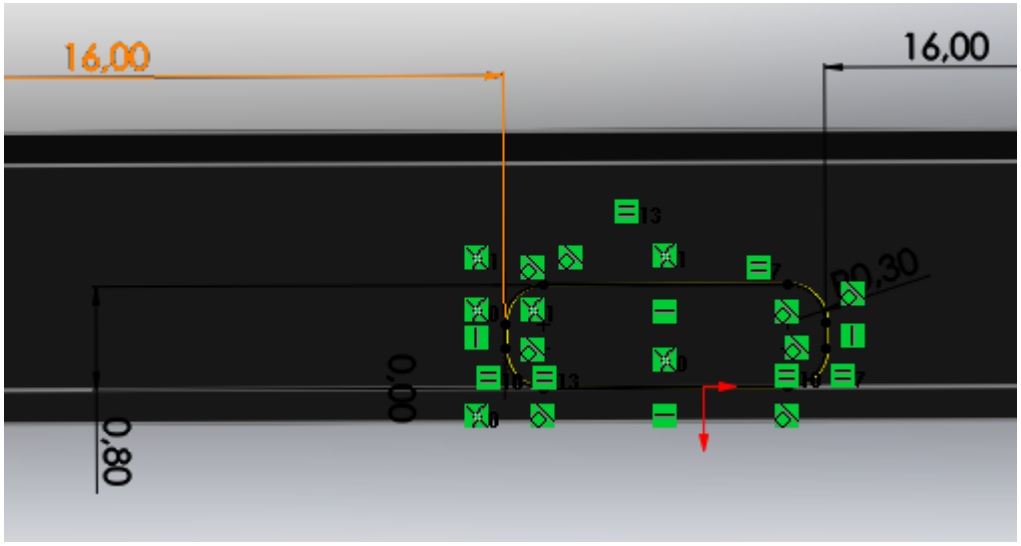
Medidas de los círculos que representaran en el modelado los tornillos de las bisagras.

Fuente: (Autor, 2020)

Por último, queda hacer el pasador inferior, el cual servirá para asegurar las dos tapas del módulo, utilizamos las mismas funciones de extruir, extruir corte, redondeo y demás con las siguientes medidas.

Estas serían las medidas de la base del pasador, dentro de ese cuadrado dibujamos otro cuadrado, lo extruimos y en una de sus caras le aplicamos un extruir corte para hacer un orificio en el cual irá asegurado el pasador de la tapa superior.

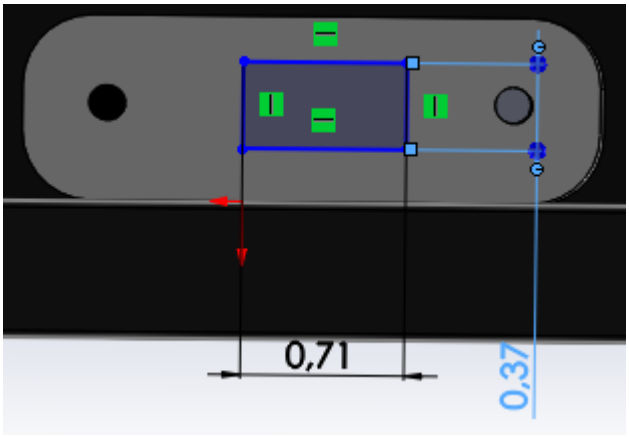
Figura12: Medidas de la base del pasador.



Medidas y ubicación de la base del pasador con sus respectivos redondeos.

Fuente: (Autor, 2020)

Figura13: Medidas del pasador.

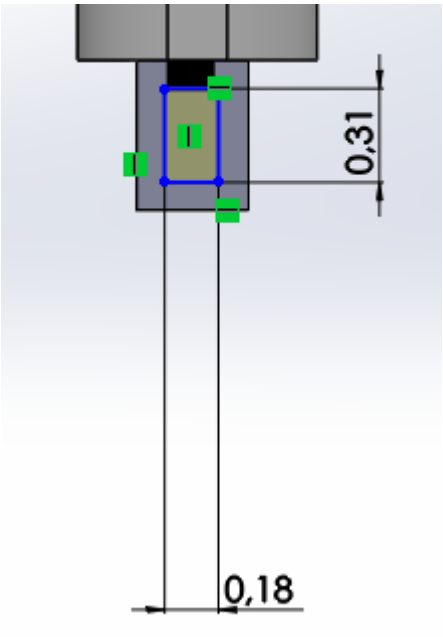


Base del orificio donde encajará el pasador del Módulo didáctico de entrenamiento.

Fuente: (Autor, 2020)

Después de extruir el cuadrado anterior a 0.5 cm y procedimos a diseñar el orificio donde entrará el pasador, en una de las caras laterales del cuadrado extruido anteriormente vamos a dibujar otro cuadrado y después hacemos un extruir corte para crear el orificio y se deberá ver de la siguiente forma.

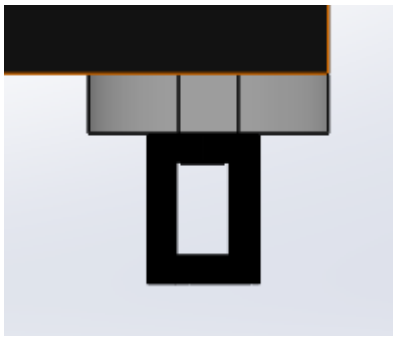
Figura14: Medidas del orifico para el pasador.



Medidas del orificio de la base donde se encajará el pasador para asegurar el Módulo didáctico de entrenamiento.

Fuente: (Autor, 2020)

Figura15: Vista del orificio del pasador.



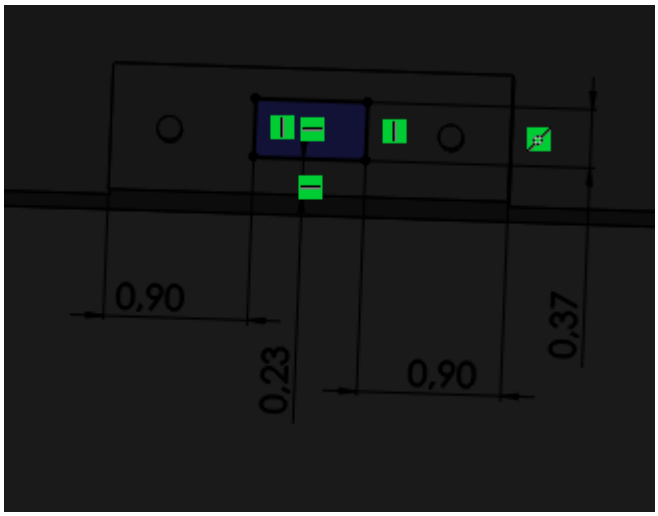
Vista terminada del orificio y su basa que servirán para acoplar el pasador.

Fuente: (Autor, 2020)

3.1.2. Parte superior del módulo.

Se hicieron los mismos pasos de la ilustración 4 a la ilustración 8, después se procedió a dibujar un cuadrado el cual se extruyó a 0.5 cm.

Figura16: Medidas del cuadrado.



Medidas del pasador superior del Módulo didáctico de entrenamiento.

Fuente: (Autor, 2020)

Lo siguiente que se hizo fue dibujar un círculoel cual se extruyó a 0.2 cm que será donde se acoplará el pasador y final mente se le aplicó redondeos y a los dos bases anteriores del circulo.

Figura17: Vista final de base de pasador superior.

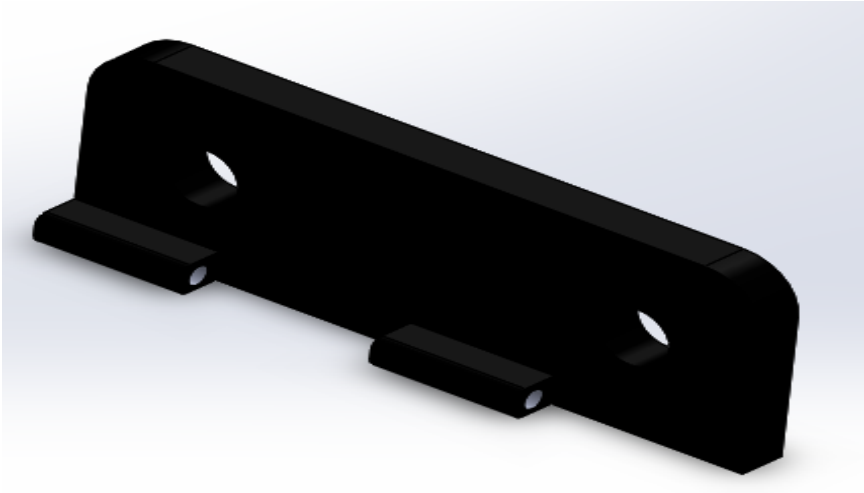
Base del pasador superior terminada y lista para ser ensamblada con el pasador.

Fuente: (Autor, 2020)

3.1.3. Bisagras.

En un croquis diferente se diseñan las bisagras para un posterior ensamble a la caja superior e inferior.

Figura18: Bisagra 1



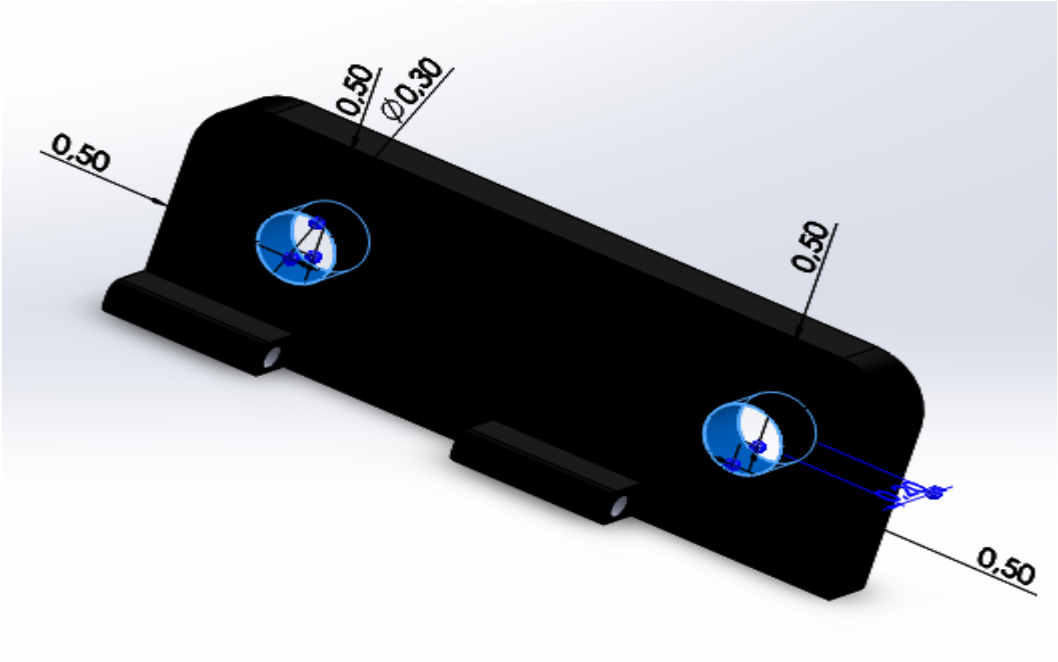
Vista final del diseño de la bisagra 1.

Fuente: (Autor, 2020)

Para lograr este diseño lo que se hizo fue dibujar un cuadrado el cual tenía unas dimensiones de 1x2.8 cm y se le aplicó un redondeo de croquis de 0.25 cm para darle ese aspecto redondeado, como se observa en la ilustración 14 y se extruye a 0.2cm.

Lo siguiente fue dibujar dos círculos de 0.3 cm de diámetro, se ubicó estos círculos a 0.5 cm de sus costados más cercanos y de la parte superior, finalmente aplicando un extruir corte para posterior mente acoplarlos en los cilindros que representan tornillos en las cajas superior e inferior.

Figura19: Medidas de círculos.

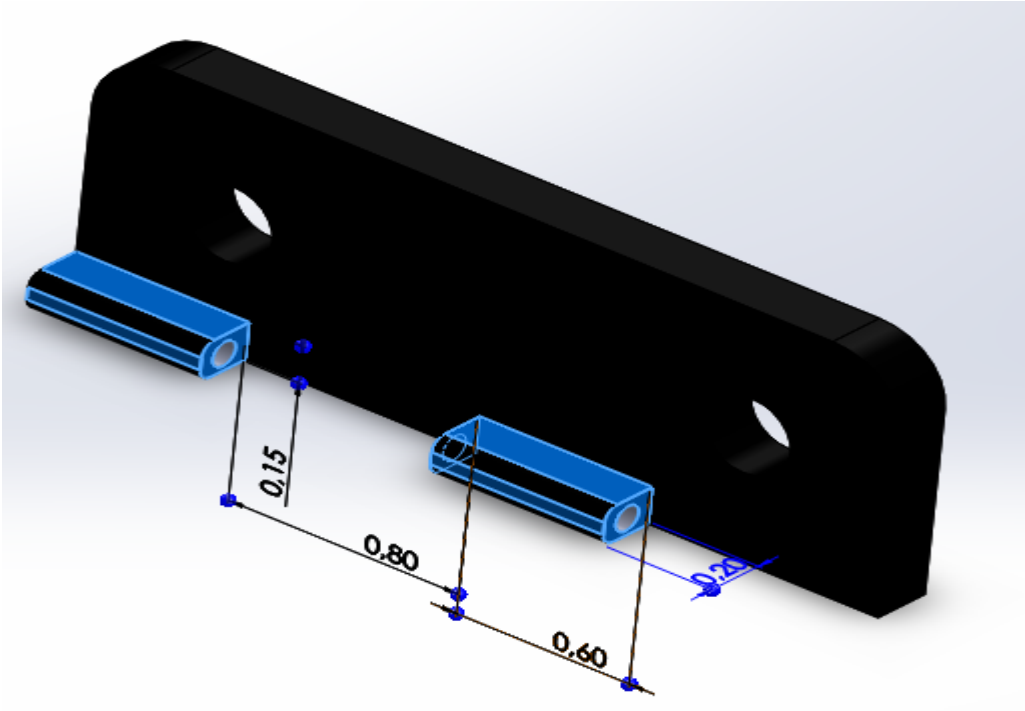


Medidas y ubicación de los círculos de la bisagra 1

Fuente: (Autor, 2020)

Se dibujó dos cuadrados al costado inferior de la bisagra para posteriormente extruirse a 0.2cm y se ubicó con las siguientes medidas.

Figura20: Medidas inferiores de la bisagra.



Medidas y ubicación de los túneles que servirán para acoplar la bisagra 2.

Fuente: (Autor, 2020)

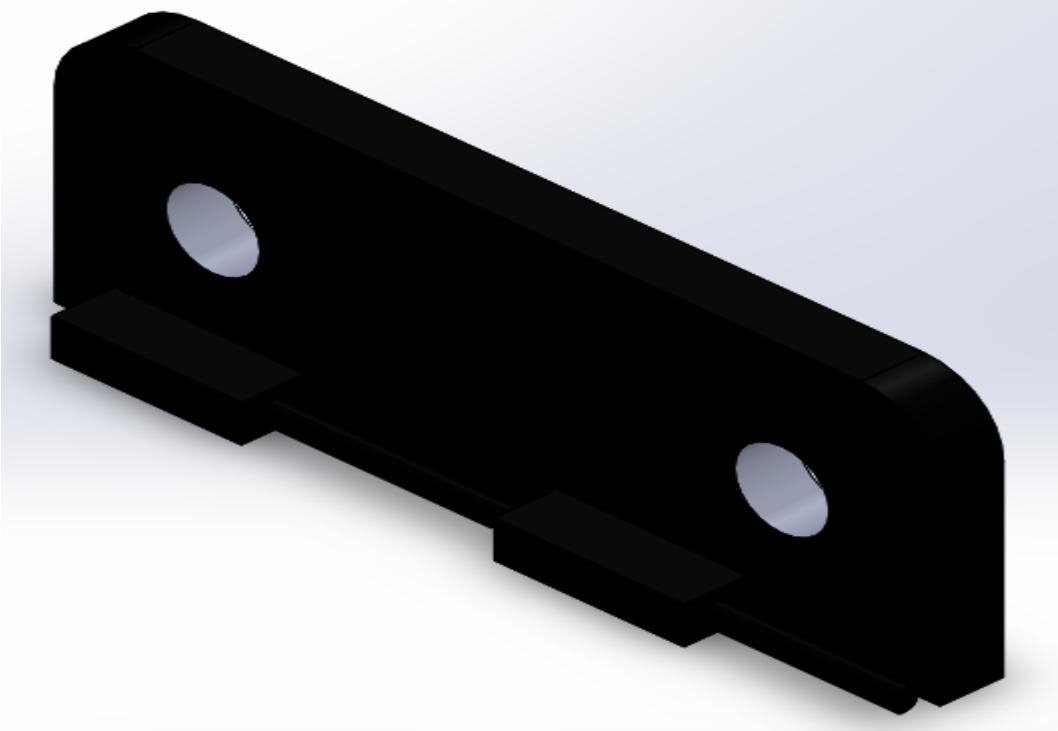
Finalmente se procede a dibujar un círculo en una de las carasde los cuadrados anteriormente extruidos, después aplicamos un extruir corte a una distancia que cree un orificio en los dos cuadrados, con esto se termina el diseño de la bisagra 1.

Cabe recalcar nuevamente que estas medidas se toman de las partes que conforman el módulo posterior a su construcción.

3.1.4. Bisagra 2.

Para la bisagra 2 se hacen los mismos pasos de la ilustración anteriormente mencionados y vistos en las ilustraciones 14, 15 y 16 pero a diferencia de 0.3 de diámetroal círculo se le diseña un diámetro de 0.28 para que acople bien a la bisagra 1 después en vez de aplicarle un extruir corte, le vamos a aplicar un extruir para lograr un cilindro el cual se le ensamblará a la bisagra 1 con la función concéntrica.

Figura21: Bisagra 2



Vista final del diseño de la bisagra 2.

Fuente: (Autor, 2020)

De esta forma básicamente es como se diseñó el módulo y sus partes incluyendo Arduino y demás artefactos para hacer las practicas, utilizando las

funcionesinsertar pieza, croquis, ensamblaje, extrusión, extruir corte, redondeo, redondeo de croquis, cota inteligente, líneas, spline, arco centro extremos, cuadrados, círculos, editar apariencias la cual sirve para darle color o como su nombre lo indica darle apariencia a las piezas y la función insertar piezas.

Se procede a continuación a demostrar en que parte se aplican las funciones faltantes por demostrar como: spline, arco centro extremos.

La función insertar piezas, ensamblaje y relaciones de posición (mencionada anteriormente) están relacionadas ya que cuando se diseñan diferentes piezas por separado para mayor comodidad lo primero que se hace es ir a la pestaña de ensamblaje-insertar componentes se saca la pieza del explorador de archivos a SolidWorks y si se quiere que esta se acople a alguna otra pieza se utilizan relaciones de posición como en las bisagras para unirlas, como en el ángulo máximo de apertura que tiene el módulo y demás.

3.1.5. Pasador.

Figura22: Pasador.

Vista del diseño final del pasador el cual segura que la parte inferior y superior no se abran si así se requiere.

Fuente: (Autor, 2020)

Para esta pieza se decide dibujar primero la pieza en 2d utilizando las funciones spline que es una línea que se puede curvar, arco centro esta función permite crear arcos y un cuadrado.

Figura23: Diagrama del pasador

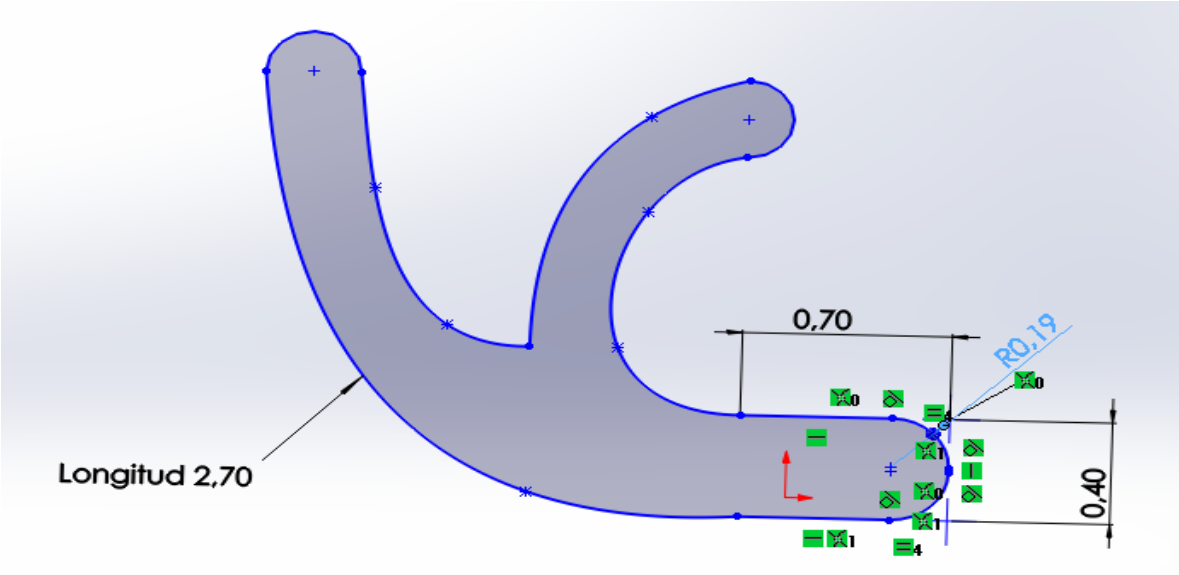
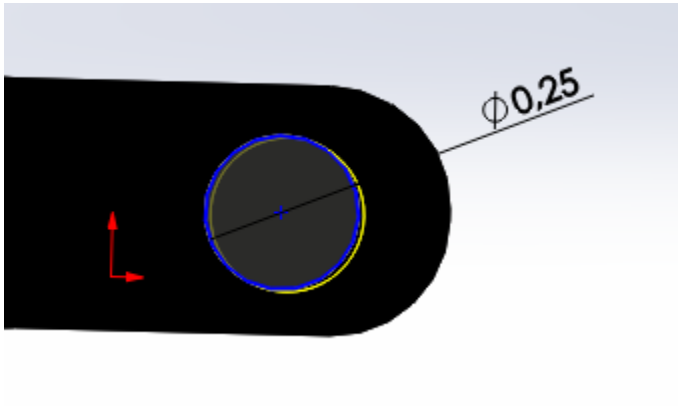


Diagrama con las medidas del eslabón y sus redondeos.

Fuente: (Autor, 2020)

Después se procede a extruir el diseño anteriormente hecho y se le dibuja un círculo el cual servirá para acoplarlo a la base del pasador y se le aplicó un extruir corte.

Figura24: Medidas del circulo.



Medidas del orificio del eslabón el cual servirá para acoplar el eslabón con la base ubicada en la parte superior del módulo.

Fuente: (Autor, 2020)

Finalmente, se le aplica un redondeo en los bordes de la pieza para darle el aspecto circular vista en la figura 21.

Como se puede observar, para el diseño del módulo completo y sus artefactos se tienen en cuenta pasos muy repetitivos que en grupo logran el resultado esperado, por lo cual no se procede a explicar cada diseño de cada móduloporque son pasos iguales y repetitivos, en conclusión, se expone cada una de las herramientas utilizadas con las cuales se puede demostrar cómo se desarrolló el módulo en su totalidad con los mismos pasos y funciones.

3.2. IMPLEMENTACIÓN DEL MÓDULO DE DESARROLLO DE DESARROLLO

Mediante el diseño previamente hecho en la herramienta SolidWorks nos guiamos para la construcción del módulo, insertando cada componente en el molde donde se eligió que estarían acoplados tomando en cuenta todas las medidas para los componentes que se agregaron en el molde según el boceto hecho.

Se elige una caja de madera la cual se divide en dos para crear dos tapas, una superior y una inferior, las cuales se unirán con bisagras comunes y se le agrega unos eslabones para que con esto la tapa superior se pueda sostener.

Se forra internamente en gamuza para darle un mejor aspecto a este y por fuera se forra con fibra de carbono para darle un aspecto óptimo.

Para la implementación de los componentes en el maletín se diseña en SolidWorks una plataforma la cual servirá para fijar todos los componentes en esta y se fabrica en acrilonitrilo butadieno estirenoel cual es un tipo de plástico muy resistente a impactos usualmente utilizado en ámbitos industriales como domésticos y también se conoce como plástico de ingeniería dado que su fabricación y procesamiento es más complejo que los plásticos comunes.

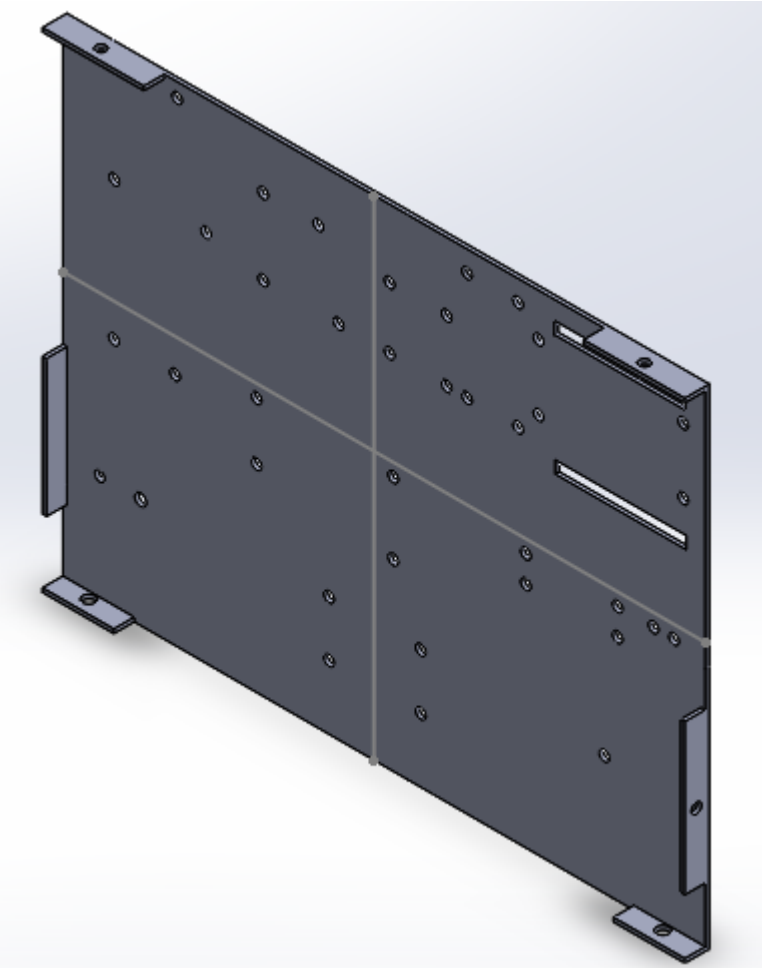
Se utiliza esta plataforma con el fin de que una vez acoplados los elementos que se agregaran en el módulo en ella, se pueda fijar esta plataforma en la base de la tapa inferior de madera por medio de 6 tornillos que irán incrustados en las aletas superiores, inferiores y laterales de esta plataforma, logrando con esto un aspecto optimo en el exterior del módulo, ya que si no se utilizara los elementos irían directamente acoplados a la tapa inferior de madera, teniendo así un aspecto nada favorable.

Estos componentes como la placa Arduino, servomotores, placa Wifi nodeMCU esp 8266, motores paso a paso, LCD keypad shield, driver puente h, L298N, etc. Que se acoplaron en esta plataforma, se acoplaron mayor mente mediante tornillos M3 los cuales cuentan con un diámetro óptimo para que este pueda ingresar por los orificios que se encuentran en cada uno de los componentes acoplados y para que ingrese por los orificios de la plataforma previamente diseñada.

Estos tornillos de medidas M3 se ajustaron por medio de tuercas comunes para facilitar el desmonte de los módulos si así se requiera, también entre estos, tornillo tuerca se le agregara un material de caucho con el fin de evitar contactos no deseados entre el tornillo o tuercas y las placas.

Se utiliza este mismo tipo de tornillos para asegurar la plataforma en el módulo, pues se considera un diámetro fuerte que asegura de manera perfecta la plataforma y sus agregados.

Figura25: Plataforma para inserción de componentes.



De acuerdo a la figura 24 ilustra la plaforma y sus orificios donde serán insertados los componentes del módulo.

Fuente: (Autor, 2020)

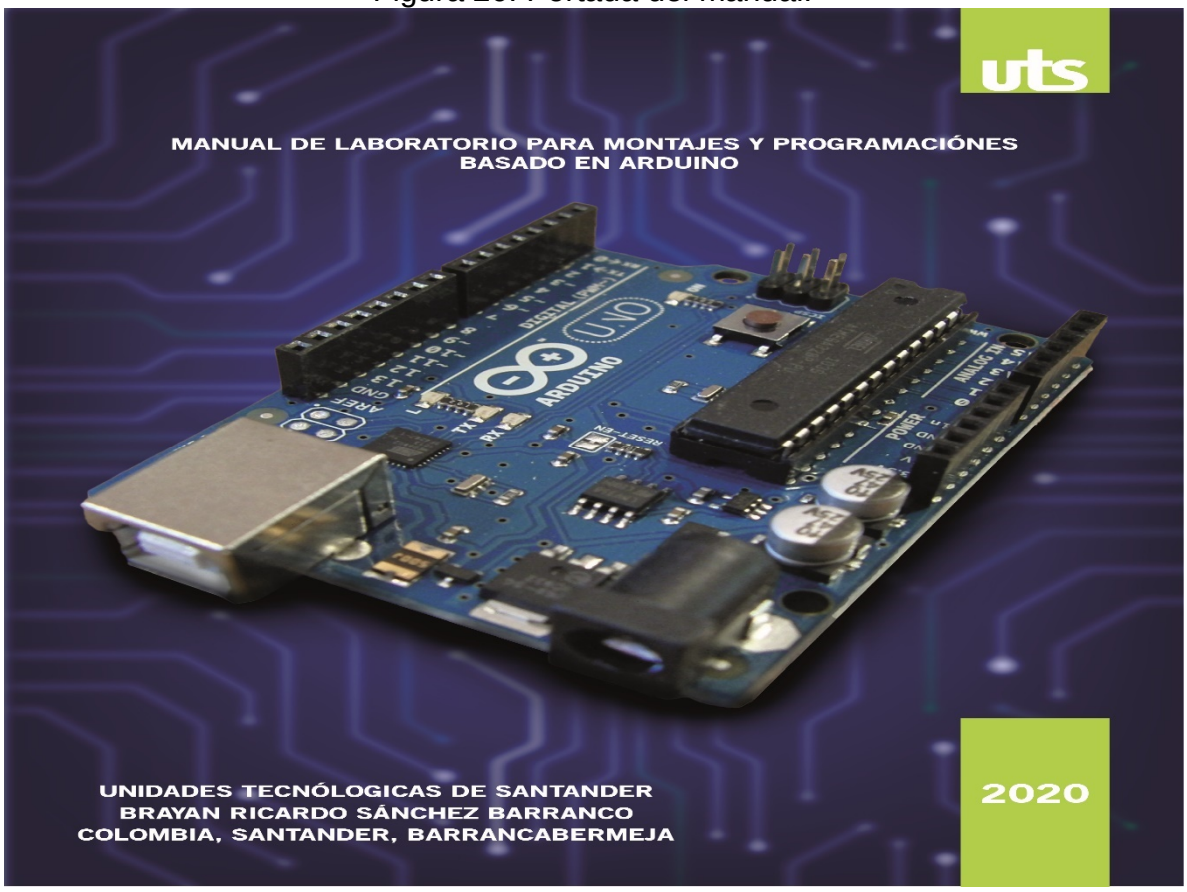
Se divide en 4 para facilidad en su fabricación ya que fue difícil encontrar maquinas que logaran fabricar esta placa completa, en el centro de estas 4 placas se le diseña una apertura y un cuadrado que sobresale con el fin de que ya diseñado se puedan acoplar estilo lego para logar mayor firmeza y se le añaden

aletas de sujeción que sirve para sujetar a las paredes del módulo logrando firmeza.

Cada orificio y apertura fue ubicado con las posiciones exactas donde se implementarán todos los componentes logrando con esto y con todo lo anterior un producto bien logrado.

3.3. MANUAL DE PRÁCTICAS DE LABORATORIO BASADO EN ARDUINO

Figura 26: Portada del manual.



De acuerdo a la figura 25 se presenta la portada del manual de prácticas de laboratorio elaborada en las herramientas software Adobe ilustrator y Adobe Photoshop.

Fuente: (Autor, 2020)

Tabla 1: Diseño del manual de prácticas de laboratorio.

Introducción
Normas de laboratorio
Iniciación en Arduino
Practicas

De acuerdo a la tabla 1 se presenta la estructura con que se diseña el manual.

Fuente: (Autor, 2020)

3.3.1. Introducción.

Se empezó dando una introducción sobre que contendrá el manual, en que plataformas se apoyó, software utilizado para el diseño de diagramas, su fin y el público al que va dirigido.

3.3.2. Normas de laboratorio.

Se empezó mostrando cuales son las normas de laboratorio, funciones del laboratorista, derechos de sus usuarios, deberes de los mismos, de los estudiantes, de los docentes, las normas de obligatorio cumplimiento, con el fin de informar a los usuarios finales del manual y de los ambientes de formación.

3.3.3. Iniciación en Arduino.

Después se elaboró un contenido que expresa el significado de las diferentes constantes, entradas y salidas de la placa Arduino, puerto serie, variables, funciones, estructuras y nomenclaturas mediante textoe imágenes que utiliza Arduino y su entorno de programación Arduino ide que eslaherramienta software la cual se utiliza para programar el microcontrolador Arduino uno que fue el que mayormente se utilizó en el manual y la placa Wifi nodeMCU esp 8266 que es compatible con Arduino.

Se empieza explicando todo este contenido con el fin de poner en contexto a los usuarios de este manual con el fin de que cuando se llegue al apartado de prácticas, en el cual se utiliza líneas de código en la herramienta software Arduino ide y otras herramientas de este software, montajes entre las diferentes salidas de la placa Arduino y módulos, pueda entender el por qué y para que la utilización de las distintas estrategias utilizadas en las 10 prácticas que contiene este manual.

3.3.4. Practicas.

Para el apartado de prácticas y montajes que se utilizaron en las 10 diferentes practicas se utilizó el software Fritzing que permite diseñar distintos circuitos entre variados módulos que se pueden encontrar en sus bibliotecas como Arduino o por ende en sus bibliotecas online gratis con el fin de dar una vista más clara sobre el conexionado de los diferentes montajes que se realizaron en este manual.

Se evidencia en este manual la realización de las 10 prácticas mediante fotos expresadas al final de cada práctica, donde se evidencia una vista real de cada una de las practicas.

La estructura para el apartado de prácticas es el siguiente.

Tabla 2: Estructura del apartado de prácticas.

Nombre de la práctica con el número de su posición
Competencia
Resultado de aprendizaje
Materiales
Contenido
Diagramas
Bibliografía

De acuerdo a la tabla 2 se evidencia la forma en las que estas prácticas se encuentran desarrolladas.

Fuente: (Autor, 2020)

Finalmente, el manual cuenta con un contenido de calidad, ilustrativo y muy didáctico que fue el fin de este con un tamaño aproximado de 115 páginas. (Véase anexo 1)

3.4. DESARROLLO DE LAS PRÁCTICAS, PRUEBAS Y ANÁLISIS DE FUNCIONAMIENTO.

Se desarrolla, se explica y se demuestra la aplicación de diferentes estructuras, variables, constantes, librerías, etc. En el manual, en este se explica la definición de cada código, letra, y que pueden significar estas, su uso final, como estas pueden facilitar procesos de programación, como por ejemplo las librerías, las cuales nos da una serie de comandos intrínsecos de estas que nos evitan varias líneas de código.

También se demuestra muchas veces la estructura del entorno de programación Arduino ide, como manejarlo y darle utilidad.

Se realizan prácticas con gran parte de las, constantes, estructuras y contextos en general que el estudiante puede darle aplicaciones importantes, como el uso de motores, microcontroladores etc.

Se fue muy didáctico por medio de muchas ilustraciones (ciento catorce ilustraciones) que dan una vista clara de ciertos indicadores que en programación pueden estar fallando ya que en Arduino ide cuando por ejemplo se agrega una librería o un comando este texto tiende a tomar un color especifico lo cual es indicador de que está bien agregado o no lo está y poco a poco a través de las distintas prácticas en el manual que van de un grado de dificultad bajo a uno más alto, va aprendiendo a identificar el alumno cuando puede estar fallando algo y cuando no.

A través de estas ilustraciones también se muestra diferentes diagramas que van desde explicar un circuito interno de x motor hasta el diagrama de conexiones de la práctica que se esté realizando siendo esto presente en todas las diez prácticas.

Se implementan este tipo de diagramas creados en el software Fritzing ya que se presta para observar mejor donde va cada cable, cosa que en una foto real no se alcanza a apreciar de la mejor manera dada la cantidad de cables, sin embargo, con el ánimo de que el estudiante se lleve una vista de cómo quedan esas prácticas en la realidad también se le añade una foto real de como quedan estas prácticas para que una vista practica no sea un faltante en este manual.

Para todo lo anterior (véase anexo 1)

3.4.1. Practica 1: Encendiendo una fila de leds.

Esta práctica busca ser un introductorio a este mundo maker, pues involucra estructuras de programación básicas como declaración de variables, declaración de pines como salida, ciclos de repetición, temporizadores y demás.

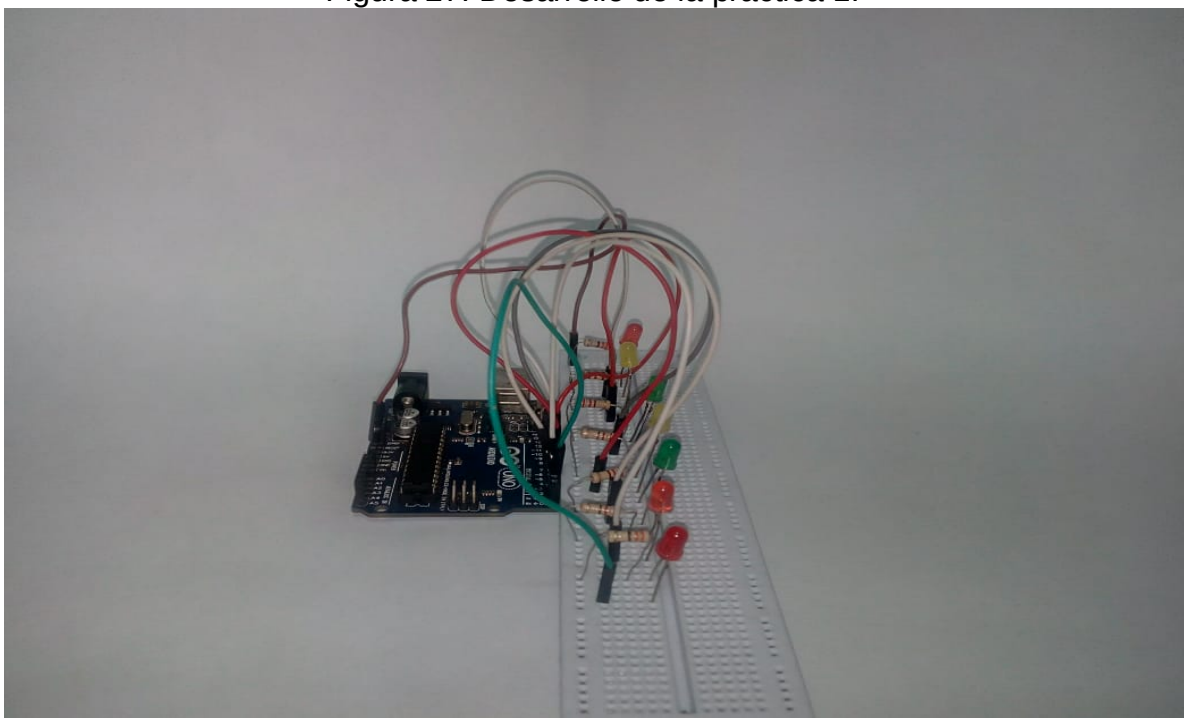
Para el desarrollo de esta práctica se utilizaron los siguientes materiales.

- Arduino uno
- Protoboard
- 8 resistencias de 220
- 8 leds
- cables

El fin de esta práctica es realizar una secuencia de luces, básicamente para programar esto primero se declaran las variables como el led1 hasta el 8, los pines que donde irán conectados estos leds se declaran como salidas y finalmente mediante un ciclo for y digitalWrite mandaran cada cuanto tiempo y cuantas veces se apagaran o prenderán según lo hayamos programado.

Se suministran sus respectivos diagramas de montaje que indican una trayectoria cerrada que inicia en el pin, pasa por el led, luego la resistencia y por último termina en gnd. Además, vistas graficas.

Figura 27: Desarrollo de la practica 1.



Vista del montaje paraje el desarrollo de la practica 1.

Fuente: (Autor, 2020)

3.4.2. Practica 2: Semáforo.

En esta práctica se puede decir que es muy similar a la anterior, a través de tiempos programar cada cuanto se encenderá o apagará un led.

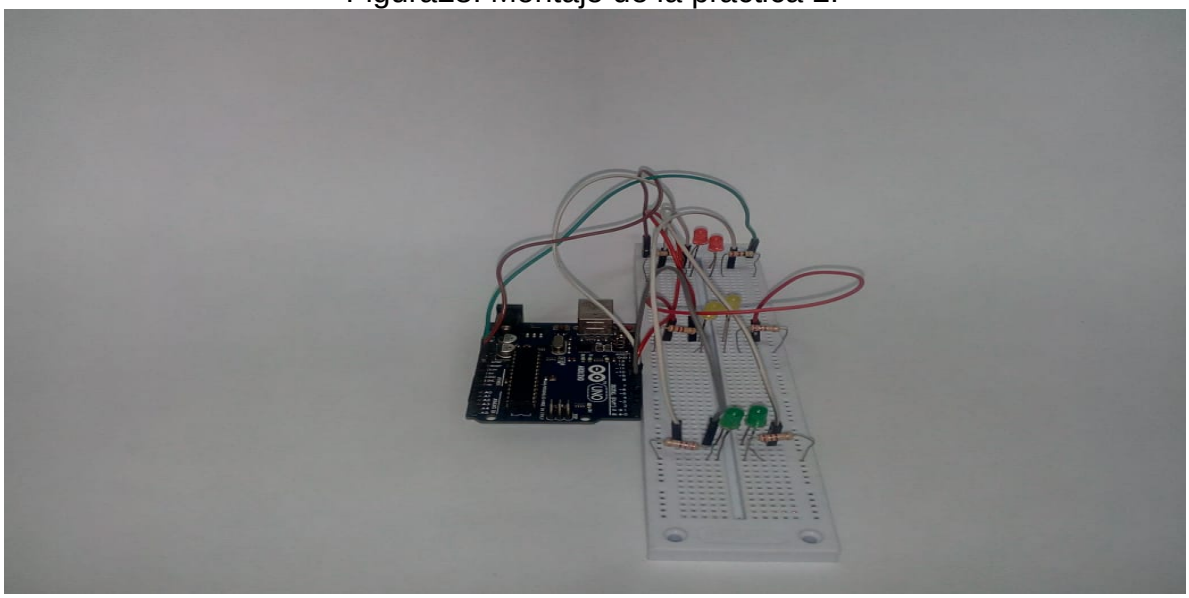
Para esta práctica se utilizaron los siguientes materiales.

- Arduino uno
- Protoboard
- 6 resistencias de 220
- 6 leds
- cables

Cabe resaltar que la diferencia a con la anterior es sus conexiones ya que en esta se busca que por ejemplo cuando se encienda la luz roja, en el segundo semáforo esté encendida la verde, de igual manera todo esto se puede lograr solo con programación y prácticamente las mismas conexiones que la practica 1 pero se quiso hacer de esta manera ya que la segunda practica hacer parte aún en la parte introductoria al lector a con este mundo maker y con este tipo de conexión empleado se hace más ameno.

En esta práctica se utiliza un software llamado Croco dile que nos brinda otra perspectiva respecto a conexiones y también Fritzing.

Figura28: Montaje de la practica 2.



Vista del conexionado para el desarrollo de la practica 2.

Fuente: (Autor, 2020)

3.4.3. Practica 3: Interrupciones externas.

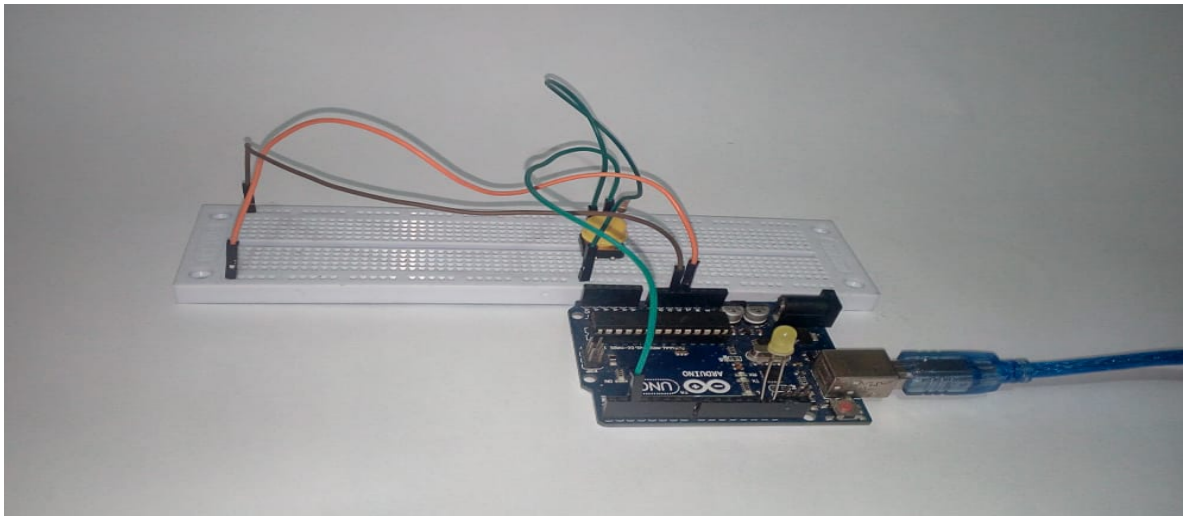
Para esta práctica se utilizan los siguientes materiales.

- Arduino uno
- Protoboard
- Pulsador
- 1 Resistencia
- Leds
- Cables

El análisis más concreto de esta práctica, como ya lo explicamos en el manual (véase anexo 1) es que si se tiene un microcontrolador y un pulsador conectado a él, se programa el Arduino como si el pulsador no estuviera conectado a él, el Arduino hace lo que quiero que haga y además programo el código que se ejecutaría cuando se presione el pulsador aparte, entonces la interrupción va a ser la encargada de escuchar ese cambio de estado en el pin, le diré escucha el pin que quiero que el sketch haga otra cosa, el sketch va a ejecutarse normal mente y cuando se produzca la pulsación se va a producir una interrupción, entonces Arduino va a dejar de hacer lo que está haciendo, va a atender esa interrupción y después continuará donde se quedó para seguir con su ejecución normal.

Ese sería el concepto de interrupción externa en esta práctica, la cual implementa comandos nuevos los cuales nos ayudan a programar estos artículos periféricos.

Figura29: Montaje de la practica 3.



Vista real del conexionado para el desarrollo de la practica 3.

Fuente: (Autor, 2020)

3.4.4. Practica 4: Secuencia de luces con control remoto.

Para esta práctica se utilizan los siguientes materiales.

- Arduino uno
- Protoboard
- Fototransistor infrarrojo HX1838
- 6 Resistencia
- 6 Leds
- Cables

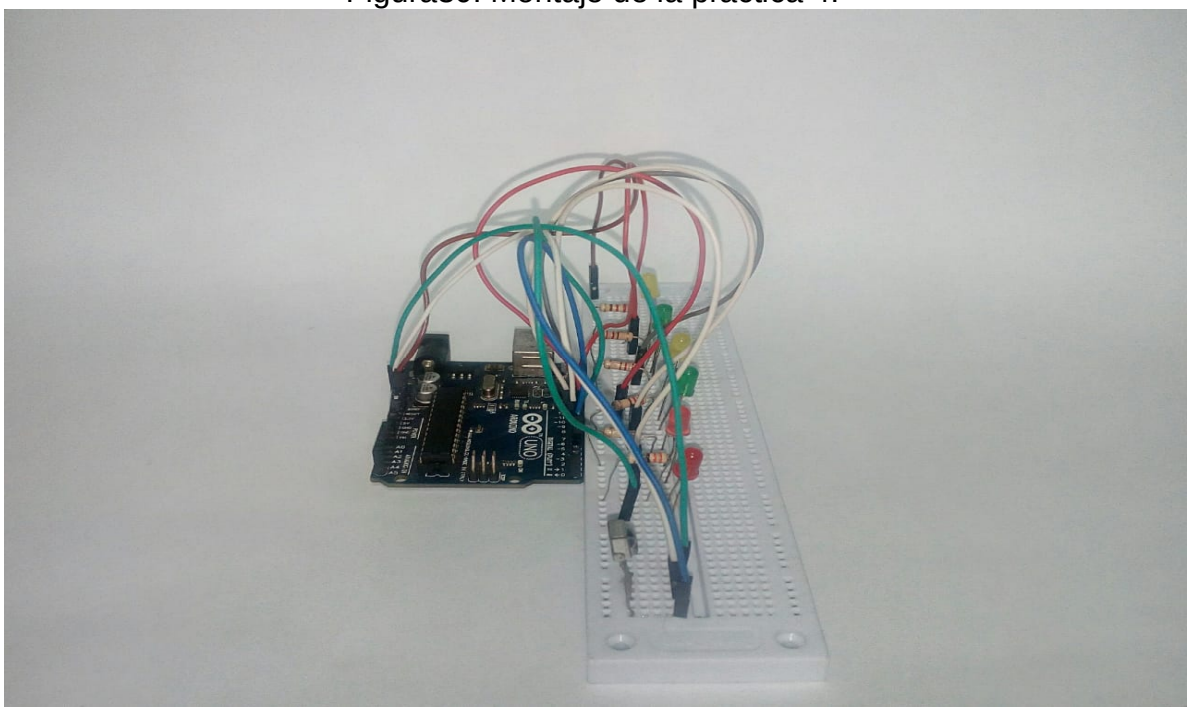
En esta práctica se incluye por primera vez la una librería y se explica por medio de ilustraciones como agregarla.

También podemos ver comandos que incluye esta librería los cuales ayudan a leer el código que envía el control remoto y que capta el infrarrojo HX1838 para posterior mente verlo en el monitor serial de Arduino ide e incluirlo en la programación, lo cual también se explica en esta práctica. (véase anexo uno)

Esta programación se trabaja a través de casos los cuales a través de funciones de (si esto pasa haz esto otro) los famosos if, else if.

Por ejemplo, se programa de que cuando capte cierto código el programa lo interprete como un 1 y cuando esto se lea ejecutará una acción.

Figura30: Montaje de la practica 4.



Conexionado para el desarrollo de la practica 4.

Fuente: (Autor, 2020)

3.4.5. Practica 5: Conexiones bluetooth.

Para esta práctica se utilizaron los siguientes materiales.

- Arduino uno
- Protoboard
- 1 Resistencia
- 1 Led
- Cables
- Modulo Bluetooth HC_06

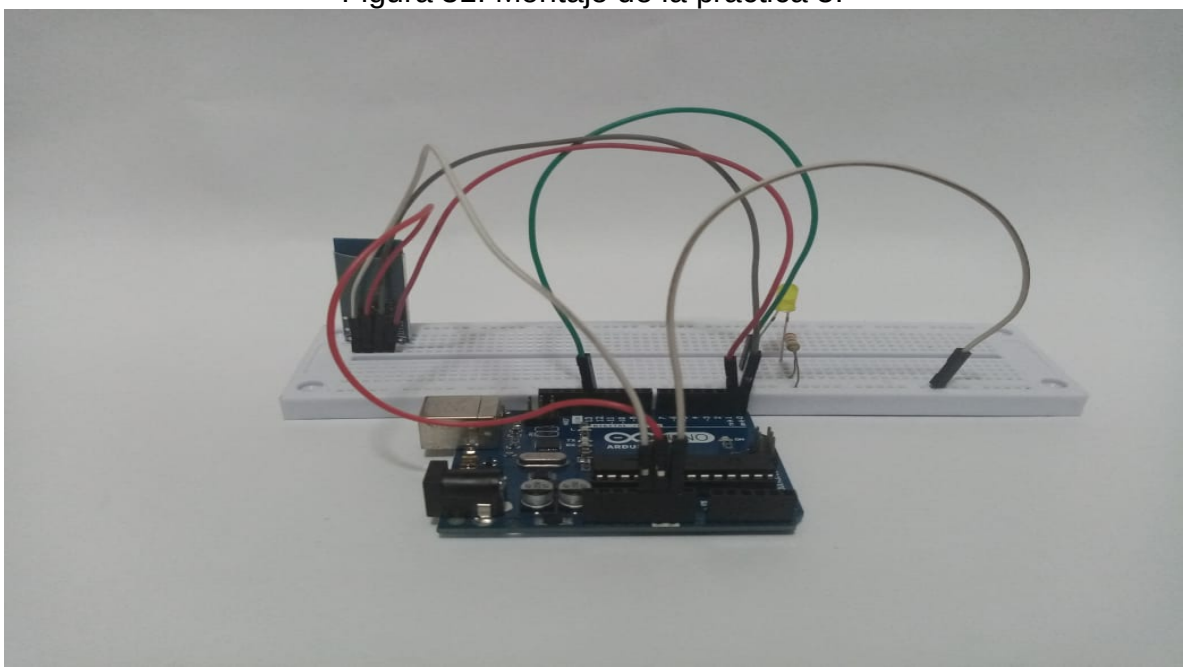
Esta práctica es parecida a la anterior, tiene el mismo patrón pues, en esta práctica se desarrolla una aplicación para celular en la página web app inventor que conecte a través de bluetooth la app y el módulo bluetooth hc-06.

El análisis de funcionamiento es bastante sencillo y es el siguiente: La app le va a mandar un numero x a él modulo bluetooth y este lo enviará a Arduino, si aquel número x cumple con las condiciones mandadas por un if ejecutará una acción o no.

Con este principio podemos manejar desde un led hasta la velocidad de un motor y demás.

Como en todas las prácticas, como ya mencionamos anteriormente, cuenta con diagramas de conexiones e imágenes ilustrativas con el fin de lograr un aprendizaje dinámico.

Figura 31: Montaje de la practica 5.



Conexionado para el desarrollo de la practica 5.

Fuente (Autor, 2020)

3.4.6. Practica 6: Sensor de temperatura y humedad.

Para esta práctica se utilizaron los siguientes materiales.

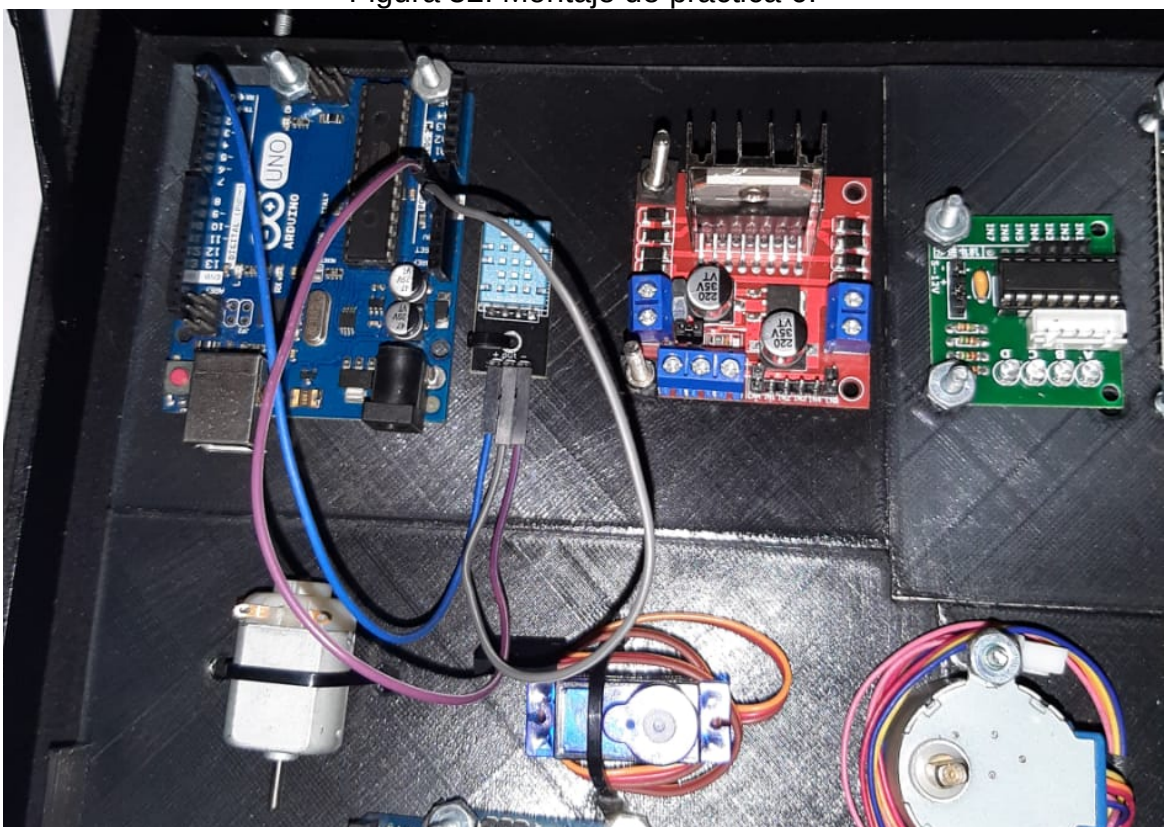
- Arduino uno
- Protoboard
- Cables
- Modulo sensor de temperatura y humedad DHT11

Primeramente, se vuelve a ilustrar el procedimiento para incluir una librería en Arduino ide, después se comienzan a declarar variables para que son las que definirán humedad, temperatura y el sensor.

Se usan como ya se ha visto anteriormente comandos predefinidos de la librería que se agregó la cual es DHT.

El análisis de funcionamiento se hace mayormente por comandos de esta librería, los cuales lo utilizamos para declarar el tipo de dht que estamos utilizando, ya que existen diferentes versiones de este tipo de sensor, también utilizamos ciertos comandos ya definidos que sirven para leer los valores de temperatura y humedad del sensor para posteriormente imprimirlos a través de un Serial.print y poder observar esos datos.

Figura 32: Montaje de practica 6.



Conexionado para el desarrollo de la practica 6.

Fuente: (Autor, 2020)

3.4.7. Acondicionamiento de señal y control de servomotor.

Para esta práctica se utilizaron los siguientes materiales.

- Arduino uno
- Cables
- Servomotor Tower pro SG90

- Protoboard
- Potenciómetro

Esta práctica la dividimos en dos, una la cual controlamos el servomotor solo con Arduino y la otra parte lo controlamos también con un potenciómetro.

Se incluyen las librerías respectivas para la programación del servo, declaramos el servo, donde irá conectado y su rango el cual nos indicará cuando el servo está en 0° y cuando está en 180°, es importante aclarar que en todos los servos este rango no es igual por lo cual se debe consultar con el fabricante cual es el rango de este.

Por medico de comandos de esta librería le indicamos cual serán los rangos del servo y el pin al cual será conectado (el de signal).

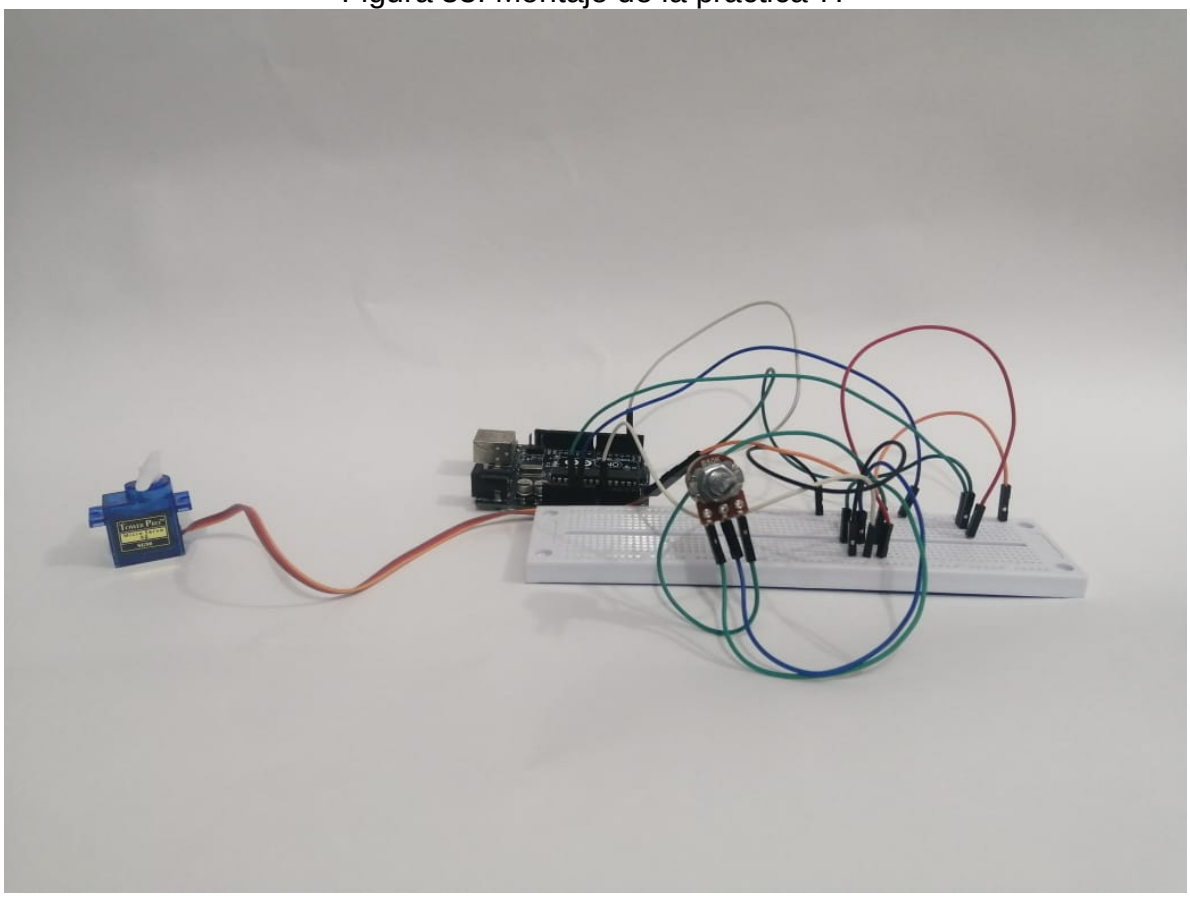
Después en el loop programamos el tiempo en el cual el servo girara desde 0° a 180° igualmente a través de comandos de la librería agregada Servo.h y básicamente ese sería el funcionamiento en esta programación.

En su segunda parte se le inserta las variables para el Angulo del potenciómetro y la variable la cual definirá el pin al cual será conectado este.

Finalmente leemos la entrada analógica a la cual conectamos el potenciómetro y después con la función map leemos en un conjunto el valor leído del potenciómetro, después le indicamos los valores máximos y mínimos que puede tomar una entrada analógica y finalmente indicamos el rango en el cual queremos

que se mueva el potenciómetro que es de 0° a 180°. Ese sería el análisis de funcionamiento de la practica 7 encontrada en el manual. (Véase anexo 1)

Figura 33: Montaje de la practica 7.



Conexionado para el desarrollo de la practica 7.

Fuente: (Autor, 2020)

3.4.8. Practica 8:Acondicionamiento de señal y control de motor paso a paso.

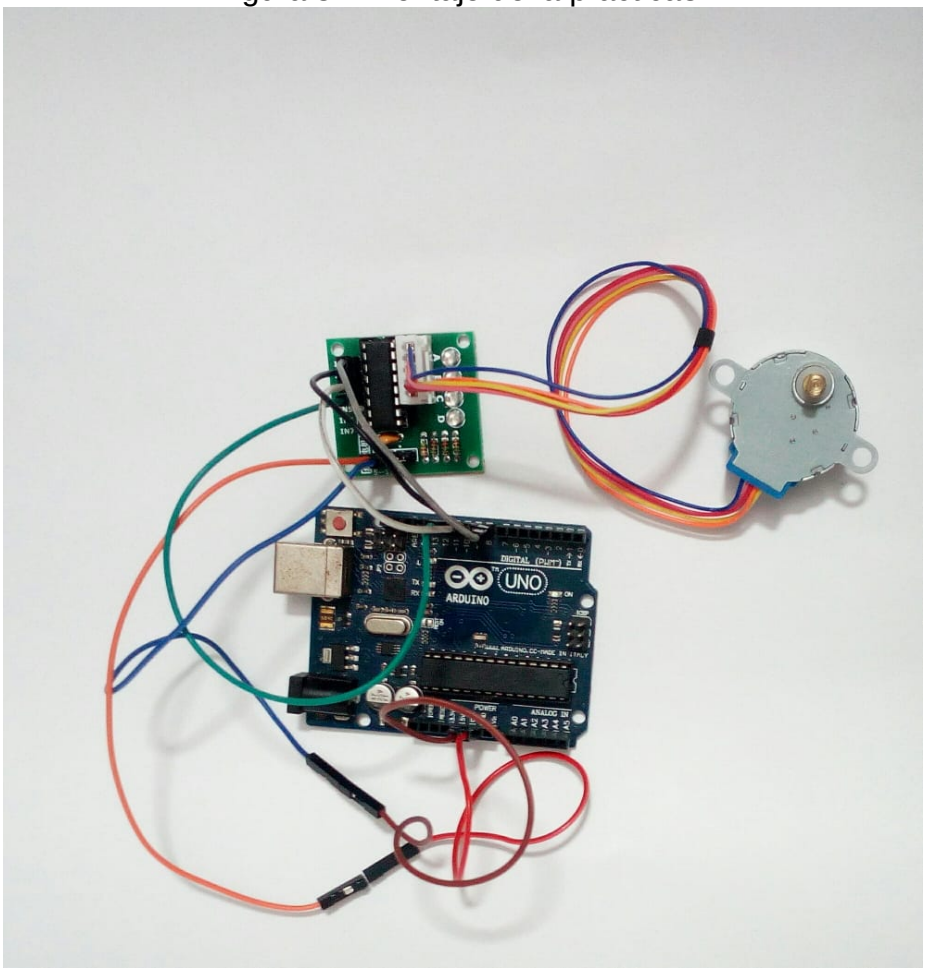
Para esta práctica se utilizaron los siguientes materiales.

- Arduino uno
- Cables
- Motor paso a paso unipolar 28BYJ-48
- Controlador de motores ULN2003

Como indica su nombre es un motor que funciona paso a paso y para el paso a paso necesitaremos energizar las bobinas en una determinada secuencia y el rotor no gira constantemente, lo hace solo de a pequeños pasos, por eso el valor de frecuencia máxima de 100hz indica que la aplicación de los pulsos a cada bobina no debe ser menor a 10ms porque pues el periodo de una señal es el inverso de la frecuencia de alii que 100hz sea un periodo de 10 ms como se menciona en el manual. (Véase anexo 1)

Básicamente se maneja estas bobinas como si se quisiesen controlar leds en una determinada secuencia según su uso y para aprovechar las máximas capacidades del paso a paso o funciones específicas a través de secuencias ya definidas en la tabla 1, 2, 3 del manual. (Véase anexo 1)

Figura 34: Montaje de la practica8.



Conexionado para el desarrollo de la practica 8.

Fuente: (Autor, 2020)

3.4.9. Practica 9: Acondicionamiento de señal y control de motor DC.

Para esta práctica se utilizaron los siguientes materiales.

- Arduino uno
- Cables

- Motor DC
- Controlador de motores DC paso a paso L298N

A través del controlador de motores DC y paso a paso L298N manejamos el sentido de giro y velocidad a través de pwm.

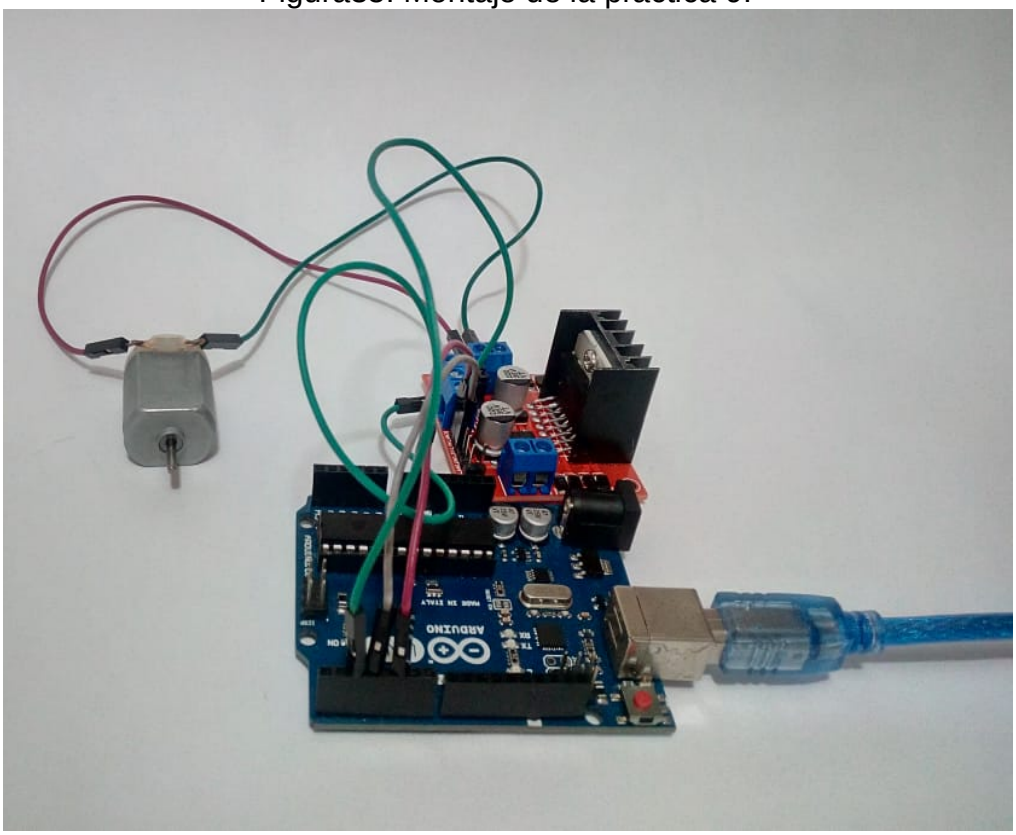
El sentido de giro resulta fácil de comprender. Esta shield puede controlar dos motores dc y a través de 3 puertos para cada uno. Un puerto llamado ena o enb y otros 4 llamados in 1, in 2, in 3 e in 4.

Para el análisis de funcionamiento de esta práctica se puede observar en la tabla 4 del manual (véase anexo 1) que cuando se alimenta los puertos en y se alimenta una in y la segunda in se deja en estado bajo, de igual manera en sentido contrario.

Podemos manejar este motor dc dándole estados altos a la iny en, agregándole tiemposse hace funcionar el motor.

También se controla la velocidad de este por medio de los pines pwm y creando una variable velocidad, con un ciclo for se le hizo que reduzca o aumente su velocidad paulatinamente.

Figura35: Montaje de la practica 9.



Conexionado para el desarrollo de la practica 9.

Fuente: (Autor, 2020)

3.4.10. Practica 10: Recepción de datos de temperatura y humedad a través de un módulo wifi a una aplicación vía internet.

Para esta práctica se utilizaron los siguientes materiales.

- Arduino uno
- Cables
- Modulo wifi esp8266 nodeMCU
- Sensor de humedad y temperatura DHT11

Para el análisis de funcionamiento primero que todo se programó los gauges que ilustrarían los valores de humedad y temperatura en la app en Blynk, esta da un código el cual se inserta en la programación de Arduino.

Después se agregan las librerías necesarias para esta práctica y la board del esp 8266 lo cual se ilustra en el manual. (Véase anexo 1)

Se declaran variables tipo Char para declarar las variables donde insertaremos el nombre del wifi, la contraseña de esta y el código que nos envió Blynk.

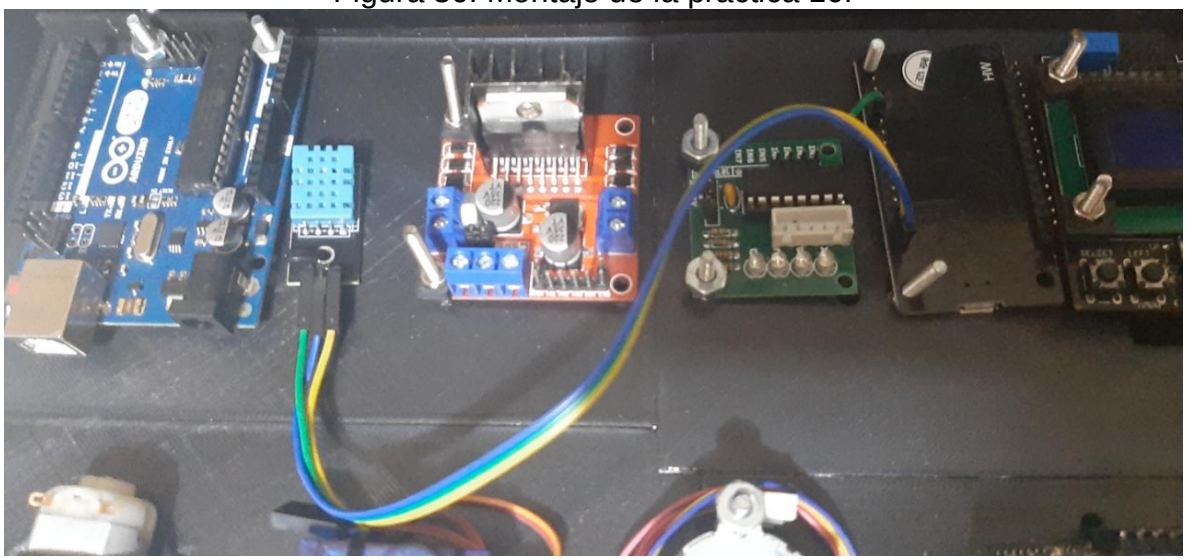
Después se agregan los comandos respectivos de la biblioteca Dht para leer los datos de temperatura como en la practica 7.

También programamos en la programación con los comandos ilustrados la función de si el dht no envía datos numéricos imprimirá un mensaje que indicará que se está presentando un error.

Con los comandos de la librería de Blynk relacionamos los pines que declaramos en la app con los valores de temperatura y humedad.

Finalmente hacemos iniciar Blynk dht y el temporizador que revisará los datos suministrados por el dht cada segundo.

Figura 36: Montaje de la practica 10.



Conexionado para el desarrollo de la practica 10.

Fuente: (Autor, 2020)

4. RESULTADOS

Posterior a la socialización y divulgación del Proyecto, se hace entrega del módulo de entrenamiento para el desarrollo de aplicaciones electrónicas orientadas a la industria basado en microcontrolador Arduino.

Se estima acogida y disposición por parte del cuerpo docente para la aplicación y ejecución de las actividades teóricas y prácticas diseñadas en el formato presentado.

Cabe resaltar, que esta propuesta permitirá al docente complementar sus estrategias de enseñanza con los contenidos ilustrativos y didácticos en las

clases, lo cual conlleva a una mayor comprensión y recepción de contenido por parte de los estudiantes.

Se realiza consultoría con la empresa Ferretería la principal la cual expresa acogida y gusto del módulo didáctico debido a ser un producto que está dentro de oferta de productos manejada por esta empresa y al ser el módulo algo compacto y visualmente bien logrado comentan que dada a esta característica puede ser de fácil almacenaje y ventas. Estiman futura cohesión entre la empresa Ferretería la Principal y el módulo.

5. CONCLUSIONES

Al utilizar herramientas de modelado como SolidWorks se facilita la construcción de los diseños hechos previamente concluyendo con esto que el hacer un diseño previo nos ahorra tiempo ya que al momento de la construcción no se tendrá que recurrir a la prueba y error, sino que se entrará a construir lo diseñado de manera exacta y hace que la construcción sea más ordenada ya que al hacer el diseño se saben los paso a paso de construcción del diseño en la realidad.

Se observó la facilidad y orden con la que se pudo construir el módulo didáctico gracias al diseño previamente hecho en SolidWorks, también se concluye que la implementación de módulos didácticos como este dinamiza el aprendizaje ya que al contar con módulos que facilitan la realización de diferentes practicas al tenerlos tan a la mano y de manera accesible expande las capacidades indudablemente de los estudiantes y las oportunidades de acceder a estos de manera más fácil.

Se puede concluir que la realización de manuales interactivos logra un conocimiento más fácil y ameno de adquirir para el usuario inexperto ya que la inserción de imágenes ilustrativas, diagramas y tablas ayudan más que solo un texto y dan una idea más clara de lo que se está queriendo lograr.

Al poner en práctica la teoría previamente hecha al inicio del manual se concluye que esos conocimientos se acentúan y fortalecen ya que al hacer cosas de forma práctica y al aplicar lo aprendido en prácticas reales se logra tener todas las vistas y aplicaciones del conocimiento adquirido, en esta etapa el usuario puede estar en

la capacidad de optimizar procesos ya que al tener la teoría y experiencia juntas tiene todas las bases para lograr un buen producto.

6. RECOMENDACIONES

Principalmente se recomienda la inserción de másmódulos como el presentado ya que este facilita y pone más al alcance de las manos de los estudiantes todos estos módulos que sin duda serán muy importantes para las competencias que puedan desarrollar estos.

En el caso de las placas que necesiten desacoplarse del módulo para un mejor uso se recomienda no olvidar el uso de aislantes en los orificios de estas ya que al contar con tornillos metálicos M3 puede existir el riesgo de un contacto no deseado.

Se recomienda ubicar el módulo didáctico en un lugar fresco y seco ya que como contiene elementos electrónicos la humedad los puede afectar seriamente.

Se recomienda utilizar las herramientas necesarias para el uso del módulo con el ánimo de preservar esta gran herramienta para el uso del alumnado.

7. REFERENCIAS BIBLIOGRÁFICAS

Aguilar, & Marquez. (s.f.). *Ambientes de aprendizaje en arduino.* Obtenido de http://www.somedicyt.org.mx/simposio/images/docs/simposio/2015/memorias/ambientes-de-aprendizaje-para-la-ciencia-usando-tecnologia-arduino.pdf

Ayala, & Yupa. (Octubre de 2013). *Universidad Politecnica Salesiana.* Obtenido de https://dspace.ups.edu.ec/bitstream/123456789/5522/1/UPS-GT000511.pdf

Baeza, J. P. (2009). *Manual de Arduino.* Obtenido de Universidad de Alicante: http://electroship.com/documentos/Arduino_user_manual_es.pdf

Bonilla, M. F., Gutierrez, S. M., Orueta, G. D., & Gil, M. A. (Diciembre de 2017). *Dyra New technologies.* Obtenido de https://www.dyna-newtech.com/busqueda-NT/buenas-practicas-para-evitar-vulnerabilidades-en-diseno-con-plataformas-de-hardware-abierto-tipo-ard

Condori, O. &. (2018).

Correz, D. R. (18 de noviembre de 2016). *Unir.* Obtenido de https://reunir.unir.net/bitstream/handle/123456789/4540/RUIZ%20CORRES%2C%20DANIEL.pdf?sequence=1&isAllowed=y

Crespo, E. (6 de Octubre de 2016). *Aprendiendo Arduino*. Obtenido de https://www.aprendiendoarduino.com/

Esqueda, J. A., López, A. S., Espino, V. R., & Salazar, R. M. (1 de Abril de 2014). *Dialnet.* Obtenido de https://dialnet.unirioja.es/servlet/articulo?codigo=4741421

Gil, Y. Z. (2015). *Universidad Pedagogica Nacional.* Bogota. Obtenido de http://repository.pedagogica.edu.co/bitstream/handle/20.500.12209/1991/TE-18155.pdf?sequence=1&isAllowed=y

Herrador, R. E. (13 de Noviembre de 2009). *Guía de Usuario de Arduino.* Obtenido de http://electroship.com/documentos/Arduino_user_manual_es.pdf

Herrero, & Sanchez. (Julio de 2015). *Researchgate.* Obtenido de Universidad Alfonso X el Sabio: https://www.researchgate.net/publication/320531618_Una_mirada_al_mundo_Arduino

Paucara, R. O. (2016).

Rivera, & Turizo. (06 de Junio de 2015). *Ventana informatica.* Obtenido de http://revistasum.umanizales.edu.co/ojs/index.php/ventanainformatica/article/view/1098

Riveros. (2017). Obtenido de http://ria.utn.edu.ar/bitstream/handle/123456789/1835/LTE_%20Riveros%2C%20Gabriel.pdf?sequence=1&isAllowed=y

Sierra, Rojas, & Garcia. (26 de Agosto de 2019). *Universidad Tecnlogica de Panama.* Obtenido de https://revistas.utp.ac.pa/index.php/memoutp/article/view/2304

Tapia, & Manzano. (Octubre de 2013). *Universidad Politecnica Salesiana.* Obtenido de https://dspace.ups.edu.ec/handle/123456789/5522

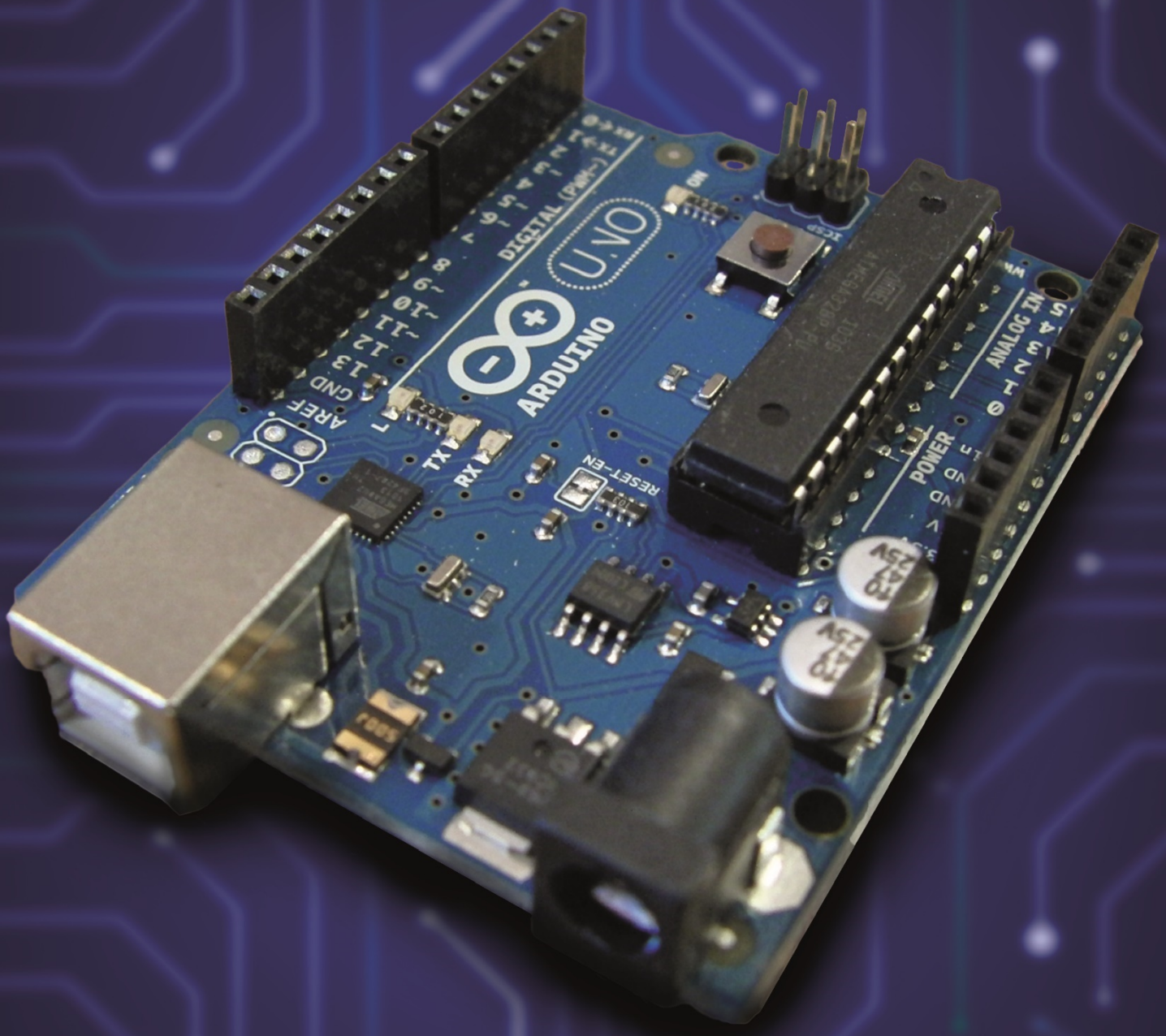
Vargas, C. &. (2015). Obtenido de http://www.ecorfan.org/bolivia/researchjournals/Tecnologia_e_innovacion/vol2num4/Revista-de-Tecnologia-e-Innovacion--Volumen-4-164-169.pdf

Wikipedia. (23 de 09 de 2019). *Arduino*. Obtenido de https://es.wikipedia.org/w/index.php?title=Arduino&oldid=119633331

8. ANEXOS

8.1. ANEXO 1. MANUAL DE LABORATORIO PARA MONTAJES Y PROGRAMACIONES BASADO EN ARDUINO.

MANUAL DE LABORATORIO PARA MONTAJES Y PROGRAMACIÓNES BASADO EN ARDUINO



UNIDADES TECNÓLOGICAS DE SANTANDER
BRAYAN RICARDO SÁNCHEZ BARRANCO
COLOMBIA, SANTANDER, BARRANCABERMEJA

2020

FACULTAD DE CIENCIAS NATURALES E INGENIERÍAS

TECNOLOGÍA EN OPERACÍON Y MANTENIMIENTO ELECTROMECANICO

MANUAL DE LABORATRORIO PARA MONTAJES Y PROGRAMACIONES BASADO EN ARDUINO

AUTOR

BRAYAN RICARDO SANCHEZ BARRANCO

Colombia, Santander, Barrancabermeja

Fecha de presentación: (30/05/2020)

LISTA DE FIGURAS

LISTA DE TABLAS

8.1.1. Introducción.

Este manual de entrenamiento aborda los conceptos de programación en Arduino, características del microcontrolador y montajes en diferentes prácticas que se desarrollaran yendo desde un nivel básico hasta algo un poco más avanzado. Este manual es una excelente guía para todo aquel que quiera iniciar, sembrar bases en programación y automatizar procesos gracias a este apasionante entorno de Arduino.

Nos apoyaremos en imágenes ilustradas gracias a Fritzing (Elaboración de esquemáticos y montajes) http://fritzing.org

El manual ha sido elaborado por Brayan Ricardo Sanchez Barranco usando para esto principalmente información y recursos de:

http://www.arduino.cc (página oficial de Arduino)

http://zonamaker.com (página web)

https://aprendiendoarduino.wordpress.com (página web)

IDENTIFICACIÓN	
UNIDAD ACADÉMICA	FACULTAD DE CIENCIAS NATURALES E INGENIERIAS
ASIGNATURA: MICROCONTROLADORES	

8.1.2. NORMAS DE LABORATORIO.

COMPETENCIA	RESULTADOS DE APRENDIZAJE
Conocer las normas que rigen el laboratorio y los formatos requeridos para la evaluación de las prácticas.	• Conoce las normas institucionales de trabajo en el laboratorio. • Conoce cada uno de los formatos requeridos para la evaluación de las prácticas del laboratorio.

ACTIVIDADES

Para dar a conocer las normas, el docente lee y explica las funciones, deberes de los usurarios, derechos de los usuarios, las normas de seguridad y normas de obligatorio cumplimiento. También se presentan los formatos de pre-informe e informe que se emplean para la evaluación de los laboratorios.

REGLAMENTO DE LABORATORIO

FUNCIONES DEL LABORATORISTA

Son funciones de los auxiliares, técnicos e ingenieros encargados de los laboratorios de las Unidades Tecnológicas de Santander, las siguientes:

- Firmar el Paz y Salvo requerido por los estudiantes.
- Ordenar y reubicar los equipos y manuales dentro del laboratorio.
- Facilitar el área física del laboratorio y los implementos para el desarrollo de las prácticas.
- Alistar el laboratorio para dar inicio a las labores académicas.
- Elaborar las normas adecuadas que permitan compartir las responsabilidades con los docentes, usuarios del laboratorio y el laboratorista.
- Recopilar y unificar los pedidos de las necesidades semestrales de las asignaturas que se ofrezcan en el laboratorio y remitirlas a la Coordinación o Departamento respectivo para mantener los materiales necesarios para las prácticas.
- Revisar periódicamente el estado del laboratorio, de los materiales y equipos que en allí se encuentren, notificando inmediatamente a la Coordinación o Departamento respectivo, en caso de observar algún deterioro, desperfecto o falta de alguno de ellos.
- Recibir los materiales y equipos que ingresen al laboratorio, firmando el cargo correspondiente.
- Elaborar un inventario de equipos y materiales existentes en el laboratorio, cada fin de semestre académico y remitirlo a las instancias respectivas.
- Velar por el cumplimiento de las normas de trabajo de obligatorio cumplimiento y de las normas de seguridad dentro del laboratorio.
- Mantener el laboratorio aseado, en perfecto estado las herramientas, equipos y módulos de trabajo, es decir, en óptimas condiciones para realizar las prácticas en forma eficiente.
- Proporcionarle al docente y a los estudiantes de la asignatura, los materiales y el equipo necesario para la realización de las prácticas respectivas.
- Custodiar los elementos y equipos de los laboratorios.
- Realizar mantenimiento preventivo y correctivo a los equipos existentes en el laboratorio.
- Brindar un eficiente, oportuno y amable servicio a los estudiantes.
- Coordinar y planificar en conjunto con el jefe inmediato el servicio general de los laboratorios y/o talleres de los respectivos programas.

- Coordinar y planificar los horarios de los servicios de laboratorios.
- Reintegrar al almacén general de la institución el equipo, o enseres que del laboratorio sean dados de baja de acuerdo con la autorización del jefe inmediato.
- Retirar del almacén general los pedidos que para su dependencia se haya hecho.
- Colaborar con la organización de las muestras técnicas y de divulgación que la institución realice o participe respectivamente.
- Sugerir los cambios y las modificaciones que crea conveniente para el funcionamiento y la modernización del laboratorio que se tiene a cargo.
- Al finalizar la práctica, se debe verificar que los equipos, estén en perfectas condiciones.
- Responder por las herramientas y equipos del laboratorio a cargo.
- Colaborar en la instalación e implementación de equipos y máquinas adquiridas por la institución para la actualización de los módulos de trabajo.
- Resolver dudas pertinentes a las prácticas que se estén realizando en ausencia del docente.
- Las demás funciones que le sean asignadas por el superior inmediato acorde con la naturaleza del cargo.

DERECHOS DE LOS USUARIOS

Son derechos de los usuarios de un laboratorio de las UTS los siguientes:

- Utilización de los recursos disponibles para préstamo dentro de los horarios establecidos.
- El usuario tendrá derecho a solicitar asesoría en el momento que lo requiera y será brindada por el auxiliar encargado o por el profesor de la materia.
- El usuario dispondrá del servicio para uso exclusivo de su formación académica.
- Las solicitudes de servicio de soporte técnico o soporte al usuario se harán directamente a la persona encargada de la atención y registro de estos requerimientos y podrán efectuarse

mediante solicitud verbal de acuerdo con los procedimientos establecidos por el reglamento del laboratorio respectivo.

- Los equipos y materiales que van a utilizar los docentes y estudiantes deben encontrarse en perfecto orden y aseo.
- Préstamo de los elementos o equipos necesarios para realizar las practicas del laboratorio.
- Solicitar el buen estado de los elementos, equipos y bancos de trabajo.
- La disponibilidad de los laboratorios en los horarios estipulados.
- Exigir la verificación del funcionamiento de los equipos y elementos solicitados.
- La explicación por parte del docente de la correcta manipulación de los equipos.
- Los estudiantes tienen derecho a la clase práctica, orientada por el docente y el conocimiento con anterioridad de las prácticas a realizar.
- Recibir un trato cortes según los principios básicos de las relaciones humanas.
- Recibir las advertencias necesarias que le permitan trabajar cumpliendo todas las normas de obligatorio cumplimiento y de seguridad que disponga cada laboratorio según su reglamento interno.
- El estudiante para solicitar el préstamo de equipos y elementos dispone de 15 minutos después del inicio del laboratorio.
- Solicitar el permiso correspondiente si tuviera que ausentarse o no asistir, siempre y cuando sea por una causa justificada.

DEBERES DE LOS USUARIOS

Son deberes de los usuarios de los laboratorios de las UTS los siguientes:

- El usuario deberá comprometerse a dar un trato adecuado a los equipos, hardware, software, muebles y elementos que hagan parte del laboratorio, respetando y acatando las normas

establecidas en el presente reglamento.

- Todo usuario de un laboratorio deberá al momento de solicitar el servicio presentar el documento que lo acredite como usuario, ya sea carné de estudiante, de empleado de las UTS, o cedula de ciudadanía cuando se trate de algún tipo de convenio interinstitucional.
- Todo usuario se hace responsable ante las UTS por los daños que se ocasionen a los equipos, muebles y enceres durante el tiempo de su utilización.
- El usuario recibirá el equipo en perfectas condiciones; si detecta cualquier irregularidad en el funcionamiento, daño o faltante de algún elemento propio del equipo, deberá reportarlo inmediatamente antes de hacer uso de este.
- Queda rotundamente prohibido a cualquier usuario utilizar los equipos para prácticas o fines diferentes a aquellos para los cuales fueron prestados por la institución, haciéndose además responsable del deterioro de los equipos por uso negligente, así como de cualquier tipo de lesión en su persona o en terceros que pueda derivarse de estos actos.
- El usuario deberá presentarse a los laboratorios de las UTS vistiendo las ropas adecuadas y cumpliendo con los requisitos de seguridad industrial necesarios para realizar la práctica académica en cada laboratorio; la institución y el laboratorio no asumen responsabilidades por la omisión, desconocimiento o violación de esta regla por parte de sus usuarios.

DE LOS ESTUDIANTES

- Dejar en perfecto orden y aseo todos los equipos, materiales, y manuales utilizados en la práctica.
- Pagar o reponer en caso de pérdida o daño el (los) material(es) y equipo(s) que se encontraban a su cargo durante la práctica.
- Debe mantener el orden y la disciplina durante la práctica.
- Debe hacer un buen uso de los equipos y materiales a su cargo durante las prácticas de

laboratorio.

- Preservar, cuidar y mantener en buen estado el material de enseñanza, instalaciones, equipos, dotación y bienes de los laboratorios.
- Cumplir con las normas de respeto y convivencia para el logro de una formación integral.
- Cumplir con las normas de seguridad del laboratorio que disponga cada laboratorio según su reglamento interno.
- En caso de no conocer el manejo de los equipos es necesario pedir las instrucciones pertinentes antes de realizar cualquier conexión y de usarlos.
- Cuidar lo que se conserve bajo su cuidado o a lo cual tenga acceso, así como impedir o evitar la sustracción, destrucción, ocultamiento y utilización indebida de los equipos que se encuentren en el laboratorio.
- Verificar antes de iniciar una práctica el estado de su puesto de trabajo y del equipo a utilizar en la experiencia.
- Tratar con respeto, imparcialidad y rectitud a las personas con que tenga relación por razón del servicio.
- Avisar inmediatamente al asistente, o persona encargada de las salas acerca de las anomalías que se presenten en los equipos.
- Informar al docente o encargado del laboratorio sobre el mal uso que otros usuarios hagan de los equipos.
- Acatar las instrucciones de la persona encargada del laboratorio y respetar sus decisiones de acuerdo con lo dispuesto en este reglamento.

DE LOS DOCENTES

- Durante la primera práctica deberán dar las indicaciones a los estudiantes, referentes al buen uso del material y equipos de laboratorio, así como de sus deberes, obligaciones y

cumplimiento de las normas de seguridad dentro del laboratorio.

- Dar las indicaciones necesarias para la realización de las prácticas de laboratorio y la explicación para su ejecución.
- Durante las prácticas de laboratorio, por ningún motivo deben abandonar a los estudiantes a su cargo, ni ocupar el tiempo de las prácticas en las actividades ajenas a las mismas.
- Dar la explicación respectiva de la práctica a realizar, así como también la aclaración de las dudas que tengan los estudiantes.

NORMAS DE OBLIGATORIO CUMPLIMIENTO

Se establecen las siguientes normas de estricto cumplimiento:

- Cumplir con el horario de laboratorio establecido, para la realización de las prácticas.
- Está prohibido el ingreso de comidas, bebidas, cigarrillos a los laboratorios.
- Está prohibido el ingreso de estudiantes en pantaloneta, bermuda, sandalias o en chanclas a los laboratorios.
- Tendrán acceso al laboratorio los estudiantes que se encuentren debidamente matriculados en el período académico correspondiente.
- Para préstamo de equipos y/o elementos del laboratorio se debe presentar carnet debidamente estampillado.
- Para el inicio de la práctica de laboratorio debe estar presente el docente de la asignatura quien se hará responsable de la sala.
- Está prohibido facilitar o propiciar el ingreso al laboratorio de personas no autorizadas.
- En lo posible, el docente y el encargado deben permanecer todo el tiempo en el laboratorio, durante la realización de las prácticas.
- Quince (15) minutos después de iniciar la práctica de laboratorio no se permite el ingreso de

estudiantes al aula.

- Después de quince (15) minutos de haber comenzado la práctica de laboratorio no se despachará ninguna lista de pedido de equipos y/o elementos a los estudiantes (seguridad del laboratorio).
- Quince (15) minutos antes de la hora prevista para la terminación de la práctica de laboratorio, el estudiante debe devolver los equipos y/o elementos dados en préstamo.
- El material asignado a cada práctica debe permanecer en el mismo lugar. No se debe coger material destinado a prácticas distintas a la que se está realizando.
- En caso de dudas en el momento de conectar un equipo, se debe preguntar a la persona indicada, así se evitará el pago innecesario.
- El estudiante debe seguir los pasos establecidos por el docente para la práctica.
- Se permite el uso del laboratorio si está autorizado por el Coordinador del programa o el laboratorista a cargo.
- Todo estudiante debe estar debidamente preparado para la realización de la práctica.
- La ausencia injustificada de una práctica de laboratorio se calificará con cero, cero (0,0).
- La no presentación del pre-informe y del informe el día de la práctica se calificará con cero (0.0).
- Cada equipo de trabajo es responsable del material que se le asigne, en caso de pérdida o daño deberá responder por ello. Antes de empezar con el procedimiento experimental o utilizar algún aparato, revisar todo el material, y en caso de desconocer su funcionamiento pregunte al docente o al encargado del laboratorio.
- La pérdida o deterioro por mal uso de un elemento, aparato o equipo, se cobra al estudiante responsable. En caso de no encontrarse un responsable único, el grupo de la práctica correspondiente asumirá la responsabilidad y cubrirá los costos de reparación o de sustitución del equipo.
- No se permite el traslado de computadores, sillas o de cualquier otro material o equipo que se

encuentre en el laboratorio, sin la debida autorización del funcionario encargado del mismo.

- Al finalizar la práctica el material y la mesa de trabajo deben dejarse limpios y ordenados.
- En los laboratorios con computadores, estos son para uso exclusivamente académico, evite instalar programas de índoles ajenas a las de la academia.
- En los laboratorios con computadores, se prohíbe la utilización de software que no esté amparado legalmente mediante la respectiva licencia para la Universidad.
- Se prohíbe el cambio de la configuración del software instalado.

CRITERIOS PARA LA EVALUACIÓN DE ASIGNATURAS DE LABORATORIO

Para evaluar los laboratorios se considerarán los siguientes instrumentos con sus respectivos porcentajes: **Pre-informe y/o quices(20%), Informes (40%), Parcial Escrito y/o práctico(40%)**.

Pre-informe y/o quiz: Para la revisión de los conceptos previos se evaluará con el pre-informe o un quiz.El pre-informe se presenta al iniciar cada experiencia, es un documento escrito a mano que se elabora teniendo en cuenta la información suministrada en el manual de guías de laboratorio. En el pre-informe el equipo de trabajo refleja lo que va a ser su actividad en la práctica del día. A continuación, se presenta el formato para la elaboración del pre-informe:

UNIDADES TECNOLÓGICAS DE SANTANDER

FORMATO DE PREINFORME DE LABORATORIO

IDENTIFICACIÓN

PRACTICA Nº: El número que identifica la práctica **NOMBRE DE LA PRÁCTICA:** El título debe ser conciso, pero completo, en forma tal, que se entienda claramente el objeto del experimento. Por ejemplo: “Carga Específica del Electrón”.	**FECHA:**

INTEGRANTES **NOMBRE:** Los estudiantes que conforman el equipo de trabajo	**CÓDIGO:**
NOMBRE:	**CÓDIGO:**
NOMBRE:	**CÓDIGO:**

PROGRAMA: El programa académico que estudia. Por ejemplo: “Tecnología Electromecánica”	**GRUPO:**El Grupo de la asignatura. Por ejemplo: “E111” **Nº grupo:**El grupo que lo identifica dentro del laboratorio. Es un número de 1 a 7.	**DOCENTE:**

RESULTADOS DE APRENDIZAJE

Son los resultados de aprendizaje trabajados desde el Programa de Asignatura.

MARCO TEÓRICO

En este espacio se describen las leyes, principios y teorías en las que se basa y se fundamenta la práctica a desarrollar.

MATERIALES Y EQUIPOS
Aquí se relacionan todos los materiales a utilizar para el desarrollo de la práctica.
PROCEDIMIENTO (MONTAJE Y EJECUCIÓN)
En este espacio se debe realizar un resumen del procedimiento de la práctica, deberá hacerse en diagrama de flujo, mapa conceptual y dibujo del montaje, o simulación.
NOTA DE SEGURIDAD
Hace referencia a los cuidados que se deben tener con algunos equipos y/o materiales del laboratorio. También el cuidado que se debe tener con los altos voltajes y corrientes que se usan en algunas prácticas.
REFERENCIAS BIBLIOGRÁFICAS
Para reportar un libro texto, escriba en su orden y teniendo en cuenta los signos de puntuación, los siguientes elementos: Autor, título, número de edición, lugar de publicación, nombre de la editorial, año de publicaciones, paginación. Ejemplo: GROSSMAN, Stanley. INTRODUCCIÓN AL ALGEBRA LINEAL. Editorial Mc Graw Hill Interamericana. México. 512,5L269i

Otras variables para tener en cuenta, en el momento de la evaluación de los laboratorios son: la puntualidad, el trabajo en equipo, el comportamiento y seguimiento de las normas de seguridad, el manejo y destreza para realizar los diferentes montajes. Estos criterios se consideran dentro de la nota del pre informe.

El informe: Se entrega una semana después de haber realizado la práctica. Es un documento

escrito a mano. Existen varios formatos para reportar los resultados de un experimento, las secciones que se indican a continuación son aquellas que se encuentran en la mayoría de los artículos que se publican. A continuación, se presenta el formato para la elaboración del informe.

UNIDADES TECNOLÓGICAS DE SANTANDER

FORMATO INFORME DE LABORATORIO

<table>
<tr><th colspan="3">IDENTIFICACIÓN</th></tr>
<tr><td colspan="2">PRACTICA Nº: El número que identifica la práctica
NOMBRE DE LA PRÁCTICA: El título debe ser conciso, pero completo, en forma tal, que se entienda claramente el objeto del experimento. Por ejemplo: “Carga Específica del Electrón”.</td><td>FECHA:</td></tr>
<tr><td colspan="2">INTEGRANTES
NOMBRE: Los estudiantes que conforman el equipo de trabajo</td><td>CÓDIGO:</td></tr>
<tr><td colspan="2">NOMBRE:</td><td>CÓDIGO:</td></tr>
<tr><td colspan="2">NOMBRE:</td><td>CÓDIGO:</td></tr>
<tr><td>PROGRAMA: El programa académico que estudia. Por ejemplo: “Tecnología Electromecánica”</td><td>GRUPO: El Grupo de la asignatura. Por ejemplo: “A053, C064”
N° grupo: El grupo que lo identifica dentro del laboratorio. Es un número de 1 a 12.</td><td>DOCENTE:</td></tr>
</table>

RESUMEN

El resumen debe indicar la finalidad del experimento y presentar en forma breve pero muy clara en qué consiste la práctica realizada (máximo 5 renglones). **Nota:** Esta parte debe ser elaborada por el estudiante con sus propios análisis y argumentos.

TABLAS DE DATOS Y GRAFICAS

Los datos se refieren a aquellas cantidades que se derivan de mediciones y que se han de utilizar en el proceso de los cálculos. En esta sección se muestran los resultados obtenidos. Los gráficos y tablas que se muestren deben estar numerados y tener una leyenda o título, o sea, deben estar identificados.

Los resultados deben presentarse preferiblemente en forma de gráficos, sin embargo, si se requiere se hace necesario la inclusión de las tablas de datos. Los datos del experimento deben estar diferenciados de otros datos que puedan incluirse para comparación y tomados de otras fuentes.

Si en el laboratorio no se hacen mediciones, es decir, se basa en observaciones solamente, entonces se realizan las anotaciones a cerca del desarrollo de la experiencia.

EVALUACIÓN Y CALCULOS

En este espacio el estudiante responde al cuestionario propuesto por el docente. Este cuestionario le ayuda a obtener las conclusiones y a realizar el análisis de resultados de la experiencia. La evaluación debe ser contestada apoyándose en la bibliografía consultada y en la ejecución de la experiencia.

En esta parte también se deben mostrar las ecuaciones utilizadas y los cálculos realizados Todos los símbolos deben definirse en el momento en que aparecen por primera vez. Los resultados deben ser claros y precisos que indiquen lo que el estudiante pudo observar, no lo que los libros dicen, que se ha debido observar.

ANALISIS DE RESULTADOS Y/O ANALISIS DE GRAFICAS

En este espacio se describe la relación (contrastación) entre los resultados obtenidos en la práctica y la teoría expuesta en los libros de textos o en el aula de clases, si hay discrepancia respecto a los valores aceptados o esperados, se deben indicar las causas y algunas sugerencias que puedan mejorar el método experimental.

La discusión de resultados generalmente suele corresponder a un argumento lógico, basado en los resultados y no una repetición de estos. En ocasiones, puede ser útil, comparar los resultados obtenidos con los reportados en la literatura, mirar si hay discrepancia respecto a los valores aceptados o esperados, indicando las causas y algunas sugerencias que puedan mejorar el método experimental.

Otros aspectos a tratar son las dificultades encontradas durante la realización del experimento que hayan podido influir en los resultados, si son o no válidas las aproximaciones hechas, son entre otros, temas que también pueden tratarse como discusiones de resultados.

OBSERVACIONES

Se pretende realizar observaciones que mejoren la práctica o aquellos detalles de los cuales se percató cuando realizó la experiencia y que pueden ser importantes en la obtención de los resultados.

CONCLUSIONES
Debe hacerse una síntesis breve de los conocimientos verificados y de lo aprendido al cumplir con los objetivos de la práctica. De ninguna manera serán fragmentos copiados de textos o conclusiones extraídas de otras experiencias realizadas.
REFERENCIAS BIBLIOGRÁFICAS
Para reportar un libro texto, escriba en su orden y teniendo en cuenta los signos de puntuación, los siguientes elementos: Autor, título, número de edición, lugar de publicación, nombre de la editorial, año de publicaciones, paginación. Ejemplo: GROSSMAN, Stanley. INTRODUCCIÓN AL ALGEBRA LINEAL. Editorial Mc Graw Hill Interamericana. México. 512,5L269i

NOTA IMPORTANTE: El docente en la primera clase deberá socializar los criterios de evaluación planteados en el Plan de Asignatura y también, motivar y explicar la importancia de la puntualidad, la disciplina en el desarrollo de la práctica y del cumplimiento de las normas del laboratorio para alcanzar los resultados de aprendizaje previstos para la asignatura. Además, explicará a los estudiantes los formatos con un ejemplo elaborado. En los laboratorios que usen software, el docente debe capacitar a los estudiantes para que lo implementen de forma correcta en las prácticas.

Inasistencia: La inasistencia a una práctica de laboratorio, automáticamente descalifica el pre informe y el informe, por lo que se asume que no presenta ninguna de estas evidencias, obteniendo una nota de 0.0 (cero punto cero) en cada uno de ellos. Para recuperar una práctica

el estudiante debe presentar la incapacidad del médico de la EPS y el VoBo del Coordinador del Programa. Para presentar la excusa y recuperar la experiencia el estudiante cuenta con 8 días (una semana) contados a partir del día de la clase.

El parcial escrito: Se hace teniendo como referente los resultados de aprendizaje y las habilidades previstas en cada práctica de laboratorio. Se diseñan en los formatos de evaluación de las asignaturas teóricas. Se presenta de manera individual o grupal. Esta evaluación puede ser teórica, teórico – práctica, o una evaluación tipo problema con datos de laboratorio. Este examen tiene un valor del 40% del corte, se hace uno solo y comprende las prácticas realizadas antes de la fecha del parcial.

8.1.3. INICIACIÓN EN ARDUINO	
COMPETENCIA	**RESULTADOS DE APRENDIZAJE**
Conocer algoritmos básicos de programación para el control y configuración de puertos en un microcontrolador.	Explica los bloques funcionales del microcontrolador (memoria, procesador y puertos). Utiliza el lenguaje de programación para desarrollar aplicaciones que involucren el manejo de puertos.

8.1.3.1 Estructura de un programa

Los programas de Arduino se pueden dividir en tres partes principales: Estructura, Valores (variables y constantes) y Funciones.

8.1.3.1.1 Funciones

Una función es un bloque de código con un nombre y un conjunto de estamentos que son ejecutados cuando se llama a dicha función. Las funciones de usuario se crean para realizar tareas repetitivas reduciendo de esa manera el tamaño del programa.

Las funciones están asociadas a un tipo de valor "type", este es el valor que devolverá la función una vez termine su ejecución.

type nombreFuntión (parámetros)
{
Estamentos o instrucciones;
}

Si la función no devuelve ningún valor, se le asignará el tipo "Void" o función vacía. No es obligatorio pasarle parámetros a la función, se pueden crear funciones independientes que obtengan sus propios parámetros para trabajar.

8.1.3.1.2 Void Setup

El setup es la primera función en ejecutarse dentro de un programa en Arduino. Es, básicamente, donde se setean las funciones que llevará a cabo el microcontrolador.

Aquí es donde establecemos algunos criterios que requieren una ejecución única. Por ejemplo, si nuestro programa va a usar comunicación serial, en el setupestablecemos el comando Serial.begin para indicarle al programa que vamos a iniciar la comunicación serial.

Si vamos a utilizar un pin determinado como salida de voltaje, usamos el pinMode para indicarle a Arduino que determinado pin funcionará como salida, usando el parámetro OUTPUT.

El concepto es ese: colocamos aquello que debemos ejecutar una sola vez. Creo que no es sensato que coloquemos el Serial.begin en el loop donde se va a hacer lo mismo una y otra vez. Aquello es algo que, con que suceda una sola vez, es suficiente.

8.1.3.1.3 Void Loop

Loop en inglés significa lazo o bucle. La función loop en Arduino es la que se ejecuta un número infinito de veces. Al encenderse el Arduino se ejecuta el código del setup y luego se entra al loop, el cual se repite de forma indefinida hasta que se apague o se reinicie el microcontrolador.

8.1.3.2 Llaves

Las llaves definen el principio y el final de un bloque de instrucciones. Se usan para delimitar el inicio y fin de funciones como *setup ()* o para delimitar el alcance de los bucles y condicionales del programa.

```
Función ()
{
Estamentos o instrucciones;
}
```

8.1.3.3 Punto y Coma

El punto y coma ";" se utiliza para separar instrucciones en el lenguaje de programación de Arduino. También se utiliza para separar elementos en una instrucción de tipo "bucle for".

Cualquier tipo de instrucción que se utilice para programar un sensor en Arduino deberá terminar con un punto y como si no automáticamente mandara un error el compilador.

8.1.3.4 Línea de comentarios //

Las líneas de comentarios tienen la misma función que los bloques de comentarios, la única diferencia es que las líneas de comentarios

suelen usarse para comentar instrucciones ya que solo afectan a una línea.

8.1.3.5 Variables

Las variables son elementos donde se almacenan valores numéricos que serán usados por el programa. Como su nombre indica, las variables van a cambiar de valor con la evolución del programa y nos van a permitir crear la lógica del programa en función de estos cambios.

int variable_entrada = 0; //declara una variable y le asigna el valor 0
variable_entrada = analogRead (2); //la variable toma el valor de la entrada analógica 2

¿Y que es int? Es Los int (enteros) son el tipo de datos primario para el almacenamiento de números.

En el Arduino Uno (y otras placas basadas en el ATMega) un int almacena un valor de 16-bit (2-byte). Esto produce un rango de -32,768 to 32,767 (valor mínimo de -2^15 y un valor máximo de (2^15) - 1).

Hay más rangos y estos se ilustrarán más adelante.

8.1.3.5.1 Utilización de Variables

Las variables pueden ser declaradas en diferentes lugares del programa, en función del lugar donde sean declaradas, las variables van a ser de tipo global o local, determinando esto el ámbito de aplicación o la capacidad de ciertas partes del programa para hacer uso de lasmismas.

Una variable global es aquella que puede ser vista y utilizada por cualquier función y estamento de un programa. Las variables globales se declaran al comienzo del programa, antes de la función *setup()*.

Una variable local es aquella que se define dentro de una función o como parte de un bucle. Solo será visible y podrá utilizarse dentro de la función o bucle donde es declarada.

La existencia de variables globales y locales, permite el uso de variables con el mismo nombre en partes diferentes del programa, estas variables podrán almacenar valores distintos sin que se produzca ningún conflicto, ya que a las

variables locales solo tendrán acceso las funciones donde se declaren dichas variables.

8.1.3.5.2 Tipos de Variables

Los datos que guardamos en las variables pueden ser de diferentes tipos, vamos a listar algunos de ellos.

- **Char**, se utilizan para almacenar caracteres, ocupan un byte.
- **Byte**, pueden almacenar un número entre 0 y 255.
- **in**t, ocupan 2 bytes (16 bits), y por lo tanto almacenan número entre 2^{-15} y 2^{15}-1, es decir, entre -32,768 y 32,767.
- **unsigned int**, ocupa también 2 bytes, pero al no tener signo puede tomar valores entre 0 y 2^{16}-1, es decir entre 0 y 65,535.
- **Long**, ocupa 32 bits (4 bytes), desde -2,147,483,648 a 2,147,483,647.
- **float**, son números decimales que ocupan 32 bits (4 bytes). Pueden tomar valores entre -3.4028235E+38 y +3.4028235E+38.
- **double**, también almacena números decimales, pero disponen de 8-bytes (64 bit).

Siempre que elegimos un tipo de dato debemos escoger el que menos tamaño necesite y que cubra nuestras necesidades, ya que ocuparán espacio en la memoria de nuestra placa y podría darse el caso de que nuestro programa requiera más memoria de la disponible.

8.1.3.6 Constantes

La diferencia fundamental entre una variable y una constante es que la constante va a ser un valor de solo lectura que no va a poder ser modificado con la evolución del programa.

Si queremos definir una constante, podemos hacerlo de manera muy similar a como definíamos las variables, tan solo hay que indicar al programa que se trata de un valor "no modificable", para ello hay que

añadir antes del "tipo" la palabra "*const*", que va a hacer que la variable sea de solo lectura.

const float pi = 3.1415;*//crea una constante de tipo "float" y le asigna el valor 3.1415*

Es posible crear constantes usando *#define*, esta sentencia va a nombrar las constantes antes de ser compiladas por Arduino. Las constantes creadas de esta manera no van a ocupar memoria de programa, ya que el compilador va a remplazar estas constantes con el valor definido en el momento de compilar.

#define pi 3.1415 *//crea una constante usando #define* a *la que le asigna el valor 3.1415*

Hay que tener cuidado a la hora de usar *#define*, ya que la sintaxis es diferente, no hay que poner el operador de asignación "=" ni cerrar sentencia con el punto y coma ";".

Usar #define puede ser problemático, ya que si existe en cualquier parte del programa alguna constante o variable con el mismo nombre que tengamos asignado en #define, el programa lo va a sustituir con el valor que tengamos asignado a esta constante, por ello se debe tener cuidado a la hora de asignar constantes con *#define* y no crear constantes muy genéricas cómo*#define x 10, ya* que "x" es muy común usarlo como variable dentro de alguna función o bucle.

8.1.3.7 Estructuras de control

En programación, las estructuras de control permiten modificar el flujo de ejecución de las instrucciones de un programa.

Con las estructuras de control se puede:

- De acuerdo con una condición, ejecutar un grupo u otro de sentencias (If-Then-Else)

- De acuerdo con el valor de una variable, ejecutar un grupo u otro de sentencias (Select-Case)

- Ejecutar un grupo de sentencias mientras se cumpla una condición (Do-While)

- Ejecutar un grupo de sentencias hasta que se cumpla una condición (Do-Until)

- Ejecutar un grupo de sentencias un número determinado de veces (For-Next)

Todos los lenguajes de programación modernos tienen estructuras de control similares. Básicamente lo que varía entre las estructuras de control de los diferentes lenguajes es su sintaxis; cada lenguaje tiene una sintaxis propia para expresar la estructura.

8.1.3.7.1 Estructuras de decisión

if: es un estamento que se utiliza para probar si una determinada condición se ha alcanzado, como por ejemplo averiguar si un valor analógico está por encima de un cierto número, y ejecutar una serie de declaraciones (operaciones) que se escriben dentro de llaves, si es verdad. Si es falso (la condición no se cumple) el programa salta y no ejecuta las operaciones que están dentro de las llaves.

if… else: viene a ser una estructura que se ejecuta en respuesta a la idea “si esto no se cumple haz esto otro”. Por ejemplo, si se desea probar una entrada digital, y hacer una cosa si la entrada fue alta o hacer otra cosa si la entrada es baja.

else: puede ir precedido de otra condición de manera que se pueden establecer varias estructuras condicionales de tipo unas dentro de las otras (anidamiento) de forma que sean mutuamente excluyentes pudiéndose ejecutar a la vez. Es incluso posible tener un número ilimitado de estos condicionales. Recuerde sin embargo

que sólo un conjunto de declaraciones se llevará a cabo dependiendo de la condición probada.

switch..case: Al igual que if, switch..case controla el flujo del programa especificando en el programa que código se debe ejecutar en función de unas variables. En este caso en la instrucción switch se compara el valor de una variable sobre los valores especificados en la instrucción case.
break es la palabra usada para salir del switch. Si no hay break en cada case, se ejecutará la siguiente instrucción case hasta que encuentre un break o alcance el final de la instrucción.

default es la palabra que se usa para ejecutar el bloque en caso que ninguna de las condiciones se cumpla.

8.1.3.7.2 Estructuras de repetición

for: La declaración for se usa para repetir un bloque de sentencias encerradas entre llaves un número determinado de veces. Cada vez que se ejecutan las instrucciones del bucle se vuelve a testear la condición. La declaración for tiene tres partes separadas por (;). La inicialización de la variable local se produce una sola vez y la condición se testea cada vez que se termina la ejecución de las instrucciones dentro del bucle. Si la condición sigue cumpliéndose, las instrucciones del bucle se vuelven a ejecutar. Cuando la condición no se cumple, el bucle termina.

Cualquiera de los tres elementos de cabecera puede omitirse, aunque el punto y coma es obligatorio. También las declaraciones de inicialización, condición y

expresión puede ser cualquier estamento válido en lenguaje C sin relación con las variables declaradas.

while: Un bucle del tipo while es un bucle de ejecución continua mientras se cumpla la expresión colocada entre paréntesis en la cabecera del bucle. La variable de prueba tendrá que cambiar para salir del bucle. La situación podrá cambiar a expensas de una expresión dentro el código del bucle o también por el cambio de un valor en una entrada de un sensor.

do..while: El bucle do while funciona de la misma manera que el bucle while, con la salvedad de que la condición se prueba al final del bucle, por lo que el bucle siempre se ejecutará al menos una vez.

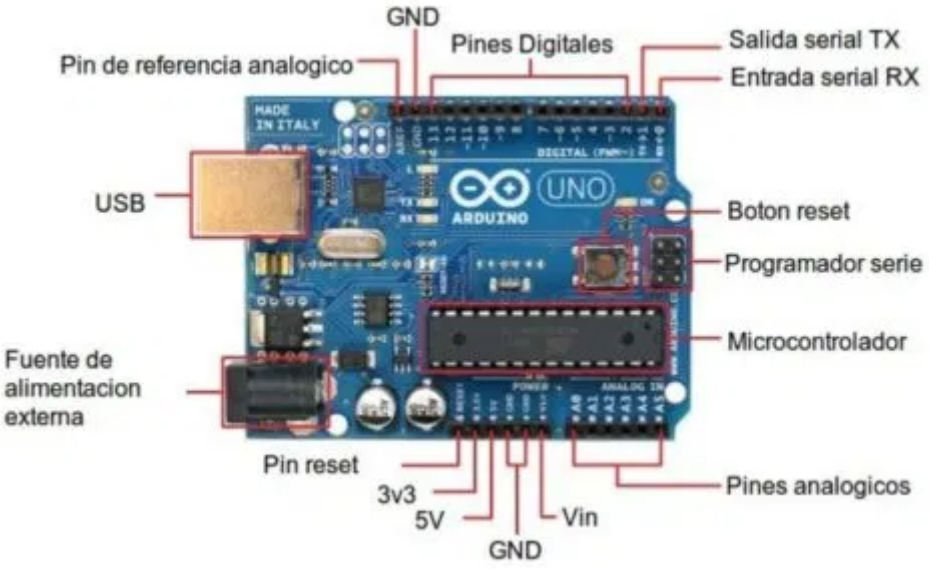
goto: transfiere el flujo de programa a un punto del programa que está etiquetado.

break: se usa en las instrucciones do, for, while para salir del bucle de una forma diferente a la indicada en el bucle.

continue: se usa en las instrucciones do, for, while para saltar el resto de las instrucciones que están entre llaves y se vaya a la siguiente ejecución del bucle comprobando la expresión condicional.

8.1.3.8 Entradas y salidas E/S

Figura37: Entradas y salidas del Arduino uno



De acuerdo a la Figura 37 se ilustra cada componente y característica de la placa Arduino 1

Fuente: (Mecafenix, 2020)

8.1.3.8.1 Entradas y salidas digitales

Los sistemas digitales, como por ejemplo un microcontrolador, usan la lógica de dos estados representados por dos niveles de tensión eléctrica, uno alto, H y otro bajo, L (de High y Low, respectivamente, en inglés). Por abstracción, dichos estados se sustituyen por ceros y unos, lo que facilita la aplicación de la lógica y la aritmética binaria. Si el nivel alto se representa por 1 y el bajo por 0, se habla de lógica positiva y en caso contrario de lógica negativa.

En Arduino para tratar las entradas y salidas digitales usamos las siguientes funciones:

- **pinMode()** – configura en el pin especificado si se va a comportar como una entrada o una salida.

- **digitalWrite()** – Escribe un valor HIGH o LOW en el pin digital especificado. Si el pin está configurado como OUTPUT pone el voltaje correspondiente en el pin seleccionado. Si el pin está configurado como INPUT habilita o deshabilita la resistencia interna de pull up del correspondiente pin.

- **digitalRead()** – lee el valor del pin correspondiente como HIGH o LOW.

8.1.3.8.2 Entradas y salidas analógicas

Una señal eléctrica analógica es aquella en la que los valores de la tensión o voltaje varían constantemente y pueden tomar cualquier valor.

Un sistema de control (como un microcontrolador) no tiene capacidad alguna para trabajar con señales analógicas, de modo que necesita convertir las señales analógicas en señales digitales para poder trabajar con ellas.

La señal digital obtenida de una analógica tiene dos propiedades fundamentales: Valores. Que valoren voltios define 0 y 1. En nuestro caso es tecnología TTL (0 – 5V)

Resolución analógica: nº de bits que usamos para representar con una notación digital una señal analógica:

En el caso de un Arduino Uno, el valor de 0 voltios analógico es expresado en digital como B0000000000 (0) y el valor de 5V analógico es expresado en digital como B1111111111 (1023). Por lo tanto, todo valor analógico intermedio es expresado con un valor entre 0 y 1023, es decir, sumo 1 en binario cada 4,883 mV.

Arduino Uno tiene una resolución de 10 bits, es decir, unos valores entre 0 y 1023.

Arduino Due tiene una resolución de 12 bits, es decir, unos valores entre 0 y 4095.

8.1.3.8.2.1 Salidas analógicas PWM

Arduino Uno tiene entradas analógicas que gracias a los conversores analógico digital puede entender ese valor el microcontrolador, pero no tiene salidas analógicas puras y para solucionar esto, usa la técnica de PWM.

Las Salidas PWM (Pulse Width Modulation) permiten generar salidas analógicas desde pines digitales. Arduino Uno no posee salidas analógicas puras.

La modulación por ancho de pulsos (también conocida como PWM, siglas en inglés de pulse-Width Modulation) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica (una sinodal o una cuadrada, por ejemplo), ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación con el período.

duty cycle = (tiempo que la salida está a uno o HIGH) / (periodo de la función)

En Arduino la frecuencia de PWM es de 500Hz. Pero es un valor que puede modificarse en caso que lo necesitemos.

8.1.3.8.2.2 Funciones de I/O analógicas en Arduino

Los microcontroladores de Arduino contienen en la placa un conversor analógico a la digital de 6 canales. El conversor tiene una resolución de 10 bits, devolviendo enteros entre 0 y 1023. Los pines analógicos de Arduino también tienen todas las funcionalidades de los pines digitales. Por lo tanto, si necesitamos más pines digitales podemos usar los pines analógicos. La nomenclatura para los pines analógicos es A0, A1, etc.

En Arduino los pines analógicos se definen y tienen las propiedades siguientes:

En Arduino para tratar las entradas y salidas digitales usamos las siguientes funciones:

- analogReference() – configura la referencia de voltaje usada para la entrada analógica.

- analogRead() – lee el valor del pin analógico especificado.

- analogWrite() – escribe un valor analógico (onda PWM) al pin especificado. No en todos los pines digitales se puede aplicar PWM.

8.1.3.9 Puerto serie

El microcontrolador Arduino tiene la una serie de pines con la capacidad de comunicarse con otros dispositivos a través de la comunicación serial. A través de esta comunicación serie también podemos hacer conexión con el ordenador por medio de USB gracias a chip FDTI que incorpora la placa Arduino la cual realiza la conversión USB-serie.

La capacidad de interactuar con diferentes dispositivos hace de Arduino una herramienta muy potente ya que podemos llevar a cabo proyectos complejos con muchos dispositivos y que estos interactúen entre sí.

8.1.3.9.1 Inicialización de la comunicación serie Serial.begin

Para poder utilizar el puerto serie hay que inicializarlo estableciendo la velocidad de la conexión. Esta inicialización se hace siempre dentro de lafunción*setup()* .
Un valor típico para realizar la conexión es9600baudios , aunque se pueden asignar otros muchosvalores.

```
Void setup ()
{
Serial.begin (9600);//abre el puerto serie estableciendo la velocidad en 9600
}       //baudios
```

Según el modelo de Arduino que estemos usando, puede tener 1 o más puertos para la comunicación serie, estos puertos estarán numerados y deberán de abrirse de manera independiente según los que queramos usar.

En el Arduino MEGA se disponen de 4 puertos para la conexión serie, si se desean abrir los 4 puertos el programa quedará de la siguiente manera:

```
Void setup ()
{
Serial.begin(9600);

Serial1.begin(9600);

Serial2.begin(9600);

Serial3.begin(9600);
}
```

Debe de tenerse en cuenta que, si se inicializa la comunicación serie, los pines asociados al puerto serie que estemos utilizando no podrán ser usados para otro propósito.

8.1.3.9.2 Escritura en el puerto serie Serial.print

Si queremos que Arduino muestre información a través del puerto serie, debemos de usar instrucciones que "impriman" en pantalla dicha información.

Para imprimir estos datos se usa el comando *Serial.print,* que mandará a través del puerto serie el dato o la cadena de caracteres que le indiquemos. Esta instrucción tiene algunas variantes, que veremos a continuación.

Serial.print(dato, tipo de dato)

Esta es la instrucción más común a la hora de enviar datos a través del puerto serie, tan solo hay que indicar el dato que queremos enviar y el formato en el que queremos que muestre dicho dato.

*nota: el tipo de dato es un campo opcional, si no le indicamos ningún tipo, mostrará el dato en formato decimal.

El "tipo de dato" puede tomar los valores BIN (binario), OCT (octal), DEC (decimal) y HEX (hexadecimal). En versiones antiguas de Arduino, también estaba disponible el sacar los datos en formato "BYTE", pero este formato fue eliminado, si queremos sacar un byte en pantalla, podemos utilizar la función

Serial.write(valor).

Serial.print(78,BIN); // manda el dato "1001110"

Serial.print(78, OCT); // manda el dato "116"

Serial.print(78, DEC); // manda el dato "78"

Serial.print(78, HEX); // manda el dato"4E"

Si estamos trabajando con datos que tienen decimales (*float*) y queremos mostrar un número concreto de ellos, podemos hacerlo poniendo en tipo de dato el número de decimales que queremos mostrar. Por defecto mostrará dos decimales.

Serial.println(1.23456, 0); *I/manda por el puerto serie el valor "1"*

Serial.println(1.23456, 1);*//manda por el puerto serie el valor "1.2"*

Serial.println(1.23456, 2);*//manda por el puerto serie el valor "1.23"*

Serial.println(1.23456, 3);*//manda por el puerto serie el valor "1.234"*

Serial.println(1.23456, 4);*//manda por el puerto serie el valor "1.2346"*

Además de mostrar datos en pantalla, también es posible mandar cadenas de caracteres, para ello tan solo hay que encerrar el texto que queramos mostrar entre comillas.

Serial.print("Hola mundo"); //muestra en pantalla "hola mundo"

Cuando mandamos datos a través del puerto serie y queremos visualizar estos datos en pantalla, es recomendable introducir espacios y saltos de línea, ya que si no lo hacemos los datos nos van a aparecer de manera continua y no vamos a poder diferenciar unos de otros. Para ordenar estos datos, podemos introducir tabulaciones o saltos de línea con los siguientes comandos:

Serial.print("\t");*//introduce una tabulación entre los datos*

Serial.print("\n");*//introduce un salto de línea*

Si queremos crear datos en líneas diferentes, se puede optar por una variante del *Serial.print* que introduce automáticamente un salto de línea, haciendo que el siguiente dato que se vaya a escribir aparezca en la siguiente línea.

Serial.println(dato, tipo de dato)

Como en el caso anterior, el tipo de dato será un campo opcional, si no se rellena este campo, el dato aparecerán en formato decimal.

La función *Serial.println* es equivalente a poner:

Serial.print (dato, tipo de dato); Serial.print ("\n");

8.1.3.10 Otras instrucciones de interés

El lenguaje utilizado por Arduino, como otros lenguajes de programación cuenta con diversas instrucciones básicas que dan fácil resolución a diversas situaciones presentadas a la hora de la programación, se mostrará algunas de las instrucciones básicas que trae Arduino por defecto, pero no siendo éstas las únicas ya que por medio de librerías podremos añadir más funciones, como librerías de matemáticas y demás.

8.1.3.10.1 Delay

Esta instrucción detiene la ejecución del programa un tiempo determinado. El tiempo habrá que indicarlo entre paréntesis en milisegundos (1 segundo = 1000 milisegundos).

Hay que tener mucho cuidado cuando se usa esta función, ya que se trata de una función bloqueante. Esta función va a detener el flujo de programa, haciendo que durante este tiempo no se detecten eventos como pueden ser presionar un pulsador o la activación de un sensor, esto puede ocasionar graves problemas en la aplicación, por lo que siempre que se pueda hay que evitar usar la instrucción *delay ()*.

8.1.3.10.2 Delay microsegundos

La instrucción *delayMicroseconds ()* funciona igual que *delay ()*, con la diferencia de realizar las temporizaciones usando microsegundos (µs) en lugar de milisegundos (ms).

Esta instrucción nos va a permitir realizar temporizaciones mucho menores, haciendo que el tiempo que detenemos el flojo de programa sea prácticamente imperceptible.

delayMicroseconds (10);*//espera durante 0'00001 segundos*

8.1.3.10.3 Min (x, y)

La instrucción *min (x, y)* va a comparar dos valores devolviendo el menor de ellos.

dato = min (sensor, 100);*//dato toma el valor más pequeño entre "sensor" o 100*

En el ejemplo anterior "dato" va a tomar el valor del sensor siempre que este valga menos de 100, si es mayor tomará el valor 100. Esa instrucción puede usarse para limitar superiormente el valor de “dato".

8.1.3.10.4 Max (x, y)

La instrucción *max (x, y)* va a comparar dos valores devolviendo el mayor de ellos.

dato =max (sensor, 100);

Esta instrucción funciona igual que *min (x, y)*, la diferencia es que va a entrega el mayor de los dos números, esto puede ser utilizado para inferiormente el valor que tomará la variable “dato".

BIBLIOGRAFÍA

(s.f.). Obtenido de Aprendiendo Arduino: https://aprendiendoarduino.wordpress.com/2017/06/20/estructuras-de-control-3/

(18 de 01 de 2020). Obtenido de Descubre Arduino: https://descubrearduino.com/estructura-de-programa/

Aguilar, & Marquez. (s.f.). *Ambientes de aprendizaje en arduino.* Obtenido de http://www.somedicyt.org.mx/simposio/images/docs/simposio/2015/memorias/ambientes-de-aprendizaje-para-la-ciencia-usando-tecnologia-arduino.pdf

Aprendiendo Arduino. (s.f.). Obtenido de https://aprendiendoarduino.wordpress.com/2017/10/14/estructuras-de-control-4/

Arduino. (s.f.). Obtenido de Variables: https://www.arduino.cc/en/Tutorial/Variables

Arduino. (21 de febrero de 2019). Obtenido de https://www.arduino.cc/reference/en/language/variables/data-types/unsignedint/

Arduino Bolivia. (2018). Introduccion a la plataforma arduino. *Arduino Bolivia*, 26.

Barragan, H. (7 de Julio de 2011). *Wiring*. Obtenido de http://wiring.org.co/reference/es/Serial_println_.html

Crespo, M. D. (s.f.). *Blogger*. Obtenido de Arduino en español: http://manueldelgadocrespo.blogspot.com/p/ifelse.html

Edmundo, R. (2017). *Uso de Arduino en Programación Electrónica.* Obtenido de http://ria.utn.edu.ar/bitstream/handle/123456789/1835/LTE_%20Riveros%2C%20Gabriel.pdf?sequence=1&isAllowed=y

González, A. G. (27 de Noviembre de 2019). *Panamahitek*. Obtenido de http://panamahitek.com/la-velocidad-de-la-comunicacion-serial-en-arduino/

Herrero, & Sanchez. (Julio de 2015). *Researchgate.* Obtenido de Universidad Alfonso X el Sabio: https://www.researchgate.net/publication/320531618_Una_mirada_al_mundo_Arduino

panamahitek. (s.f.). Obtenido de http://panamahitek.com/el-setup-y-el-loop-en-arduino/

Programar facil. (s.f.). Obtenido de https://programarfacil.com/blog/arduino-blog/if-else-arduino/

Wikipedia. (23 de Septiembre de 2019). Obtenido de https://es.wikipedia.org/wiki/Arduino

Wikipedia. (23 de 09 de 2019). *Arduino*. Obtenido de https://es.wikipedia.org/w/index.php?title=Arduino&oldid=119633331

8.1.4. Practica 1: Encendiendo una fila de leds

COMPETENCIA	RESULTADOS DE APRENDIZAJE
Conocer algoritmos básicos de programación para el control y configuración de puertos en un microcontrolador.	Explica los bloques funcionales del microcontrolador (memoria, procesador y puertos). Utiliza el lenguaje de programación para desarrollar aplicaciones que involucren el manejo de puertos.

MATERIALES
• Arduino uno • Protoboard • 8 resistencias de 220 • 8 leds • cables

Arduino es una herramienta muy fuerte ya que con ella podemos hacer muchos artículos con muchas funciones prácticas, con Arduino podemos controlar, automatizar, monitorear etc., procesos o aplicaciones que nos podrán ayudar con las funciones que deseemos optimizar.

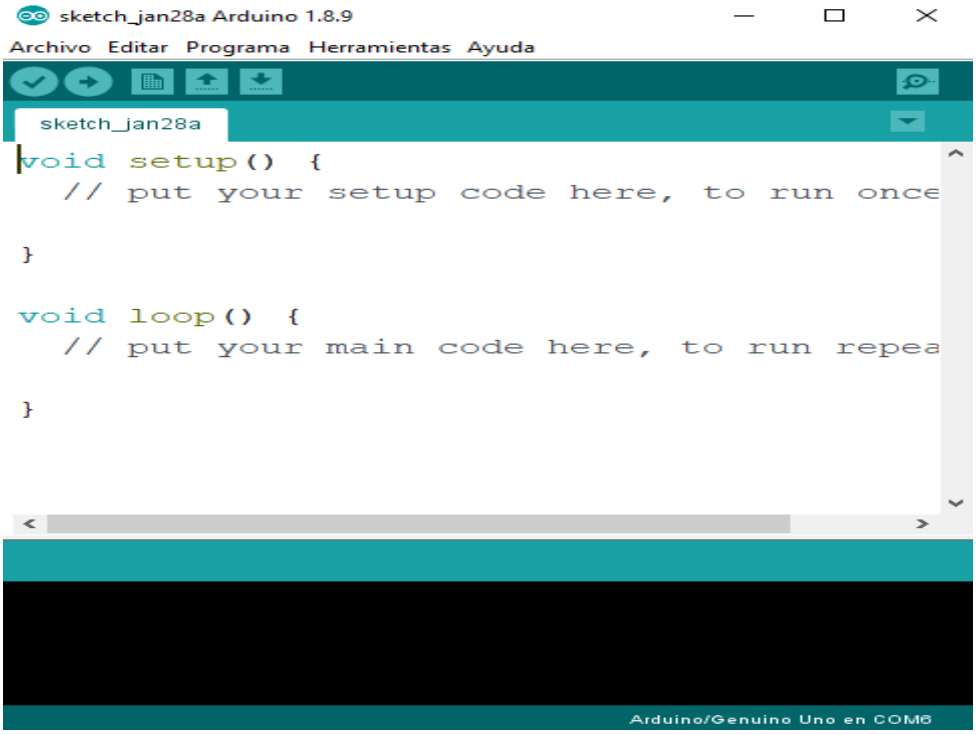
Iremos desde un nivel básico como encender un led, hasta controlar servomotores, motores de paso, DC y demás.

Empezaremos primero con la programación y después con su respectivo montaje.

Primero que todo abriremos el programa Arduino, que podrán descargar en el siguiente link: https://www.arduino.cc/en/Main/Software

Ya abierto el programa de Arduino se verá de esta manera:

Figura38:Sketch practica 1



Interfaz de Arduino ide.

Fuente: (Autor, 2020)

Como podemos ver, en el entorno Arduino muestra su estructura Void setup y Void loop explicadas anterior mente, en este espacio donde está Void setup y Void loop podremos escribir nuestras líneas de código que nos servirán para programar nuestro proceso.

Entonces empecemos, como en todo programa empezamos con las variables.

Figura39: Sketch practica 1

```
int tiempo = 90;

int led1 = 2;
int led2 = 3;
int led3 = 4;
int led4 = 5;
int led5 = 6;
int led6 = 7;
int led7 = 8;
int led8 = 9;
```

Declaración de variables led1 a led8.

Fuente: (Autor, 2020)

Primero que todo declaramos una variable **int** este tipo de variable puede almacenar un numero de valor entre -32769 y 32767 y nos servirá tanto como para crear la variable tiempo como para declarar las variables “led” igual a sus respectivos pines.

Después configuramos los puertos usados como salida con la opción **pinMode** que permite configurar a cada pin, de forma individual, como entrada o como salida.

Figura40: Sketch practica 1

```
void setup()
{
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);
  pinMode(led5, OUTPUT);
  pinMode(led6, OUTPUT);
  pinMode(led7, OUTPUT);
  pinMode(led8, OUTPUT);
}
```

Declaración de pines como salidas.

Fuente: (Autor, 2020)

Ahora entraremos en si con la secuencia de luces, para esto utilizaremos el ciclo for que generalmente siempre se programa en la seccion void loop ya que Es el lugar donde tenemos que poner los comandos que se ejecutarán mientras la placa Arduino esté habilitada y comenzará con el primer comando, el microcontrolador irá hasta el final y saltará inmediatamente al principio para repetir la misma secuencia.

El ciclo for se divide en 3 partes las cuales son: índice, condición e incremento, cada una separada por una coma de esta manera **for** (índice; condición; incremento).

El índice es una variable que ira almacenando el número de veces que se repita el ciclo y como es una variable se puede llamar como se desee.

La condición es la pauta que se indicará para que se siga ejecutando el ciclo o haga terminar el bucle y que el resto de programa siga ejecutándose a continuación.

El incremento es el que modificará el índice en cada repetición del bucle.

Yendo a la práctica lo vemos de esta manera.

Figura41: Sketch practica 1

```
void loop()
{
  for(byte s1 = 0; s1 < 12; s1++)
  {
    digitalWrite(led1, HIGH);
    delay(tiempo);
    digitalWrite(led1, LOW);

    digitalWrite(led2, HIGH);
    delay(tiempo);
    digitalWrite(led2, LOW);

    digitalWrite(led3, HIGH);
    delay(tiempo);
    digitalWrite(led3, LOW);

    digitalWrite(led4, HIGH);
    delay(tiempo);
```

De acuerdo a la figura 41 se implementa un ciclo for y se declaran estados altos y bajos a los diferentes pines.

Fuente: (Autor, 2020)

Figura42: Sketch practica 1

```
    digitalWrite(led5, LOW);

    digitalWrite(led6, HIGH);
    delay(tiempo);
    digitalWrite(led6, LOW);

    digitalWrite(led7, HIGH);
    delay(tiempo);
    digitalWrite(led7, LOW);

    digitalWrite(led8, HIGH);
    delay(tiempo);
    digitalWrite(led8, LOW);
}
```

Declaración de estados altos y bajos a los diferentes pines.

Fuente: (Autor, 2020)

Lo cual evidenciamos que **byte**se puede usar para guardar valores enteros desde 0 hasta 255 s1 como indicador de que comienza desde 0 y s2 para indicar que va hasta valores menores que 12 para que se cumpla las condiciones y el s1++ indica con ++ que incrementamos s1 en uno, sería equivalente a s1=s1+1 Mientras se cumpla la condición, se ejecutarán las líneas de código que hay entre las llaves a continuación, e incrementara el s1 en 1 como lo hemos programado.

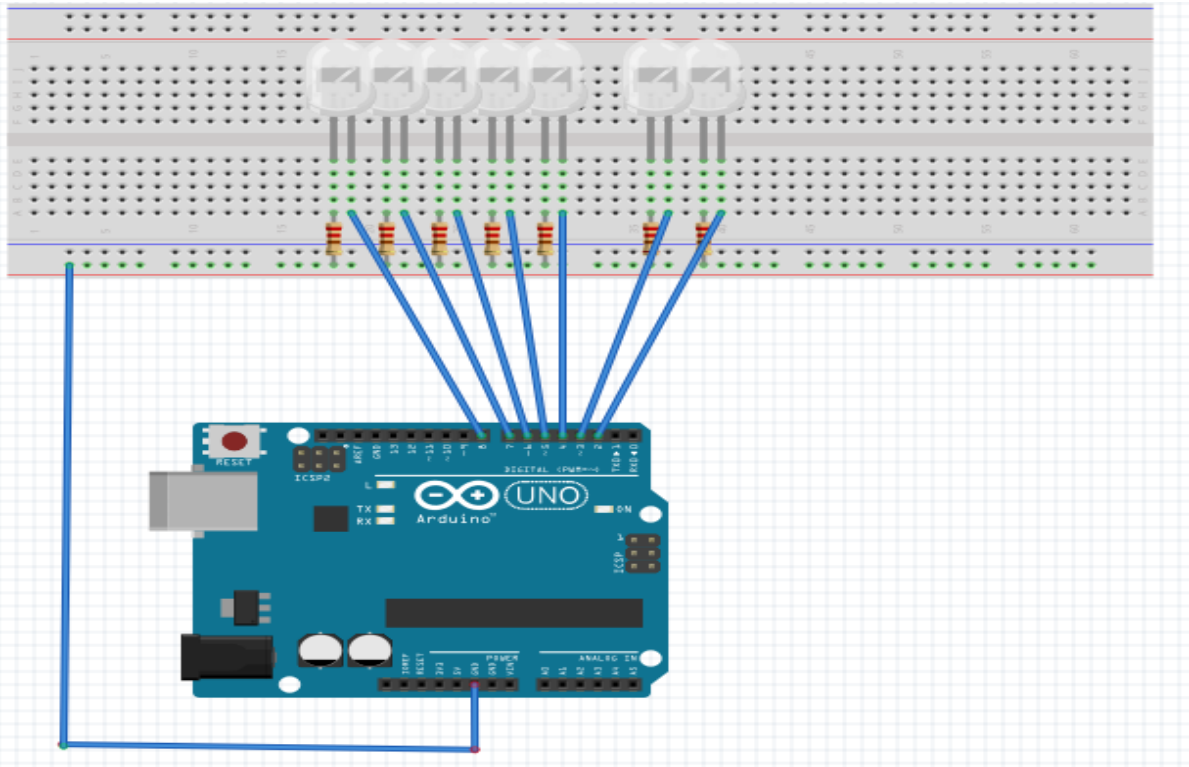
Con las sucesivas pasadas del bucle, se irá incrementando la variable s1 hasta que ya no se cumpla la condición, porque el valor de la variable s1 será 13, el bucle no se ejecutará, saltará la parte incluida entre llaves y continuará la ejecución del resto del programa.

Con la opción digitalWrite se le indica que se apague o encienda y con la opción Delay el tiempo en que cada led estará encendido con la variable tiempo que ya habíamos indicado desde el principio de la línea de código.

Procederemos al ensamblaje en Arduino-Protoboard utilizando la herramienta Fritzing para ser más ilustrativo el montaje.

Siguiendo el conexionado de los pines del Arduino al protoboard, de la GND al protoboard podremos lograr el ensamblaje de la práctica.

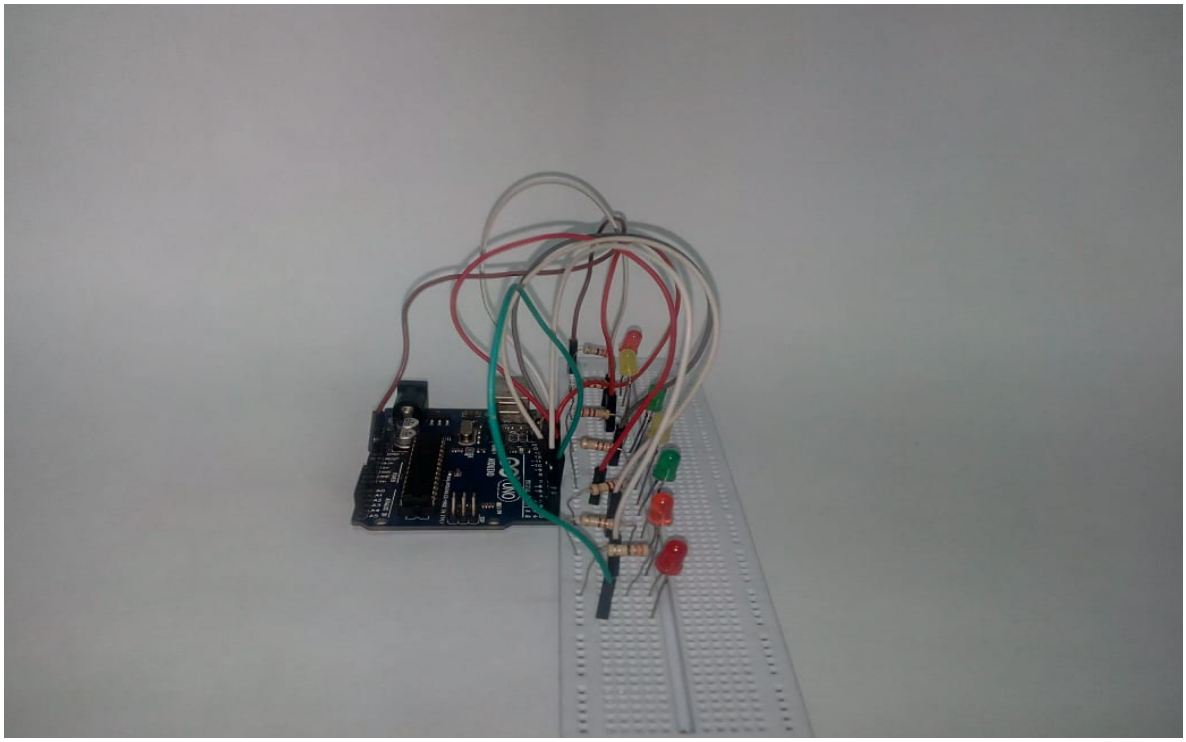
Figura43: Diagrama de practica 1



De acuerdo a la figura 43podemos observar la conexión entre Arduino, Protoboard, Resistencia y Leds.

Fuente: (Autor, 2020)

Figura44: Montaje practica 1



Vista real de como se ve el diagrama de conexiones de la figura 43.

Fuente: (Autor, 2020)

BIBLIOGRAFÍA

(s.f.). Obtenido de Aprendiendo Arduino: https://aprendiendoarduino.wordpress.com/2017/06/20/estructuras-de-control-3/

(18 de 01 de 2020). Obtenido de Descubre Arduino: https://descubrearduino.com/estructura-de-programa/

Acurio, J. (2020). *SlideShare.* Obtenido de https://es.slideshare.net/mobile/Jaime_hernan/ciclo-for

Aprendiendo Arduino. (s.f.). Obtenido de

https://aprendiendoarduino.wordpress.com/2017/10/14/estructuras-de-control-4/

Arduino. (s.f.). Obtenido de Variables: https://www.arduino.cc/en/Tutorial/Variables

Arduino. (21 de febrero de 2019). Obtenido de https://www.arduino.cc/reference/en/language/variables/data-types/unsignedint/

panamahitek. (s.f.). Obtenido de http://panamahitek.com/el-setup-y-el-loop-en-arduino/

Wikipedia. (23 de 09 de 2019). *Arduino*. Obtenido de https://es.wikipedia.org/w/index.php?title=Arduino&oldid=119633331

8.1.5. Practica 2: Semáforo.

COMPETENCIA	RESULTADOS DE APRENDIZAJE
Conocer algoritmos básicos de programación para el control y configuración de puertos en un microcontrolador. Diseñar aplicaciones que involucren la generación y control de tiempos.	Utiliza el lenguaje de programación para desarrollar aplicaciones que involucren el manejo de puertos. Desarrolla aplicaciones en las cuales se implementan circuitos de control de tiempos.

MATERIALES

- Arduino uno
- Protoboard
- 6 resistencias de 220
- 6 leds
- cables

Programaremos un semáforo de vehículos y uno de peatones los cuales trabajaran juntos.

Iniciaremos explicando su código y el porqué de este.

Empezamos con # define, la cual nos ayuda a poner un nombre a un valor que es constante antes de compilar el programa.

En esta ocasión empezamos definiendo los colores de los semáforos los cuales serán amarillo, verde y rojo. El verde lo asignaremos con el pin 2, el amarillo con el pin 3 y el rojo con el pin 4.

Después en el voidsetup que es donde se setean la funciones que llevara a cabo el microcontrolador, con pinMode definimos que verde, amarillo y rojo que vendrían siendo el pin 2, 3 y 4 como salidas de voltaje.

Se vería así:

Figura45: Sketch practica 2

```
# define verde     2
# define amarillo  3
# define rojo      4
void setup() {
  pinMode (verde,    OUTPUT);
  pinMode (amarillo, OUTPUT);
  pinMode (rojo,     OUTPUT);
}
```

De acuerdo a la figura 45 se definen los colores con sus respectivos pines y se declaran de salida.

Fuente: (Autor, 2020)

Despues entramos en el ciclo repetitivo que seria el voidloop. Empezamos con digitalWrite, que permite escribir valores logicos digitales en un pin de salida que ya debe estar definido ybeso ya se hizo.

Escribo en digitalWrite (rojo, true); lo cual quiere decir como rojo encendido, le damos una duracion con delay de 5 segundos.

Despues volvemos a escribir en digitalWrite (rojo, false); para que este se apague.

Asi hacemos con el amarillo y con el verde, con el fin de que cada 5 segundos se encienda uno y se apague el anterior.

La programacion se vería de esta manera.

Figura46: Sketch practica 2

```
void loop() {
      //:::::::::::: ROJO::::::::::
   digitalWrite(rojo,     true);
   delay         (5000);
      digitalWrite(rojo,     false);

      //:::::::::::: AMARILLO:::::::
   digitalWrite(amarillo,true);
   delay         (5000);
       digitalWrite(amarillo,false);

     //:::::::::::: VERDE::::::::::
     digitalWrite(verde,    true);
   delay         (5000);
     digitalWrite(verde,    false);
}
```

Declaración de estados altos y bajos en el bucle loop.

Fuente: (Autor, 2020)

Para que el semáforo de los vehículos y de los peatones estén conectados siendo que por ejemplo cuando el semáforo de los vehículos esté en verde indicando que estos pueden seguir, el de los peatones esté en rojo y cuando el de los vehículos esté en rojo el de los peatones en verde entraremos a jugar con las conexiones del sistema.

Presentamos el siguiente diagrama para mayor compresión del conexionado.

Figura47: Diagrama ilustrativo del semáforo

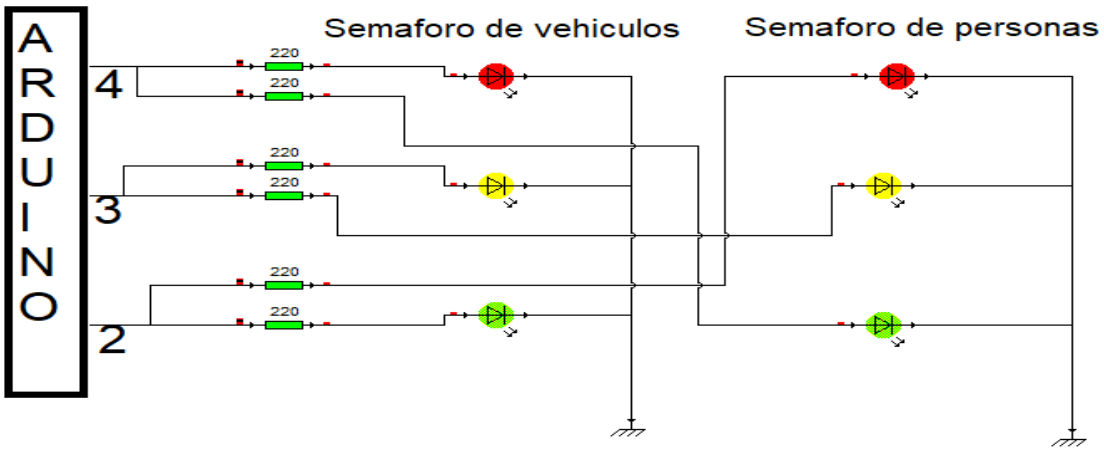


Diagrama ilustrativo de conexiones.

Fuente: (Autor, 2020)

En el podemos observar que cuando la luz roja del semáforo de los vehículos está encendida, la luz verde del semáforo de las personas está encendida, indicando a estos que pueden seguir.

De la misma manera con la luz amarilla del semáforo de los vehículos, cuando está se encienda, los del semáforo de las personas también, lo estará.

Y, por último, cuando la luz de los vehículos verde esté encendida indicando a estos que pueden seguir, la luz roja del semáforo de las personas estará encendida, indicando a las personas que se deben detener.

A continuación, se presentará el mismo conexionado, visto desde la protoboard-Arduino.

Figura48: Diagrama de practica 2

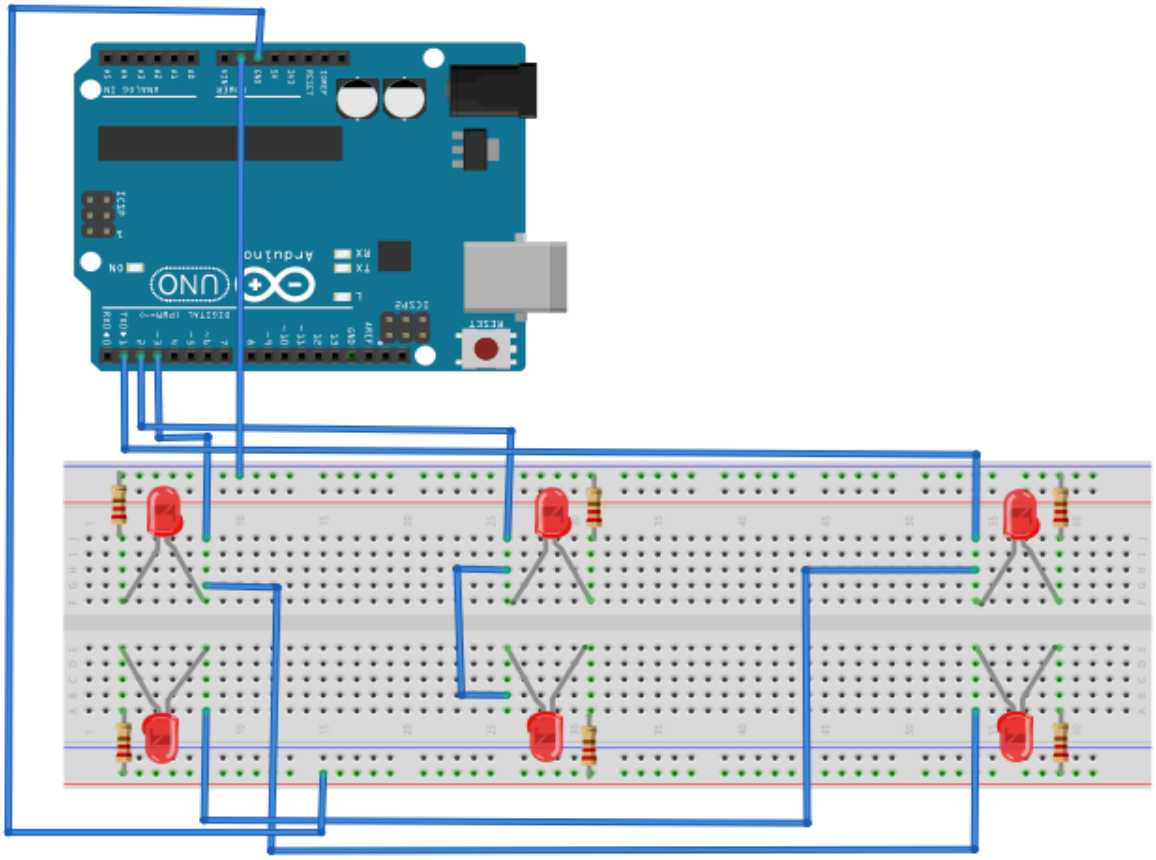
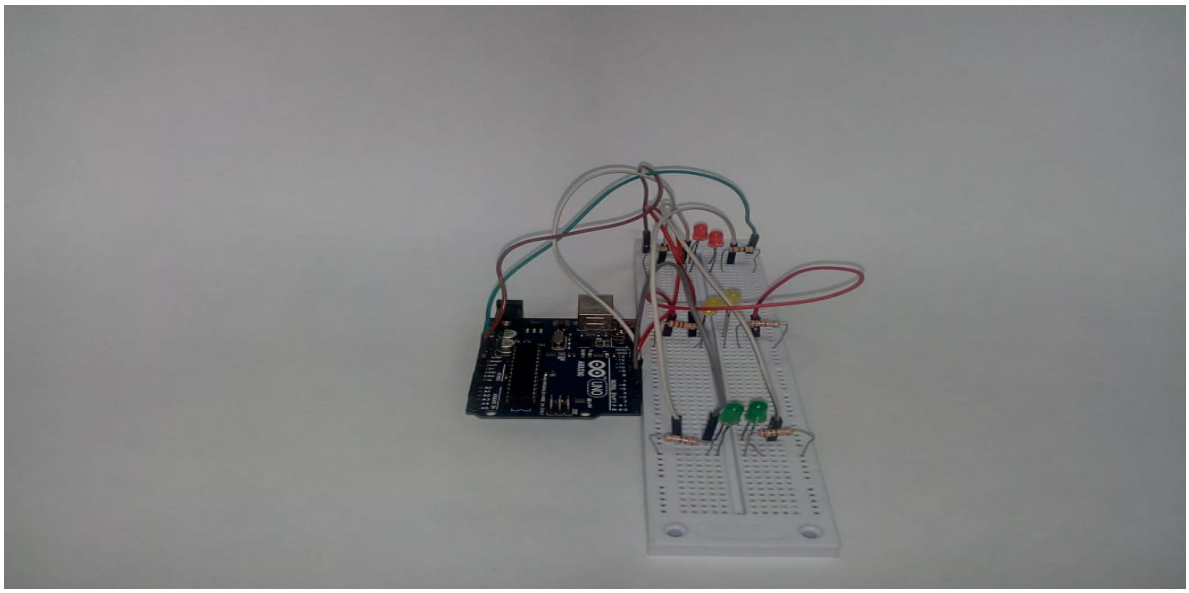


Diagrama ilustrado en la herramienta Fritzing.

Fuente: (Autor, 2020)

A continuación, una vista más real.

Figura49: Montaje practica 2



Vista real de como se ve el conexionado presentado en la figura 48 y 47.

Fuente: (Autor, 2020)

BIBLIOGRAFÍA

(s.f.). Obtenido de Aprendiendo Arduino: https://aprendiendoarduino.wordpress.com/2017/06/20/estructuras-de-control-3/

(18 de 01 de 2020). Obtenido de Descubre Arduino: https://descubrearduino.com/estructura-de-programa/

Acurio, J. (2020). *SlideShare.* Obtenido de https://es.slideshare.net/mobile/Jaime_hernan/ciclo-for

Aprendiendo Arduino. (s.f.). Obtenido de https://aprendiendoarduino.wordpress.com/2017/10/14/estructuras-de-control-4/

Arduino. (s.f.). Obtenido de Variables: https://www.arduino.cc/en/Tutorial/Variables

Arduino. (21 de febrero de 2019). Obtenido de https://www.arduino.cc/reference/en/language/variables/data-types/unsignedint/

panamahitek. (s.f.). Obtenido de http://panamahitek.com/el-setup-y-el-loop-en-arduino/

Wikipedia. (23 de 09 de 2019). *Arduino*. Obtenido de https://es.wikipedia.org/w/index.php?title=Arduino&oldid=119633331

8.1.6. Practica 3: Interrupciones externas.

COMPETENCIA	RESULTADOS DE APRENDIZAJE
Conocer algoritmos básicos de programación para el control y configuración de puertos en un microcontrolador. Diseñar aplicaciones que involucren la generación y control de tiempos mediante el uso del módulo temporizador y de interrupción externa en un microcontrolador.	Utiliza el lenguaje de programación para desarrollar aplicaciones que involucren el manejo de puertos. Desarrolla aplicaciones en las cuales se implementan circuitos de control de tiempos y de interrupciones por hardware.

MATERIALES

- Arduino uno
- Protoboard
- Pulsador
- 1 Resistencia
- Leds
- Cables

¿Qué son las interrupciones hardware?

La explicaremos en un ejemplo sencillo, tengo un microcontrolador y un pulsador conectado a él, programo el Arduino como si el pulsador no estuviera conectado a él, el Arduino hace lo que quiero que haga y además programo el código que se ejecutaría cuando se presione el pulsador aparte, entonces la interrupción va a ser la encargada de escuchar ese cambio de estado en el pin, le diré escucha el pin que quiero que el sketch haga otra cosa, el sketch va a ejecutarse normal mente y cuando se produzca la pulsación se va a producir una interrupción, entonces Arduino va a dejar de hacer lo que está haciendo, va a atender esa interrupción y después continuará donde se quedó para seguir con su ejecución normal.

No todas las placas soportan la misma cantidad de interrupciones, para esto se debe consultar en la página oficial de Arduino cuantas interrupciones soporta la placa que usted disponga.

Entendido esto vamos con la programación.

Primero que todo vamos a declarar una para el led que nos servirá como ejemplo del funcionamiento de las interrupciones por hardware y nos servirá como indicativo y la pondremos en el pin 13, otra para el pulsador que lo hemos conectado en el pin dos, además declararemos otra variable para el estado de nuestro led e inicialmente la pondremos con un LOW de apagado.

Vamos al Void setup, primero que todo declaramos el pin 13 como salida, segundo declaramos el pulsador como entrada ya que este envía datos a Arduino.

Ahora vamos a asociar interrupciones y para ello vamos a usar la función attachInterrupt y voy a decirle cual es la interrupción que voy a usar, aquí se debe poner el numero de la interrupción, no el pin que voy a usar, en la página de Arduino se encuentra el número de interrupción del pin de la placa que estés usando, en nuestro caso como estamos usando la placa Arduino uno, el número asociado al pin dos es el 0, también existe una función la cual es digitalPinToInterrupt y entre paréntesis el nombre de la variable al que tenemos conectado el pulsador, el detectara que la tenemos en el pin dos y devolverá el valor de 0, ahora debemos indicar que es lo que queremos que ocurra cuando presionemos el pulsador, vamos a llamarla controladorpulsador y después le indicamos a Arduino que es esa expresión y por ultimo debemos indicar cual es el evento que provocará la interrupción y digitamos la función RISING, esta función indica que pasa de valor bajo a valor alto es un flanco de subida, por ejemplo entra una función LOW y sale HIGH, y justo en ese momento se produce la interrupción, en resumen cuando de una función Low se presione el pulsador cambie de estado y la vuelva High.

Le agregamos un digitalWrite (led, estado) lo que quiere decir que como estado es igual a Low mantenga el led apagado.

Así, va hasta ahora la programación.

Figura50: Sketch practica 3

```
int led =  13;
int pulsador = 2;
int interrupcion = 0;
int estado = LOW;
void setup(){
  pinMode(led,OUTPUT);
  pinMode(pulsador  , INPUT);
  attachInterrupt(digitalPinToInterrupt(pulsador), controladorpulsador, RISING);
digitalWrite(led, estado);
}
```

Declaración de variables, declaración de pines de salida y comandos para el interruptor.

Fuente: (Autor, 2020)

Ahora vamos a decirle a Arduino que es controladorpulsador entonces declaramos una función Void ya que cuando se trata de interrupciones deben ser funciones que no devuelvan nada.

Entonces será Void controladorpulsador y dentro de sus corchetes vamos a escribir que es lo que quiero que ocurra cuando se presione el pulsador y lo que queremos es que cambie el estado del led.

Entonces digitamos nuestra variable estado y le decimos que estado =! estado que es lo contrario esa expresión expresa algo inverso, por ejemplo, tengo algo high y lo vuelvo Low, tengo algo Low y lo vuelvo high.

Ahora vamos con la función loop.

Le indicamos un digitalWrite (led, estado) esto quiere decir que el loop siempre va a estar vigilando el estado y cuando este cambie se dará cuenta inmediatamente.

Por último, cabe resaltar que siempre que se trabaja con interrupciones y variables que se almacenan en dos estados es mejor almacenarla en la memoria RAM por que esta se está compartiendo en dos sitios y para hacer eso simplemente tenemos que incluir la función volatile en la variable que estaría en dos sitios en nuestro caso la variable estado así que se la incluimos. En total la programación quedaría de esta manera.

Figura51: Sketch practica 3

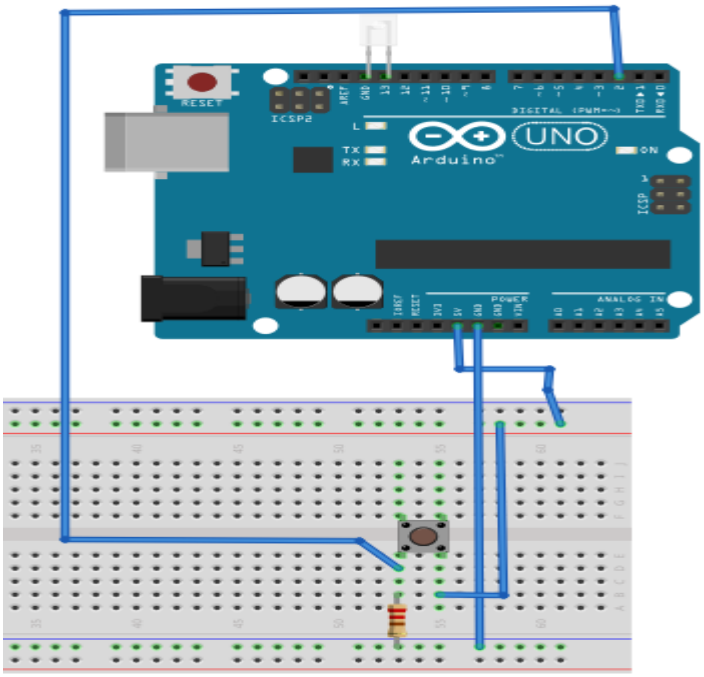
```
int led =  13;
int pulsador = 2;
int interrupcion = 0;
volatile int estado = LOW;
void setup(){
  pinMode(led,OUTPUT);
  pinMode(pulsador  , INPUT);
  attachInterrupt(digitalPinToInterrupt(pulsador), controladorpulsador, RISING);
digitalWrite(led, estado);
}
void controladorpulsador (){
  estado = !estado;
}
 void loop (){
  digitalWrite(led, estado);
 }
```

De acuerdo a la figura 51 se presenta la figura 50 pero con las modificaciones ya incluidas.

Fuente: (Autor, 2020)

Ahora presentamos su conexionado.

Figura52:Diagrama practica 3

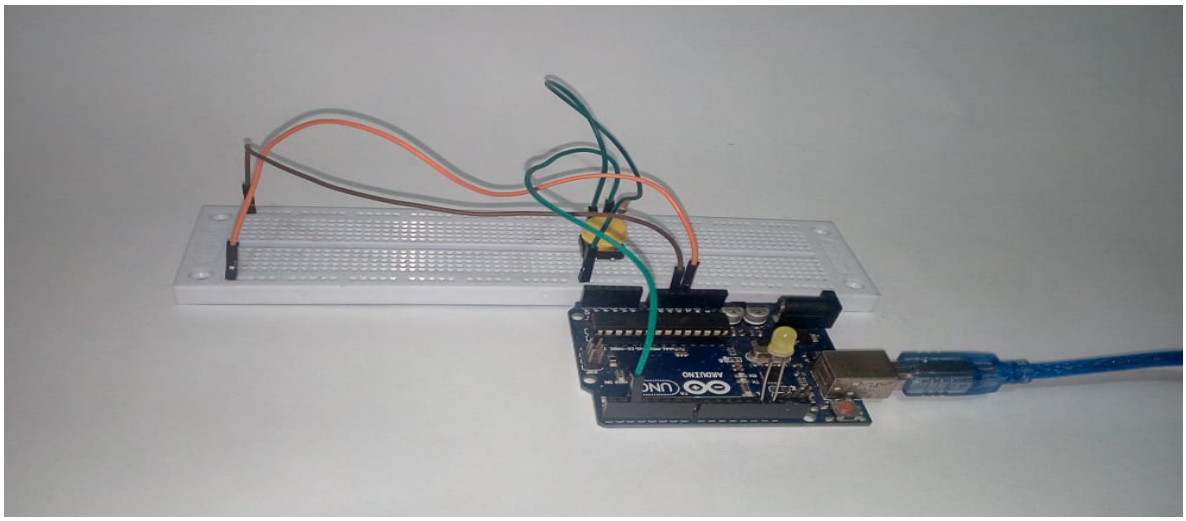


De acuerdo a la figura 52 se puede observar la conexión del pulsador a su pin por el cual enviará la señal y después su conexión de alimentación y tierra.

Fuente: (Autor, 2020)

Una vista más real.

Figura53: Montaje practica 3



Vista real del conexionado de la figura 52 antes presentado.

Fuente: (Autor, 2020)

BIBLIOGRAFÍA

(s.f.). Obtenido de Aprendiendo Arduino: https://aprendiendoarduino.wordpress.com/2017/06/20/estructuras-de-control-3/

(18 de 01 de 2020). Obtenido de Descubre Arduino: https://descubrearduino.com/estructura-de-programa/

Acurio, J. (2020). *SlideShare.* Obtenido de https://es.slideshare.net/mobile/Jaime_hernan/ciclo-for

Aprendiendo Arduino. (s.f.). Obtenido de https://aprendiendoarduino.wordpress.com/2017/10/14/estructuras-de-control-4/

Arduino. (s.f.). Obtenido de Variables: https://www.arduino.cc/en/Tutorial/Variables

Arduino. (21 de febrero de 2019). Obtenido de https://www.arduino.cc/reference/en/language/variables/data-types/unsignedint/

Crespo, M. D. (s.f.). Obtenido de http://manueldelgadocrespo.blogspot.com/p/description-digital-pins-with.html?m=1

LLamas, L. (26 de Abril de 2016). Obtenido de https:/www.luisllamas.es/que-son-y-como-usar-

interrupciones-en-arduino/

panamahitek. (s.f.). Obtenido de http://panamahitek.com/el-setup-y-el-loop-en-arduino/

Wikipedia. (23 de 09 de 2019). *Arduino*. Obtenido de https://es.wikipedia.org/w/index.php?title=Arduino&oldid=119633331

8.1.7. Practica 4: Secuencia de luces con control infrarrojo.

COMPETENCIA	RESULTADOS DE APRENDIZAJE
Conocer algoritmos básicos de programación para el control y configuración de puertos en un microcontrolador. Diseñar aplicaciones orientadas a la transmisión y recepción de datos a sistemas periféricos con base en microcontroladores.	Utiliza el lenguaje de programación para desarrollar aplicaciones que involucren el manejo de puertos. Desarrolla aplicaciones en las cuales se implementan circuitos de control de tiempos. Identifica las aplicaciones electrónicas que necesitan la transmisión y recepción de datos digitales entre circuitos integrados.

MATERIALES

- Arduino uno
- Protoboard
- Fototransistor infrarrojo HX1838
- 6 Resistencia
- 6 Leds
- Cables

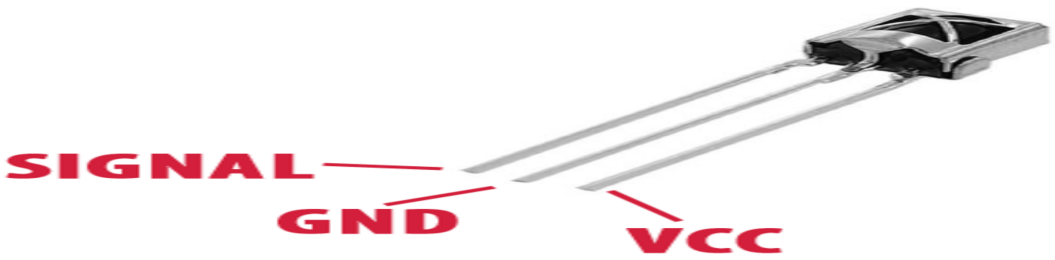
Para esto utilizaremos el **Fototransistor infrarrojo HX1838** es un sensor que nos permitirá recibir señales infrarrojas de otros dispositivos tales como; controles remotos, celulares o incluso emisores infrarrojos que armemos para nuestros proyectos de electrónica.
Lo primero que se debe entender es como conectar el sensor infrarrojo.

La extremidad del sensor denominada **signal**: Se conecta a una entrada digital de Arduino, esta es la que se encarga de enviar el código de la tecla que se presionó en el control.

GND: Es la tierra debe conectarse a Negativo.

VCC: Debe conectarse a una alimentación DC positiva en un rango de 2.7 a 5.5 Volts.

Figura54: infrarrojo HX1838

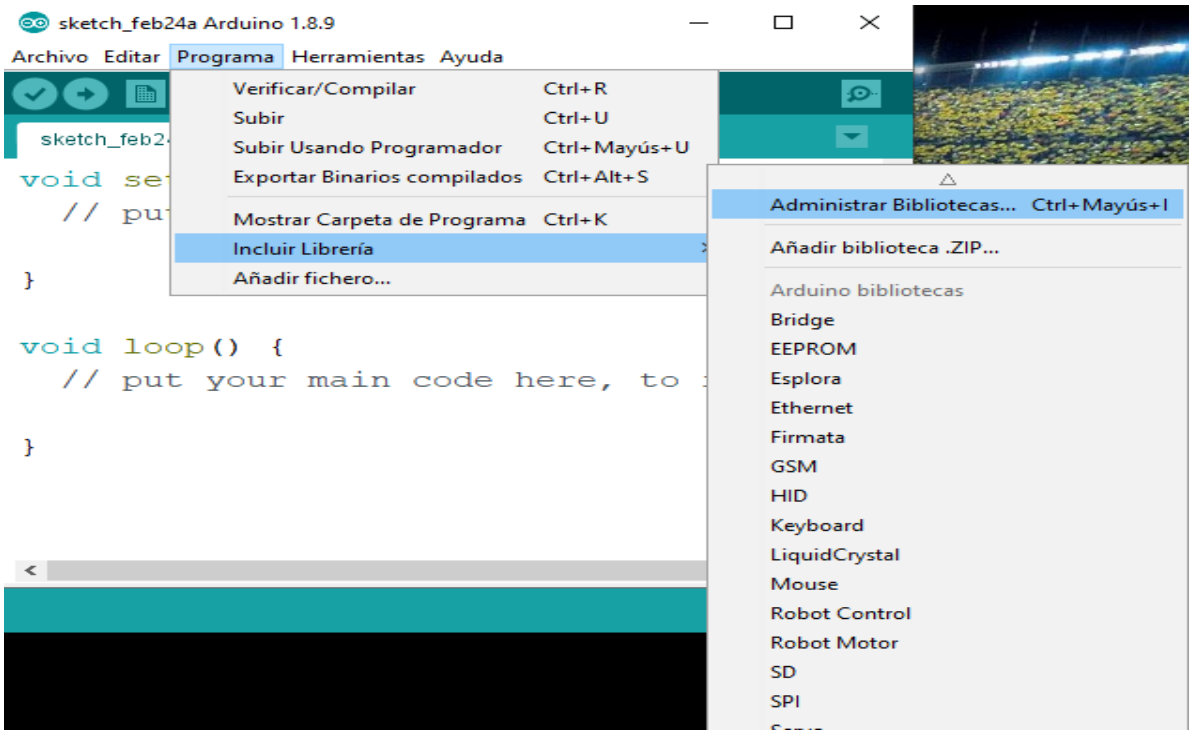


De acuerdo a la figura 54 se puede observar los pines del infrarrojo, los cuales indican: señal, tierra y alimentación.
Fuente: (electronics, 2020)

Ahora procedamos para la programación.

Para esto debemos abrir nuestro programa Arduino y debemos incluir una librería llamada IRremote, lo haremos de la siguiente manera.

Figura55: Como añadir bibliotecas practica 4

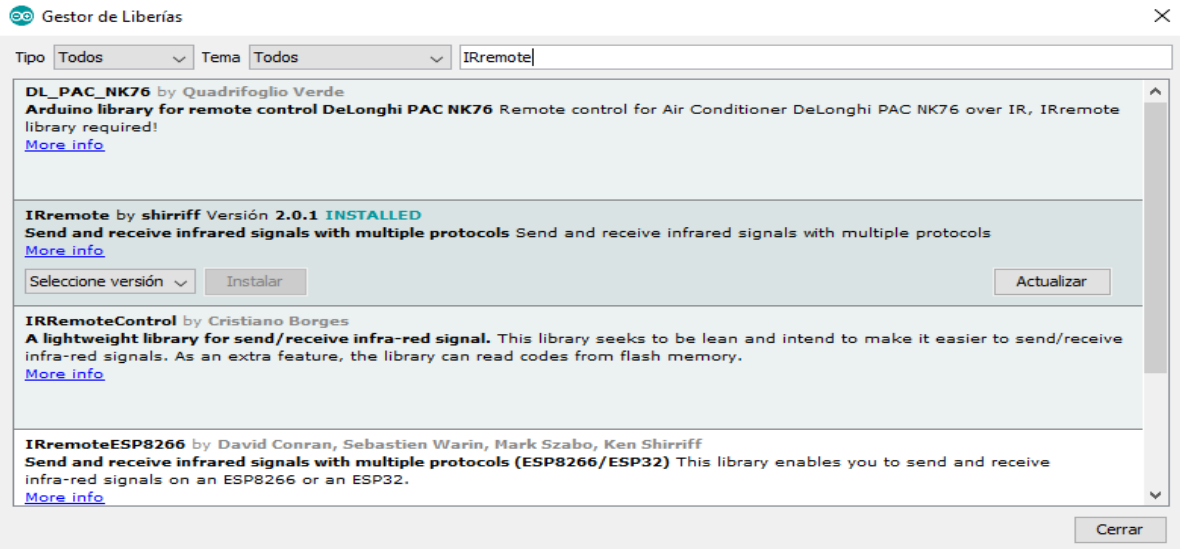


Según la figura 55 se ilustra el primer paso para añadir una librería.

Fuente: (Autor, 2020)

Después de pulsar en Administrar bibliotecas nos aparecerá un recuadro y en el escribiremos IRremote.

Figura56: Gestor de librerías practica 4



Vista de la librería a instalar (IRremote by shirriff) en el gestor de librerías.

Fuente: (Autor, 2020)

Pulsarán donde dice instalar y su librería IRremote estará instalada.

Pasando a la línea de códigos.

Figura57: Sketch practica 4

```
#include <IRremote.h>
int RECV_PIN = 2;
int led1 = 3;
int led2 = 4;
int led3 = 5;
int led4 = 6;
int led5 = 7;
int led6 = 8;
int itsONled[] = {0,0,0,0};
#define code1 54113

#define code2 25191
```

Inclusión de librerías, declaración de pines y definición del código proveniente del control remoto captado por el infrarrojo.

Fuente: (Autor, 2020)

Escribimos #include <IRremote.h> para agregar la librería previamente instalada a la programación.

Después definimos que el pin de salida del sensor infrarrojo va a el pin 2 de Arduino.

Almacenamos que led1 = 3, lo que quiere decir que va al pin 3 de Arduino y así sucesivamente hasta llegar al led6.
Almacenamos itsONled [] = {0, 0, 0, 0} los cuatro ceros representan los dos modos que representaran cada caso

Ejemplo:

Case1 tiene dos estados: encendido y apagado

Case2 tiene dos estados: encendido y apagado

En conjunto suman 4 a ello los 4 0 para los 4 estados.

Con #define code1 y #define code2 definimos el código de la tecla que se pulsa en el control y recibe el sensor esto se haya en monitor serial que se explicará más adelante.

Dependiendo de la programación los códigos de las teclas pulsadas en el control se pueden decodificar de diferente manera, diferenciándose los valores decodificados entre estos por sus características como de que algunos son alfanuméricos, otros numéricos y otros también numéricos, pero de diferentes cifras.

Siguiendo con la programación.

Figura58: Sketch practica 4

```
IRrecv irrecv(RECV_PIN);
decode_results results;
void setup()
{
Serial.begin(9600); // velocidad para comunicación serial
irrecv.enableIRIn(); // comienza a funcionar el receptor
pinMode(led1, OUTPUT);
pinMode(led2, OUTPUT);
pinMode(led3, OUTPUT);
pinMode(led4, OUTPUT);
pinMode(led5, OUTPUT);
pinMode(led6, OUTPUT);

}
```

Decodificación de códigos, se establece la velocidad de comunicación serial, se inicia el receptor y se declaración de salidas.

Fuente: (Autor, 2020)

Con IRrecv irrecv (RECV_PIN) creamos la variable u objeto para el receptor IR, aclarando antes que RECV_PIN es igual al pin 2.

Luego creamos la variable result que es una estructura donde se guardarán los datos relacionados cuando se recibe un dato a través del sensor.

Nos vamos a el apartado de Void setup.
Establecemos que la velocidad de comunicación serial a con el dispositivo será de 9600 baudios.

Con irrecv. enableIRIn comenzaremos con la recepción de datos.

Declaramos desde led1 hasta led 6 como salidas.

Figura59: Sketch practica 4

```
void loop() {
if (irrecv.decode(&results))
{
unsigned int value = results.value;
switch(value) {
```

Bucle loop, funciones if que definirán la ejecución del sketch.

Fuente: (Autor, 2020)

Empezamos con Void loop.

La función if traduciría: Si (“y la condición”) se vería así: if (irrecv. decode(&results)), entonces se entendería que se cumple la condición de (irrecv. decode(&results)) cuando unsigned int value = results.value; lo que quiere decir es que cuando la variable value de resultados positivos solamente ya que unsigned int solo almacena valores positivos sea igual a results.value la condición se cumple.

Seguimos con la función switch case.

Switch compara el valor de la variable que en nuestro caso sería la variable value con con los valores que especificamos en nuestras funciones case, si estas son iguales se ejecuta el codigo de esa declaracion case.

Entonces veamos la programación completa hasta llegar al break que lo usamos para salir de la instrucción switch.

Figura60: Sketch practica 4

```
switch(value)
{
case codel:
if(itsONled[1] == 1)
{
digitalWrite(led1, LOW);
delay(90);
digitalWrite(led2, HIGH);
delay(90);
digitalWrite(led3, LOW);
delay(90);
digitalWrite(led4, HIGH);
delay(90);
digitalWrite(led5, LOW);
delay(90);
digitalWrite(led6,HIGH);
delay(200);
digitalWrite(led1,HIGH);
delay(90);
digitalWrite(led2, LOW);
delay(90);
digitalWrite(led3, HIGH);
delay(90);
digitalWrite(led4, LOW);
delay(90);
digitalWrite(led5, HIGH);
delay(90);
digitalWrite(led6,LOW);
delay(200);
```

Indicación de si se cumple un if reproduzca lo que está entre paréntesis.

Fuente: (Autor, 2020)

Figura61: Sketch practica 4

```
// apague este cuando el boton es presionado
itsONled[1] = 0; // y se fija su estado en apagado
}
else { //sino el primer led está apagado
digitalWrite(led1, LOW);
digitalWrite(led2, LOW);
digitalWrite(led3, LOW);
digitalWrite(led4, LOW);
digitalWrite(led5, LOW);
digitalWrite(led6, LOW);
 // enciende cuando el botón es presionado
itsONled[1] = 1; // y fija su estado en encendido
}
break;
```

Indicación de si se cumple else reproduzca lo que está entre paréntesis

Fuente: (Autor, 2020)

Entonces como bien decíamos antes la función switch compara la variable entre paréntesis que en nuestro caso sería value, con las instrucciones indicadas en case, proponemos case code1 lo cual quiere decir que empezaremos a trabajar con el código que envía el control remoto a nuestro sensor infrarrojo y que a través de monitor serial descifraremos, esto se enseñará más adelante. Entonces decimos que: if (itsONled [1] == 1) lo que se traduce es que si la variable led es igual a 1 indica que se debe encender, en otras palabras, que se ejecute lo que está entre paréntesis hacia abajo.

Cuando se termina el código se espera la próxima instrucción la cual será que desde nuestro control remoto pulsemos el mismo botón lo cual nos expresará que apaguemos los led's, nos dice: itsONled[1] = 0; esto traduce que cuando la variable led es igual a 0 ejecute ésta otra instrucción por eso el **else** la cual es apagar todo, después ratifica que si volvemos a pulsar expresando que

itsONled[1] = 1; se ejecutará el programa desde la primera línea de código del case code 1.

Concluimos con un break el cual como antes se dijo, sirve para salir de la función switch.

Figura62: Sketch practica 4

```
case code2:
if(itsONled[2] == 1) // si led esta encendido
{
digitalWrite(led1, LOW);
digitalWrite(led2, LOW);
digitalWrite(led3, LOW);
digitalWrite(led4, HIGH);
digitalWrite(led5, HIGH);
digitalWrite(led6, HIGH);;
// apague cuando el boton es presionado
itsONled[2] = 0; //  se fija su estado en apagado
}
else { //ejecute ésta otra linea de codigo//
digitalWrite(led1, LOW);
digitalWrite(led2, LOW);
digitalWrite(led3, LOW);
digitalWrite(led4, LOW);
digitalWrite(led5, LOW);
digitalWrite(led6, LOW);
// enciende cuando el botón es presionado
itsONled[2] = 1; // y fija su estado en encendido
}
break;
}
Serial.println(value); // muestra el valor del cada boton
irrecv.resume(); // recibe el proximo valor
```

Indicación de si se cumple un if y else reproduzca lo que está entre paréntesis dividiendo estas programaciones por casos.

Fuente: (Autor, 2020)

Para case code2 sería igual, solo que ciertos valores estarían predeterminados por code2 que vendría siendo el código que desciframos del segundo botón que pulsemos.

Para terminar Serial.println imprime el valor de cada botón en el puerto serial por eso el “value” y con el irrecv. Resume después de recibir, este deberá ser convocada para reiniciar el receptor y prepararla para recibir otro código.

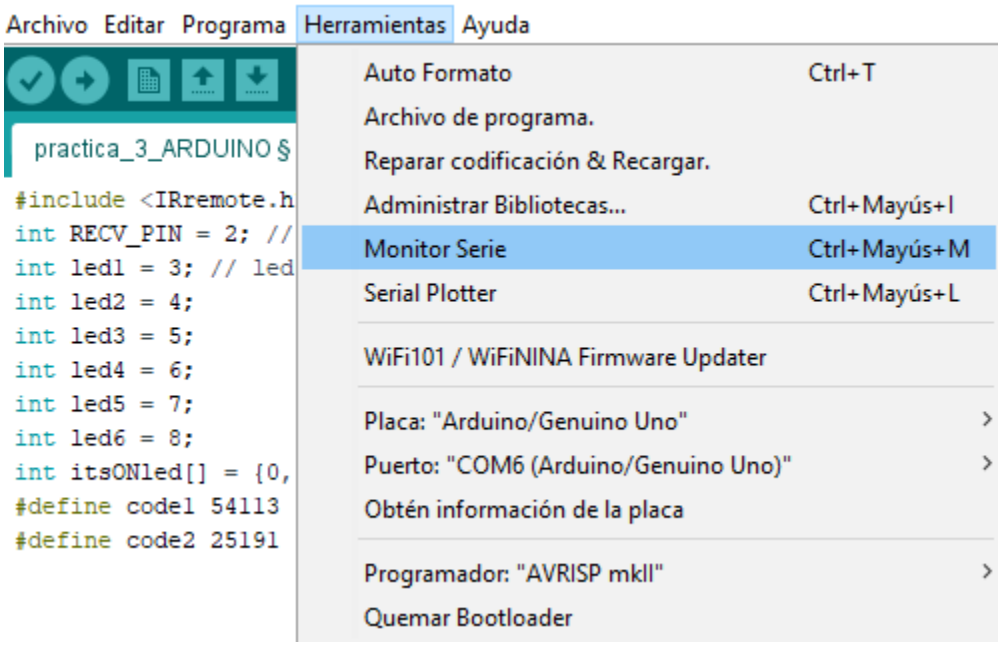
Iniciaremos con la explicación del monitor serial.

Figura63: Herramientas de Arduino ide.

Fuente: (Autor, 2020)

En nuestro entorno Arduino pulsamos donde dice herramientas.

Figura64: Como poner monitor serie

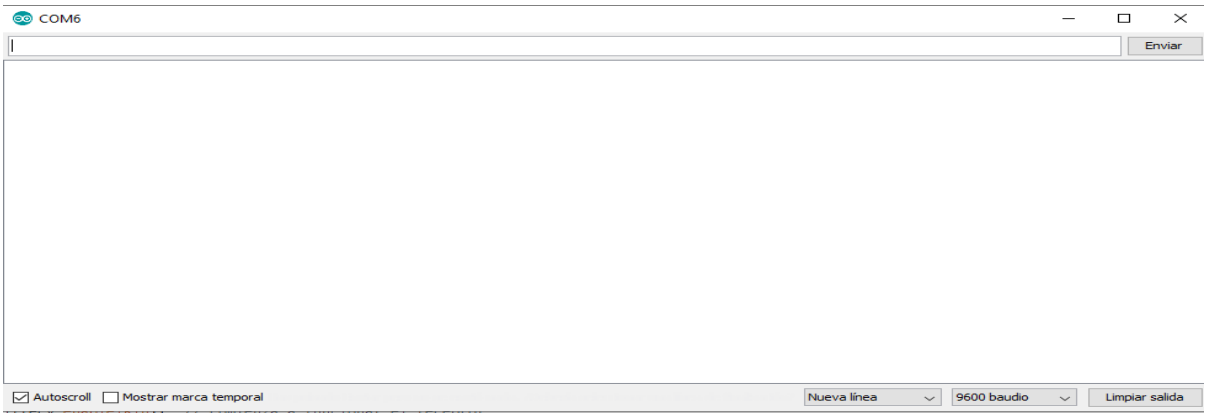


Pasos a seguir para entrar a monitor serie.

Fuente: (Autor, 2020)

Pulsamos en monitor serie y nos aparecerá una ventada donde está el monitor serial, otra manera de hacer que Arduino muestre el monitor serial es como el entorno de Arduino nos lo indica, con Ctrl+Mayus+M.

Figura65: Monitor serie

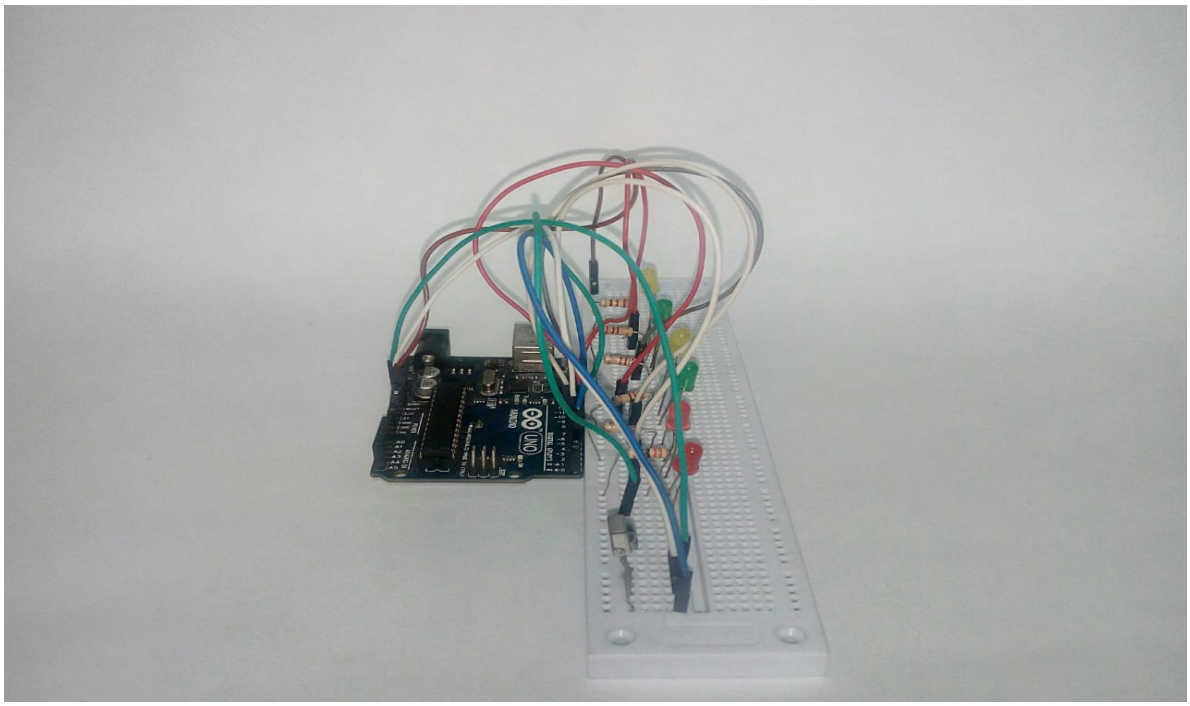


Fuente: (Autor, 2020)

Se verá de esta forma, en él se mostrará los códigos cuando pulsemos el botón del control remoto como también podremos cambiar la velocidad de ensamble donde dice 9600 baudios y allí nos aparecerán distintas velocidades que podremos escoger a gusto.

Así se ve en realidad todo el circuito de nuestra practica ya montado.

Figura66: Montaje practica 4

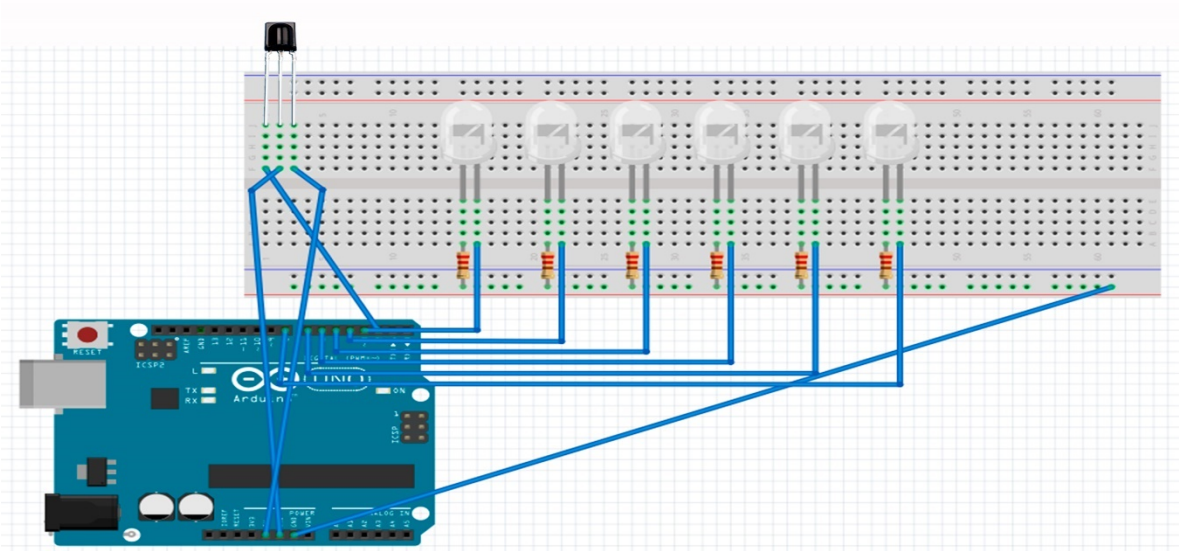


Vista real del diagrama de la figura 67.

Fuente: (Autor, 2020)

Ahora el circuito en diagrama para mejor vista.

Figura67: Diagrama practica 4



De acuerdo a la figura 67 se monta un conexionado de led clásico el cual empieza en su respectivo pin y acaba en tierra para cerrar el circuito y el infrarrojo se conecta a los pines correspondidos según la figura 54.

Fuente: (Autor, 2020)

BIBLIOGRAFÍA

(s.f.). Obtenido de Aprendiendo Arduino: https://aprendiendoarduino.wordpress.com/2017/06/20/estructuras-de-control-3/

(18 de 01 de 2020). Obtenido de Descubre Arduino: https://descubrearduino.com/estructura-de-programa/

Acurio, J. (2020). *SlideShare.* Obtenido de https://es.slideshare.net/mobile/Jaime_hernan/ciclo-for

Aprendiendo Arduino. (s.f.). Obtenido de https://aprendiendoarduino.wordpress.com/2017/10/14/estructuras-de-control-4/

Arduino. (s.f.). Obtenido de Variables: https://www.arduino.cc/en/Tutorial/Variables

Arduino. (21 de febrero de 2019). Obtenido de https://www.arduino.cc/reference/en/language/variables/data-types/unsignedint/

Crespo, M. D. (s.f.). Obtenido de http://manueldelgadocrespo.blogspot.com/p/description-digital-

pins-with.html?m=1

Naylamp mechatronics. (s.f.). Obtenido de https://naylampmechatronics.com/blog/36_Tutorial-Arduino-y-control-remoto-Infrarrojo.html

panamahitek. (s.f.). Obtenido de http://panamahitek.com/el-setup-y-el-loop-en-arduino/

Programar facil. (s.f.). Obtenido de https://programarfacil.com/blog/arduino-blog/if-else-arduino/

Wikipedia. (23 de 09 de 2019). *Arduino*. Obtenido de https://es.wikipedia.org/w/index.php?title=Arduino&oldid=119633331

8.1.8. Practica 5: Conexiones bluetooth.

COMPETENCIA	RESULTADOS DE APRENDIZAJE
Conocer algoritmos básicos de programación para el control y configuración de puertos en un microcontrolador. Diseñar aplicaciones orientadas a la transmisión y recepción de datos a sistemas periféricos con base en microcontroladores.	Utiliza el lenguaje de programación para desarrollar aplicaciones que involucren el manejo de puertos. Desarrolla aplicaciones en las cuales se implementan circuitos de control de tiempos. Identifica las aplicaciones electrónicas que necesitan la transmisión y recepción de datos digitales entre circuitos integrados. Explica el funcionamiento del módulo de comunicaciones y el uso de sus registros asociados, en el desarrollo de programas.

MATERIALES
• Arduino uno • Protoboard • 1 Resistencia • 1 Led • Cables • Modulo Bluetooth HC_06

La combinación Arduino-Bluetooth nos abre una gran cantidad de aplicaciones posibles, viendo que desde un dispositivo poder indicarle instrucciones a Arduino a través de Bluetooth para que este las ejecute resulta muy útil, para esto primeramente se debe contar con un entorno en el cual el móvil pueda comunicarse a través de Bluetooth con Arduino, en este manual nos apoyaremos en una herramienta llamada App Inventor la cual nos permite desarrollar aplicaciones en la cual podremos enlazar comunicación entre el móvil y Arduino para que este pueda enviar instrucciones.

Explicaremos como crear una App básica con el fin de sentar bases para poder crear una App con más aplicaciones y utilidades, con éstas app podemos desde programar secuencias de leds hasta controlar motores, servomotores y demás.

En esta práctica utilizaremos el módulo Bluetooth

Empecemos.

Montajes Arduino-bluetooth y creación de App en App Inventor para aplicaciones bluetooth.

Empezamos por dirigirnos a la página web con el siguiente link: http://appinventor.mit.edu/

Figura68: Creación de app en App inventor

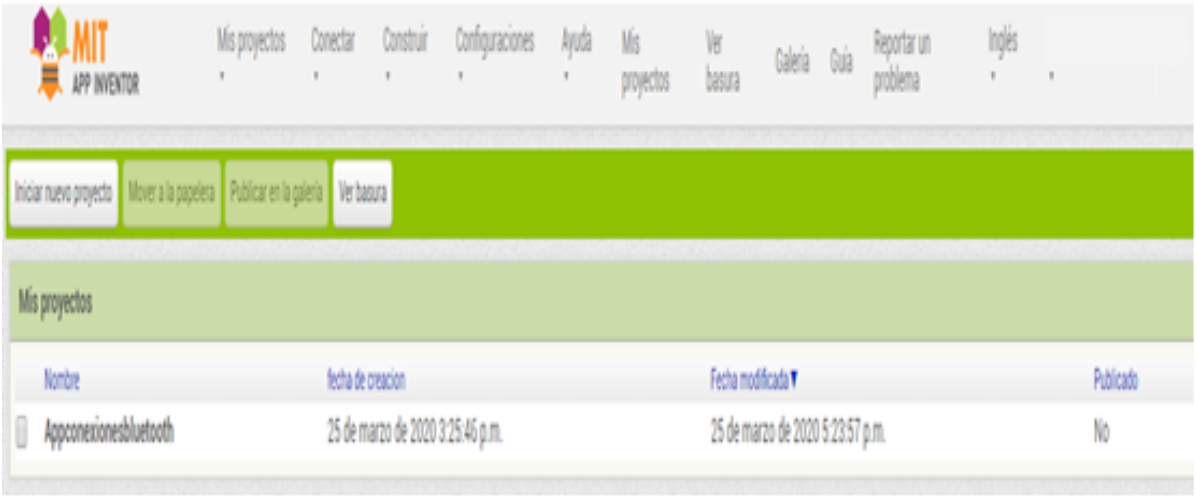
Interfaz de la página MIT APP INVENTOR.

Fuente: (Autor, 2020)

Allí podremos ver un botón naranja que dice ¡crea aplicaciones! Pulsamos allí y nos dirigirá a otra ventana en la cual nos pedirá ingresar con una cuenta de Google para poder continuar.

Después de ingresar nuestra cuenta nos aparecerá esta ventana.

Figura69: Creación de app en App inventor

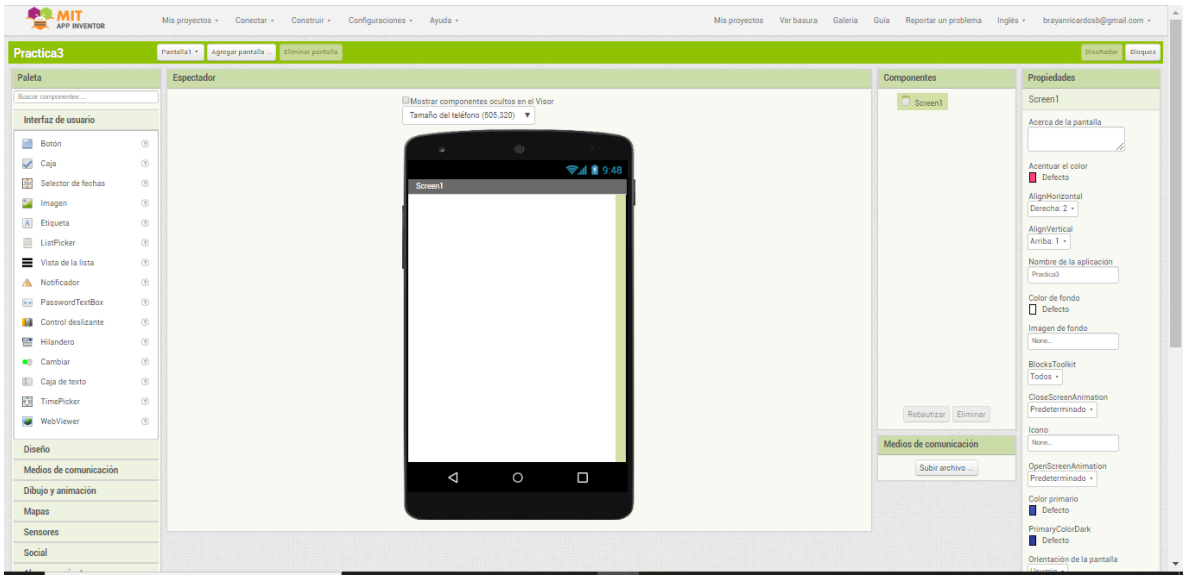


Biblioteca de creaciones.

Fuente: (Autor, 2020)

Pulsaremos en iniciar nuevo proyecto y nos pedirá el nombre que le queramos poner a la App y le daremos okay y nos saldrá una ventana así.

Figura70: Creación de app en App inventor



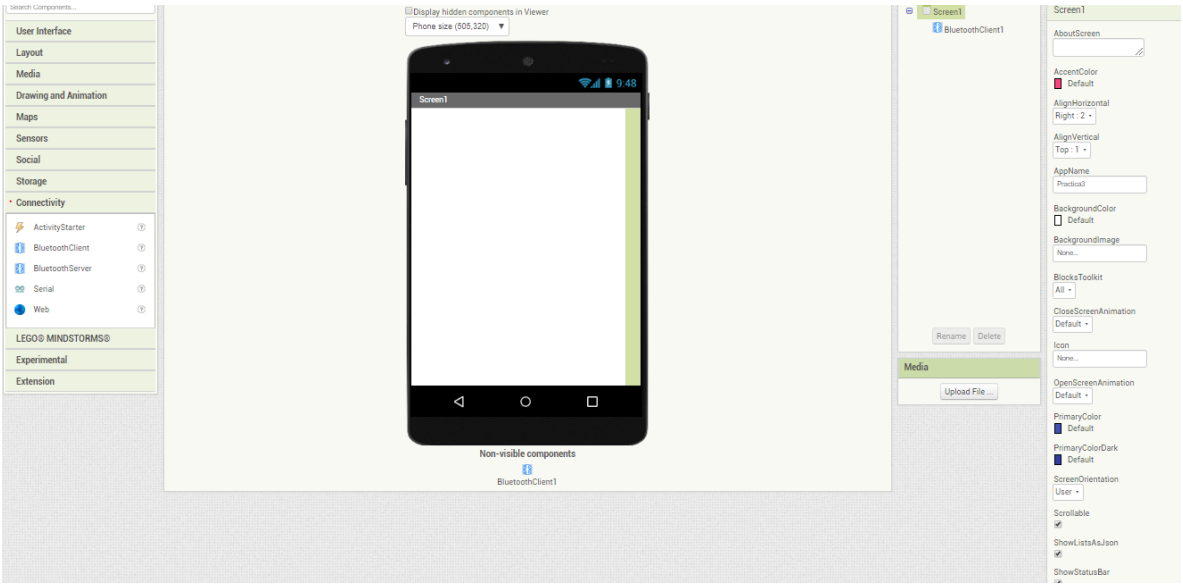
Interfaz de diseño de la página App inventor.

Fuente: (Autor, 2020)

Empezaremos con el diseño de la App.

Nos iremos al apartado de Connectivity y arrastraremos la opción Bluetooth Client hasta la pantalla y se deberá ve de esta manera.

Figura71: Creación de app en App inventor



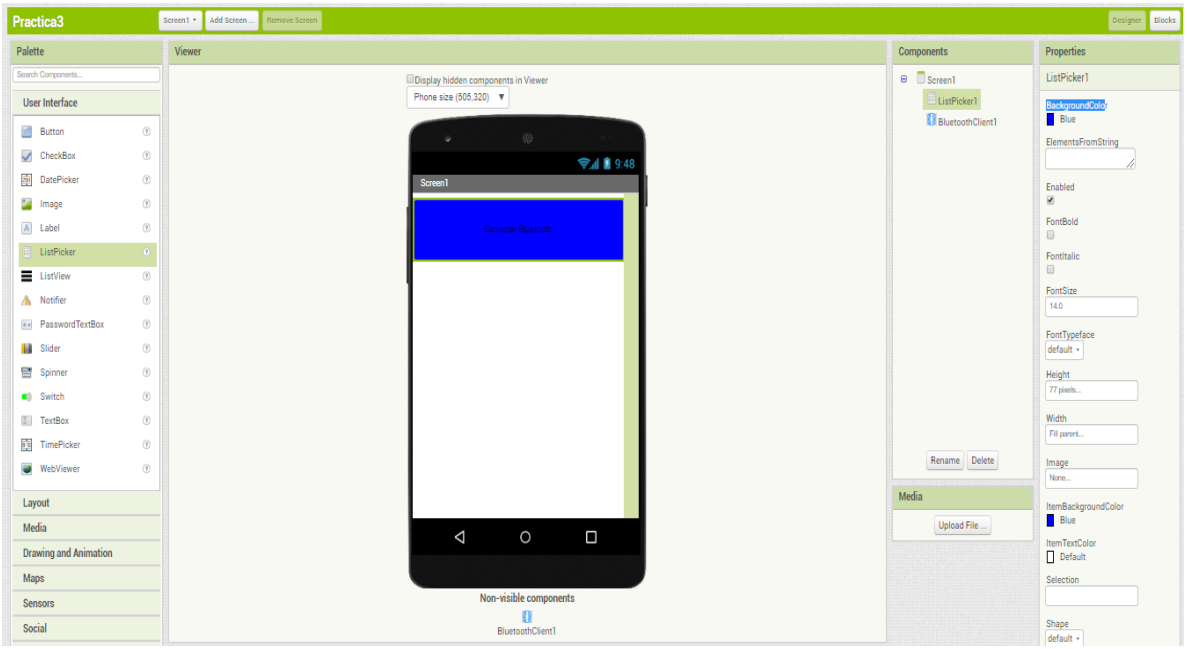
Fuente: (Autor, 2020)

Después como pueden ver al costado izquierdo hay una opción que dice “ListPicker” pulsaremos en ella y la arrastraremos hasta la pantalla.

Al costado derecho podremos ver diferentes opciones entre las que se encuentran su altura, su ancho y el texto que le queramos insertar al botón, por ejemplo, en altura le podemos indicar que sea de 77 pixeles para que el recuadro del botón quede con tal altura y en el apartado de ancho que se encuentra debajo de altura, indicarle que llene los padres para que se expanda como se muestra en pantalla.

Para insertarle texto al botón pulsaremos donde dice texto al costado inferior derecho y escribiremos en el apartado donde dice “tex”, para cambiarle de color pulsamos donde dice BackgroudColor y elegimos el color que nos guste, se deberá ver de esta manera.

Figura72: Creación de app en App inventor



Utilización de herramientas de diseño.

Fuente: (Autor, 2020)

Después por cuestión estética dejaremos un espacio en blanco y nos dirigiremos en el apartado que dice Layout, pulsaremos y arrastraremos hasta la pantalla el apartado que dice HorizontalArrangement le cambiaremos el ancho a toda la pantalla y le ajustaremos la altura de unos 40 pixeles.

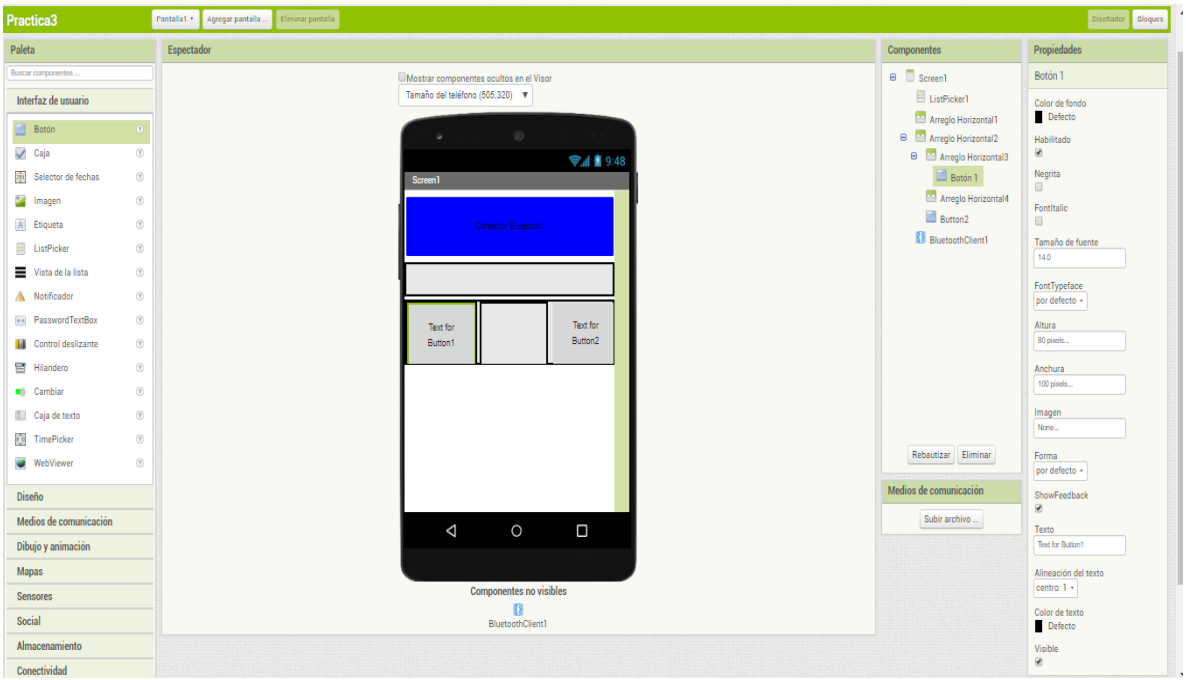
Estos Layout se utilizan más que todo para darle orden a la aplicación

Sacamos otros dos HorizontalArrangement, el primero lo pondremos de ancho en toda la pantalla y de altura unos 80 pixeles, vamos a interfaz de usuario, arrastramos nuestro primer botón a la pantalla configurándole una altura de 80

pixeles y un ancho de 100 pixeles, el segundo HorizontalArrangement lo insertaremos dentro del primer HorizontalArrangement esto nos servirá para separar los dos botones de apagar y encender, después insertamos el segundo botón en el lado derecho y lo configuramos de la misma forma que el anterior botón.

Hasta ahora se deberá ver así.

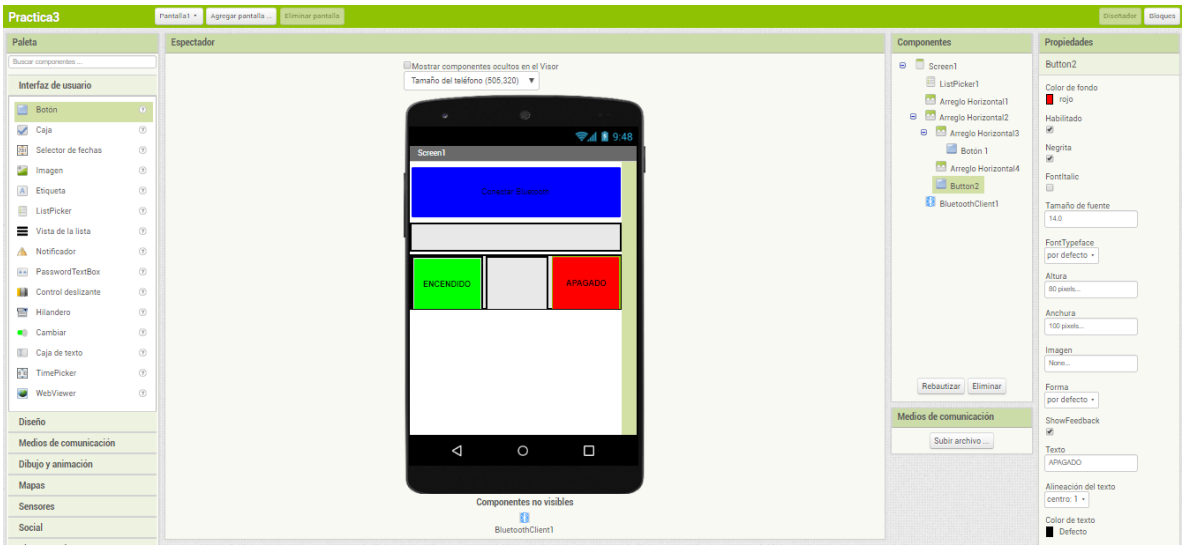
Figura73: Creación de app en App inventor



Fuente: (Autor, 2020)

Pulsamos en nuestro primer botón en el que dice tex for button 1, vamos al apartado de texto y le ponemos “**Encendido**” o “**On**” como guste, igual con el otro botón y se le configura el color si se desea, se deberá ver parecido a esto.

Figura 74: Creación de app en App inventor



Fuente: (Autor, 2020)

Ahora iniciaremos con la programacion de la aplicación, iremos al costado superior derecho en la opcion que dice “Bloques”, iniciaremos programando el botón 1 y le vamos a decir que cuando sea oprimido nos envie un texto que sea el numero uno y cuando se oprima el de apagado que nos envie un texto con el numero 2, ya despues en el Arduino programamos que con uno se encienda y que que con dos se apague Pin 13.

Nos vamos a bloques, botón 1.

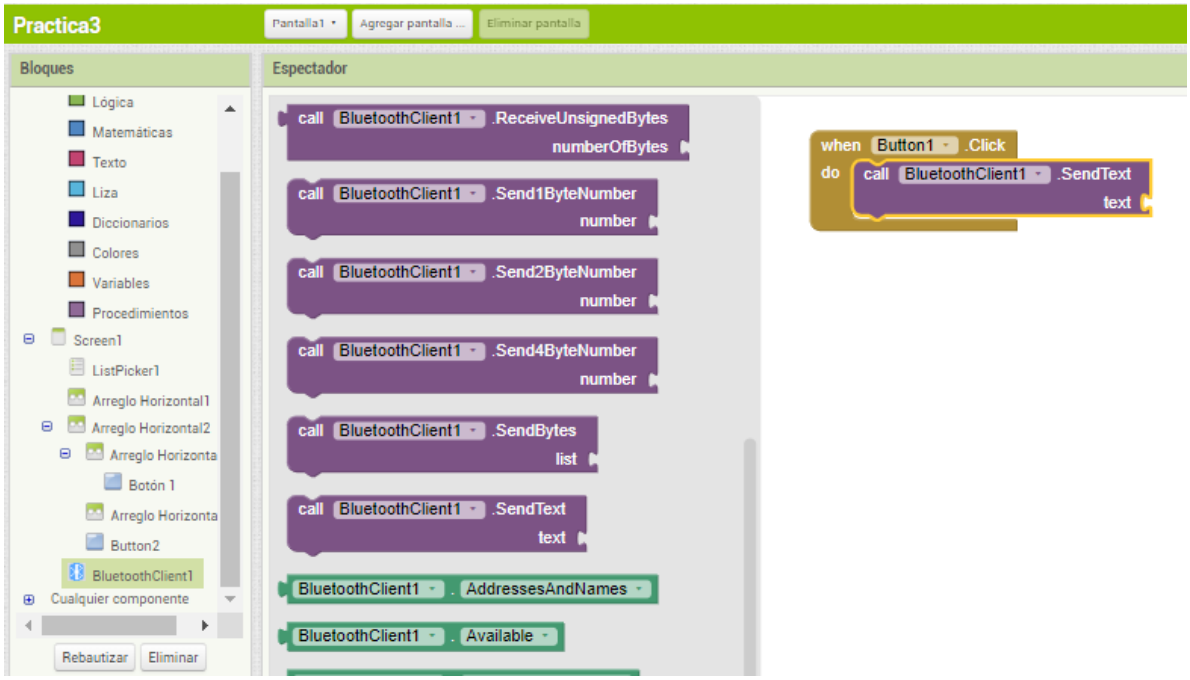
Figura75: Creación de app en App inventor

Programación de la app por bloques.

Fuente: (Autor, 2020)

Escogemos la primera opción que dice cuando pulsamos botón 1 haga, y la arrastramos a la pantalla, entonces ¿Qué queremos que haga?, nos dirigimos a él modulo bluetooth y buscamos la siguiente instrucción “llama al Bluetooth clientes y envía el siguiente texto” lo colocamos por dentro de la anterior instrucción y se deberá ver de esta manera.

Figura76: Creación de app en App inventor

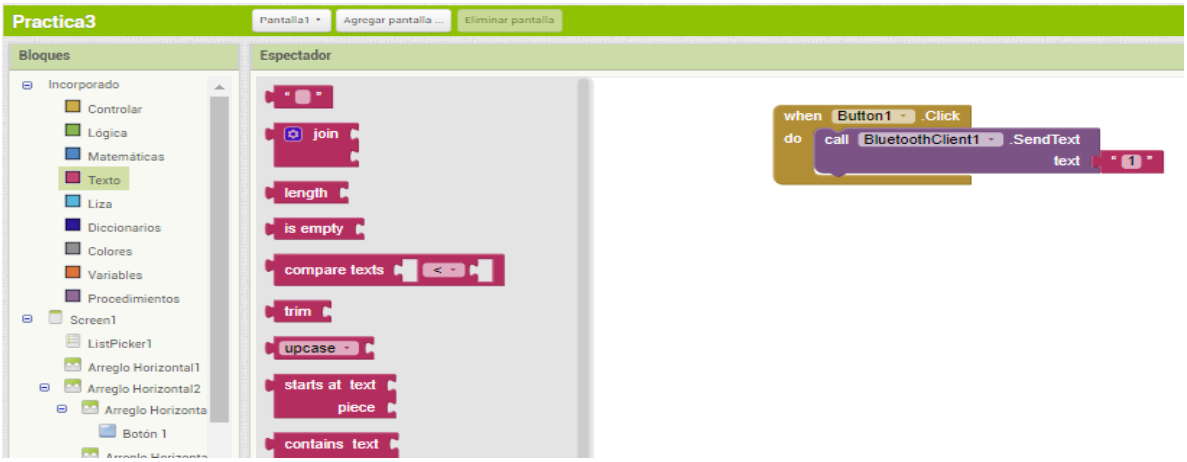


Programación de la app por bloques.

Fuente: (Autor, 2020)

Entonces la programación diría hasta ahora: cuando pulse el botón 1 llame al módulo Bluetooth para que envíe el siguiente texto ¿Y qué texto queremos enviar? Nos vamos a la opción texto y seleccionamos escribiendo “1” finalmente se deberá ver así.

Figura77: Creación de app en App inventor

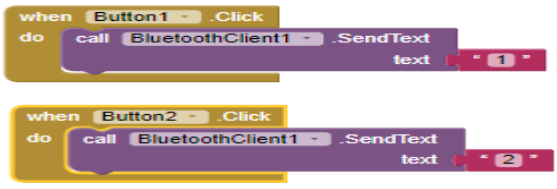


Programación de la app por bloques.

Fuente: (Autor, 2020)

Ya nos queda programado, ahora ¿Cómo programamos el botón 2? Pulsamos clic derecho en el primer bloque, el amarillo y le pulsamos en duplicar, cambiamos el Button1 por el 2 y en tex cambiamos uno por dos. Ya con esto tenemos programado los dos botones.

Figura78: Creación de app en App inventor

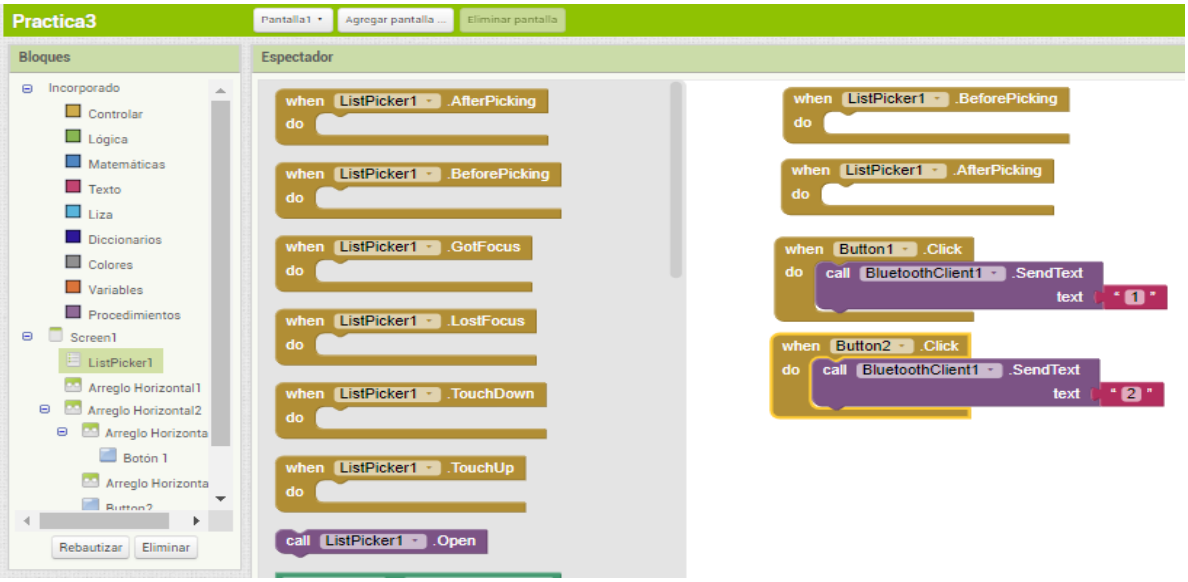


Programación de la app por bloques.

Fuente: (Autor, 2020)

Pero el módulo Bluetooth aun no lo hemos encendido así que debemos hacer dos procedimientos para que el módulo Bluetooth quede correctamente programado, vamos a la opción ListPicker pues fue la que utilizamos para que crear el recuadro azul que dice Conectar Bluetooth y ese cuadro no lo hemos programado y vamos a hacer dos cosas, sacamos dos bloques, el de before que es el de antes y el de after que es el de después, vamos a programar cosas para ese recuadro antes y después de que sea oprimido.

Figura79: Creación de App inventor

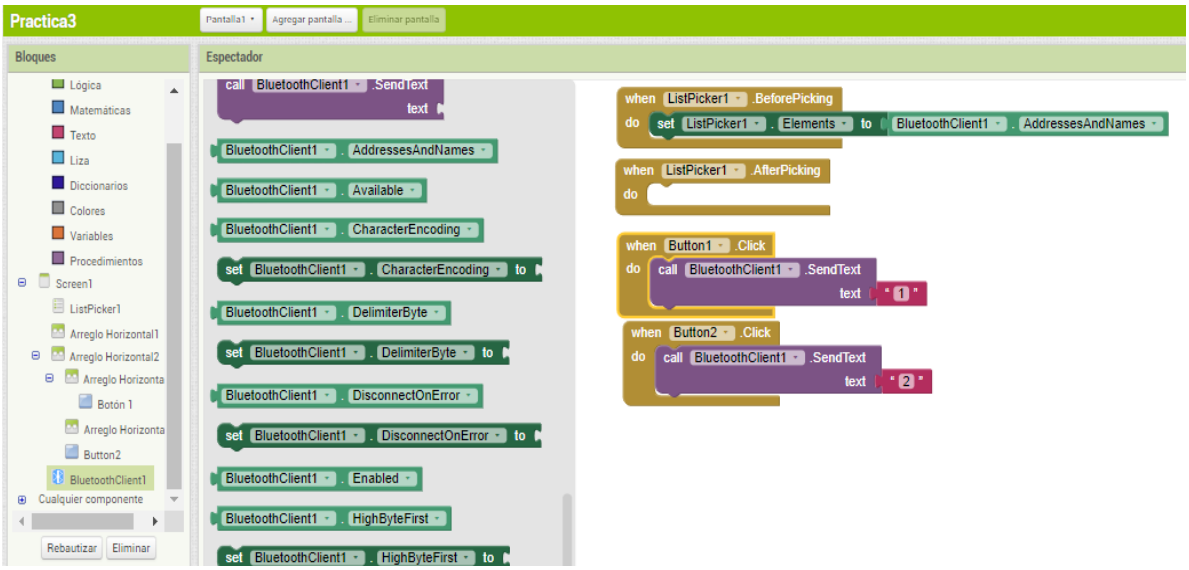


Programación de la app por bloques.

Fuente: (Autor, 2020)

Entonces empezamos con el antes de que sea oprimido, buscamos en ListPicker la opción set ListPicker 1 elements to y que queremos que haga, vamos a la opción Bluetooth client y escogemos la opción Bluetooth client1 addressesAndnames.

Figura80: Creación de app en App inventor

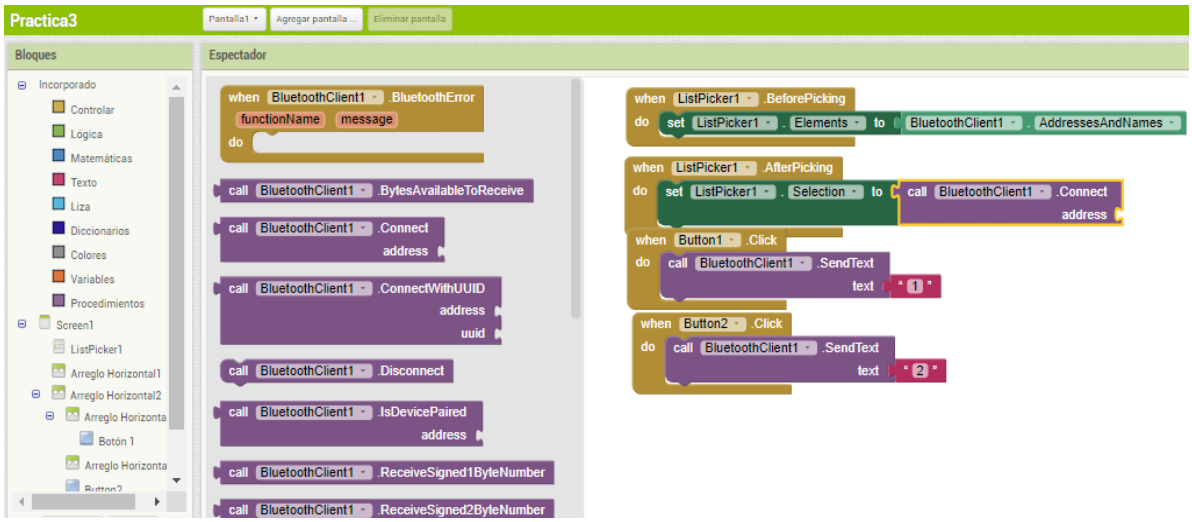


Programación de la app por bloques.

Fuente: (Autor, 2020)

Esta parte de la programación quiere decir que antes de que sea oprimido él ya vaya colocando en la lista un elemento llamado Bluetooth y vaya colocando los nombres de los módulos bluetooth que vaya encontrando, esto quiere decir que antes de hacer todo este paso él va a alistar todos los módulos Bluetooth que tenga sincronizados al teléfono Android, solo nos va alistar esos los que previamente ya haya sincronizado, ahora que quiero que haga después de que oprima el botón, nos vamos a la opción de antes set ListPicker 1 elements pero en vez de el elements colocamos la opción selection, me vengo nuevamente a Bluetooth y busco el llamado del cliente con la conexión, la dirección se lo inserto, se deberá ver de ésta forma.

Figura81: Creación de App inventor.

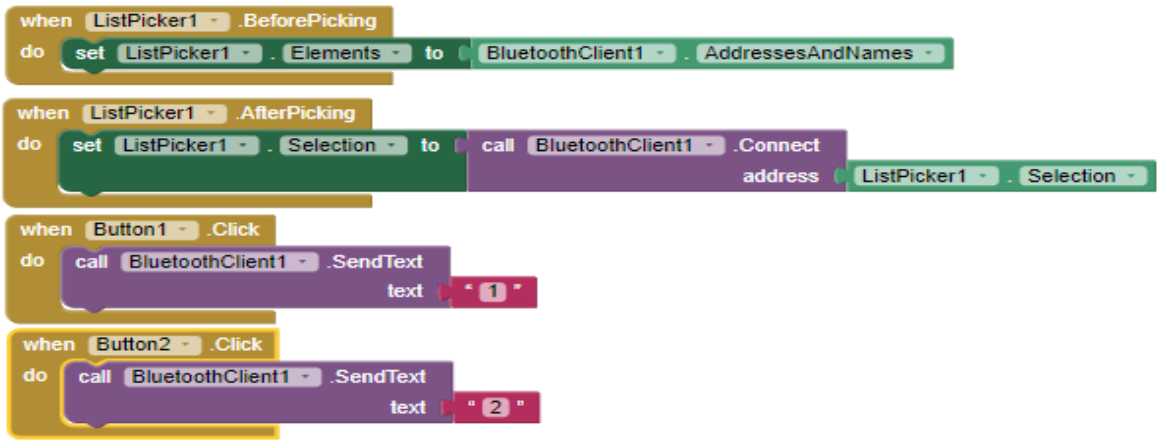


Programación de la app por bloques.

Fuente: (Autor, 2020)

Entonces que quiero que me adhiera, selecciono ListPicker y escojo el que antes habiamos seleccionado, el set ListPicker 1 selection y se lo insterto, esto para que el elemento que yo haya selecccionado me lo conecte.

Figura82: Creación de app en App inventor

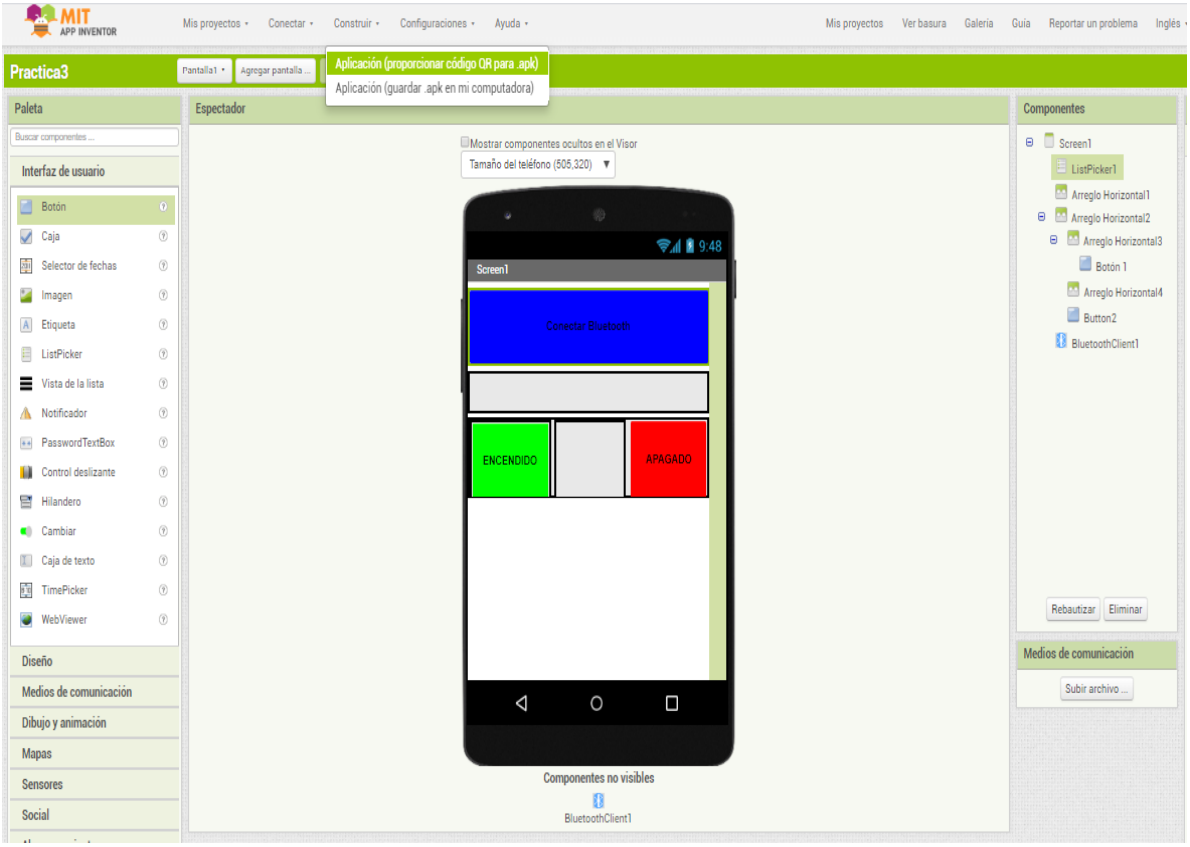


Programación de la app por bloques.

Fuente: (Autor, 2020)

Listo terminamos la programación, para este caso programamos dos botones, pero pueden ser los que se deseen.

Figura83: Creación de app en App inventor



Creación del archivo APK para poder instalarlo en nuestro celular.

Fuente: (Autor, 2020)

Ahora procedemos a crear el instalador, el instalador se basa en un APK para Android, pulsaremos en construir y en aplicación guardar en mi computadora.apk, nos descargará el APK a nuestro pc, después se procederá a pasar ese APK al celular e instalarlo como cualquier otra app de Android.

Ahora nos iremos a programar el Arduino para que este reconozca lo que la app le envía.

Figura84: Sketch practica 5

De acuerdo a la figura 84 se puede observar: declaración de variables, de salidas, declaración de velocidad para serial Begin y comandos if.

Fuente: (Autor, 2020)

Empezamos declarando una variable que se llama led13 y una variable de tipo entero llamada estado que inicialmente es con 0, Nota: una variable de tipo entero son las que almacenan valores positivos o negativos, pero no decimales.

En el Void setup lo que hacemos es inicializar el puerto serial a 9600 baudios que es la velocidad de transmisión y le asignamos el PIN “led13” como salida.

Luego para el Void loop revisamos que el puerto serial esté habilitado y que tenga por lo menos un dato en el buffer, es decir, que sea mayor a cero, si esto es así entonces almacenamos a la variable estado lo que leamos por el puerto serial, el puerto serial hay que recordar que está conectado al Bluetooth, o sea que lo que

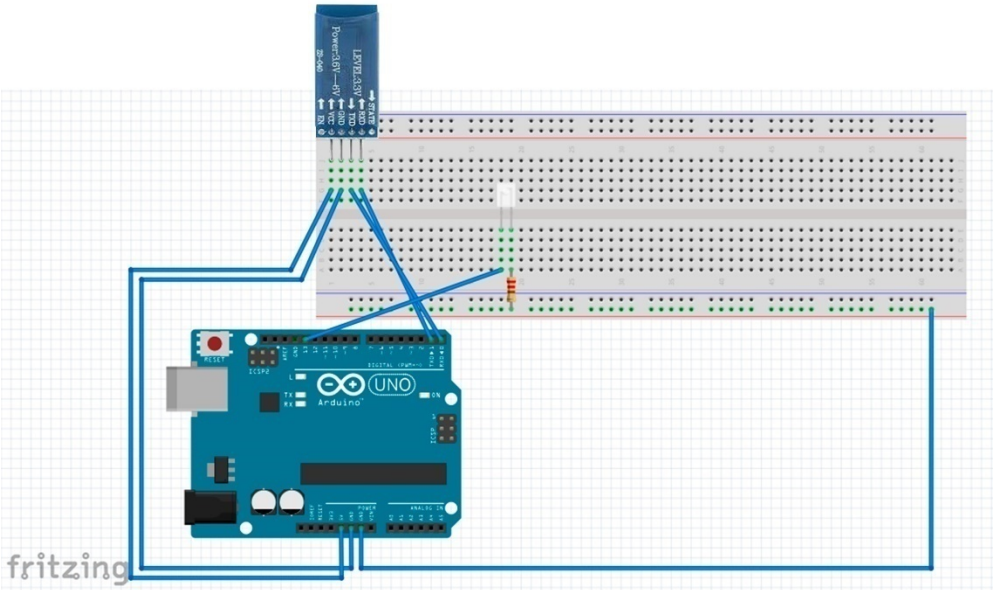
esté llegando por el puerto Bluetooth es lo que va a ser almacenado en la variable estado y va a hacer lo que habíamos programado anteriormente en nuestra app, de que si se oprime el botón 1 envía el número uno y de que si se pulsa el botón 2 se enviará el número dos, los cuales traducen que el uno es encendido y el dos apagado, para efectos de ésta práctica ese es el significado que le damos a ese número uno o dos pero eso fácilmente se podría traducir en el encendido de un motor, la posición de un servo ya configurando la app desde la página app inventor los botones de la aplicación.

Continuando con la programación hacemos un condicional, si la variable estado es igual a uno encendemos el led que está conectado al pin 13 y de igual forma, si la variable estado es igual a dos, apague el led conectado al pin 13.

Si yo quisiese incluir más botones en nuestra aplicación para distintos usos que se nos ocurra lo único que tendría que hacer seria incluir más condicionales y listo.

Ahora veamos una vista de cómo sería su conexionado.

Figura85: Diagrama practica 5



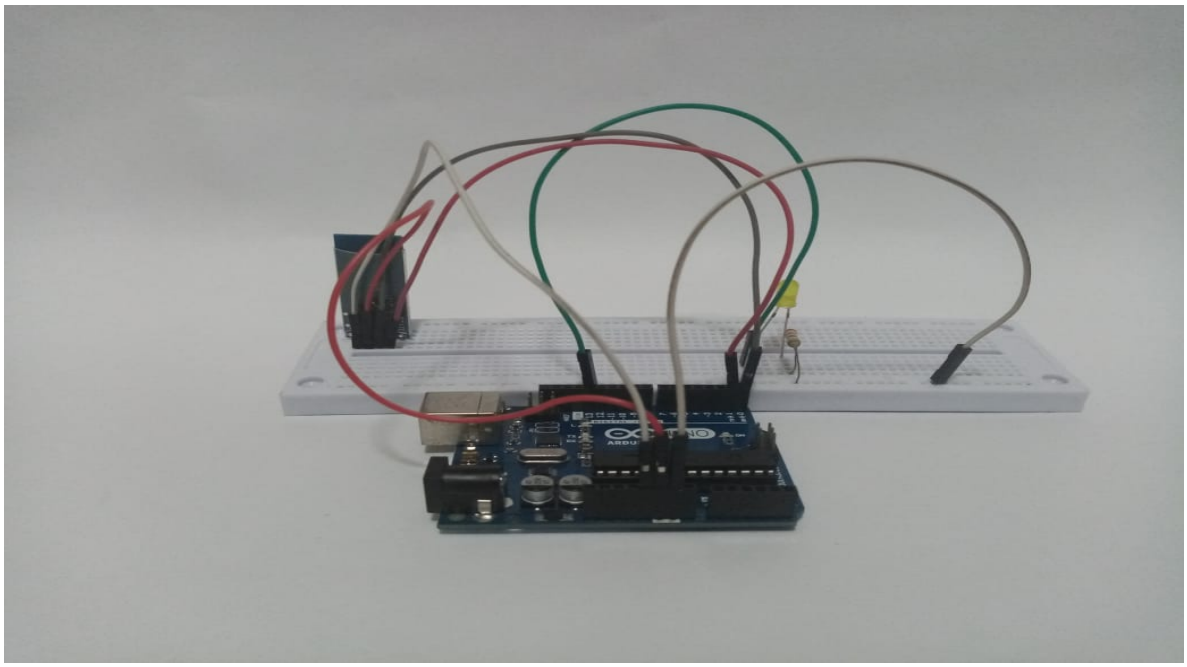
Conexión del módulo bluetooth con a sus respectivos pines: alimentación, tierra, signal, etc. Conexión clásica de led.

Fuente: (Autor, 2020)

La unica nota a aclarar es que al momento de subir la programcion en arduino debemos de tener el modulo HC_06 desconectado para que este no genere conflictos con arduino y cabe aclarar que los pines de recepcion y transmision deben ir cruzados pues el que transmite debe llegar al que recibe.

Ahora una vista real.

Figura86: Montaje practica 5



Vista real del conexionado visto anteriormente.

Fuente: (Autor, 2020)

BIBLIOGRAFIA

(s.f.). Obtenido de Aprendiendo Arduino: https://aprendiendoarduino.wordpress.com/2017/06/20/estructuras-de-control-3/

(18 de 01 de 2020). Obtenido de Descubre Arduino: https://descubrearduino.com/estructura-de-programa/

App Inventor. (2020). Obtenido de https://appinventor.mit.edu/explore/ai2/tutorials

Aprendiendo Arduino. (s.f.). Obtenido de https://aprendiendoarduino.wordpress.com/2017/10/14/estructuras-de-control-4/

Arduino. (s.f.). Obtenido de Variables: https://www.arduino.cc/en/Tutorial/Variables

Arduino. (21 de febrero de 2019). Obtenido de https://www.arduino.cc/reference/en/language/variables/data-types/unsignedint/

Crespo, M. D. (s.f.). Obtenido de http://manueldelgadocrespo.blogspot.com/p/description-digital-

pins-with.html?m=1

panamahitek. (s.f.). Obtenido de http://panamahitek.com/el-setup-y-el-loop-en-arduino/

Programar facil. (s.f.). Obtenido de https://programarfacil.com/blog/arduino-blog/if-else-arduino/

Wikipedia. (23 de 09 de 2019). *Arduino*. Obtenido de https://es.wikipedia.org/w/index.php?title=Arduino&oldid=119633331

8.1.9. Practica 6: Sensor de temperatura y humedad.

COMPETENCIA	RESULTADOS DE APRENDIZAJE
Conocer algoritmos básicos de programación para el control y configuración de puertos en un microcontrolador. Diseñar aplicaciones orientadas a la transmisión y recepción de datos a sistemas periféricos con base en microcontroladores.	Utiliza el lenguaje de programación para desarrollar aplicaciones que involucren el manejo de puertos. Desarrolla aplicaciones en las cuales se implementan circuitos de control de tiempos. Identifica las aplicaciones electrónicas que necesitan la transmisión y recepción de datos digitales entre circuitos integrados. Explica el funcionamiento del módulo de comunicaciones y el uso de sus registros asociados, en el desarrollo de programas.

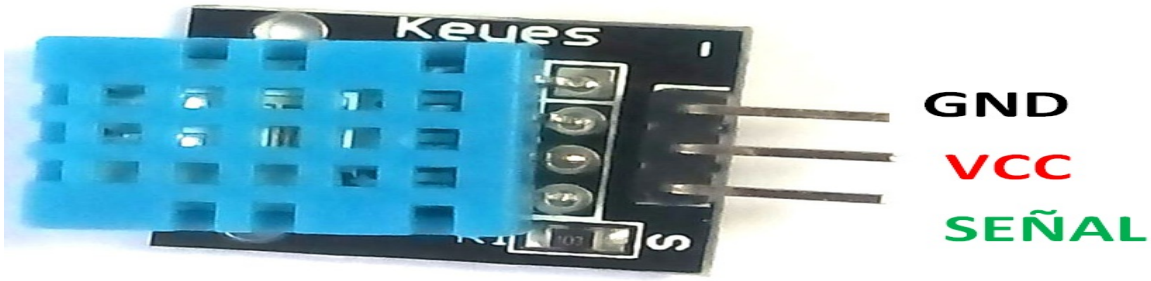
MATERIALES

- Arduino uno
- Protoboard
- Cables
- Modulo sensor de temperatura y humedad DHT11

Para esta práctica utilizaremos el módulo sensor de temperatura y humedad DTH1, básicamente para esta práctica solo necesitaremos un protoboard, una placa Arduino y el módulo sensor.

Empecemos por explicar las características del módulo sensor de temperatura y humedad DTH11.

Figura87: Modulo DHT11



Presentación y pines del dht11.

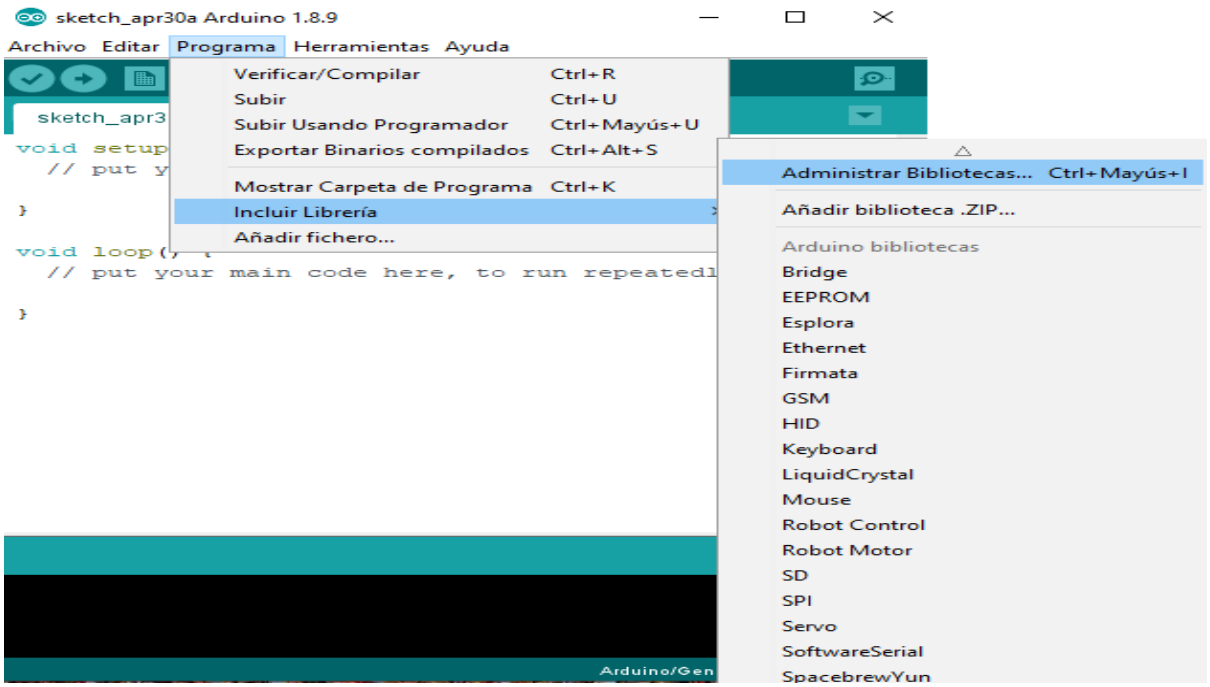
Fuente: (Hetpro, 2020)

Donde GND es tierra, VCC donde conectaremos su alimentación y señal donde este envía datos, este módulo se alimenta de 5 volts.

Vámonos a la programación, para programar este módulo tenemos que primero instalar la librería del mismo con el fin de facilitar su programación. Abrimos el entorno de Arduino, nos vamos a: programa, incluir librerías, administrar librerías, en ese orden.

Después nos aparecerá el gestor de librerías y vamos a la barra de búsqueda y escribimos DHT sensor library y posteriormente la instalamos.

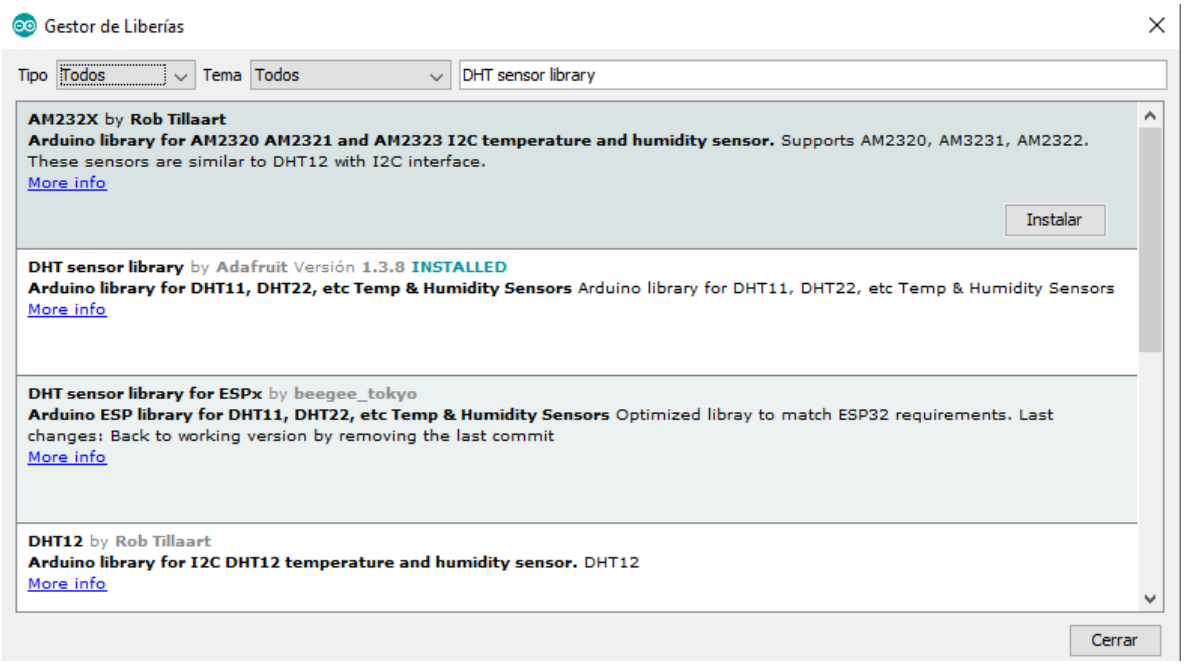
Figura 88: Incluir librería practica 6



Pasos para inclusión de librerías.

Fuente: (Autor, 2020)

Figura89: Gestor de librería practica 6

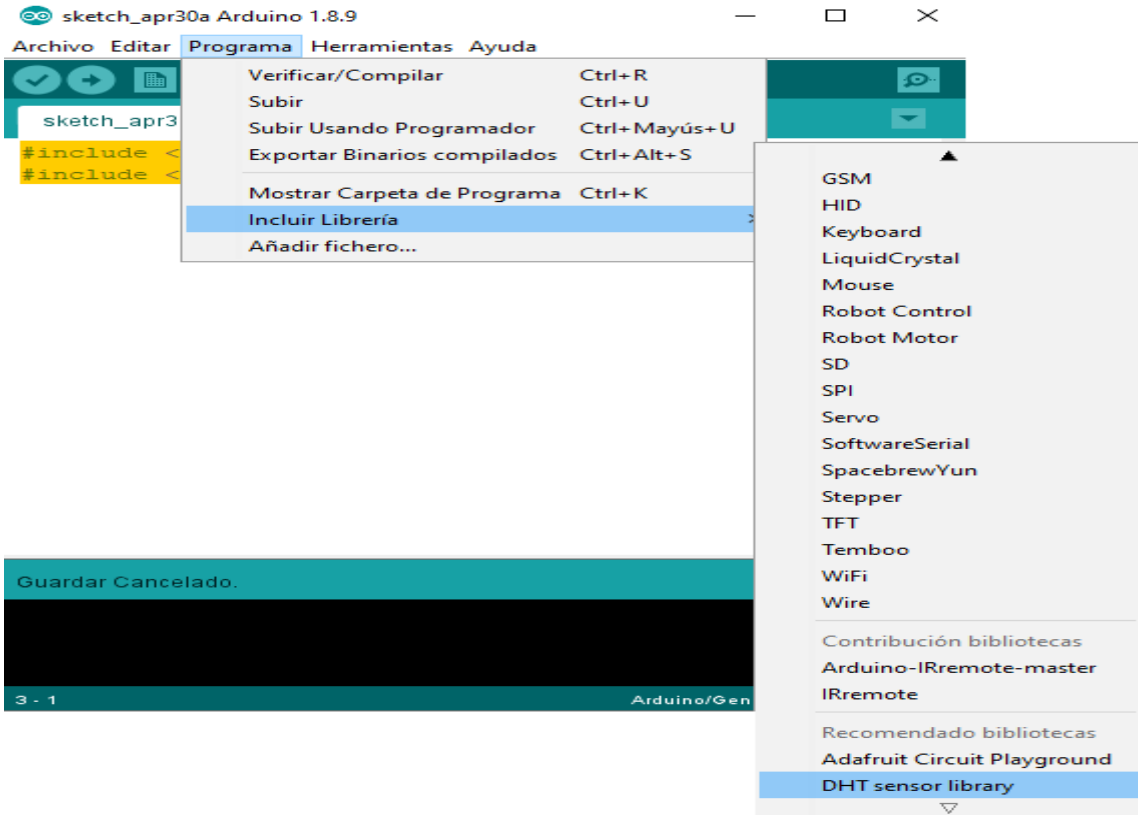


Fuente: (Autor, 2020)

Ahora despues de instalarla vamos de nuevo a programa, incluir librería la buscamos y pulsamos en ella y donde se escribe las lineas de codigo nos deberá escribir .

```
#include <DHT.h>
#include <DHT_U.h>
```

Figura90: Librería DHT



Inclusión de librerías.

Fuente: (Autor, 2020)

Empezamos como siempre a declarar variables para acceder al pin digital, todas las variables necesarias y demás.

Crearemos la primera variable int que se llamará sensor y le vamos a dar valor dos en referencia al pin al que vamos a conectar desde el pin de señal del sensor.

También incluiremos otras dos variables int, que serán para leer y recoger la información del sensor, un de estas se llamará temperatura y la otra humedad.

Ahora vamos a configurar el Arduino utilizando el set up lo que vamos a hacer es leer los datos que exporta el sensor al Arduino y la voy a mostrar por el monitor serial, siendo así la forma más sencilla de comprobar el funcionamiento de este sensor.

Establecemos serial begin a 9600 baudios para poder utilizar el monnitor serial y mostrar la informacion que vayamos leyendo, la temperatura y la humedad.

Ahora vamos a utilizar un tipo de dato definido en la librería agregada anteriormente que me va permitir simplificar todo el uso del sensor.

Vamos a crear una variable arriba del set up de tipo HT esta está definida dentro de la librería, tengo que indicarle dos cosas, la variable y de que tipo es el sensor, pues no solamente está el DHT11, existen mas.

Entonces le ponemos DHT11 que es una constante que está ya definida dentro de la librería agregada, y representa la id de nuestro modulo sensor DHT11 y de esta forma le decimos a arduino que el sensor está conectado al pin dos y que el sensor es un DHT11.

Una vez hecho eso dentro de nuestro setup vamos a inicializar el sensor entonces le ponemos dht. begin y con eso nuestro sensor comenzaria a medir, recopilar los diferentes impulsos captados del exterior y mandarlos a arduino.

Figura91: Sketch practica 6

```
#include <DHT.h>


int sensor = 2;
int temperatura, humedad;

DHT dht (sensor, DHT11);
 void setup()
 {
  Serial.begin(9600);
  dht.begin();
 }
```

Según la figura 91 podemos observar: inclusión de librerías, comandos de la librería DHT.h, declaración de variables y inicializaciones.

Fuente: (Autor, 2020)

Ahora nos vamos a void loop, necesitamos leer la informacion que arduino recibe de el sensor, entonces en la librería DHT hay una funcion que permite leer la humedad y la temperatura.

Le decimos a arduino que la variable temperatura es igual a dht.readTemperature, la cual nos leerá la temperatura y asi mismo con la humedad que su funcion propia se define como dht.readHumididy.

Ahora vamos a mostrarla por el serial atra ves de serialprint que es la funcion que nos imprimirá nuestros datos deseados en el monitor serial.

Entonces primero hacemos un serialprint que nos imprima la palabra “Temperatura: ”, despues hacemos otro que nos imprima la variable antes creada temperatura que será igual a la temperatura que el sensor capte, ese dato es el

que mostrará, siguiendo haremos otro serialprint donde nos imprima el texto “°C” para que por ahora llevemos: Temperatura: “valor captado por el sensor” °C.

Asi mismo haremos con la humedad, primero imprimiremos la palabra Humedad, segundo imprimiremos la variable humedad y despues imprimiremos el “%” ya que la humedad la dictaremos porcentualmente.

Le daremos un delay que será el tiempo en el que el monitor serial imprimirá de nuevo los datos.

Figura92: Sketch practica 6

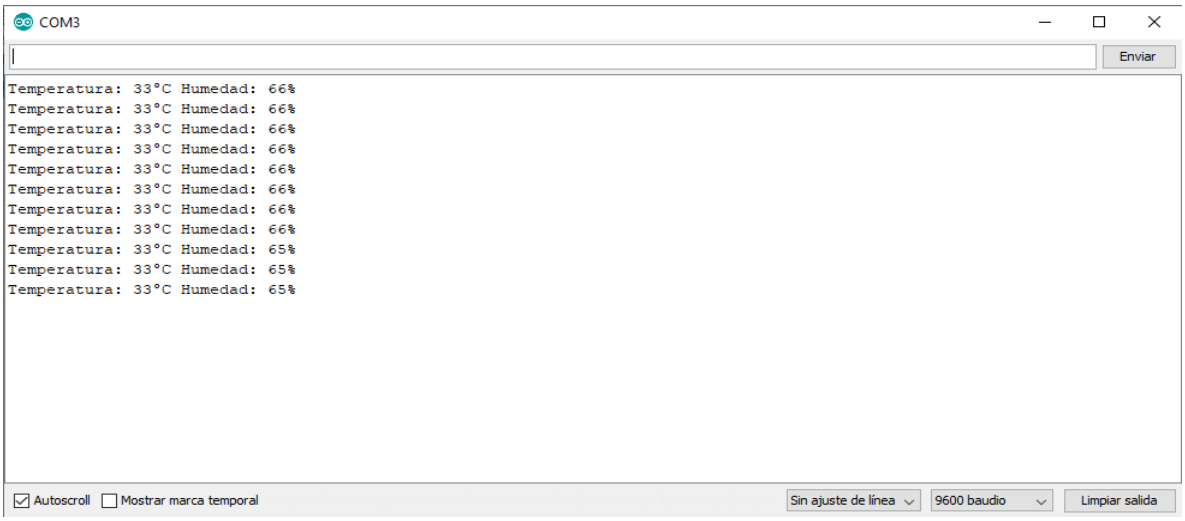
```
#include <DHT.h>


int sensor = 2;
int temperatura, humedad;

DHT dht (sensor, DHT11);
 void setup()
 {
  Serial.begin(9600);
  dht.begin();
 }
void loop()
{
  humedad = dht.readHumidity();
  temperatura = dht.readTemperature();

  Serial.print("Temperatura: ");
  Serial.print(temperatura);
  Serial.print("°C Humedad: ");
  Serial.print(humedad);
  Serial.println("%");
  delay(500);

}
```

Inclusión de comandos para realizar lecturas de temperatura y humedad e impresión de datos.

Fuente: (Autor, 2020)

Y así quedaria la programacion, lo subimos a la placa y abrimos el monitor serial y veremos los datos que capta el sensor.

Figura93: Monitor serial practica 6



Vista de resultados en el monitor serie.

Fuente: (Autor, 2020)

Ahora veremos su conexionado.

Figura94: Diagrama practica 6

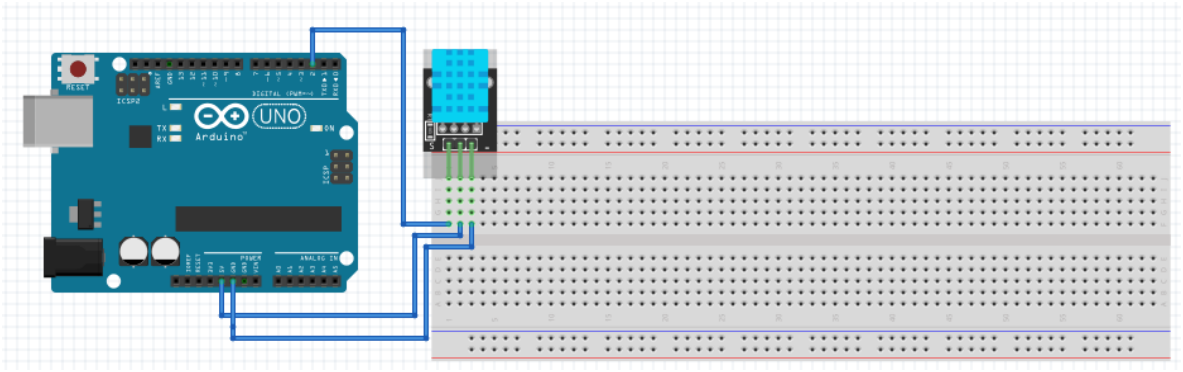
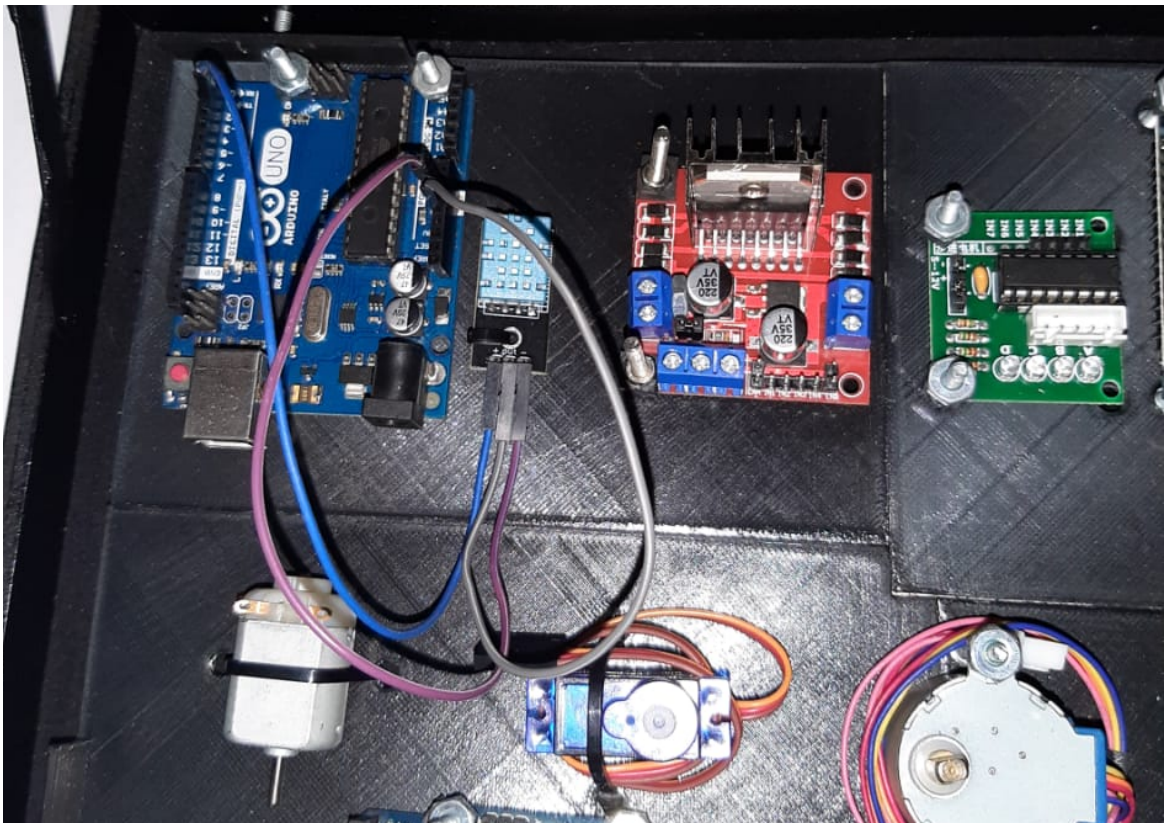


Diagrama de conexiones del dht a sus respectivos pines.

Fuente: (Autor, 2020)

Figura 95: Montaje practica 6



Vista real del conexionado presentado en la figura 95

Fuente: (Autor, 2020)

BIBLIOGRAFIA

(s.f.). Obtenido de Aprendiendo Arduino: https://aprendiendoarduino.wordpress.com/2017/06/20/estructuras-de-control-3/

(18 de 01 de 2020). Obtenido de Descubre Arduino: https://descubrearduino.com/estructura-de-programa/

Aprendiendo Arduino. (s.f.). Obtenido de https://aprendiendoarduino.wordpress.com/2017/10/14/estructuras-de-control-4/

Arduino. (s.f.). Obtenido de Variables: https://www.arduino.cc/en/Tutorial/Variables

Arduino. (21 de febrero de 2019). Obtenido de https://www.arduino.cc/reference/en/language/variables/data-types/unsignedint/

Crespo, M. D. (s.f.). Obtenido de http://manueldelgadocrespo.blogspot.com/p/description-digital-pins-with.html?m=1

panamahitek. (s.f.). Obtenido de http://panamahitek.com/el-setup-y-el-loop-en-arduino/

Programar facil. (s.f.). Obtenido de https://programarfacil.com/blog/arduino-blog/if-else-arduino/

Valle, L. d. (s.f.). *Programar Facil.* Obtenido de https://programarfacil.com/blog/arduino-blog/sensor-dht11-temperatura-humedad-arduino/

Wikipedia. (23 de 09 de 2019). *Arduino*. Obtenido de https://es.wikipedia.org/w/index.php?title=Arduino&oldid=119633331

8.1.10. Practica 7: Acondicionamiento de señal y control de servomotor.

COMPETENCIA	RESULTADOS DE APRENDIZAJE
Conocer algoritmos básicos de programación para el control y configuración de puertos en un microcontrolador. Diseñar aplicaciones orientadas a la transmisión y recepción de datos a sistemas periféricos con base en microcontroladores. Manejo de periféricos usando puertos.	Utiliza el lenguaje de programación para desarrollar aplicaciones que involucren el manejo de puertos. Desarrolla aplicaciones en las cuales se implementan circuitos de control de tiempos. Identifica las aplicaciones electrónicas que necesitan la transmisión y recepción de datos digitales entre circuitos integrados.

Desarrollo de algoritmos básicos aplicados a circuitos en contextos específicos con el microcontrolador utilizando puertos.	Explica el funcionamiento del módulo de comunicaciones y el uso de sus registros asociados, en el desarrollo de programas.

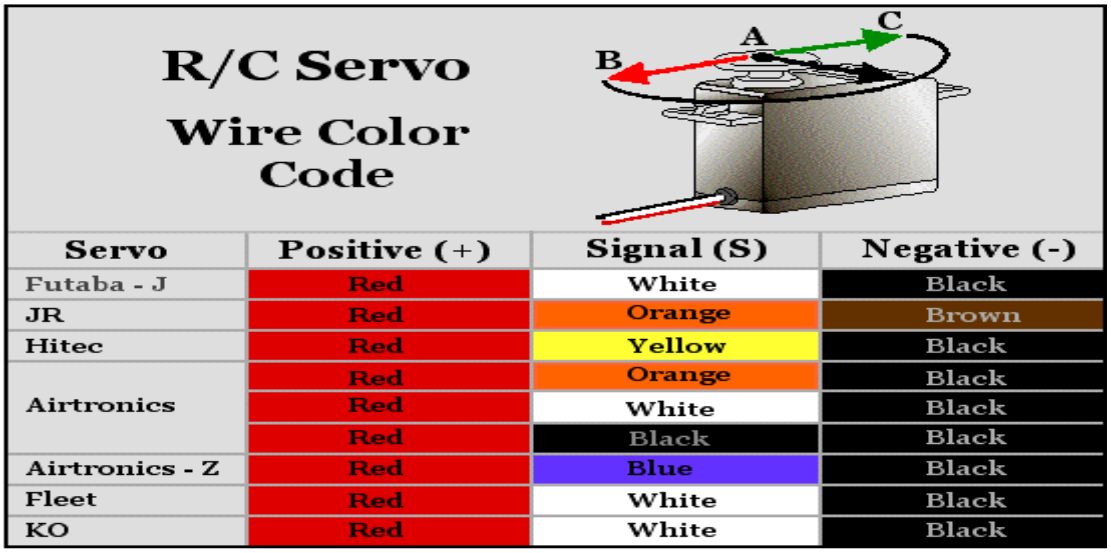
MATERIALES

- Arduino uno
- Cables
- Servomotor Tower pro SG90
- Protoboard
- Potenciómetro

En cualquier proyecto electromecánico es importante la elección de un motor para la ejecución de procesos que deseemos hacer, cuando se necesita precisión y desempeño los servos son la una de las mejores opciones.

Para esta práctica necesitaremos un servomotor, protoboard, potenciómetro de 100k y una placa Arduino.

Figura96: Colores de entradas del servomotor



Servo	Positive (+)	Signal (S)	Negative (-)
Futaba - J	Red	White	Black
JR	Red	Orange	Brown
Hitec	Red	Yellow	Black
Airtronics	Red	Orange	Black
	Red	White	Black
	Red	Black	Black
Airtronics - Z	Red	Blue	Black
Fleet	Red	White	Black
KO	Red	White	Black

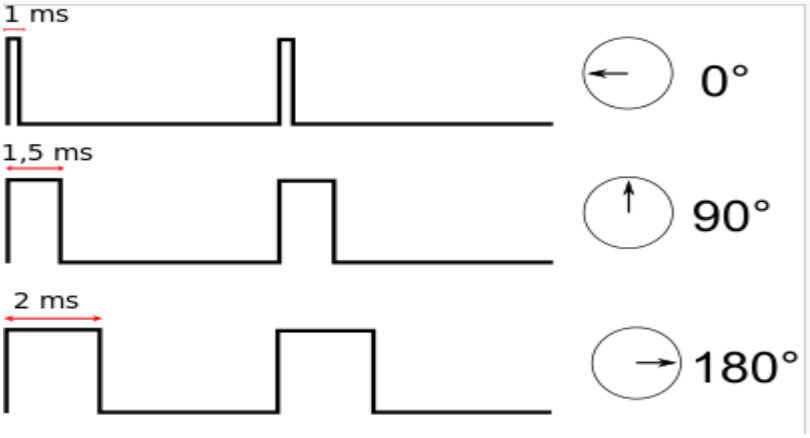
Colores de los cales del servomotor y a que pin pertenecen.

Fuente: (casado, s.f.)

Empecemos explicando las entradas del servo, este consta de tres, pueden variar respecto a colores y se determina cual es el positivo, que es por donde va la alimentacion.

El negativo que vendria siendo tierra y signal es por donde este recibirá indicaciones.

Figura97: Señal de posicionamiento del servomotor



De acuerdo a la figura 97 se puede observar que modificando el ancho de la señal podemos controlar la posición del servomotor.

Fuente: (casado, s.f.)

La mayoria servomotores permiten un giro de 0 a 180 grados es decir que si le enviamos un pulso de 1.5 milisegundos de duracion el servomotor girará a un angulo de 90°, en resumen, modulando el ancho del pulso entre uno y dos milisegundos tendremos la totalidad de su rango de giro de 0 a 180 grados.

Ahora vamos a la programacion, arduino incluye una librería especial para servomotores la cual nos facilitará en gran media a programacion del servo.

Debemos incluir la librería de servo asi que escrbimos: #include <Servo.h>, ahora escribimos la siguiente linea: Servo servomotor1, este es el nombre que usaremos para referenciar nuestro servo, puede ser cualquier otro nombre (“servomotor1”), no debemos olvidar esta linea pues al crear el objeto obtrendremos que nos ayudaran a controlar el servo de manera mas facil.

Ahora declaramos una variable int PINSERVO = 2 , int PULSOMIN = 1000, INT PULSOMAX = 2000.

A la variable PINSERVO le asignamos el numero 2 por que es el pin al cual va conectado el servo en el arduino despues tenemos PULSOMIN y PULSOMAX que como pudimos observar anterior mente 1 milisegundo equivale a 0° y 2 milisegundos equivalen a 180° pero lo expresaremos en microsegundos, por eso el 1000 y el 2000.

Ahora vamos al set up escribiremos servomotor1.attach y entre parentesis PINSERVO, PULSOMIN, PULSOMAX.

La funcion attach sirve para inicializar el servo, el primer parametro será el pin donde está conectado el servo, el segundo el valor del pulso minimo y el tercero el valor del pulso maximo.

La funcion attach requiere que los valores de los pulsos esten en microsegundos, por eso colocamos los valores de esa manera anteriormente.

Figura98: Sketch practica 7

```
#include <Servo.h>

Servo servomotor1;

int PINSERVO = 2;
int PINMIN = 1000;
int PINMAX = 2000;

void setup(){
  servomotor1.attach(PINSERVO, PINMIN, PINMAX);
}
```

Inclusión de librerías, declaración de nombre del servo, declaración de variables, comandos de la librería Servo.h

Fuente: (Autor, 2020)

Ahora nos vamos al void loop escribimos servomotor1.write(0).

La funcion write envia el angulo al cual debe posicionarse el servo, en este caso 0° y eso es todo, la funcion write se encarga de modular la frecuencia con la duracion y ancho del pulso adecuados.

Ahora despues de esa linea demosle un delay de 5000 y escribimos servomotor1.write(180) con un delay de 5000.

Despues de 5 segundos de la primera funcion write enviamos al servo a una posicion de 180°, despues se ejecutará la ultima linea del loop que es el delay devolviendose de nuevo al pincipio, es decir, estaremos haciendo que el servomotor se mueva de 0 a 180° en intervalos de 5 segundos y listo esa seria la programacion.

Figura99: Sketch practica 7

```
void loop(){
  servomotor1.write(0);
  delay(5000);
  servomotor1.write(180);
  delay(5000);
}
```

Declaración de tiempo en el cual el servomotor pasará de estar en una posición 0° a una 180°.

Fuente: (Autor, 2020)

Ahora, tendriamos que hacer un pequeño ajuste pues en todos los motores el pulso minimo y maximo no son los mismos, como por ejemplo para el que estamos programando su intervalo es de mil a dos mil que nos indicaria el 0 a 180° habrá alguno que su intervalo sea de por ejemplo 500 a 2400 esto lo podremos saber del fabricante el cual nos indicará el pulso minimo y maximo del motor o verificando a prueba y error hasta que sintamos el tope mecanico del servo.

Ahora montando su conexionado.

Figura100: Diagrama servomotor

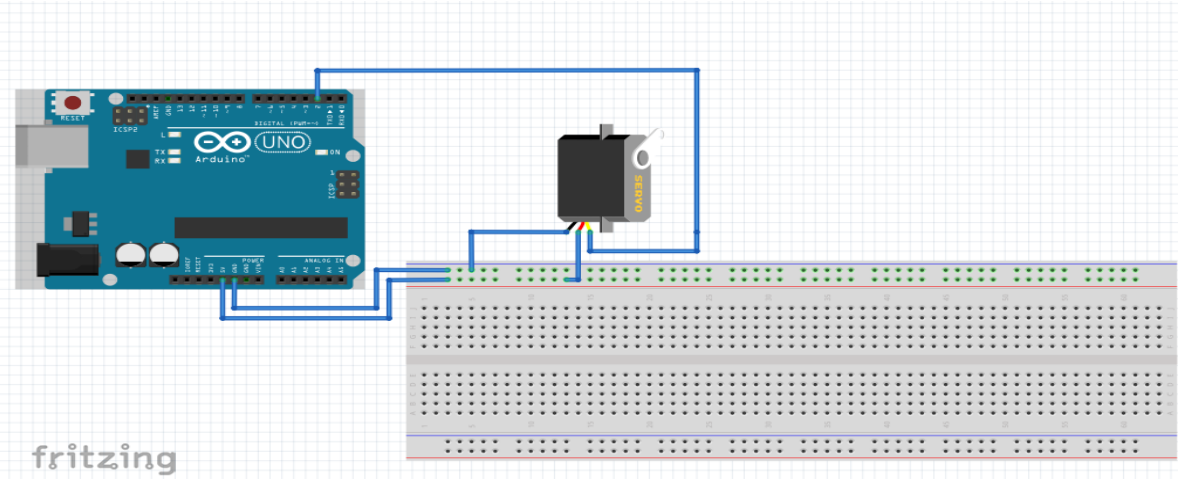


Diagrama de conexiones del servomotor a la placa Arduino uno, siguiendo la indicación de la figura 86.

Fuente: (Autor, 2020)

Es una conexión sencilla y con esta podremos ver el funcionamiento del servo.

Ahora vamos a hacer el programa que usa el potenciometro para generar el angulo de giro del servomotor.

Despues de la declaracion de variables int PINSERVO = 2, int PULSOMIN = 1000, INT PULSOMAX = 2000, difiniremos dos variables que se necesitan y el pin donde está conectado el potenciometro.

Escribiremos int VALORPOT, int ANGULO, int POT = 0.

Cabe recordar que las entradas analogicas no requieren de una inicializacion.

Borramos el codigo que estaba en el void loop y agregamos lo siguiente: VALORPOT = analogread(POT), despues digitamos ANGULO = map(VALORPOT, 0, 1023, 0, 180), despues escribimos servomotor1.write(ANGULO) y para concluir le agregamos un delay de 20.

Usaremos la funcion analog read para leer la entrada analogica A0 a la cual conectaremos el potenciometro y lo asignaremos a la variable VALORPOT.

Ahora utilizamos una funcion nueva llamada map. Map realiza la conversión de un valor de un determinado rango a otro valor proporcional en otro rango.

El primer parametro es el valor que deseamos convertir en este caso el valor convertido del potenciometro, los siguientes dos parametros son los valores que puede tomar la variable, como se trata de una entrada analogica se sabe que esta varia desde 0 a 1023 por eso se incluyen, los ultimos dos parametros es el rango al cual debe convertirse, como la funcion write toma valores de 0 a 180 se ponen dichos valores, el resultado de la funcion map se asigna a la variable ANGULO.

Hemos convertido los valores que nos entrega dicho potenciometro a traves de la entrada analogica a un valor de angulo el cual se aplicará al servo, finalmente a la funcion servomotor1.write le asignamos el valor de angulo y un delay pequeño de 20 el cual sirve para darle tiempo al servo de regresar a su posicion este numero varia en funcion de la carga que le apliquemos al servo.

Finalmete asi quedarian las lineas de codigo.

Figura101: Sketch practica 7

```
#include <Servo.h>

Servo servomotor1;

int PINSERVO = 2;
int PINMIN = 500;
int PINMAX = 2400;
int VALORPOT;
int ANGULO;
int POT = 0;

void setup(){
  servomotor1.attach(PINSERVO, PINMIN, PINMAX);
}

void loop(){
  VALORPOT = analogRead(POT);
  ANGULO = map(VALORPOT, 0, 1023, 0, 180);
  servomotor1.write(ANGULO);
  delay(20);
}
```

De acuerdo a la figura 101 se incluyen las modificaciones a la figura 98 y 99.

Fuente: (Autor, 2020)

Y asi quedaria la conexión del potenciometro.

Figura102: Diagrama servomotor con potenciómetro

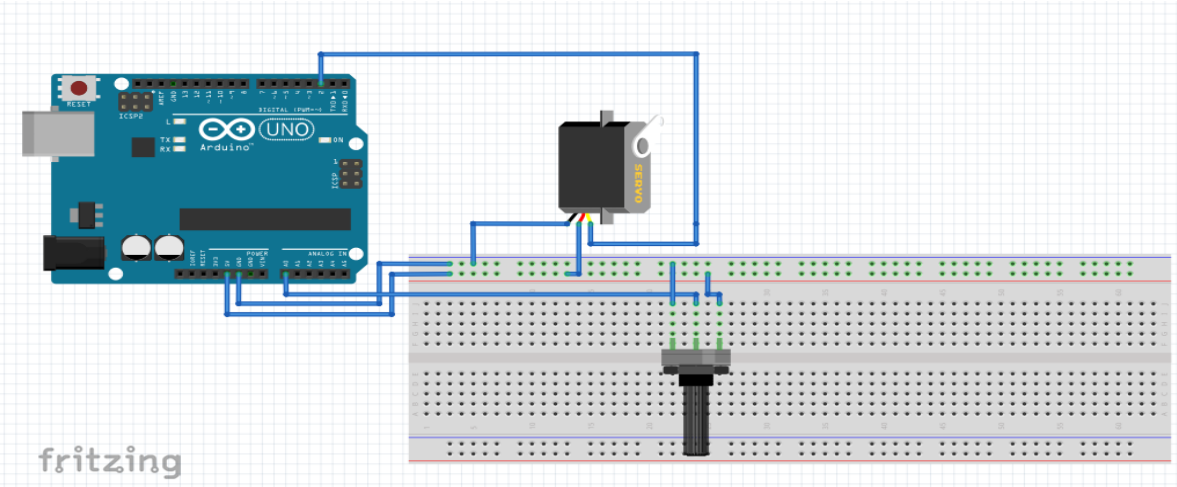
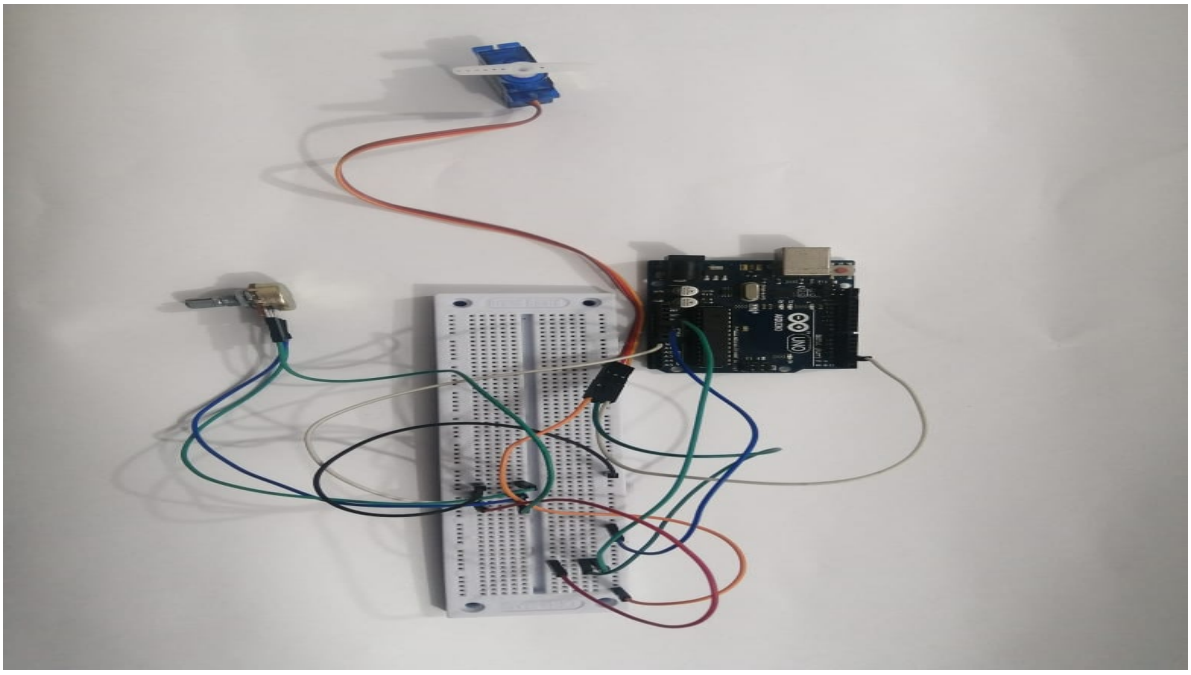


Diagrama de conexiones del servo con potenciómetro.

Fuente: (Autor, 2020)

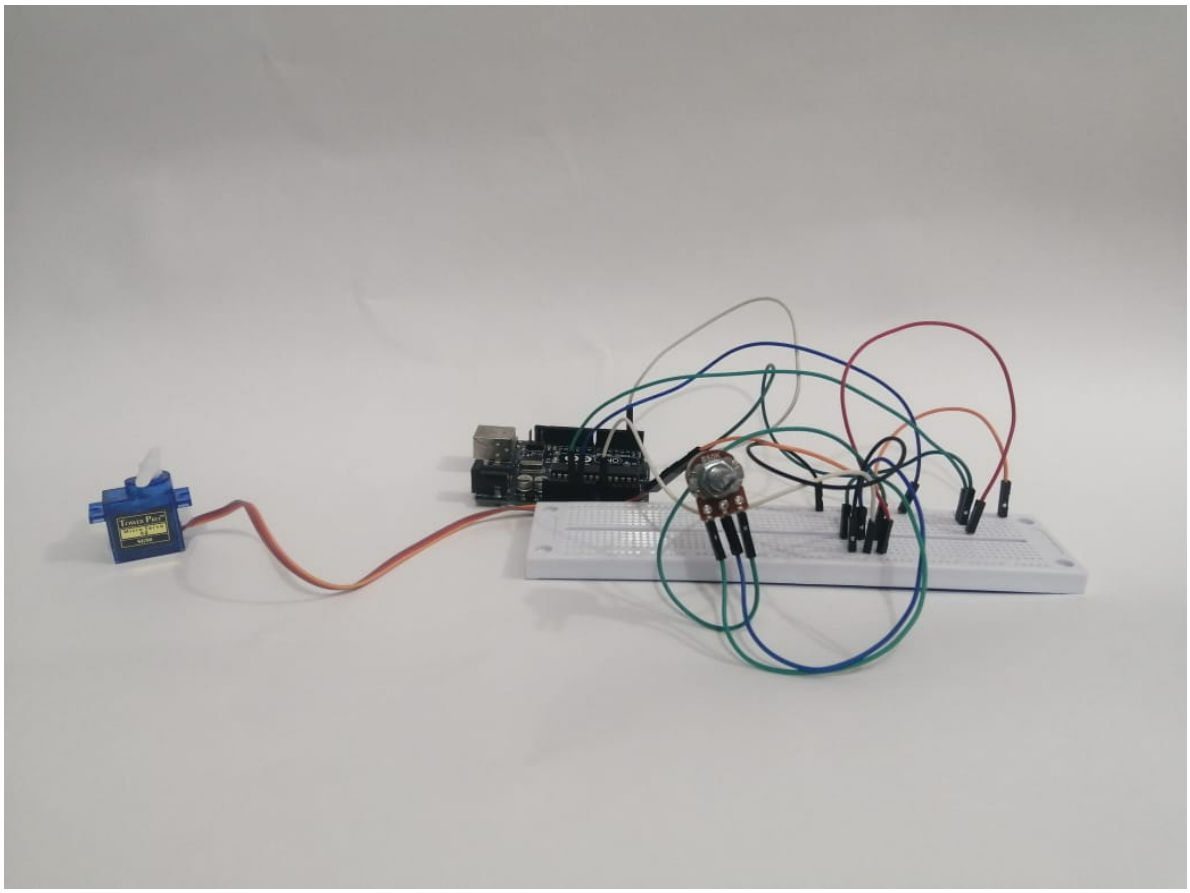
Ahora una vista real.

Figura103: Montaje 1 practica 7.



Fuente: (Autor, 2020)

Figura104: Montaje 2 practica 7.



Vista real del conexionado presentado en la figura 92

Fuente: (Autor, 2020)

BIBLIOGRAFIA

(s.f.). Obtenido de Aprendiendo Arduino: https://aprendiendoarduino.wordpress.com/2017/06/20/estructuras-de-control-3/

(18 de 01 de 2020). Obtenido de Descubre Arduino: https://descubrearduino.com/estructura-de-programa/

Aprendiendo Arduino. (s.f.). Obtenido de

https://aprendiendoarduino.wordpress.com/2017/10/14/estructuras-de-control-4/

Arduino. (s.f.). Obtenido de Variables: https://www.arduino.cc/en/Tutorial/Variables

Arduino. (21 de febrero de 2019). Obtenido de https://www.arduino.cc/reference/en/language/variables/data-types/unsignedint/

Crespo, M. D. (s.f.). Obtenido de http://manueldelgadocrespo.blogspot.com/p/description-digital-pins-with.html?m=1

Gomez, M. A. (26 de 12 de 2016). *Arduino a Muete.* Obtenido de http://arduinoamuete.blogspot.com/2016/12/introduccion-blynk.html

panamahitek. (s.f.). Obtenido de http://panamahitek.com/el-setup-y-el-loop-en-arduino/

Programar facil. (s.f.). Obtenido de https://programarfacil.com/blog/arduino-blog/if-else-arduino/

Valle, L. d. (s.f.). *Programar Facil.* Obtenido de https://programarfacil.com/tutoriales/fragmentos/servomotor-con-arduino/

Wikipedia. (23 de 09 de 2019). *Arduino*. Obtenido de https://es.wikipedia.org/w/index.php?title=Arduino&oldid=119633331

Wikipedia. (15 de enero de 2020). Obtenido de https://es.wikipedia.org/wiki/Servomotor_de_modelismo

Zgz maker space. (2019). Obtenido de https://zaragozamakerspace.com/index.php/lessons/curso-arduino-y-robotica-servomotores/

8.1.11. Practica 8: Acondicionamiento de señal y control de motor paso a paso.

COMPETENCIA	RESULTADOS DE APRENDIZAJE

Conocer algoritmos básicos de programación para el control y configuración de puertos en un microcontrolador. Diseñar aplicaciones orientadas a la transmisión y recepción de datos a sistemas periféricos con base en microcontroladores. Manejo de periféricos usando puertos. Desarrollo de algoritmos básicos aplicados a circuitos en contextos específicos con el microcontrolador utilizando puertos.	Utiliza el lenguaje de programación para desarrollar aplicaciones que involucren el manejo de puertos. Desarrolla aplicaciones en las cuales se implementan circuitos de control de tiempos. Identifica las aplicaciones electrónicas que necesitan la transmisión y recepción de datos digitales entre circuitos integrados. Explica el funcionamiento del módulo de comunicaciones y el uso de sus registros asociados, en el desarrollo de programas.

MATERIALES

- Arduino uno
- Cables
- Motor paso a paso unipolar 28BYJ-48
- Controlador de motores ULN2003

Utilizaremos un motor paso a paso unipolar ya que es más sencillo y se nos facilitará su compresión más rápidamente y se obtendrán competencias para un mejor entendimiento del motor paso a paso bipolar.

¿Qué es un motor paso y paso y para qué sirve?

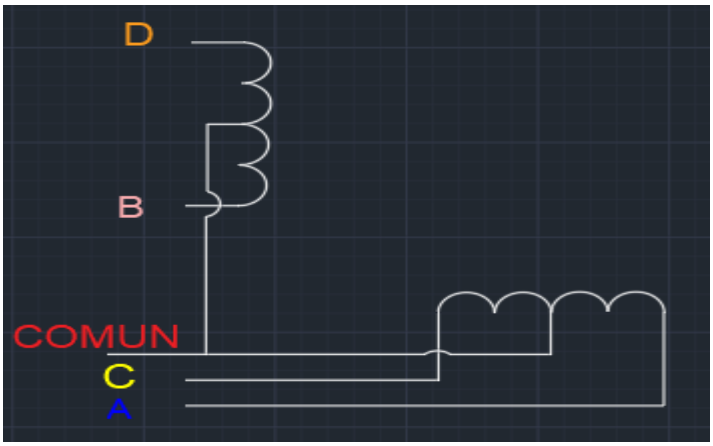
Es un artefacto con caracteristicas electromecanicas que convierte impulsos electricos en desplazamiento angular, lo cual traduce qu es capaz de girar cierta cantidad de grados, todo esto dependiendo de sus entradas de control, una de las ventajas de este motor es que es muy preciso y repetitivo en cuanto al posicionamiento.

Entre sus aplicaciones se destaca su uso en robotica, radiocontrol, impresoras digitales, autmatizacion, etc.

Como existen motores con caracteristicas diferentes, es importante utilizar representaciones graficas como la siguiente.

Diagrama unipolar de 5 hilos

Figura105: Bobinas de un motor paso a paso unipolar



Representación del conexionado entre bobinas.

Fuente: (Autor, 2020)

El circulo corresponde al rotor, en el costado izquierdo tenemos dos bobinas y en la parte inferior otraas dos. Los extremos de cada boniba se unen para formar un hilo común, por esta razon a este motor se le denomina unipolar.

Con los diferentes hilos denominados con las letras: D, B, C y A, estableceremos a traves de ellos una secuencia especifica para lograr el giro del rotor.

D = naranja, B = rosa, C = amarillo, A = azul, Rojo = comun

Las caracteristicas del Motor paso a paso unipolar 28BYJ-48 son:

Figura 106: Motor paso a paso

Fuente: (Yorobotics, 2020)

- Alimentación: 5 VDC
- Fases: 4
- Consumo: 40 mA por bobina

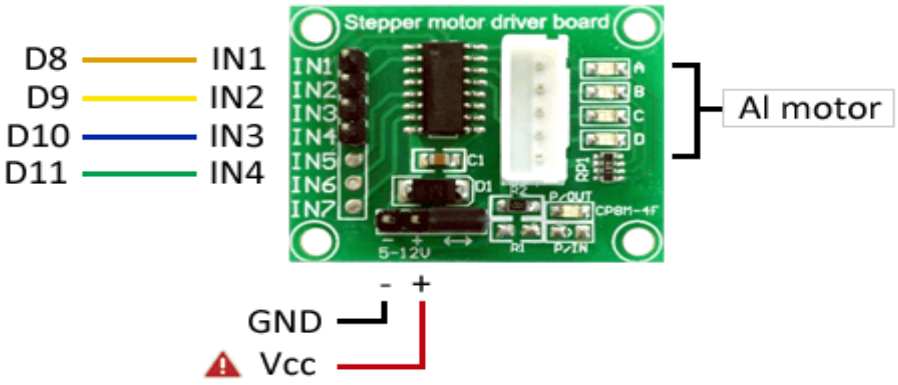
- Reducción mecánica: 1:64
- Frecuencia máxima de operación: 100Hz
  (Demora 10 ms)

La reducción mecánica quiere decir que por cada 64 vueltas que da eje del rotor, el eje externo da 1.

Para el paso a paso necesitaremos energizar las bobinas en una determinada secuencia y el rotor no gira constantemente, lo hace solo de a pequeños pasos, por eso el valor de frecuencia máxima de 100hz indica que la aplicación de los pulsos a cada bobina no debe ser menor a 10ms porque pues el periodo de una señal es el inverso de la frecuencia de alii que 100hz sea un periodo de 10 ms.

Ahora veamos la necesidad del controlador de motores ULN2003.

Figura107:Controlador de motores ULN2003.



Características del controlador de motores ULN2003.

Fuente: (Llamas, 2020)

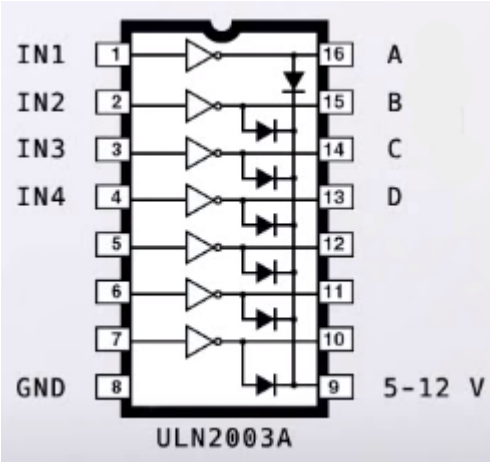
El consumo de corriente por bobina es demasiado elevado para conectarlo a un pin digital de Arduino, el corazón del controlador es el circuito integrado modelo ULN2003 que nos permitirá mediante sus pines de entrada in 1 a in 4 controlar sus respectivas salidas que permiten una gran circulación de corriente, desde esa manera desde Arduino enviamos niveles lógicos con bajo consumo de corriente y luego el integrado se encarga de activar las distintas bobinas del motor con un suministro de corriente adecuado.

El ULN2003 dispone de 7 líneas de control, pero este módulo, pero este módulo específicamente está pensando para nuestro motor que es de 5 hilos y uno de ellos es el común, con lo cual solo tendremos pines de entrada in 1 a in 4 y salida mediante el conector hacia las bobinas A, B, C y D.

Entre cada entrada y salida veras la representación gráfica un triángulo y luego un pequeño circulo, el triángulo es lo que en digitales se denomina un bajar y el circulo indica que existe una inversión en el nivel lógico, en pocas palabras, si aplicamos un uno lógico en in 1 obtendremos sobre la salida A, un cero lógico, regresivamente si en in 1 aplicamos un cero lógico, en la salida obtendremos un uno lógico, siempre el nivel lógico opuesto. Es una característica de este tipo de drive que además es un inversor.

El hilo común del motor se conecta al positivo de tensión, es el cable de color rojo, de manera que para energizar una bobina debemos indicarle un cero lógico para cerrar el circuito, así que desde Arduino enviaremos un nivel alto a una de las entradas, el ULN2003, invierte el nivel lógico a uno bajo y de esta forma logramos energizar la bobina y desde el código trabajaremos con lógica positiva y dejamos al ULN2003 que invierta el nivel.

Figura108: Diagrama del controlador de motores ULN2003.



Fuente: (Ar, 2017)

Ahora, debemos entender la secuencia que debe aplicarse a las bobinas para lograr el giro del eje, como se pudo observar anteriormente tenemos cuatro cables con las denominaciones A, B, C y D que corresponden a las cuatro fases del motor la secuencia es muy simple.

Tabla 3: Paso simple del motor paso a paso

Paso	A	B	C	D
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

Secuencia simple de energización para el paso a paso.

Fuente: (Autor, 2020)

El primer paso se logra enviando un 1 logico a la bobina A, dejando en cero el resto, luego un uno logico a la bobina B; el resto en cero y asi susesivamente como se muestra en la tabla anteriormente mostrada, la cual dice que enviamos un nivel alto a cada bobina en el orden A, B, C y D.

Existen un total de 3 formas de energizar las bobinas de un motor para lograr un giro de este, la que vimos anteriormente es la mas simple y se la denomina wave drive o paso completo simple, ya que energizamos una bobina a la vez.

Es la de paso completo con con dos bobinas y esta es la recomendada por el fabricante ya que ofrece el maximo torque que este motor puede producir.

Tabla 4: Paso máximo torque del motor paso a paso

Paso	A	B	C	D
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1

Secuencia de máximo torque de energización de motor paso a paso.

Fuente: (Autor, 2020)

La idea es que al energizar dos bobinas al mismo tiempo, su campo hace que el rotor quede apuntando justo a la mitad de las dos bobinas, como ambas generan

un campo magnetico igual pero con 90° de diferencia, el rotor se situa justo entre las mismas, generando en el rotor mas impulso.

La tercera opcion de secuencia se denomina de medio paso en la cual tendremos un total de 8 pasos para completar el ciclo y esto permite que el rotor gire con un angulo menor haciendo a este mas preciso en su girar, la secuencia es muy intuitiva y se deriva de las dos anteriormente vistas.

Tabla 5: Paso preciso del motor paso a paso

Paso	A	B	C	D
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1

Secuencia de energización de bobinas precisa paso preciso

Fuente: (Autor, 2020)

Ahora entendido todo esto vamos a la programacion.

Haremos solo el programa para la energizacion de las bobinas paso completo simple pues como el fucionamiento es el mismo que las demas y solo varia en el

numero de pasos y cuando activa o descativa las bobinas solo haremos una forma de energizacion.

Le declaramos cuatros variables int para IN1, IN2, IN3, IN4 a los cuales iran conectados a los pines digitales de arduino 8, 9 y 10 respectivamente y para una variable llamada demora lo conectamos al pin 11 y le cargamos un valor de 20, o sea una demora entre paso y paso de 20ms, recordemos que el valor minimo para el funcionamiento del motor paso a paso es de 10 ms asi que stamos utilizando 20 para no trabajar al limite.

Figura109: Sketch practica 8

```
int IN1 = 8;
int IN2 = 9;
int IN3 = 10;
int IN4 = 11;
int demora = 20;
```

Declaración de variables.

Fuente: (Autor, 2020)

Ahora vamos con el void setup, declaramos a los pines de las variables denominadas como IN1, IN2, IN3, IN4 como salidas.

Figura110: Sketch practica 8

```
void setup() {
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
}
```

Declaración de salidas.

Fuente: (Autor, 2020)

Ahora vamos con el void loop.

Cabe primero resaltar que IN1, IN2, IN3, IN4 son iguales a A, B, C y D de la tabla y programeremos los pasos en base a ella.

Tambien cabe decir que enviar los cuatro pasos no hace que el motor gire una vuelta completa, si bien en la tabla mostramos explicitamente 4 pasos, lo hicimos para comprender el funcionamiento del motor, en realidad para hacer que el motor de una vuelta completa debemos repetir la secuencia de 4 pasos 8 veces, a los cuatro pasos se le denomina un ciclo y para qyue el motor gire se necesitan 8 ciclos, pero ademas se debe tener en cuenta que el motor tiene una reduccion mecanica de 1 a 64, lo que quiere decir que para que el rotor exterior gire se debe multiplicar 4*8*64, esto quiere decir que para que el rotor exterior de una vuelta completa se necesitan 2048 pasos que es el resultado de 4*8*64.

Entonces respecto a la programacion implementamos la funcion for para que repita los 4 pasos que le vamos a progamar 512 veces por que 512*4 = 2048 para dar asi una vuelta completa.

Figura111: Sketch practica 8

```
void loop() {

  for (int i = 0; i < 512; i++)
  {
```

Ciclos de repetición for

Fuente: (Autor, 2020)

Y ahora con digitalWrite le decimo con digitalWrite que envie un nivel alto “HIGH” y los otros tres en un nivel bajo “LOW” y asi susesivamente con un delay(demora) que como declaramos anteriormente demora es igual a 20ms.

Despues cuando salga del ciclo for le indicamos que mande a IN1, IN2, IN3, IN4 estados bajos y le damos un delay de 5 segundos para que vuelva a repetirse el proceso.

Figura112: Sketch practica 8

```
    digitalWrite(IN1, HIGH);  // paso
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
    delay(demora);

    digitalWrite(IN1, LOW); // paso 2
    digitalWrite(IN2, HIGH);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
    delay(demora);

    digitalWrite(IN1, LOW); // paso 3
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    delay(demora);

    digitalWrite(IN1, LOW); // paso 4
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    delay(demora);
  }
```

Energización de bobinas del motor paso a paso.

Fuente: (Autor, 2020)

Figura113: Sketch practica 8

```
  digitalWrite(IN1, LOW); // detiene por 5 seg.
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  delay(5000);

}
```

Des energización de salidas.

Fuente: (Autor, 2020)

Ahora veamos su conexionado la cual es muy sencilla, no tiene mayor complicacion y tambien veamos una vista mas real de esta practica.

Figura114: Diagrama de practica 8

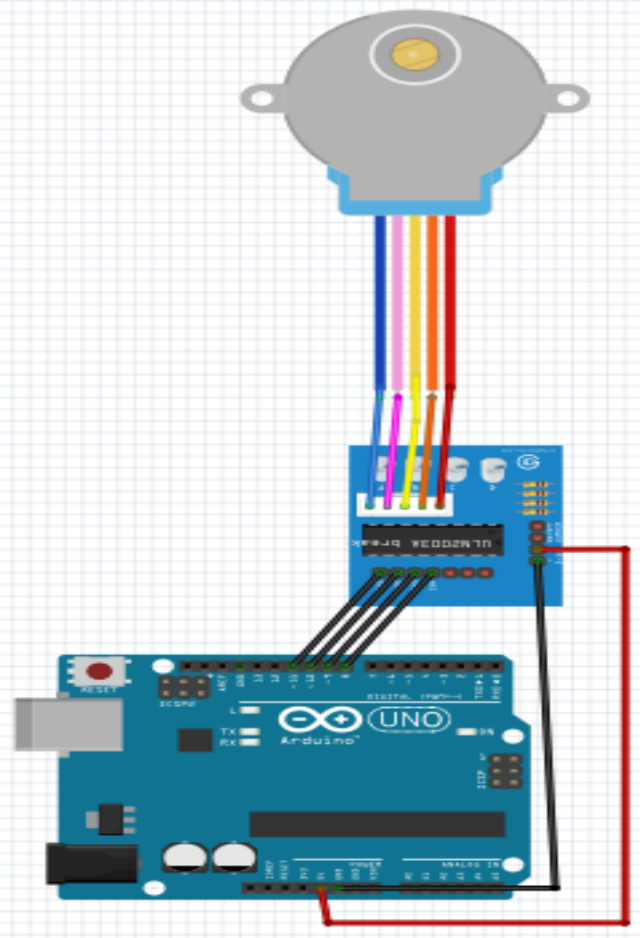
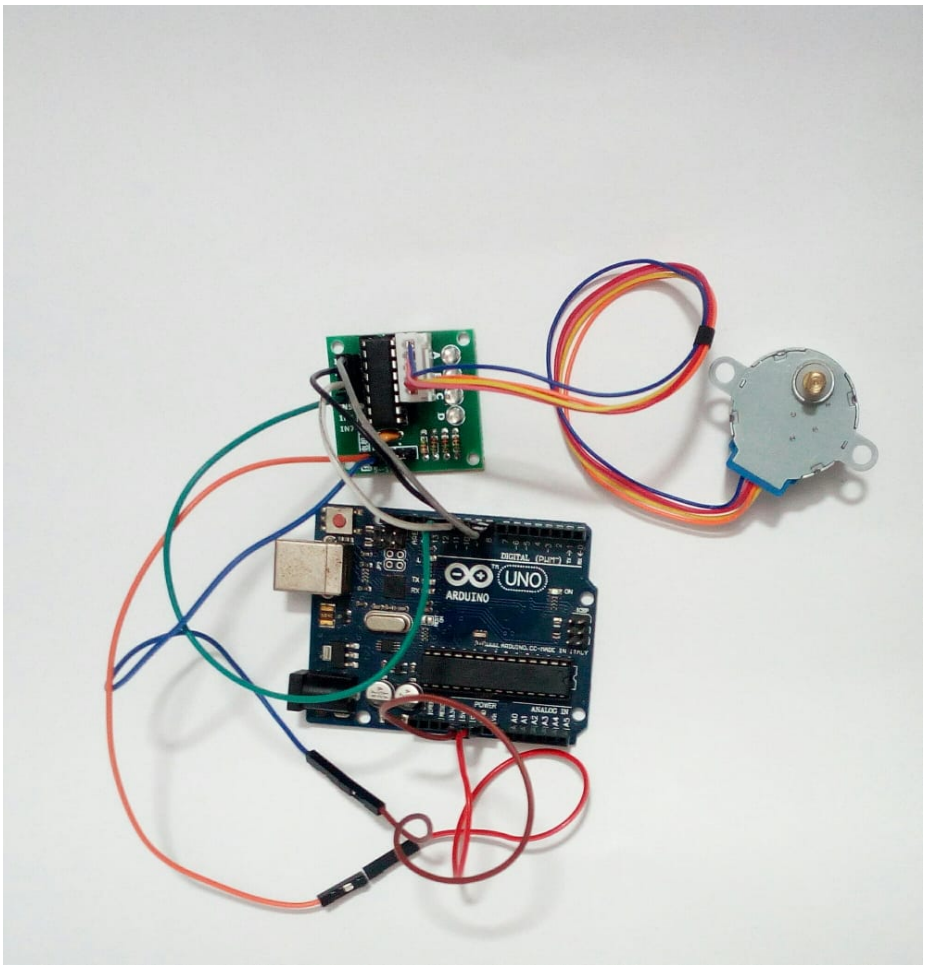


Diagrama de conexiones del paso a paso al controlador y de este a Arduino uno.

Fuente: (Autor, 2020)

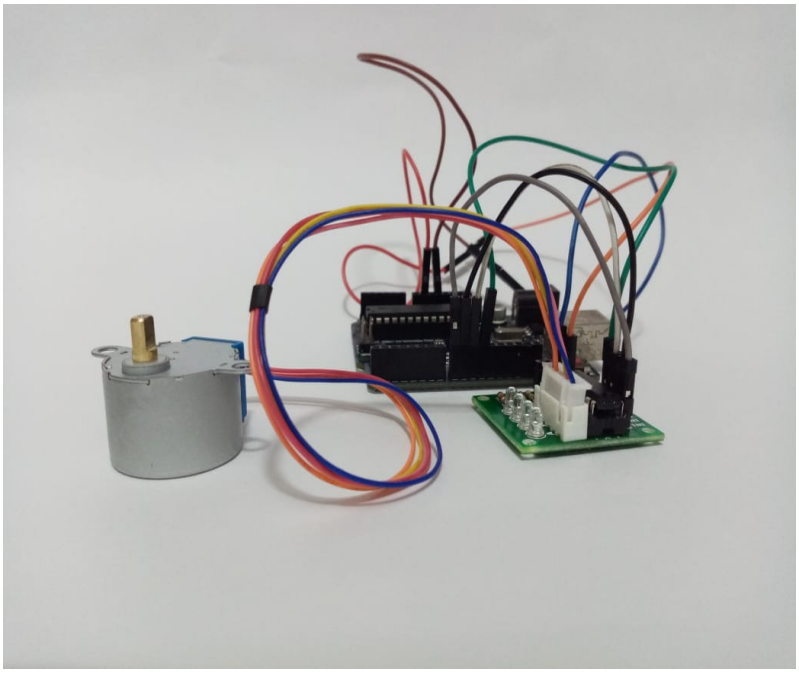
Figura115: Montaje 1 practica 8



Vista real del conexionado presentado en la figura 114.

Fuente: (Autor, 2020)

Figura116: Montaje 2 practica 8



Vista real del conexionado presentado en la figura 115.

Fuente: (Autor, 2020)

BIBLIOGRAFIA

(s.f.). Obtenido de Aprendiendo Arduino: https://aprendiendoarduino.wordpress.com/2017/06/20/estructuras-de-control-3/

(18 de 01 de 2020). Obtenido de Descubre Arduino: https://descubrearduino.com/estructura-de-programa/

330ohms. (s.f.). Obtenido de https://www.330ohms.com/products/motor-a-pasos-5v-28byj-48

Aprendiendo Arduino. (s.f.). Obtenido de https://aprendiendoarduino.wordpress.com/2017/10/14/estructuras-de-control-4/

Arduino. (s.f.). Obtenido de Variables: https://www.arduino.cc/en/Tutorial/Variables

Arduino. (21 de febrero de 2019). Obtenido de https://www.arduino.cc/reference/en/language/variables/data-types/unsignedint/

Crespo, M. D. (s.f.). Obtenido de http://manueldelgadocrespo.blogspot.com/p/description-digital-

pins-with.html?m=1

Gomez, M. A. (26 de 12 de 2016). *Arduino a Muete.* Obtenido de http://arduinoamuete.blogspot.com/2016/12/introduccion-blynk.html

Hernandez, L. d. (s.f.). *Programar facil.* Obtenido de https://programarfacil.com/blog/motor-paso-a-paso/

Llamas, L. (8 de Mayo de 2020). Obtenido de https://www.luisllamas.es/motor-paso-paso-28byj-48-arduino-driver-uln2003/

panamahitek. (s.f.). Obtenido de http://panamahitek.com/el-setup-y-el-loop-en-arduino/

Programar facil. (s.f.). Obtenido de https://programarfacil.com/blog/arduino-blog/if-else-arduino/

Valle, L. d. (s.f.). *Programar Facil.* Obtenido de https://programarfacil.com/tutoriales/fragmentos/servomotor-con-arduino/

Wikipedia. (23 de 09 de 2019). *Arduino*. Obtenido de https://es.wikipedia.org/w/index.php?title=Arduino&oldid=119633331

8.1.12. Practica 9: Acondicionamiento de señal y control de motor DC.

COMPETENCIA	RESULTADOS DE APRENDIZAJE
Conocer algoritmos básicos de programación para el control y configuración de puertos en un microcontrolador. Diseñar aplicaciones orientadas a la transmisión y recepción de datos a sistemas periféricos con base en microcontroladores.	Utiliza el lenguaje de programación para desarrollar aplicaciones que involucren el manejo de puertos. Desarrolla aplicaciones en las cuales se implementan circuitos de control de tiempos. Identifica las aplicaciones electrónicas que necesitan la transmisión y recepción de

Manejo de periféricos usando puertos. Desarrollo de algoritmos básicos aplicados a circuitos en contextos específicos con el microcontrolador utilizando puertos.	datos digitales entre circuitos integrados. Explica el funcionamiento del módulo de comunicaciones y el uso de sus registros asociados, en el desarrollo de programas.

MATERIALES

- Arduino uno
- Cables
- Motor DC
- Controlador de motores DC paso a paso L298N

¿Qué es el controlador de motores DC y paso a paso L298N?

El L298N es la denominación del circuito integrado delmódulo, se encuentra montado de manera vertical en el circuito impreso y tiene consigo un disipador de calor de aluminio pintado de color negro, este circuito integrado incluye transistores de potencia para el control de dos motores DC o un motor paso a paso, a este se le denomina doble puente H, a continuación, podemos observar el circuito de este controlador.

Figura117: Controlador de motores DC y paso a paso L298N.

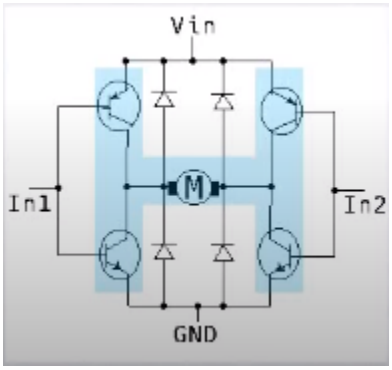


Diagrama de conexiones internas del controlador.

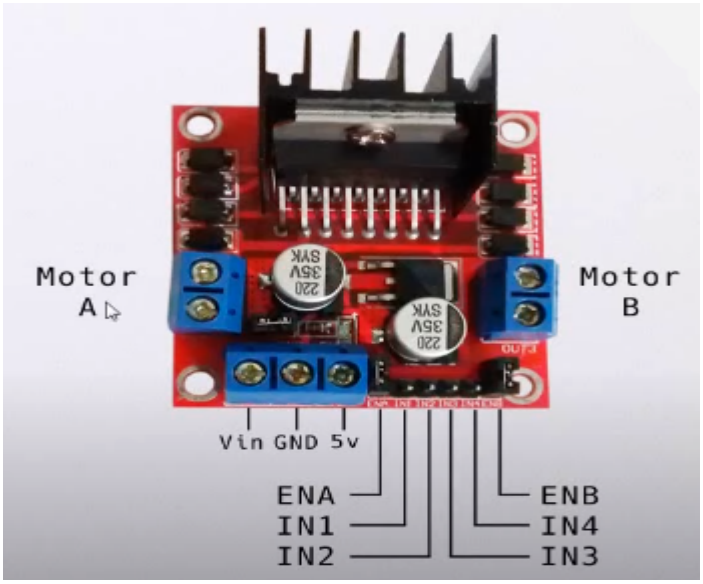
Fuente:(Ar, Youtube, 2017)

A este se le dice H por la posición en que se encuentran los cuatro transistores que manejan a un motor y con dicha distribución es posible controlar el giro del motor, mediante señales de control muy simples.

Este circuito integrado es un doble puente H así que en ese orden de ideas tendrá dos circuitos como el mostrado anteriormente, para controlar dos motores DC.

La conexión de los motores al módulo se realiza mediante dos borneras denominadas motor A y motor B, dichas salidas serán controladas por los pines ubicados en la parte inferior de este controlador y también tendrá bornera para alimentación, tierra y salida de 5v, los tres pines de la izquierda se usarán para controlar el motor A, empezando por el pin ENA, IN1, IN2 y los tres pines de la derecha se usarán para controlar el motor B.

Figura118: Controlador de motores DC y paso a paso L298N.



Características de los puertos del controlador.

Fuente:(Ar, Youtube, 2017)

Se puede observar que el modulo viene con dos jumpers sobre los pines ENA y ENB, estos son puentes que permiten un contacto electrico entre dos pines, estos jumpers lo que hacen es poner a los pines ENA y ENB, a un nivel alto, pero ENA y ENB son los pines que permiten habilitar a los motores, con estos pines a 5v habilitamos por defecto la salida de los motores, nosotros estaremos controlando ENA y ENB desde arduino asi que lo que debemos hacer es quitar esos jumpers ya que no se necesitan, para terminar con la explicacion tendremos IN1 y IN2 los cuales mediante niveles logicos distintos nos permitiran establecer el sentido de giro del motor A y asi respectivamente IN4 y IN3 el del motor B.

Tabla 6: Giro del motor dc.

ENA	IN1	IN2	MOTOR A
0			
1	0	1	
1	1	0	
ENA	IN3	IN4	MOTOR B
0			
1	0	1	
1	1	0	

Sentido de giro del motor dc según su energización.

Fuente: (Autor, 2020)

Paraque funcione el motor el pin ena debe estar en un nivel alto, tal como expresa la tabla, que, cuando este está en un nivel bajo no se activa el motor y cuando está alto si.

Tambien cabe recalcar que mediante PWM podremos controlar la velocidad de giro del motor.

Cuando IN1 o IN3 está en 0 y IN2 o IN4 estan en 1 los motores girarán a la derecha y cuando sea al contrario, girarán a la izquiera.

Ahora vamos con la programacion.

Declararemos variables, para IN1, IN2 , ENA.

En el setup le decimos a la ide de arduino que los pines que declaramos en nuestras variables serán salidas.

Figura119: Sketch practica 9.

```
int IN1 = 2;
int IN2 = 3;
int ENA = 5;

void setup(){
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(ENA, OUTPUT);
}
```

Declaración de variables y de salidas.

Fuente: (Autor, 2020)

Ahora vamos al voidloop, vamos a hacer girar el motor en un sentido y vemos que para hacer girar el motor en cualquier sentido ena debe estar en un nivel alto.

Si quisiesemos por ejemplo hacer que el motor gire a la derecha in 1 debe estar en 0 y in 2 en 1, entonces escribimos digitalWrite(ENA, HIGH), digitalWrite(IN1, LOW), digitalWrite(IN2, HIGH) y le damos un delay de tres segundos esto hará que el motor gire a la derecha.

Despues digitamos digitalWrite(ENA, LOW) y un delay de dos segundos para apagar el motor.

Ahora hagamos que el motor gire hacia la izquierda.

Digitamos, digital Write(ENA; HIGH) y ahí se enciende el moto, digitalWrite(IN1, HIGH), digitalWrite(IN2, LOW) para que el motor gire a la izquierda y le damos un delay de tres segundos.

Despues digitamos digitalWrite(ENA, LOW) y un delay de dos segundos para apagar el motor.

Figura120: Sketch practica 9

```
void loop(){
  digitalWrite(ENA, HIGH);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  delay(3000);
  digitalWrite(ENA, LOW);
  delay(2000);

  digitalWrite(ENA, HIGH);
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  delay(3000);

  digitalWrite(ENA, LOW);
  delay(2000);
}
```

Energización y sentido de giro del motor dc.

Fuente: (Autor, 2020)

Entonces basicamente hará este programa que el motor gire a la derecha, espere 3 segundos, se apague por dos segundos, se vuelva a encender y gire a la izquierda, se apague por dos segundos y asi susesivamente.

Al principio mencionabamos que ena se puede utilizar para el control de velocidad del motor, por eso ena lo conectamos al pin 5 el cual tiene capacidad pwm.

Mediantee pwm podemos enviar una señal para activar y desactivar el motor muy rapidamente, logrando el efecto en la reduccion de velocidad, para esto se tendrá que hacer un cambio sencillo en la programacion.

Lo que debemos hacer es cambiar digitalWrite a analogWrite al dna y luego asignarle un valor entre 0 y 255, siendo asi que 0 es una velocidad 0 y 255 velocidad maxima, un valor intermedio de 127 generaria una seañal pwm con ciclo de trabajo del 50% haciendo que el motor gire a la mitad de su velocidad maxima.

Para hacer un cambio develocidad gradual, podemos incluir en nuestra programacion el bucle for y modificamos el codigo asi.

Vamos al principio del codigo y declaramos una variable int y la llamaremos velocidad, en esta variable se almacenara el valor de la velocidad que será aplicada al pin ena.

Figura121: Sketch practica 9.

```
int IN1 = 2;
int IN2 = 3;
int ENA = 5;
int VELOCIDAD;
```

Declaración de variables.

Fuente: (Autor, 2020)

El void setup no cambia.

Ahora en el loop incluiremos el for (VELOCIDAD = 0, VELOCIDAD <256, VELOCIDAD ++)

Esto quiere decir que la velocidad minima es 0 y que la velocidad maxima es menor que 256 por ende 255 y velocidad++ es que iremos incrementando su velocidad en 1.

Entonces dentro de las llaves de ese ciclo for incluiremos analogwrite(ENA, VELOCIDAD), esto para que ena que es la que activa el motor, la que le da energia vaya aumentando de uno en uno gracias a la variable velocidad y a esto le damos un delay de 50 para que entre cada incremento espere esa cantidad de tiempo y se vuelva preceptible.

Despues apagamos el motor que estaba girando a la derecha y sigue la programacion de su giro hacia la izquierda

Figura122: Sketch practica 9.

```
for (VELOCIDAD = 0; VELOCIDAD < 256; VELOCIDAD++){
analogWrite(ENA, VELOCIDAD);
digitalWrite(IN1, LOW);
digitalWrite(IN2, HIGH);
delay(50);
}

digitalWrite(ENA, LOW);
delay(2000);
```

Ciclo for para control de velocidad del motor dc.

Fuente: (Autor, 2020)

Para el giro a la izquierda es exactamente lo mismo solo que se intercambian los valores de HIGH y LOW entre in1 y in2.

Figura123: Sketch practica 9.

```
  for (VELOCIDAD = 0; VELOCIDAD < 256; VELOCIDAD++){
  analogWrite(ENA, VELOCIDAD);
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  delay(50);
  }

  digitalWrite(ENA, LOW);
  delay(2000);
}
```

Control de velocidad con ciclo y energización.

Fuente: (Autor, 2020)

Ahora vamos con su conexionado.

Figura124: Diagrama practica 9.

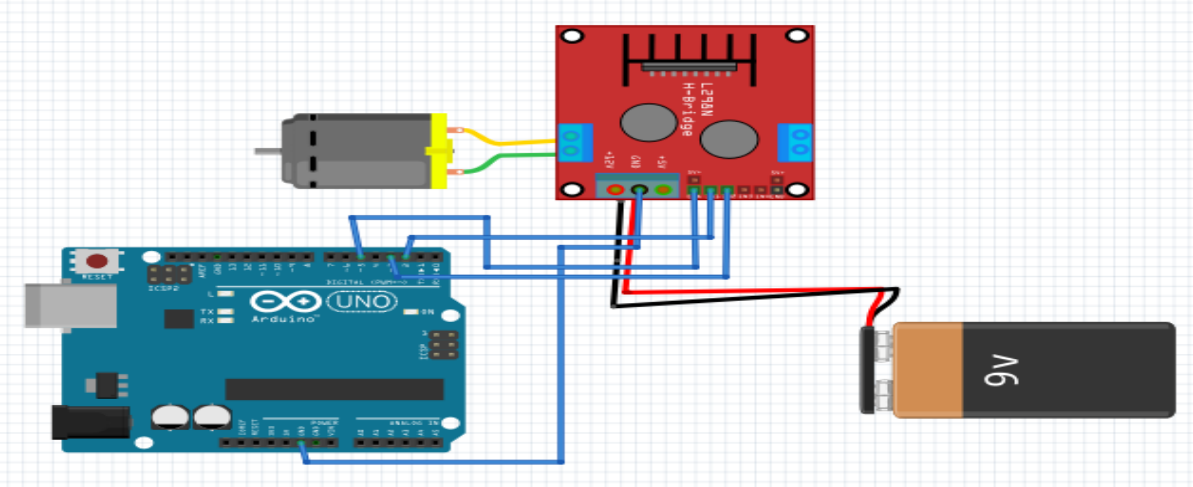


Diagrama de conexiones entre los diferentes componentes para el control y energización del motor dc.

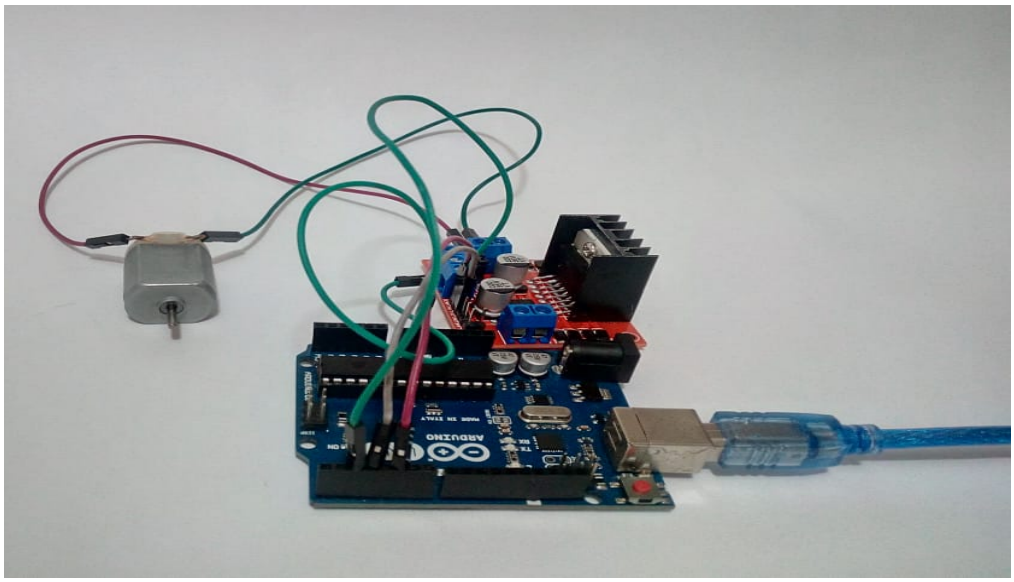
Fuente: (Autor, 2020)

Para esta práctica utilizamos un motor dc pequeño de aproximadamente 6v por lo tanto se propone una batería de 9v.

Se debe conectar la tierra del Arduino y del motor en un mismo sitio con el fin de equipotenciarlas y evitar errores de referencia.

Una vista un poco más real.

Figura125: Montaje practica 9.



Vista real del conexionado presentado en la figura 124.

Fuente: (Autor, 2020)

BIBLIOGRAFIA

(s.f.). Obtenido de Aprendiendo Arduino: https://aprendiendoarduino.wordpress.com/2017/06/20/estructuras-de-control-3/

(18 de 01 de 2020). Obtenido de Descubre Arduino: https://descubrearduino.com/estructura-de-programa/

Aprendiendo Arduino. (s.f.). Obtenido de https://aprendiendoarduino.wordpress.com/2017/10/14/estructuras-de-control-4/

Arduino. (s.f.). Obtenido de Variables: https://www.arduino.cc/en/Tutorial/Variables

Arduino. (21 de febrero de 2019). Obtenido de https://www.arduino.cc/reference/en/language/variables/data-types/unsignedint/

Ar, B. (18 de junio de 2017). Obtenido de Youtube: https://www.youtube.com/watch?v=63aitq3KTaI

Crespo, M. D. (s.f.). Obtenido de http://manueldelgadocrespo.blogspot.com/p/description-digital-

pins-with.html?m=1

Llamas, L. (26 de Mayo de 2016). Obtenido de https://www.luisllamas.es/arduino-motor-corriente-continua-l298n/

panamahitek. (s.f.). Obtenido de http://panamahitek.com/el-setup-y-el-loop-en-arduino/

Programar facil. (s.f.). Obtenido de https://programarfacil.com/blog/arduino-blog/if-else-arduino/

Wikipedia. (23 de 09 de 2019). *Arduino*. Obtenido de https://es.wikipedia.org/w/index.php?title=Arduino&oldid=119633331

8.1.13. Practica 10: Recepción de datos de temperatura y humedad a través de un módulo wifi a una aplicación vía internet.

COMPETENCIA	RESULTADOS DE APRENDIZAJE
Conocer algoritmos básicos de programación para el control y configuración de puertos en un microcontrolador. Diseñar aplicaciones orientadas a la transmisión y recepción de datos a sistemas periféricos con base en microcontroladores. Manejo de periféricos usando puertos. Desarrollo de algoritmos básicos aplicados	Utiliza el lenguaje de programación para desarrollar aplicaciones que involucren el manejo de puertos. Desarrolla aplicaciones en las cuales se implementan circuitos de control de tiempos. Identifica las aplicaciones electrónicas que necesitan la transmisión y recepción de datos digitales entre circuitos integrados. Explica el funcionamiento del módulo de

a circuitos en contextos específicos con el microcontrolador utilizando puertos.	comunicaciones y el uso de sus registros asociados, en el desarrollo de programas.

MATERIALES

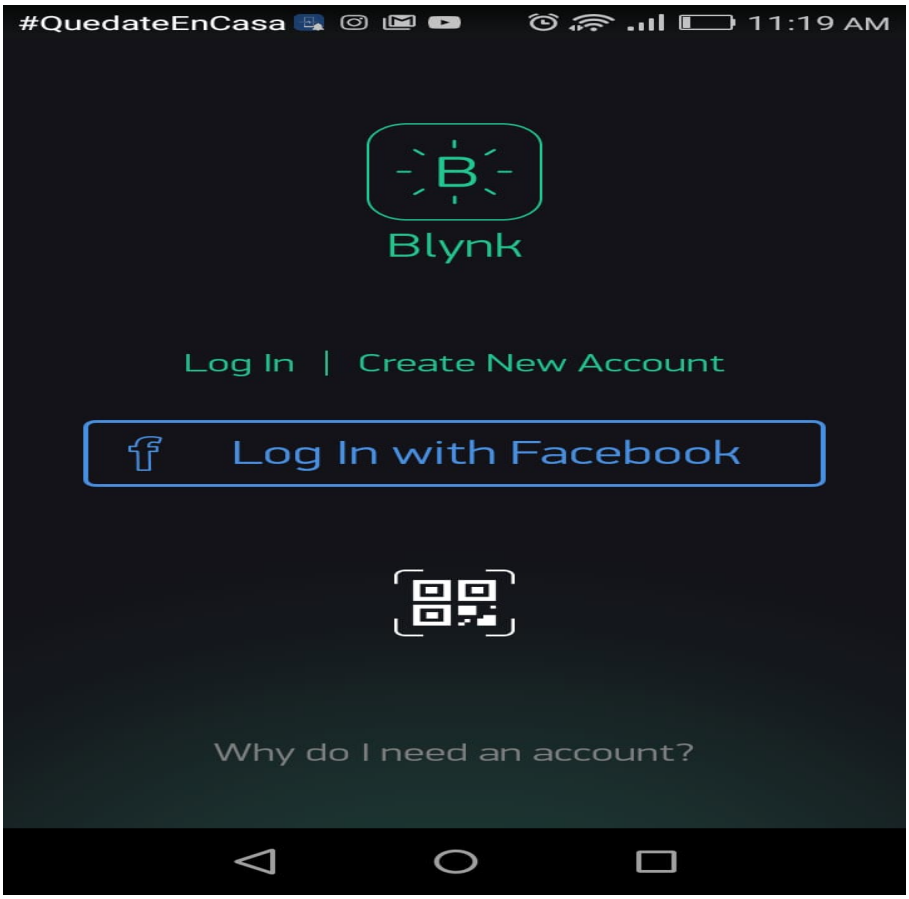
- Arduino uno
- Cables
- Modulo wifi esp8266 nodeMCU
- Sensor de humedad y temperatura DHT11

La app que utilizaremos se llama Blynk es que es una herramienta que nos permite crear una interface de control para controlar nuestros proyectos Arduino de manera remota, con esta app podremos controlar hardware, monitorear sensores que es nuestro caso y demás cosas muy útiles.

Primero que todo debemos instalar la aplicación de Blynk que se encuentra de manera gratuita en nuestra tienda de aplicaciones y la instalamos, después de tenerla instalada la abrimos y procedemos a acondicionar la aplicación, lo cual explicaremos a continuación.

Cuando abramos la aplicación nos saldrá por que cuenta queremos ingresar, si por una cuenta de correo o una de Facebook e ingresamos con alguna de ellas.

Figura126: Programación de la app en Blynk.

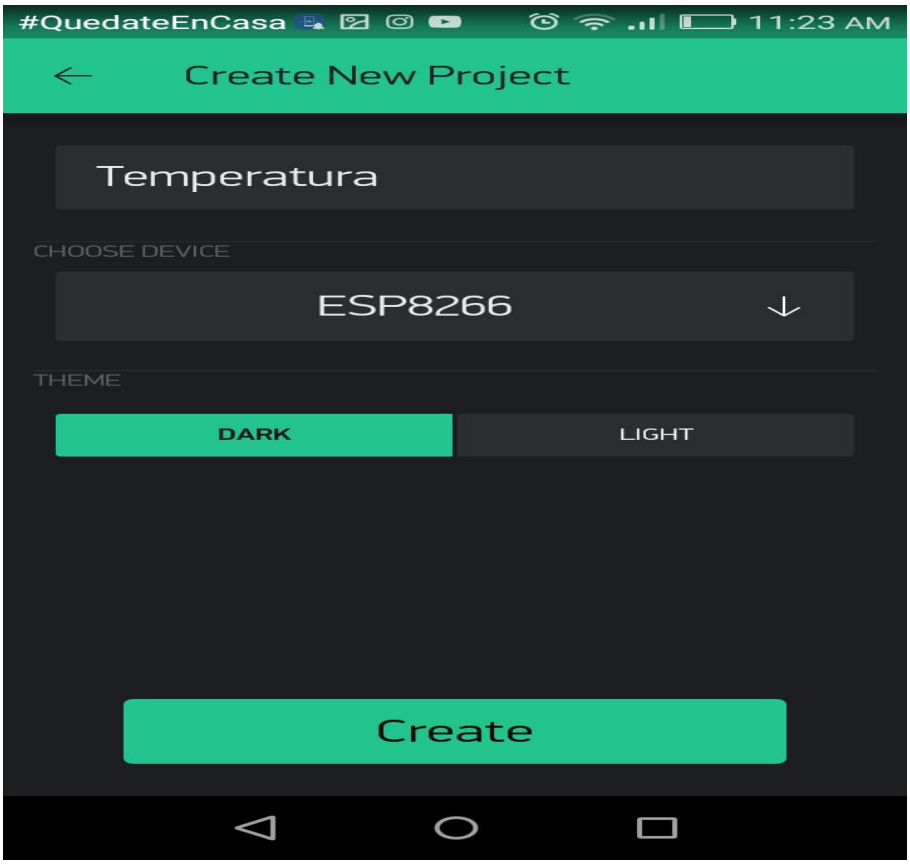


Portada de la app Blynk

Fuente: (Autor, 2020)

Después nos aparecerá en pantalla una opción para escoger el dispositivo que vamos a usar en choose device y escogemos el ESP8266, también nos da la opción para ponerle nombre a nuestro proyecto, le colocaremos el que deseemos.

Figura127: Programación de la app en Blynk.

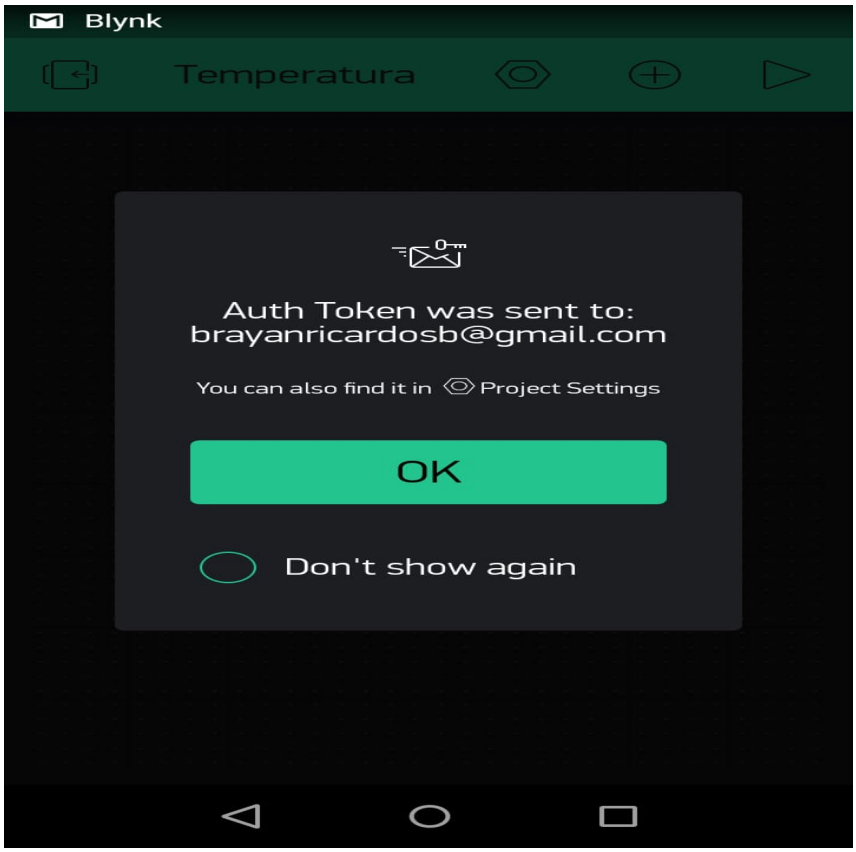


De acuerdo a la figura 117, se escoge el nombre del proyecto y el dispositivo a utilizar.

Fuente: (Autor, 2020)

Continuando después de pulsar create Blynk nos enviara un código a nuestro correo llamado token, el cual insertaremos en nuestra programación en el Arduino ide más adelante.

Figura128: Programación de la app en Blynk.

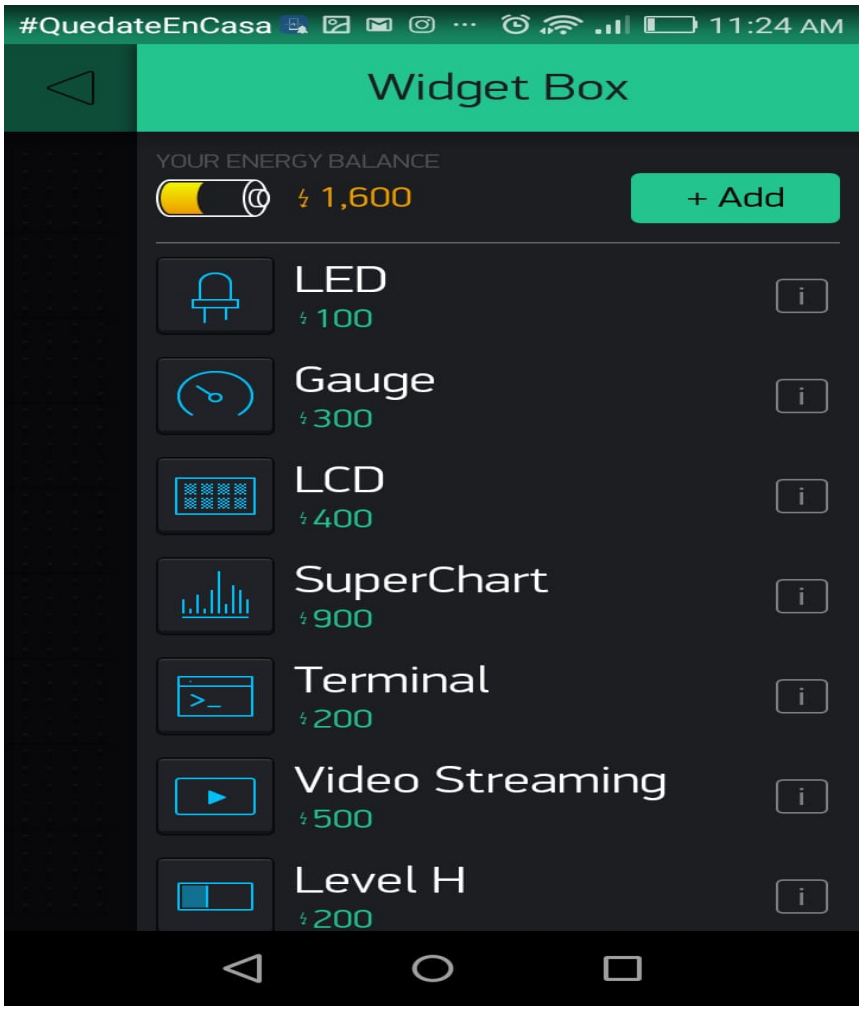


Notificación de que se ha enviado el código a nuestro correo.

Fuente: (Autor, 2020)

Le daremos okey, después de eso nos mostrará una interface donde pondremos nuestros diagramas que nos mostraran datos, pulsaremos en el botón + que está en el costado superior derecho y nos saldrá estas opciones de las cuales sacaremos dos de la dice gauge.

Figura129: Programación de la app en Blynk.

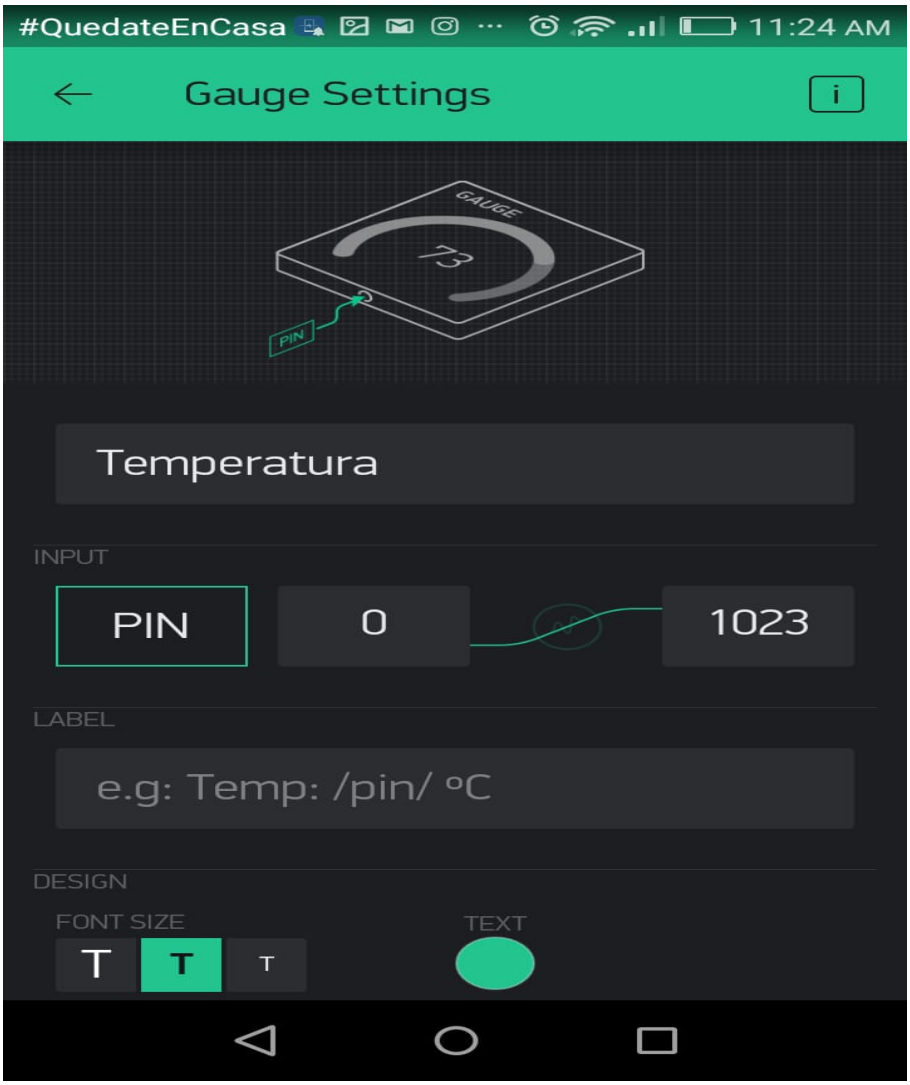


Lista de botones que se pueden escoger en Blynk

Fuente: (Autor, 2020)

Pulsaremos en un gauge y nos aparecerá una pantalla donde le podremos colocar rango para que mira la temperatura de 0 a 50, le cambiamos el nombre, cada cuanto se renueve la información y el pin de donde recibirá información.

Figura130: Programación de la app en Blynk.

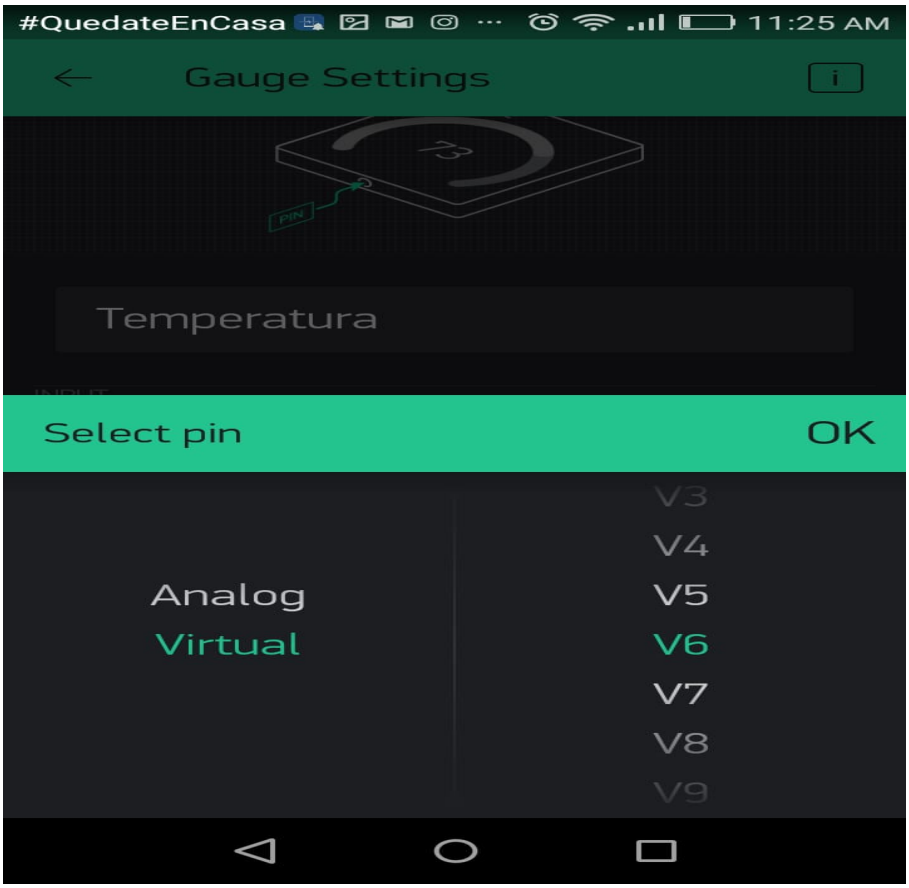


Declaración del pin a utilizar.

Fuente: (Autor, 2020)

Pulsaremos donde dice pin y escogeremos el pin virtual V6.

Figura131: Programación de la app en Blynk.

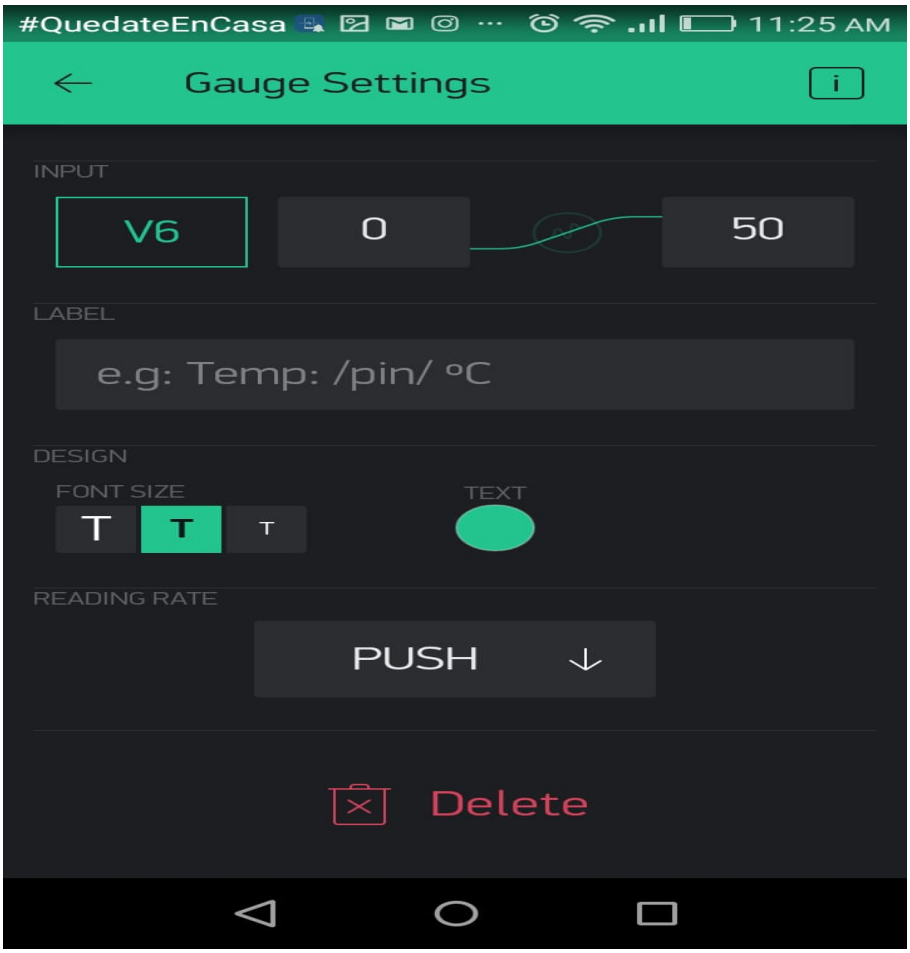


Lista de pines que se pueden utilizar.

Fuente: (Autor, 2020)

Ahora donde dice PUSH le vamos a indicar 1, que será el tiempo en que vuelva a verificar la información.

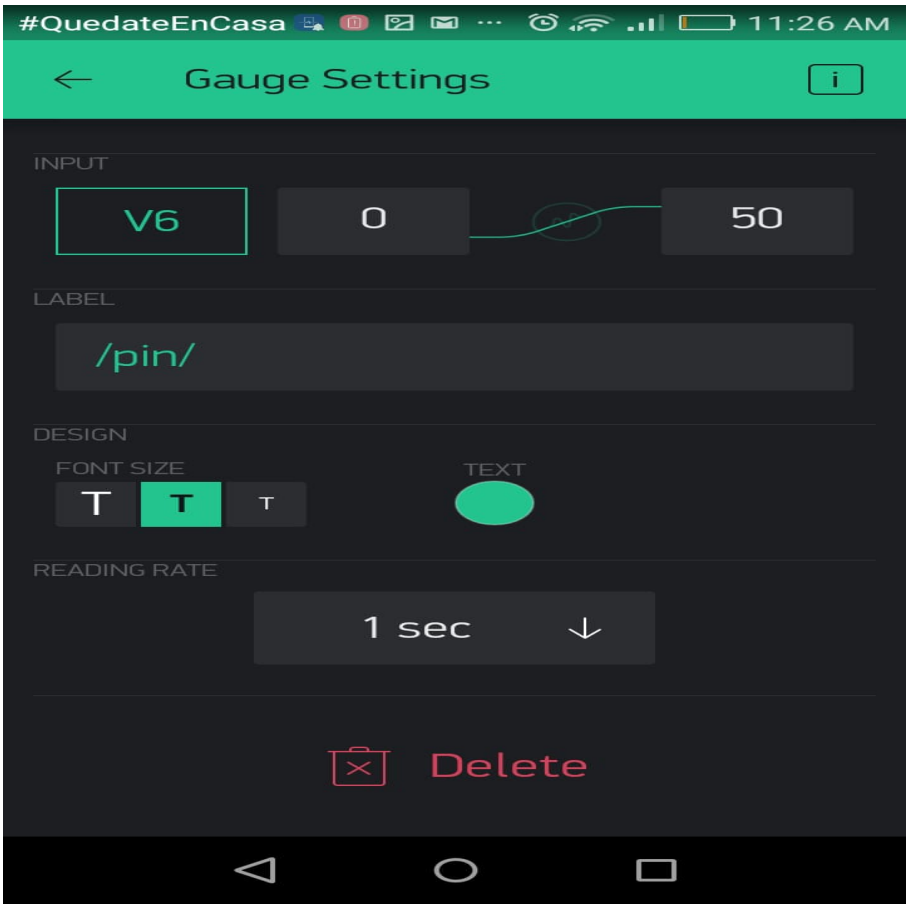
Figura132:Programación de la app en Blynk.



Declaración del tiempo en que revisará y mostrará cada lectura.

Fuente: (Autor, 2020)

Figura133: Programación de la app en Blynk.

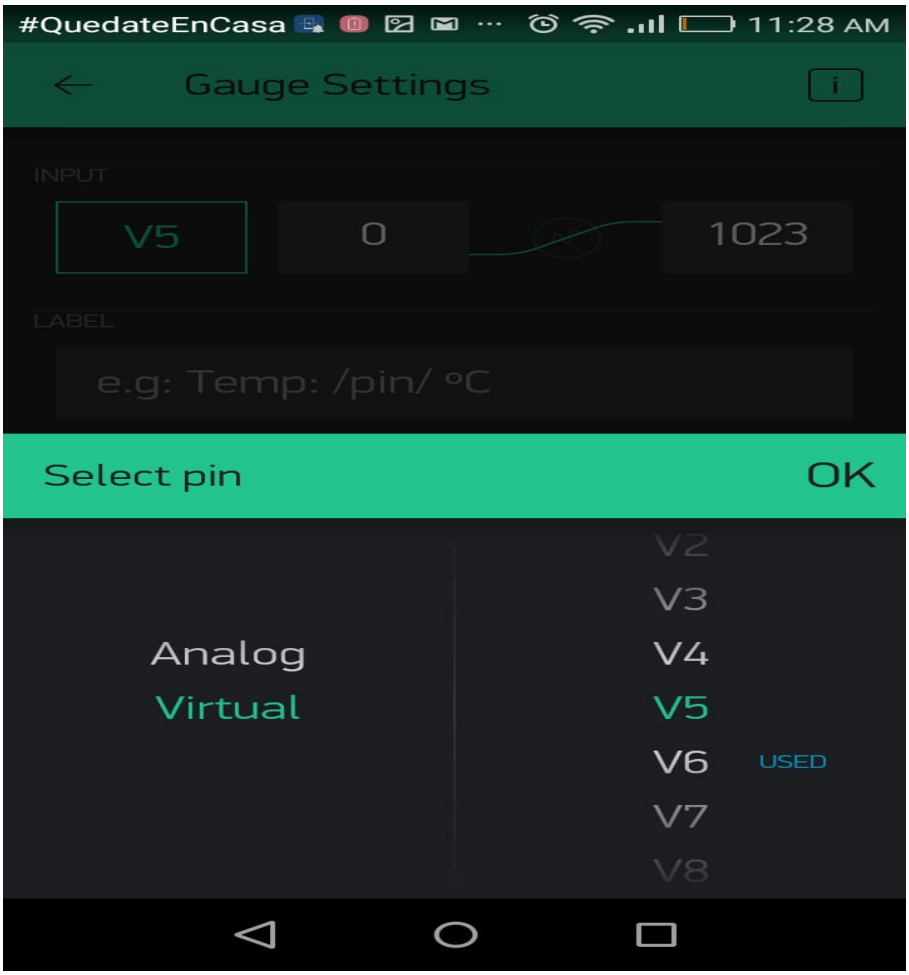


Declaración del tiempo en que revisará y mostrará cada lectura.

Fuente: (Autor, 2020)

Ahora vamos al siguiente gauge, le ponemos el pin virtual V5, le ponemos que renueve la información cada segundo, le ponemos un rango de 0 a 100% y listo.

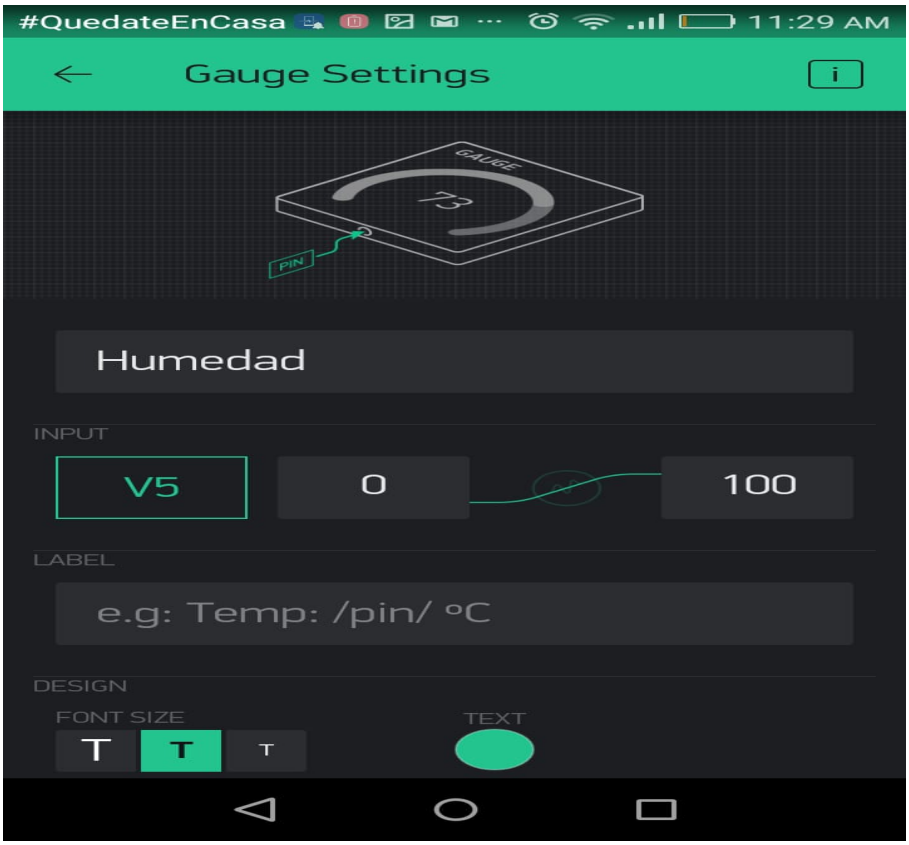
Figura134: Programación de la app en Blynk.



Lista de pines que se pueden utilizar.

Fuente: (Autor, 2020)

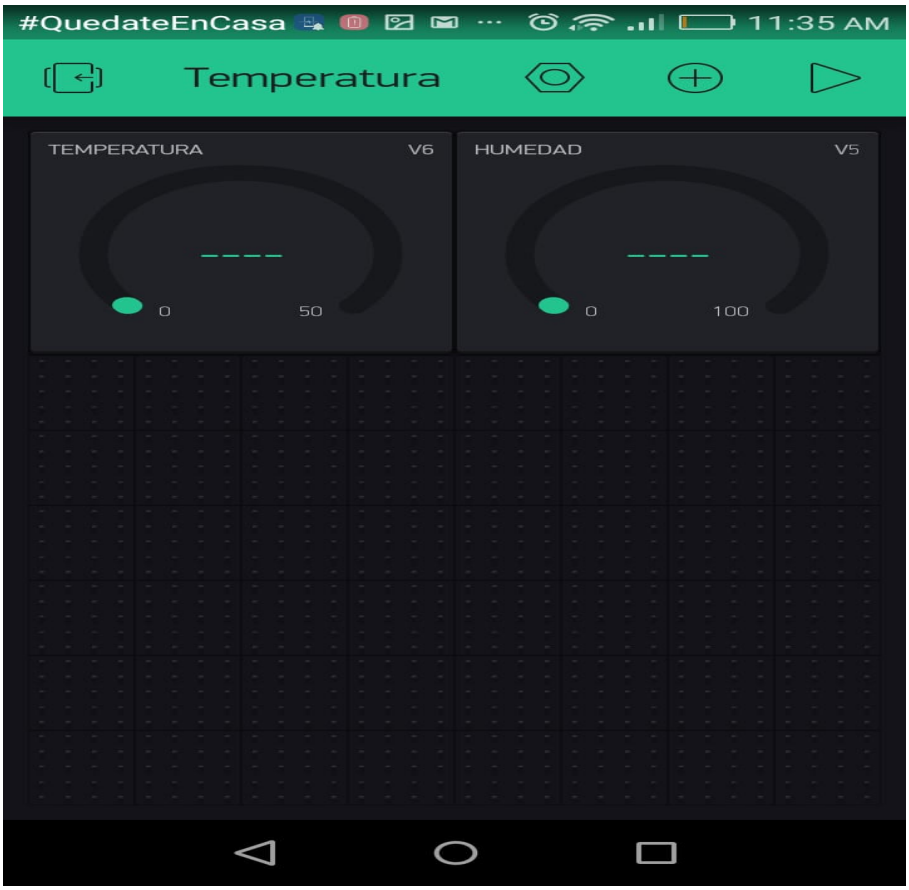
Figura135: Programación de la app en Blynk.



Fuente: (Autor, 2020)

Finalmente, la pantalla se verá así y con esto daremos por concluido el condicionamiento de la app.

Figura136: Programación de la app en Blynk.



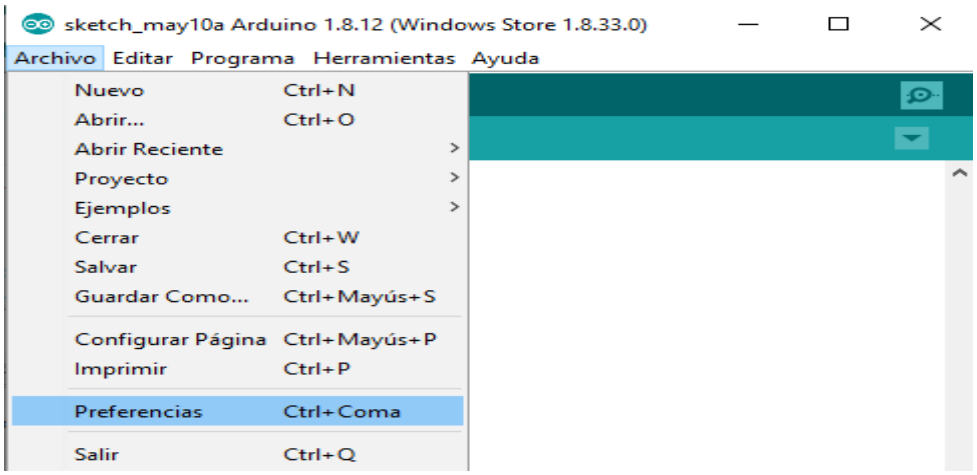
Vista final de la app.

Fuente: (Autor, 2020)

Ahora, parra la programación en el Arduino ide necesitaremos agregar unas librerías y la board del ESP8266.

Empecemos con agregar la board del esp8266, para esto debemos copiar un link en preferencias.

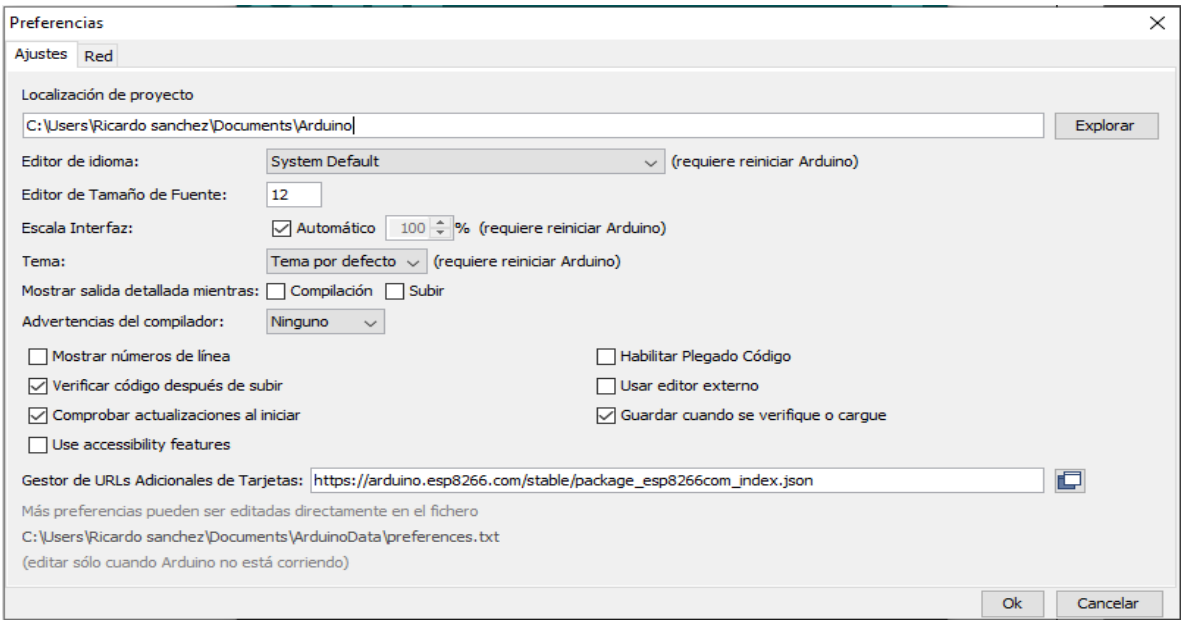
Figura137: Preferencias.



Fuente: (Autor, 2020)

Nos aparecerá esta pantalla y donde dice URLs, pegaremos este link:https://arduino.esp8266.com/stable/package_esp8266com_index.json, esto le servirá a Arduino para saber dónde descargar esa board.

Figura138: Preferencias.

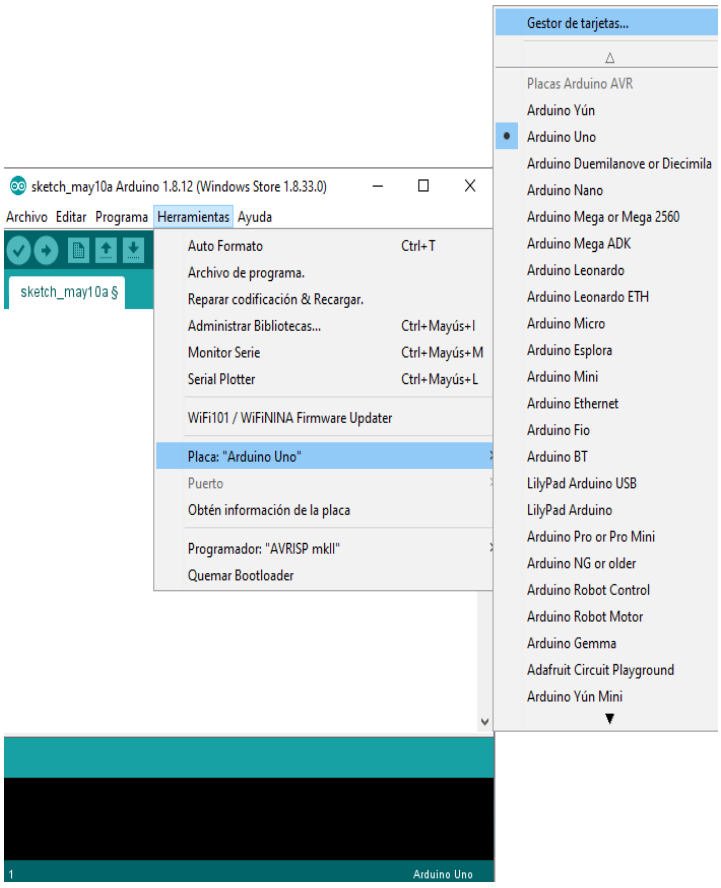


De acuerdo a la figura 138 se le indica a Arduino ide donde buscar la board del esp8266.

Fuente: (Autor, 2020)

Después, nos vamos a herramientas, placa y gestor de tarjetas.

Figura139: Gestor de tarjetas practica 10.

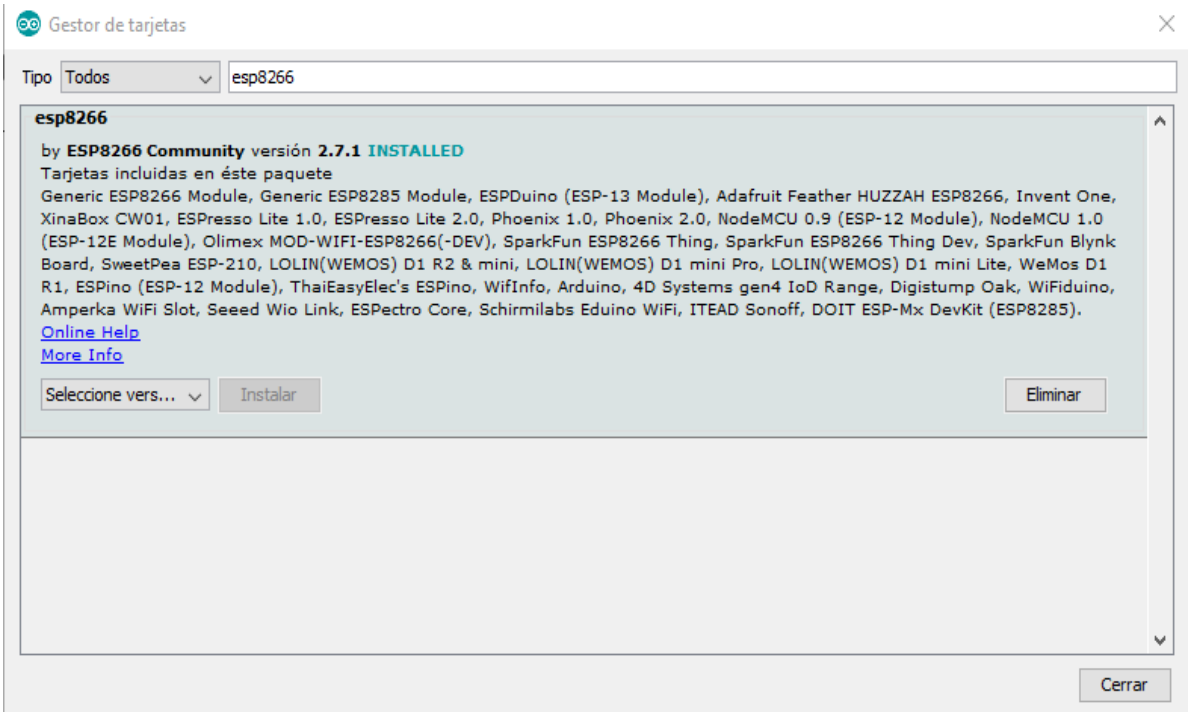


Fuente: (Autor, 2020)

Pasos para agregar tarjetas.

Nos aparecerá el gestor de tarjetas, escribimos esp8266 e instalamos la que dice esp8266 community.

Figura140: Gestor de tarjetas practica 10.



Instalación de la tarjeta ESP8266 Community.

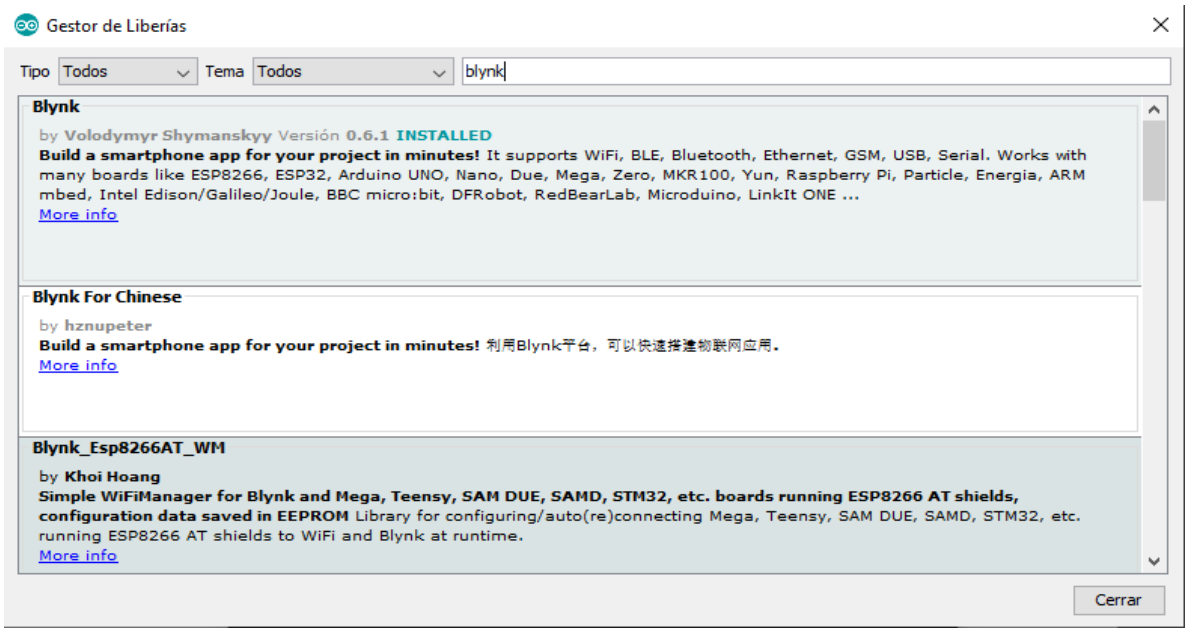
Fuente: (Autor, 2020)

Ya con esto tendremos instalada la librería del esp8266, ahora vamos instalar la librería del sensor dht11 tal cual lo hicimos en la practica 7: sensor de temperatura y humedad.

Debemos instalar dos librerías más para que nuestra programación se nos haga más simple, las cuales son Blynk y Simpletimer.h

Empecemos con la de Blynk, para ella simplemente debemos dirigirnos a programa-incluir librería-administrar bibliotecas, nos aparecerá el gestor de librería y en el vamos a escribir Blynk e instalamos la primera.

Figura141: Gestor de tarjetas practica 10.



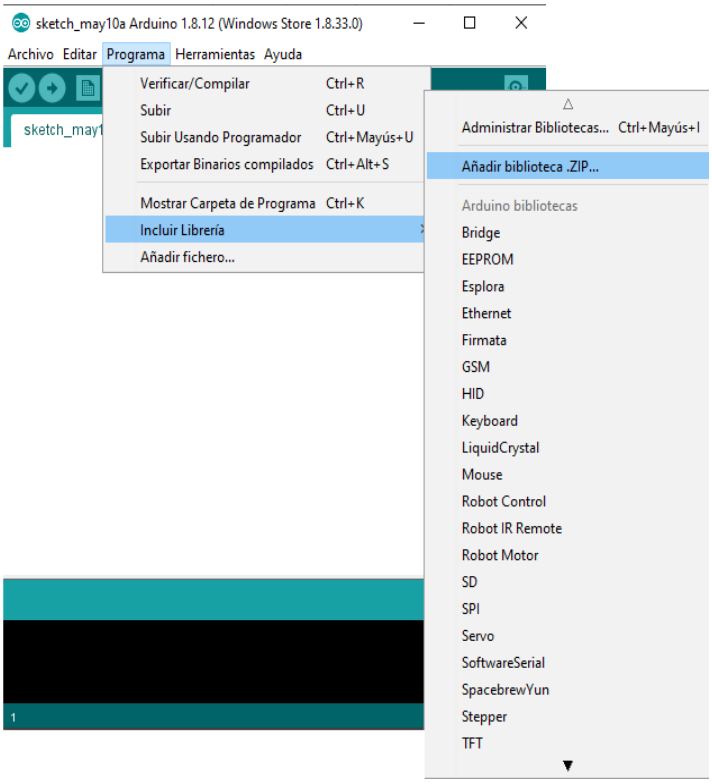
Instalación de la librería Blynk.

Fuente: (Autor, 2020)

Por último, debemos instalar la librería de simple Timer, para ello la descargaremos del siguiente link: https://github.com/jfturcot/SimpleTimer.

Una vez descargado el archivo de la librería y nos dirigimos a programa-incluir librería-añadir biblioteca.ZIP.

Figura142: Añadir bibliotecas en zip.



Pasos para agregar una librería en formato zip.

Fuente: (Autor, 2020)

Pulsamos allí, ubicamos donde se descargó la librería y le damos abrir y el Arduino ide la instalará y con esto hemos concluido la instalación de las librerías.

Ahora vamos con la programación en el Arduino ide.

Lo primero que vamos a hacer es incluir todas las librerías que vamos a necesitar y definir Blynk_print en el puerto serial.

Figura143: Añadir bibliotecas en zip.

```
#define BLYNK_PRINT Serial
#include <SPI.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <SimpleTimer.h>
#include <DHT.h>
```

Definición de las librerías a utilizar.

Fuente: (Autor, 2020)

Después vamos a definir tres variables Char que se llamaran auth, ssid y pass. En la primera colocaremos el código que nos envió Blynk al correo, en la segunda colocaremos la redwifi más cercana y de tercera la contraseña de esa red wifi, colocamos estos datos en las comillas.

Figura144: Sketch practica 10

```
char auth[] = "";
char ssid[] = "";
char pass[] = "";
```

Datos del Wifi al que se conectará el nodeMCU esp8266.

Fuente: (Autor, 2020)

Definimos el pin del dht, el tipo de sensor de temperatura y humedad, ademas de que con la funcion DHT dht como vimos en la practica 7 tenemos que indicar el pin del dht y el tipo de dht que estamos usando.

Figura145: Sketch practica 10.

```
#define DHTPIN 2
#define DHTTYPE DHT11 // DHT 11
DHT dht(DHTPIN, DHTTYPE);
```

Definición de pines del dht y tipo de dht.

Fuente: (Autor, 2020)

Ahora agregamos la función simpletimer que lo que hace esta función es enviar el tiempo de actividad cada segundo al pin virtual, después le agregamos la función Void, que lo que hace es que no devuelve ningún valor. El paso de parámetros, en este caso la variable actual, que es una variable de tipo float (número con decimales), siempre tienen que ir encerrados entre paréntesis.

En las variables tipo float vamos a definir dos variables tipo float, h y t, empleamos la función dht. readHumudity y dht. readTemperature que lo que hacen como vimos en la practica 7 es leer los datos de temperatura y humedad que capta el sensor.

Figura146: Sketch practica 10.

```
SimpleTimer timer;
void sendSensor()
{
  float h = dht.readHumidity();
  float t = dht.readTemperature();
```

Comandos de lectura para temperatura y humedad

Fuente: (Autor, 2020)

Digitamos la función si y le insertamos la función isnan, lo que hace esta función es que devuelve un uno si lo que lee no es un valor numérico, en nuestro caso si las variables h y t, si no entregan valores numéricos se ejecutará el serial print que lo que hará es imprimir el texto failed to read from dht sensor

Figura147: Sketch practica 10.

```
if (isnan(h) || isnan(t)) {
  Serial.println("Failed to read from DHT sensor!");
  return;
}
```

Declaración de mensajes de error.

Fuente: (Autor, 2020)

Entonces lo que harán las siguientes funciones es enviar datos a la aplicación del celular mediante la funcion Blynk.virtualWrite, primero le ponemos el pin y el valor a enviar.

Figura148: Sketch practica 10.

```
  Blynk.virtualWrite(V5, h);
  Blynk.virtualWrite(V6, t);
}
```

Pines para temperatura y humedad.

Fuente: (Autor, 2020)

Ahora iniciamos el void setup, iniciamos la conexión serial a 9600 baudios y iniciamos a blynk, con la variable donde colocamos el codigo que nos envio blynk, el nombre del wifi que pusimos al principio y su contraseña.

Figura149: Sketch practica 10.

```
void setup()
{
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);
```

Iniciación de Blynk.

Fuente: (Autor, 2020)

Iniciamos el dht y incluimos la variable send sensor para que la llamen cada segundo.

Figura150: Sketch practica 10.

```
  dht.begin();

  timer.setInterval(1000L, sendSensor);
}
```

Iniciación del dht y declaración de intervalo de lectura.

Fuente: (Autor, 2020)

Ahora por ultimo iniciamos el loop, iniciamos a blynk, el timer y daremos por concluida la programacion.

Figura151: Sketch practica 10.

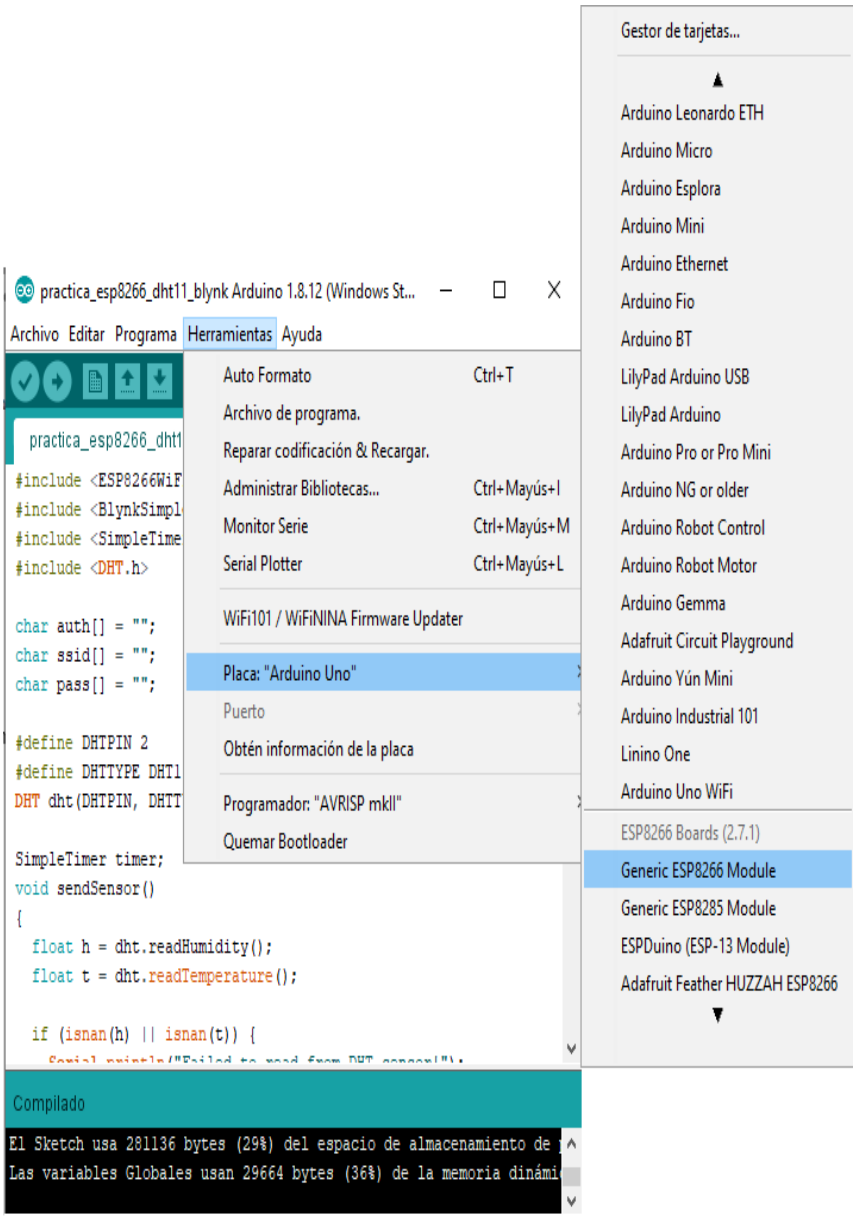
```
void loop()
{
  Blynk.run();
  timer.run();
}
                }
```

De acuerdo a la figura 151 se hace correr Blynk y el Timer.

Fuente: (Autor, 2020)

Antes de compilar el programa debemos poner como placa a compilar al esp8266.

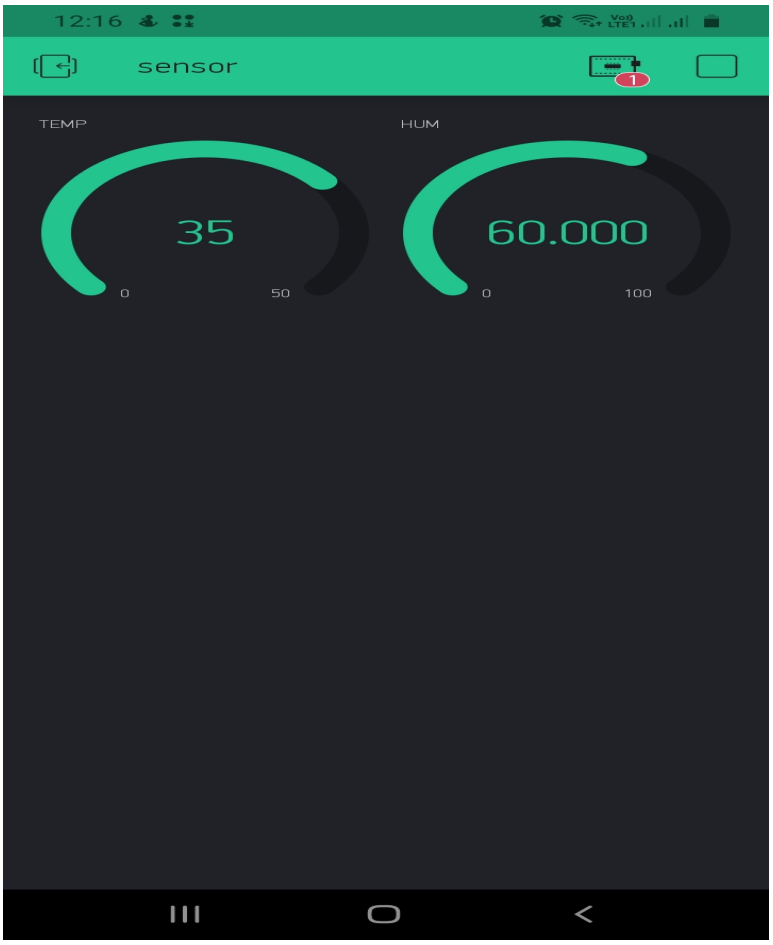
Figura152: Montando el ESP8266 en Arduino ide.



Pasos para montar el Esp8266.

Fuente: (Autor, 2020)

Figura153: Valores leídos por el Dht.



Valores medidos por el dht y enviados a través de wifi a la app Blynk.

Fuente: (Autor, 2020)

Ahora vamos con su conexionado.

Figura154: Diagrama practica 10.

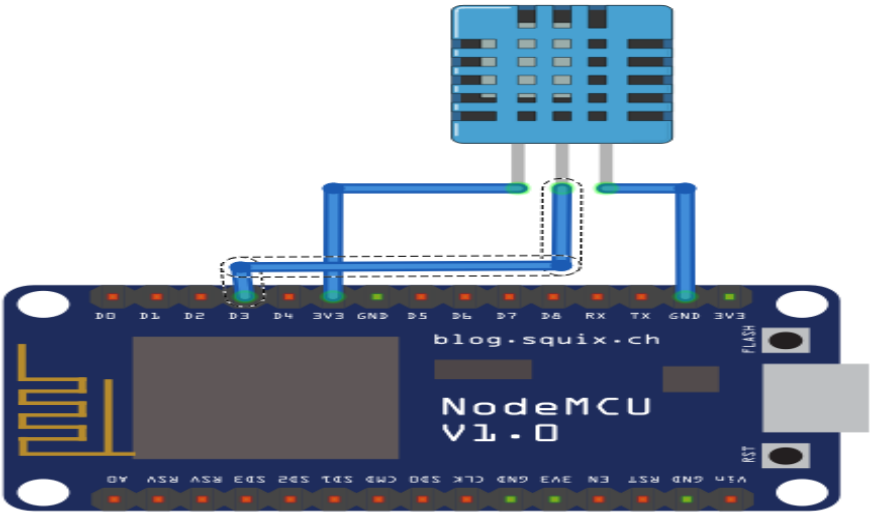
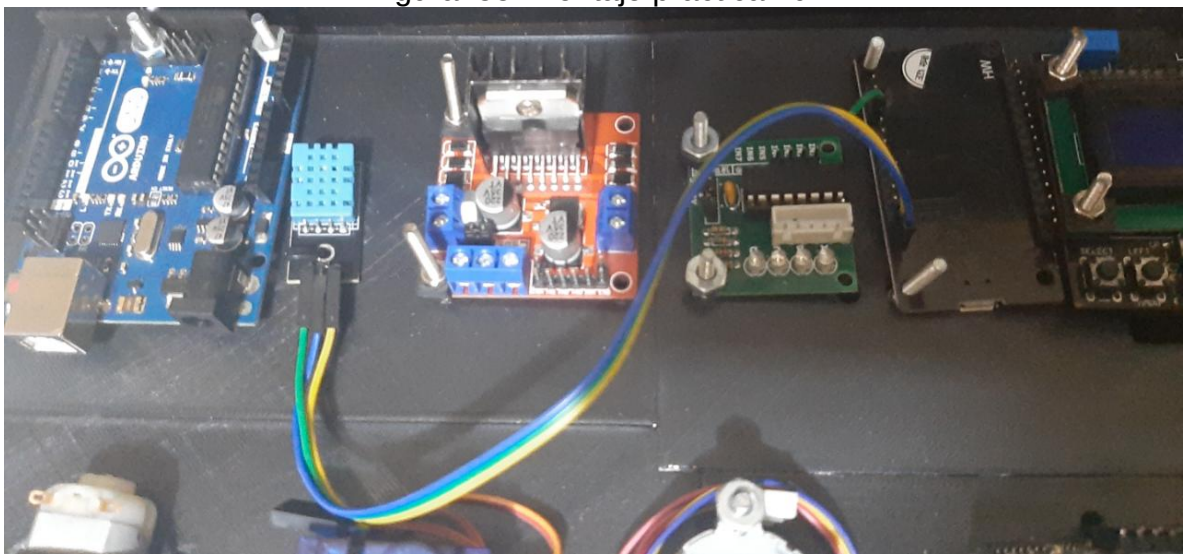


Diagrama de conexiones entre el dht y esp 8266 nodeMCU.

Fuente: (Autor, 2020)

Ahora una vista real.

Figura155: Montaje practica 10.



Vista real del conexionado presentado en la figura 155.

Fuente: (Autor, 2020)

Es realmente sencilla, cabe resaltar que en algunos modelos del dht11 la posicion de los pines pueden variar asi que hay que estar atentos a eso.

BIBLIOGRAFIA

(s.f.). Obtenido de Aprendiendo Arduino: https://aprendiendoarduino.wordpress.com/2017/06/20/estructuras-de-control-3/

(18 de 01 de 2020). Obtenido de Descubre Arduino: https://descubrearduino.com/estructura-de-programa/

Aprendiendo Arduino. (s.f.). Obtenido de https://aprendiendoarduino.wordpress.com/2017/10/14/estructuras-de-control-4/

Arduino. (s.f.). Obtenido de Variables: https://www.arduino.cc/en/Tutorial/Variables

Arduino. (21 de febrero de 2019). Obtenido de https://www.arduino.cc/reference/en/language/variables/data-types/unsignedint/

Crespo, M. D. (s.f.). Obtenido de http://manueldelgadocrespo.blogspot.com/p/description-digital-pins-with.html?m=1

Gomez, M. A. (26 de 12 de 2016). *Arduino a Muete.* Obtenido de http://arduinoamuete.blogspot.com/2016/12/introduccion-blynk.html

Ortega, F. (s.f.). *Flexbot.* Obtenido de https://www.flexbot.es/monitorizar-temperatura-humedad-movil-blynk/

panamahitek. (s.f.). Obtenido de http://panamahitek.com/el-setup-y-el-loop-en-arduino/

Programar facil. (s.f.). Obtenido de https://programarfacil.com/blog/arduino-blog/if-else-arduino/

Wikipedia. (23 de 09 de 2019). *Arduino*. Obtenido de https://es.wikipedia.org/w/index.php?title=Arduino&oldid=119633331