

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
SEDE VIÑA DEL MAR – JOSÉ MIGUEL CARRERA

CREACION DE MAQUETA ARDUINO Y MANUAL DE USUARIO

Trabajo de Titulación para optar
al Título de Técnico Universitario
en
MECÁNICA AUTOMOTRIZ

Alumnos:

Alejandro Arancibia Quiroz

Alexis Delfino Rebolledo

Profesor Guía:

Sr. Güido Almagià Flores

2020

RESUMEN

Keywords: ARDUINO - SENSOR DE LLUVIA – SENSOR DE LUZ – SENSOR DE ULTRA SONIDO – HCSR 04 – SENSOR DE VIBRACION – LEDS – DISPLAY LCD – LCD 16X2 – I2C – PROTOBOARD.

En el primer capítulo se da a conocer las características de cada uno de los componentes que forman parte de la maqueta, esto para tener una visión general de la maqueta a realizar y generar un conocimiento base acerca de la placa Arduino.

El segundo capítulo se centra principalmente en el funcionamiento de cada componente a utilizar, esto servirá para entender de qué manera trabaja cada sensor, actuador, placa controladora, entre otros componentes que forman parte de la maqueta didáctica a realizar, ampliando el conocimiento señalado en el capítulo I.

En el tercer capítulo se pueden encontrar las conexiones utilizadas de cada componente, esto para dar a conocer la unión entre elementos a modo de ejemplo, emulando diversos sistemas encontrados en los vehículos actuales y de esta manera demostrar su posible aplicación automotriz.

El cuarto capítulo se focaliza en explicar toda la programación necesaria para el correcto funcionamiento de cada componente de la maqueta, y de esta manera facilitar la comprensión de cada uno de los códigos utilizados.

INDICE

RESUMEN	2
INTRODUCCION.....	7
OBJETIVO GENERAL.....	8
OBJETIVOS ESPECIFICOS	8
Capítulo 1: “Características de los componentes del proyecto”	9
Componentes.....	10
INTRODUCCION.....	10
1.1 ARDUINO UNO OEM	11
1.1.1 Constitución de placa Arduino	13
1.2 TRANSFORMADOR DE CORRIENTE ALTERNA.....	16
1.3 SENSOR DE DISTANCIA ULTRASONICO HC-SR04.....	17
1.4 BUZZER KY-012.....	18
1.5 SENSOR DE TEMPERATURA SUMERGIBLE DS18B20	19
1.6 SENSOR DE LLUVIA.....	20
1.7 SENSOR DE LUZ POR FOTORESISTENCIA LM393	21
1.8 SENSOR DE GOLPE (VIBRACION)	22
1.9 CABLE 22AWG	23
1.10 CABLE USB PARA ARDUINO (TIPO A-B).....	24
1.11 PROTOBOARD	25
1.12 DIODO LED	26
1.13 DISPLAY LCD 16X2	27
Capítulo 2: “Funcionamiento de cada componente”	29
2.1 ARDUINO.....	29
2.1.1 Entrada Analógica	30
2.1.2 Entrada Digital.....	31
2.1.3 Salida Analógica	31
2.1.4 Salida Digital.....	32
2.2 SENSOR DE DISTANCIA ULTRASONICO HC-SR04.....	33
2.3 TRANSFORMADOR DE CORRIENTE ALTERNA.....	34
2.4 BUZZER.....	35
2.5 SENSOR DE TEMPERATURA SUMERGIBLE	36
2.6 DISPLAY LCD 16X2	37
2.7 DIODO LED	38
2.8 SENSOR DE GOLPE (VIBRACION) KY-031.....	39
2.9 SENSOR DE LUZ POR FOTORRESISTENCIA.....	39
2.10 SENSOR DE LLUVIA.....	40
Capítulo 3: “Conexionado de la maqueta”	41
3.1 INTRODUCCION.....	42

3.2 CONEXIONES DEL SISTEMA DE ASISTENCIA AL ESTACIONAMIENTO	43
3.3 CONEXIONES SISTEMA DE MONITOREO DE TEMPERATURA	45
3.4 CONEXIONES DEL SISTEMA LIMPIA PARABRISAS AUTOMATICO	47
3.5 CONEXIONES DEL SISTEMA DE ALARMA.....	49
3.6 CONEXIONES DEL SISTEMA DE ILUMINACION AUTOMATICA	51
3.7 APLICACIONES SOBRE 5[V].....	52
Capítulo 4: “Programación”	53
4.1 INTRODUCCION.....	54
4.2 PROGRAMACION.....	55
4.3 CODIGOS	56
4.4 LIBRERIAS	61
4.5 CODIGOS ESPECIFICOS UTILIZADOS EN LA MAQUETA	62
4.5.1 SISTEMA DE MONITOREO DE TEMPERATURA	62
4.5.2 SISTEMA DE ASISTENCIA AL ESTACIONAMIENTO	66
4.5.3 SISTEMA DE ALARMA.....	68
4.5.4 SISTEMA DE ILUMINACION AUTOMATICA.....	71

INDICE DE FIGURAS

Figura 1-1 KTAG Reprogramador de ECU	12
Figura 1-2 Sectores del Arduino	13
Figura 1-3 Sector Power.....	14
Figura 1-4 Sector Analógico	15
Figura 1-5 Sector Digital.....	15
Figura 1-6 Transformador de corriente alterna	16
Figura 1-7 Funcionamiento sensor de ultrasonido	17
Figura 1-8 Sensor de ultrasonido	17
Figura 1-9 Buzzer.....	18
Figura 1-10 Sensor de temperatura sumergible	19
Figura 1-11 Conexiones del sensor de lluvia	20
Figura 1-12 Sensor de luz por fotorresistencia	21
Figura 1-13 Sensor de golpe	22
Figura 1-14 Cable 22AWG	23
Figura 1-15 Jumpers.....	24
Figura 1-16 Conector Tipo A.....	24
Figura 1-17 Conector Tipo B.....	24
Figura 1-18 Protoboard	25
Figura 1-19 Diodo LED	26
Figura 1-20 Display LCD 16x2.....	27
Figura 1-21 Display LCD.....	27
Figura 2-1 Plataforma de programación Arduino	30
Figura 2-2 Entradas analógicas	30
Figura 2-3 Entradas y salidas digitales.....	31
Figura 2-4 Grafico Voltaje entregado por el sensor vs Valor analógico.....	32
Figura 2-5 Cálculo de la velocidad del sonido en [cm/μs].....	33
Figura 2-6 Calculo de la distancia mediante el sensor HCSR04	33
Figura 2-7 Reductor de voltaje.....	34
Figura 2-8 Diodo rectificador	34
Figura 2-9 Piezo eléctrico	35
Figura 2-10 Sensor de temperatura sumergible	36
Figura 2-11 Posibles caracteres a utilizar.....	37
Figura 2-12 Partes de un diodo led	38
Figura 2-13 Ilustración de contactos internos del sensor de vibración	39
Figura 2-14 Sensor de luz	39
Figura 2-15 Sensor de lluvia	40
Figura 3-1 Porta pilas con conector Jack	42
Figura 3-2 Conexión sensor ultrasónico	44
Figura 3-3 Conexiones Buzzer.....	44
Figura 3-4 Conexión I2C y LCD con Arduino	46
Figura 3-5 Conexión sensor de temperatura	46
Figura 3-6 Conexiones sensor de lluvia.....	47
Figura 3-7 Conexiones LEDS	48
Figura 3-8 Conexión sensor de vibración	49
Figura 3-9 Buzzer para alarma.....	50
Figura 3-10 Conexión sensor de luz	51
Figura 3-11 Circuito de control del relé.....	52
Figura 4-1 Reprogramador de ECU	54
Figura 4-2 Sectores de programación Arduino	56

Figura 4-3 Ejemplo uso int.....	57
Figura 4-4 Ejemplo codigo bool	57
Figura 4-5 Ejemplo codigo float	57
Figura 4-6 Ejemplo código pinMode	58
Figura 4-7 Ejemplo código Serial.begin	58
Figura 4-8 Ejemplo codigo digitalRead	59
Figura 4-9 Ejemplo codigo digitalWrite	59
Figura 4-10 Ejemplo código delay	59
Figura 4-11 Ejemplo codigo if	59
Figura 4-12 Ejemplo código else	60
Figura 4-13 Ejemplo código pulseIn.....	60
Figura 4-14 Ejemplo código analogRead.....	61
Figura 4-15 Ejemplo de librería incluida	61
Figura 4-16 Escaneo de la dirección de comunicación I2C	62
Figura 4-17 Código sistema de monitoreo de temperatura	65
Figura 4-18 Código del sistema de asistencia al estacionamiento	68
Figura 4-19 Código sistema de alarma	71
Figura 4-20 Código sistema de iluminación automática.....	74
Figura 4-21 Código sistema limpia parabrisas automático	76

INDICE DE TABLAS

Tabla 1-1 Datos de cables	23
---------------------------------	----

INTRODUCCION

Control Electrónico Automotriz es un ramo impartido por la carrera técnico universitario en mecánica automotriz, este ramo ayuda a comprender el funcionamiento de los sensores que se encuentran en los vehículos, pero no profundiza en el elemento intermediario de esta comunicación que se establece entre un sensor y un actuador.

El elemento antes mencionado es conocido en el campo automotriz como ECU (sigla en inglés de Engine Control Unit), el cual tiene un principio de funcionamiento similar al de una placa ARDUINO. El presente trabajo se centrará principalmente en explicar el funcionamiento de la placa, puesto que su comprensión es más sencilla que la de una unidad de control. Con esto se logrará suplir la desinformación existente sobre este elemento tan importante en un vehículo motorizado, facilitando la comprensión de la comunicación que existe entre los componentes de este sistema.

Otra ventaja que va a presentar la maqueta y la información de la placa ARDUINO plasmada en el presente trabajo es la oportunidad de que los estudiantes que cursan la carrera en la UNIVERSIDAD TECNICA FEDERICO SANTA MARIA SEDE VIÑA DEL MAR, puedan realizar sus propios proyectos y de esta manera se integren en el mundo de la programación.

Este trabajo va orientado como material de estudio adicional para la asignatura anteriormente mencionada, y consistirá en la confección de una maqueta que emule algunos sensores que componen el sistema de confortabilidad del automóvil. Además, en la parte escrita se detallarán todos los componentes necesarios para realizar la maqueta, se explicará el principio de funcionamiento, las conexiones que tienen cada componente y la programación necesaria para el correcto funcionamiento del sistema global a realizar.

OBJETIVO GENERAL

- Contribuir a la enseñanza de la asignatura control electrónico automotriz mediante el uso de ARDUINO

OBJETIVOS ESPECIFICOS

- Mostrar el funcionamiento del sistema Arduino básico para hacer una introducción al mundo de la programación.
- Brindar las herramientas necesarias para que el docente de la asignatura pueda comprender y transmitir los conocimientos adquiridos.

Capítulo 1: “Características de los componentes del proyecto”

Componentes

INTRODUCCION.

Dentro del mundo de la electrónica se puede encontrar una placa llamada Arduino la cual sirve para comandar distintos tipos de sensores y actuadores. A pesar de que no está directamente relacionada en el ámbito automotriz los autores del presente trabajo de título quieren demostrar que se puede utilizar ampliamente en esta área obteniendo incluso ventajas sobre esto, tanto económicas como de equipamiento en distintos sistemas que componen un vehículo siendo mayoritariamente sistemas de comodidad.

A lo largo de la historia automotriz, se han incorporado diversos componentes al sistema de confortabilidad del vehículo. Uno de estos es el llamado limpiaparabrisas, el cual fue inventado en 1903 por Mary Anderson, antes de este invento se tenían que bajar del vehículo a limpiar el parabrisas. A mediados de los años noventa se incorporó el limpiaparabrisas automático en función de la lluvia. Junto con esto se implementó el sensor de luz, el cual detecta la cantidad de luz ambiental y controla las luces del vehículo.

Otro componente que se incorporó en el mundo automotriz es el sensor de proximidad en el año 2008, el cual ayuda al momento estacionar el vehículo, indicando con un sonido la distancia a la que se encuentra de un obstáculo sólido.

Anteriormente la climatización del habitáculo era imposible debido a que el vehículo no era herméticamente cerrado por el diseño que tenían en antaño. En los años 40 se empezó a implementar el aire acondicionado, el cual únicamente bajaba la temperatura de la cabina. Actualmente se utiliza un sistema de climatización en el cual el usuario únicamente indica a que temperatura desea estar, y un sistema de aire acondicionado en conjunto con electrónica se encargan de llevar a cabo dicha acción.

Todos los avances expuestos anteriormente no se encuentran en el 100% de los vehículos existentes, por ende, se debe realizar un gasto extra al valor inicial del vehículo, tanto la adquisición de componentes extras (sensores, relés, etc.) como su coste de instalación. En vista de que en años anteriores no se disponía de esta tecnología, se buscó una manera de implementarlas en cualquier vehículo al menor costo posible, siendo este el objetivo principal del presente trabajo y la razón por la que se decidió darle solución con un simple sistema de fácil adquisición llamado ARDUINO.

1.1 ARDUINO UNO OEM

Para cumplir el cometido central del presente proyecto, se debe tener un “cerebro” central encargado de controlar diversos componentes asociados a él, teniendo como tarea principal recibir la información del entorno y tomar una decisión previamente definida. El componente encargado de esta labor se le conoce como ARDUINO. Existen varios modelos de ARDUINO, entre ellos nano, mega, UNO, R3, entre otros, siendo su principal diferencia la cantidad de entradas y salidas que estos poseen. Se determinó que el ARDUINO UNO OEM es la mejor opción por la cantidad de pines que posee, su reducido tamaño y facilidad de uso.

En el vehículo se puede encontrar un “cerebro” llamado ECU (por sus siglas en inglés engine control unit), el cual tiene un funcionamiento muy similar al sistema ARDUINO, pero este último no es capaz de suplir todas las funcionalidades de la ECU por la cantidad de componente asociados al sistema y cantidad de entradas y salidas disponibles. Por lo tanto, la configuración con la que se montaría en un automóvil sería un sistema totalmente aparte de la ECU, sin comprometer el funcionamiento de los demás sistemas. Para poder llevar a cabo una acción se debe tener el hardware y software necesarios para lo que se busca hacer.

Software corresponde al conjunto de programas, instrucciones y reglas programadas en un código que permiten ejecutar diversas tareas. Se considera como un elemento intangible en el mundo de la informática.

Hardware son todos los componentes electrónicos que forman parte del controlador y que llevan a cabo tareas específicas comandadas por los códigos escritos en el software.

Para comprender de mejor manera las definiciones anteriores se entregarán los siguientes ejemplos:

Hardware: en el área automotriz se encuentran distintos componentes que se consideran hardware, tales como las luces, aire acondicionado, sensores, actuadores, radio, etc.

Software: en un automóvil se puede encontrar la ECU, la cual se encarga de hacer funcionar los componentes anteriormente mencionados, mediante el software que esta contiene. Esta se puede modificar accediendo a ella mediante un programador OBD.



Fuente: <https://www.alientech-tools.com/es/>

Figura 0-1 KTAG Reprogramador de ECU

Arduino es una plataforma de creación electrónica de código abierto, la cual está basada en hardware y software libre, flexible y fácil de utilizar para los creadores y desarrolladores. Esta plataforma permite crear diferentes tipos de mini computadores de una sola placa a los que la comunidad de usuarios puede darles diferentes aplicaciones.

Para poder entender este concepto, primero tendrá que entender los conceptos de hardware y software libres. El hardware libre son los dispositivos cuyas especificaciones y diagramas son de acceso público, de manera que cualquiera puede replicarlos. Esto quiere decir que Arduino ofrece las bases para que cualquier persona o empresa pueda crear sus propias placas, pudiendo ser diferentes entre ellas, pero igualmente funcionales a partir de la misma base.

El software libre son los programas informáticos cuyo código es accesible por cualquier persona para darle uso y modificarlo.

La enorme flexibilidad y el carácter libre y abierto de Arduino hacen que se pueda utilizar este tipo de placas prácticamente para cualquier aplicación, desde relojes hasta básculas conectadas, pasando por robots, persianas controladas por voz, etc.

Dentro del campo automotriz existen varios sistemas que componen el funcionamiento total del vehículo, los cuales son comandados por la ECU. Por ejemplo:

El sistema de confortabilidad consiste principalmente en asistir al conductor para que pueda estar concentrado al conducir evitando las distracciones, tales como asistir al estacionamiento para eludir un posible impacto. En este caso su funcionamiento es mediante un sensor de ultrasonido, el cual indica la distancia a la que se encuentra un objeto sólido.

Gracias a esto ARDUINO puede interpretar esta señal y comandar un elemento que emita sonido, para así alertar de esta manera la distancia a la que se encuentra del objeto. Por otro lado, se encuentran los sensores de lluvia y de luz, los cuales varían sus señales dependiendo del entorno del vehículo. Por su parte el sensor de lluvia inicia el funcionamiento del motor del limpiaparabrisas de manera automática y el sensor de luz activa y desactiva las luces altas y bajas dependiendo del nivel de iluminación que percibe.

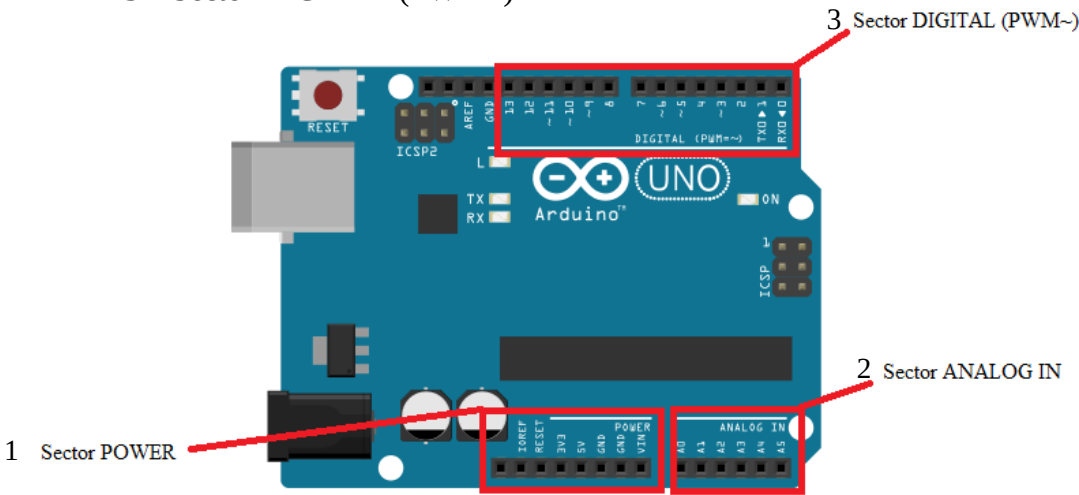
El sistema de refrigeración también forma parte de los sistemas que componen el vehículo. Es el encargado de mantener el automóvil en un funcionamiento óptimo en cuanto a la temperatura (entre 85° y 95°C). Gracias al sensor de temperatura se puede comandar mediante la ECU el paso del refrigerante hacia el radiador, y a su vez activar el electroventilador para mantener la temperatura anteriormente mencionada.

El sistema de inyección de igual manera se hace presente en el vehículo ayudando a realizar la inyección de combustible en el tiempo y cantidad adecuados, en este sistema se puede encontrar el knock sensor el cual detecta detonaciones irregulares las cuales dañan el motor. Al detectar una de estas detonaciones mediante la ECU se regula la cantidad de combustible que se inyecta en un tiempo que también define.

1.1.1 Constitución de placa Arduino

Para comprender de mejor manera como utilizar el sistema Arduino se explicarán las entradas y salidas de la placa a lo que se denominan pines para explicar esto de la forma más simple posible se separarán en 3 sectores mostrados en la figura 1-2.

- 1 Sector POWER
- 2 Sector ANALOG IN
- 3 Sector DIGITAL (PWM~)



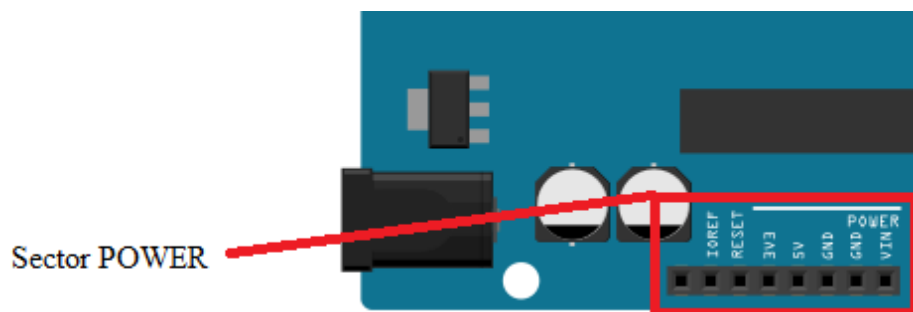
Fuente: <http://arduino.cl/que-es-arduino/>

Figura 0-2 Sectores del Arduino

El primero es POWER el cual es el encargado de recibir y/o entregar alimentación. En esta rama se encuentran los siguientes pines:

- VIN: este pin se puede utilizar de 2 formas distintas. La primera es como un pin de salida de voltaje, en este caso el Arduino esta alimentado a través del adaptador de corriente alterna o mediante un cable USB. La segunda es como entrada de voltaje de 12 a 7 [V] que gracias al regulador de la placa este será reducido a 5[V], este regulador necesita al menos 6.5 - 7[V] para poder trabajar correctamente. No se puede alimentar por 2 entradas al mismo tiempo, se debe escoger si usar un cable USB, un pin o el adaptador de corriente alterna a través del Jack. En el caso de la maqueta se alimentará a través del adaptador de corriente alterna, debido a que siempre estará en una sala de demostración cerca de una toma de corriente.
- 5 V: salida de 5[V]
- 3.3V: salida de 3.3[V]
- GND: Pines de tierra.

Todos estos pines se pueden visualizar en la figura 1-3



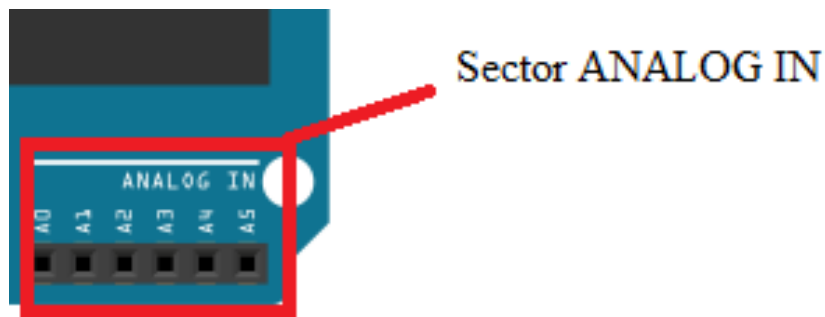
Fuente: <http://arduino.cl/que-es-arduino/>

Figura 0-3 Sector Power

El segundo sector es el ANALOG IN el cual es encargado de recibir todas las entradas analógicas, aquí se encuentran desde el pin A0 hasta el pin A5 los cuales se pueden ver en la figura 1-4.

Las entradas analógicas son las que tienen un rango de voltaje superior a las digitales (valores enteros desde 0 hasta 1023), esto quiere decir que el valor de voltaje de entrada puede variar dependiendo de la señal recibida por el sensor y que los 5[V] con los que trabaja Arduino se dividen en 1024 valores posibles.

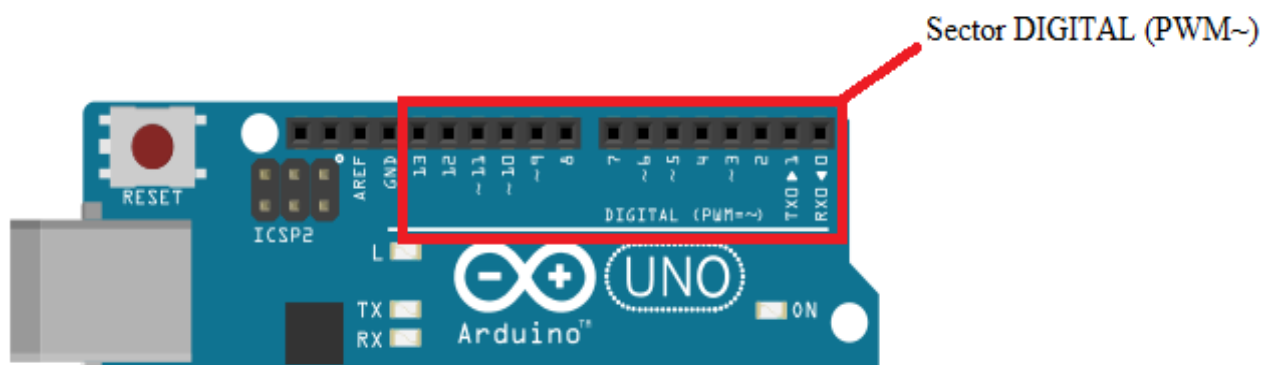
Por ejemplo, si el sensor envía un voltaje de 2,5 [V], Arduino lo traduce como un valor de 512 y con este dato determinar su posterior interpretación para el usuario, tanto en una pantalla LCD como para el monitor serie (visualizar valores en el programa Arduino).



Fuente: <http://arduino.cl/que-es-arduino/>

Figura 0-4 Sector Analógico

Para finalizar está el sector DIGITAL (PWM~) el cual se encarga de recibir y entregar un voltaje digital. Al ser digitales estos solamente tienen 2 estados posibles, los cuales son LOW y HIGH, en palabras simples “no” o “sí” respectivamente. LOW significa bajo, es decir que no recibe ni envía ninguna señal y HIGH significa alto, en este estado está recibiendo o enviando una señal. PWM (Pulse width modulation) o modulación del ancho de pulso es utilizado como salida analógica y se puede diferenciar por una onda antes del número (~) como se ve en la figura 1-5.



Fuente: <http://arduino.cl/que-es-arduino/>

Figura 0-5 Sector Digital

Además de lo dicho anteriormente se deben mencionar los parámetros con los cuales trabaja Arduino, tanto la alimentación que entrega a los componentes adicionales como el amperaje que suministra.

Características de la placa Arduino.

- Voltaje de operación: 5 Volts
- DC por pin entrada o salida de señales: 20 mA
- DC para Pin 3.3V: 50 mA

Con esto se finaliza el reconocimiento de los pines que se utilizarán en el ARDUINO, existen otros pines que son para usos más específicos los cuales se omitieron debido a que no tienen relevancia en lo que se llevara a cabo.

Ahora se procederá a nombrar y explicar la función de cada componente que estará ligado a la placa ARDUINO, según los distintos usos que se les dará en el proyecto.

1.2 TRANSFORMADOR DE CORRIENTE ALTERNA

Existen distintos tipos de alimentación para la placa Arduino, entre ellos se encuentran por USB, transformador de corriente alterna a través de un puerto llamado Jack y una conexión directa al pin VIN mencionado anteriormente. En este caso se decidió utilizar el transformador de corriente alterna que se ve en la figura 1-6 por la cantidad de cables y porque al ser una maqueta estará siempre en un lugar de demostración práctica cerca de una toma de corriente. También conocido como fuente de alimentación o adaptador de alimentación, este dispositivo se enchufa a una toma de corriente y el voltaje alterno de la toma de corriente será transformado a voltaje continuo y además baja su voltaje a las necesidades del Arduino (7 a 12 [V])



Fuente: <https://www.geekfactory.mx/tutoriales/tutoriales-arduino/alimentar-el-arduino-la-guia-definitiva/>

Figura 0-6 Transformador de corriente alterna

Voltaje de entrada: 220 [V Corriente Alterna] (En Chile se trabaja con 220[V])

Voltaje salida: 7.5 [V Corriente Continua]

1.3 SENSOR DE DISTANCIA ULTRASONICO HC-SR04

Este sensor recibe el nombre de ultrasónico, ya que supera las frecuencias audibles por el oído humano. Este será utilizado en la maqueta para medir distancias.

Puede detectar objetos con diferentes materiales, formas, colores y superficies. Los materiales pueden ser sólidos, líquidos o polvorientos, sin embargo, deben ser reflectores de sonido. Los sensores trabajan según el tiempo de transcurso del eco, es decir, se valora la distancia temporal entre el impulso de emisión y el impulso del eco.

El sensor de distancia ultrasónico asemeja el sensor de distancia utilizado en los automóviles con sistema de estacionamiento asistido, estos sensores van ubicado en la parte trasera del vehículo porque es en esa área donde existen los mayores puntos ciegos del conductor y donde más ayuda necesita para no colisionar. Los sensores ultrasónicos detectan objetos a distancias que van desde 1.7 [cm] hasta 4.5 [m]. El sensor emite un sonido y mide el tiempo que la señal tarda en regresar. El sensor recibe el eco producido y lo convierte en señales eléctricas.



Fuente: <https://upgradeperuimports.com.pe/index.php/product/sensor-de-retroceso-ultrasonico-con-pantalla-led/>

Figura 0-7 Funcionamiento sensor de ultrasonido

En la figura 1-8 se muestra el sensor ultrasónico que se utilizará en la maqueta, en donde se indican todas las partes que lo componen:



Fuente: <https://www.dx.com/p/new-hc-sr04-ultrasonic-sensor-distance-measuring-module-3-3v-5v-compatible-for-arduino-2062050#.XQKDndL0ncs>

Figura 0-8 Sensor de ultrasonido

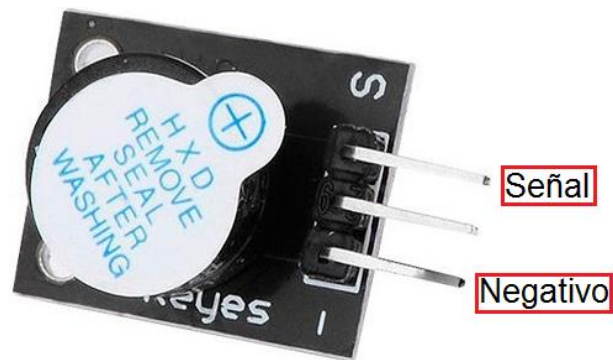
1.4 BUZZER KY-012

El zumbador o mejor conocido como buzzer es un pequeño transductor capaz de convertir la energía eléctrica en sonido. Para hacerlos funcionar solo se necesita conectar el positivo y negativo a batería o cualquier fuente de corriente directa y de esta manera sonara de forma continua. Para controlar el tiempo en el que emite sonido se conecta el positivo a una salida digital de Arduino, por la cual se manda un voltaje para cerrar el circuito y que funcione.

Estos dispositivos se utilizan principalmente para alarmas y controles de sonido con un pequeño margen de frecuencia, como en los electrodomésticos.

El buzzer asemeja el sonido que emite un vehículo con sistema de estacionamiento asistido, en los cuales al retroceder se escucha un sonido que indica que tan cerca está el vehículo de algún objeto con el que pueda colisionar. También se utilizan para advertir que un vehículo de carga está retrocediendo y de esa manera evitar un posible accidente de un peatón que este transitando cerca de él.

En la siguiente imagen 1-9 se puede observar el buzzer que se utilizara en la maqueta.



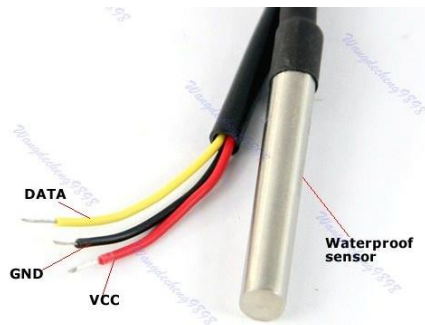
Fuente: <https://tresdprinttech.com/sonido/144-modulo-zumbador-activo.html>

Figura 0-9 Buzzer

1.5 SENSOR DE TEMPERATURA SUMERGIBLE DS18B20

El sensor de temperatura sumergible que se utilizará en la maqueta tiene una temperatura operativa se encuentra entre -50 y 125 grados Celsius. La precisión, en el rango comprendido entre -10 y 85 grados es de ± 0.5 grados.

Este componente que se muestra en la figura 1-10 asemeja un sensor de temperatura de un líquido en un vehículo, ya sea de líquido refrigerante o aceite de motor el cual nos indica la temperatura de estos para evitar daños en el motor debido a temperaturas elevadas. Otra de sus tantas aplicaciones que se le puede dar a este sensor consiste en medir la temperatura de una pecera para que los peces en su interior puedan estar a su temperatura óptima.



Fuente: <https://saber.patagoniatec.com/2014/06/ds18b20-sensor-temperatura-ds18b20-sumergible-agua-liquido-arduino-argentina-ptec/>

Figura 0-10 Sensor de temperatura sumergible

- Alimentación de 3V a 5.5V
- Rango de T°: -55°C a 125°C $\pm 0.5^\circ\text{C}$

1.6 SENSOR DE LLUVIA

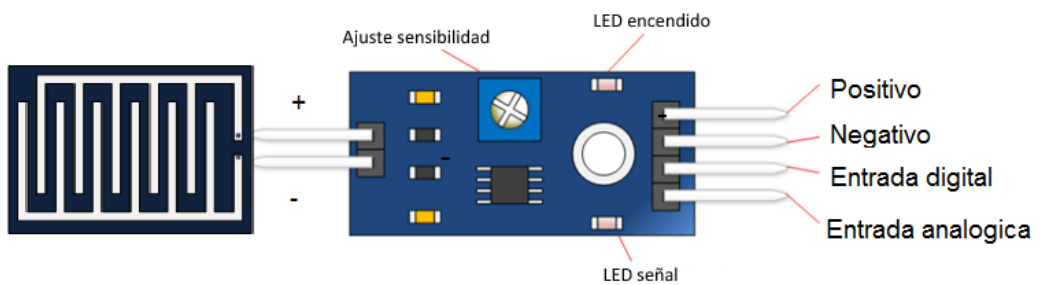
Este tipo de sensores detectan la presencia de lluvia por la variación del contacto entre componentes internos del sensor al entrar en contacto con el agua.

Constructivamente son sensores sencillos como se puede ver en la figura 1-11. Se dispone de diversos contactos, entrelazados entre sí a una pequeña distancia, sin existir contacto entre ambas. Al depositarse agua sobre la superficie, se pone en contacto eléctrico ambos conductores, lo que puede ser detectado por un sensor.

Los valores analógicos medidos varían desde 0 para una placa totalmente empapada, a 1023 para una placa totalmente seca.

La salida digital dispara cuando el valor de humedad supera un cierto valor, que se ajusta mediante el potenciómetro. Por tanto, se obtendrá una señal LOW en ausencia de lluvia, y HIGH con presencia de lluvia.

En los vehículos van ubicados en la parte superior del parabrisas y son utilizados para activar y desactivar los limpiaparabrisas al momento de detectar algún líquido en el sensor, estos forman parte del sistema de confortabilidad del vehículo esto quiere decir que no es un sensor imprescindible en el vehículo, pero sí que se agradece su existencia.



Fuente: <https://www.luisllamas.es/arduino-lluvia/>

Figura 0-11 Conexiones del sensor de lluvia

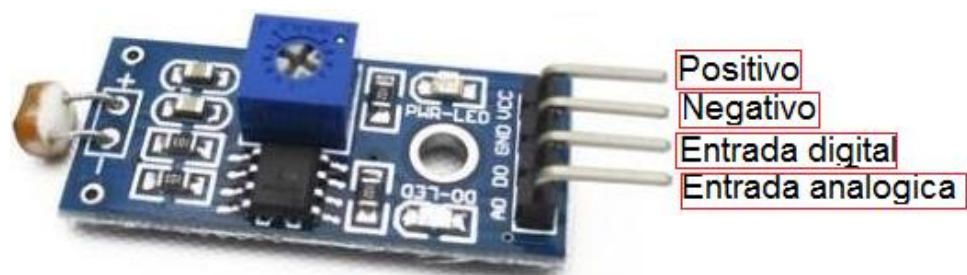
- Voltaje de trabajo: 3.3 V DC a 5 V DC.
- Señal de salida: Análoga y digital.

1.7 SENSOR DE LUZ POR FOTORESISTENCIA LM393

Este módulo de fotoresistencia es muy sensible a la luz ambiental. Este posee una resistencia que varía con la luz, por lo que, al momento de detectar mucha luz, la resistencia disminuirá, y por esto será utilizado en la maqueta.

La salida DO apreciada en la figura 1-12 se puede conectar directamente con un microcontrolador para detectar el ambiente de luz y crear un interruptor de control de luz.

El sensor de luz es utilizado en los vehículos como un sistema de confort, ya que al detectar poca luz enciende automáticamente las luces, este sensor va ubicado junto al sensor de lluvia porque al estar en la parte superior del vehículo puede detectar mejor la luz.



Fuente: <https://store.robodacta.mx/sensores/color-luz-y-vision/modulo-sensor-luz-ajustable-analogico/>

Figura 0-12 Sensor de luz por fotorresistencia

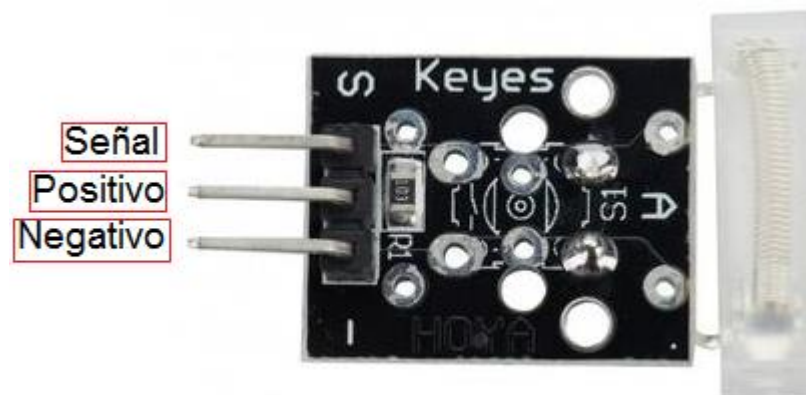
- Voltaje de funcionamiento: 3.3 - 5 V

1.8 SENSOR DE GOLPE (VIBRACION)

El Módulo KY-031 es mejor conocido como Sensor de Impacto, este sensor tiene la capacidad de detectar vibraciones en una superficie u objeto. La información de impacto es transformada por el sensor y enviada a la placa Arduino o a un microcontrolador que a su vez podrá disponer de dicha información como mejor se adapte al proyecto. Este se utilizará en la maqueta para emular el knock sensor del motor.

Este sensor asemeja a un knock sensor utilizado en el vehículo para detectar detonaciones en el motor, este sensor va ubicado a un costado del block para así recibir mejor las detonaciones y así determinar si estas son irregulares o no para informárselo a la ECU y que tome acciones al respecto.

A continuación, en la figura 1-13 se observa el sensor de impacto que se utilizará en la maqueta.



Fuente: <https://www.arcaelectronica.com/products/modulo-de-sensor-de-golpe>

Figura 0-13 Sensor de golpe

- Voltaje óptimo: 3 – 15V DC
- Salida analógica

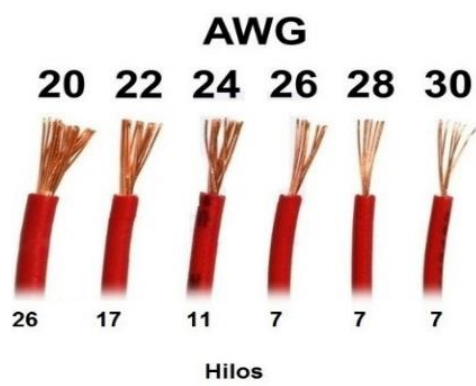
1.9 CABLE 22AWG

A continuación, se explicará el uso y nomenclatura del cable AWG (american wire gauge) o galga americana para cables, cuanto más alto es el número AWG, más delgado es el cable, como se muestra en la figura 1-14. El cable de mayor grosor (AWG más bajo) es menos susceptible a las interferencias, posee menos resistencia interna y, por esto, soporta mayor flujo de circulación de corriente a distancias más grandes como se visualiza en la tabla 1-1.

	Diámetro		Área	Resistencia por metro
AWG	Pulgadas	(mm)	(mm²)	Ω/m
20	0.031	0.812	0.519	0.0345
22	0.025	0.644	0.324	0.0548
24	0.020	0.511	0.205	0.0892

Fuente: <https://www.sab-cables.eu/productos/datos-tecnicos/cables-electricos/construccionesdevenasamericanas.html>

Tabla 1-1 Datos de cables



Fuente: https://articulo.mercadolibre.com.mx/MLM-605177660-rollo-de-cable-awg-calibre-24-de-18-metros-jumper-arduino-_JM?quantity=1&variation=18917822123

Figura 0-14 Cable 22AWG

También son conocidos como jumpers, estos cables son especiales para la placa Arduino por los conectores que posee en los terminales. Estos cables se muestran en la figura 1-15, y son los que se utilizarán en la maqueta.



Fuente: https://laniakea.mx/index.php?route=product/product&product_id=93

Figura 0-15 Jumpers

1.10 CABLE USB PARA ARDUINO (TIPO A-B)

USB es la sigla de Universal Serial Bus (Bus Universal en Serie). Se trata de un concepto de la informática para nombrar al puerto que permite conectar periféricos a una computadora.

Periférico es la denominación para nombrar al aparato o dispositivo auxiliar e independiente conectado a una computadora. Por ejemplo, un teclado, mouse, parlantes, etc.

En este caso Arduino es el periférico por lo que es necesario tener el cable USB para poner conectarlo al computador y así poder programar el funcionamiento que tendrá el ARDUINO con sus diferentes componentes.

Tipo A: Este conector se encuentra normalmente en computadores.



Fuente: <https://computerhoy.com/noticias/hardware/que-es-estandar-usb-que-tipos-hay-como-funciona-13811>

Figura 0-16 Conector Tipo A

Tipo B: Los dispositivos grandes, como impresoras o escáneres, suelen emplear este conector, esta es la entrada que utiliza ARDUINO.



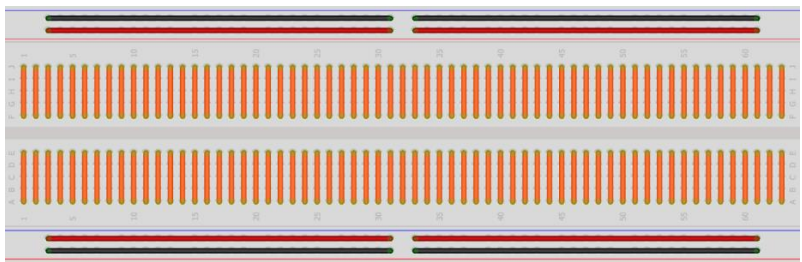
Fuente: <https://computerhoy.com/noticias/hardware/que-es-estandar-usb-que-tipos-hay-como-funciona-13811>

Figura 0-17 Conector Tipo B

1.11 PROTOBOARD

Esta ayudara a poder realizar las conexiones necesarias para hacer funcionar los sensores y actuadores en conjunto porque Arduino no tiene tantas salidas de voltaje, por ende, se conecta a la protoboard y se pueden alimentar todos los sensores y actuadores que se utilizarán.

La protoboard (*breadboard* en inglés) es una placa que posee unos orificios conectados eléctricamente entre sí siguiendo un patrón horizontal y vertical como se puede ver en la figura 1-18. Es empleada para realizar pruebas de circuitos electrónicos, insertando en ella componentes electrónicos y cables.



Fuente: <https://edutech.atlantistelecom.com/notices/2017/04/como-funciona-la-protoboard-166.php>

Figura 0-18 Protoboard

1.12 DIODO LED

Un led o diodo emisor de luz es un dispositivo semiconductor (diodo) que emite luz, actualmente se están utilizando en los vehículos debido a su bajo consumo y gran capacidad lumínica, en la maqueta estos representarán el sistema de iluminación delantero del vehículo.

Este fenómeno es una forma de electroluminiscencia, es decir, el color depende del material semiconductor empleado en la construcción del diodo.

Los diodos led tendrán la finalidad de demostrar el funcionamiento de los sensores, con esto se quiere decir que será el actuador de estos, una vez el sensor envíe una señal a ARDUINO, este mandará una señal al led para que se encienda.

En la figura 1-19 se muestran los diodos leds que se utilizarán en la maqueta.



Fuente: <http://www.teslaelectronics.cl/inicio/36-diodo-led-3mm.html>

Figura 0-19 Diodo LED

1.13 DISPLAY LCD 16X2

El display mostrado en la figura 1-20 será utilizado para mostrar la temperatura que se medirá con el sensor correspondiente, esta será mostrada tanto en grados Celsius como en grados Fahrenheit.

Las siglas LCD significan “Liquid Cristal Display” o pantalla de cristal líquido. Es una pantalla plana basada en el uso de una sustancia liquida atrapada entre dos placas de vidrio, haciendo pasar por este una corriente eléctrica a una zona específica, para que así se iluminen los sectores que se desean.



Fuente: <https://lualtec.es/display-lcd-1602a-azul.html>

Figura 0-20 Display LCD 16x2

El término LCD 16x2 se refiere a que cuenta con dos filas horizontales, de 16 caracteres (letras y números) cada una, que se utiliza para mostrar información, por lo general alfanumérica como se puede apreciar en la figura 1-20. Cada carácter se compone de puntos llamados pixeles los cuales se iluminan dependiendo de que se quiera mostrar en el display.

Las capacidades de estos dispositivos son altas, pues se puede mostrar todo tipo de información sin importar qué tipo de símbolos o caracteres sean, el idioma o el lenguaje, pues el sistema puede mostrar cualquier carácter alfanumérico, símbolos y algunas figuras como se muestra en la imagen 1-21, el número de píxeles que tiene cada símbolo o carácter varía dependiendo del modelo del dispositivo y cada artefacto está controlado por un microcontrolador que está programado para dirigir el funcionamiento y la imagen mostrada en la pantalla.



Fuente: <http://i1os.com/douHaIy39oM.video+related>

Figura 0-21 Display LCD

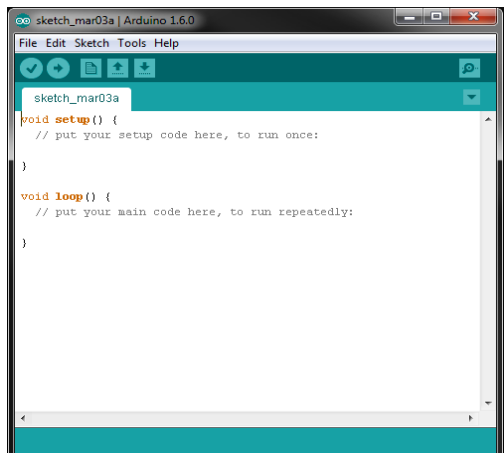
Características del display

- 16 caracteres x 2 líneas
- Caracteres de 5x8 puntos (píxeles)
- Puede mostrar letras, números, caracteres especiales, y hasta 8 caracteres creados por el usuario.
- Backlight (luz trasera) de LED color azul
- Caracteres color blanco
- Voltaje de alimentación: 5 [V]

Capítulo 2: “Funcionamiento de cada componente”

2.1 ARDUINO

El Arduino es una placa basada en un microcontrolador. Los microcontroladores son circuitos integrados que se pueden programar grabando instrucciones, las cuales se escriben con el lenguaje de programación que pueden utilizar en el entorno Arduino. Estas instrucciones permiten crear programas que interactúan con los circuitos de la placa. A continuación, en la figura 2-1, se muestra la plataforma de programación de Arduino utilizada en la realización de la maqueta.



Fuente: <https://aprendiendoarduino.wordpress.com/2016/03/29/entorno-de-programacion-de-arduino-ide/>

Figura 0-1 Plataforma de programación Arduino

El microcontrolador de Arduino posee una interfaz de entrada, que es una conexión en la que se puede conectar en la placa diferentes tipos de periféricos. La información de estos periféricos que se conecten se trasladará al microcontrolador, el cual se encargará de procesar los datos que recibe a través de ellos.

2.1.1 Entrada Analógica

Los microcontroladores poseen la capacidad de leer niveles de voltaje. Por ejemplo, un Arduino común y corriente podrá leer voltajes entre 0 y 5 voltios por defecto (no se debe superar los 5 voltios porque puede causar daños irreparables a la placa). Para ello posee una serie de pines llamados Entradas Analógicas como se ve en la figura 2-2.

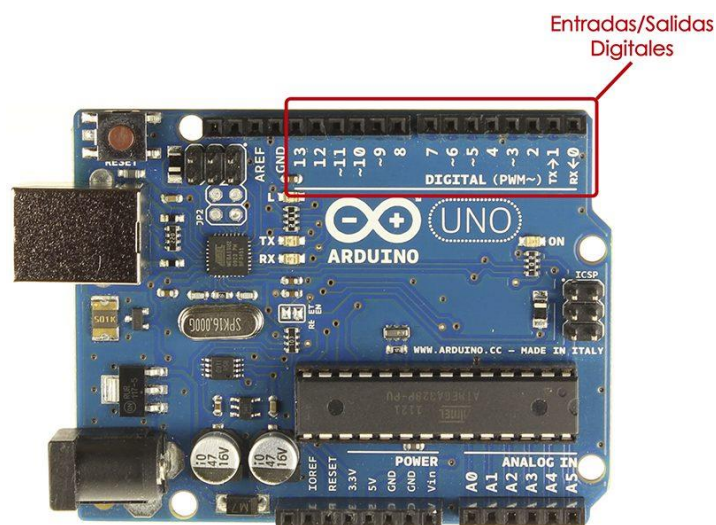


Fuente: <http://panamahitek.com/entradas-y-salidas-analogicas-y-digitales/>

Figura 0-2 Entradas analógicas

2.1.2 Entrada Digital

Un microcontrolador reconocerá un HIGH cuando se aplique más de 3 voltios (en los microcontroladores que operan a 5 voltios, como la mayoría de los Arduinos) a un terminal configurado como entrada digital; de lo contrario, el microcontrolador reconocerá el estado lógico como LOW. Las entradas digitales se pueden ver físicamente en la figura 2-3.



Fuente: <http://panamahitek.com/entradas-y-salidas-analogicas-y-digitales/>

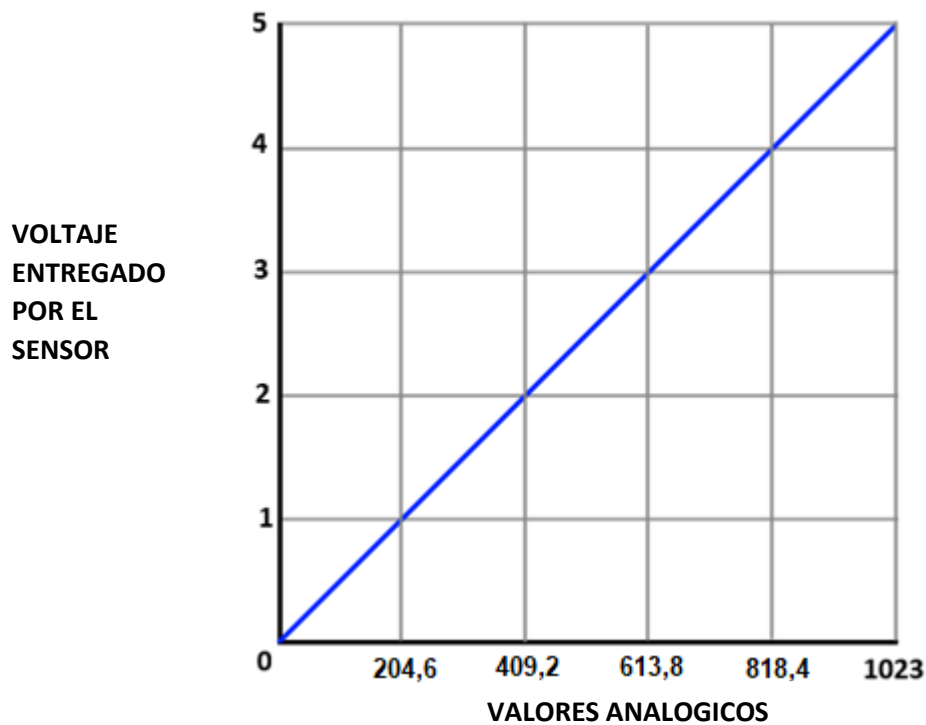
Figura 0-3 Entradas y salidas digitales

También cuenta con una interfaz de salida, que es la que se encarga de llevar la información que se ha procesado en el Arduino a otros periféricos. Estos periféricos pueden ser pantallas o altavoces en los que reproducir los datos procesados.

2.1.3 Salida Analógica

Las salidas analógicas son señales de voltaje generadas por el microcontrolador. Por ejemplo, si el usuario desea que un pin en específico entregue 3 voltios, lo podrá lograr. Si se desea que dicho voltaje suba o baje, esto será posible con la programación adecuada.

Para ello se utiliza un circuito llamado Convertidor Digital Analógico (ADC).



Fuente: <http://diymakers.es/sensores-en-entradas-analogicas-de-arduino/>

Figura 0-4 Grafico Voltaje entregado por el sensor vs Valor analógico

Para obtener estos valores existe una formula la cual es:

$$\text{Voltaje entregado por el sensor} = (\text{Valor analógico} / 1023) * \text{Voltaje de Arduino}$$

Para obtener el valor analógico, se despeja en la fórmula y se obtiene la siguiente ecuación:

$$\text{Valor analógico} = (\text{Voltaje entregado por el sensor} * 1023) / \text{Voltaje de Arduino}$$

Donde Voltaje entregado por el sensor es la señal de voltaje enviada por el sensor, ADC son los valores que puede tomar dicho voltaje, 1023 es el valor más alto que se puede interpretar del sensor y Voltaje de Arduino es el voltaje con el cual opera Arduino, es decir 5[V]

2.1.4 Salida Digital

Las salidas digitales son las terminales de un microcontrolador que permiten entregar un voltaje a una carga. Normalmente, las terminales de salida son las mismas terminales de entrada, con la diferencia de que dentro de la programación del microcontrolador se debe establecer la función que cumplirá determinado terminal en un momento dado.

2.2 SENSOR DE DISTANCIA ULTRASONICO HC-SR04

El sensor de distancia ultrasónico basa su funcionamiento en diversos cálculos y en la información recabada por el sensor al medir el tiempo entre el envío y la recepción de un pulso sonoro. Se conoce que la velocidad del sonido es 343 [m/s], pero en la fórmula se necesita este dato en [cm/μs] para encontrar la distancia en centímetros, por lo cual se realizó la siguiente operatoria matemática:

$$343 \frac{m}{s} \cdot 100 \frac{cm}{m} \cdot \frac{1}{1000000} \frac{s}{\mu s} = \frac{1}{29.2} \frac{cm}{\mu s}$$

Fuente: <https://www.luisllamas.es/medir-distancia-con-arduino-y-sensor-de-ultrasonidos-hc-sr04/>

Figura 0-5 Cálculo de la velocidad del sonido en [cm/μs]

En el cálculo anterior se logró determinar que el sonido tarda 29,2 microsegundos en recorrer un centímetro. Por lo tanto, se podrá obtener la distancia a partir del tiempo entre la emisión y la recepción del pulso enviado por el sensor mediante la siguiente ecuación.

$$distancia (cm) = \frac{tiempo (us) * 1cm}{29.2us * 2}$$

$$distancia (cm) = \frac{tiempo (us) * 1cm}{58.4us}$$

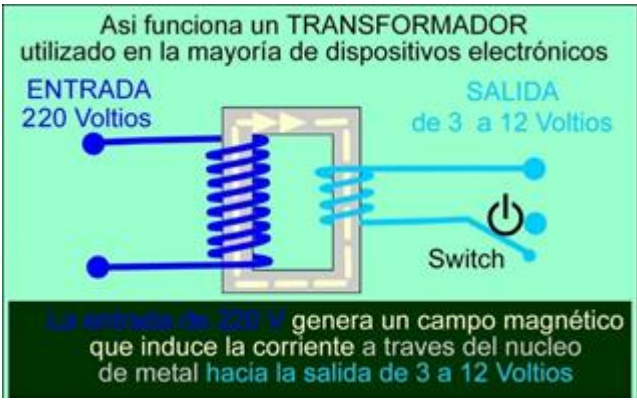
Fuente: <https://koalab.tech/aprende/componentes/sensor-ultrasonico-hc-sr04/>

Figura 0-6 Calculo de la distancia mediante el sensor HCSR04

Como se pudo apreciar en la figura 2-6, la distancia equivale al tiempo que tarda en volver la señal multiplicado por 1 [cm] y dividido en 58.4 microsegundos. El motivo de dividir por dos el tiempo (además de la velocidad del sonido en las unidades apropiadas, que se ha calculado antes) es porque se ha medido el tiempo que tarda el pulso en ir y volver, por lo que la distancia recorrida por el sonido es el doble de la que se quiere medir.

2.3 TRANSFORMADOR DE CORRIENTE ALTERNA

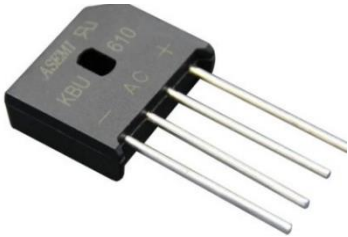
Dentro del transformador de corriente alterna hay dos bobinas de alambre que envuelven un núcleo de hierro como se muestra en la figura esquemática 2-7. La primera bobina recibe la corriente alterna de 220 [V] proveniente del toma corriente y genera un campo eléctrico en el núcleo de hierro. La segunda bobina convierte al nuevo campo eléctrico generado en una corriente alterna más pequeña. La medida del voltaje alterno resultante depende del número de espiras en la segunda bobina en relación con el número en la primera. Si la segunda bobina tiene la mitad de espiras que la primera, la corriente alterna resultante tendrá la mitad del valor de la que ingresó al adaptador. Por lo tanto, si hubiera 100 espiras en la primera bobina, la segunda sólo debería tener 50.



Fuente: <https://comofuncionaque.com/como-funciona-un-transformador/>

Figura 0-7 Reductor de voltaje

Dentro de la caja del adaptador, detrás de las dos bobinas y del núcleo de hierro, hay dos diodos envueltos en goma como se muestra en la figura 2-8, que rectifican la corriente alterna en corriente continua, permitiendo que la corriente fluya en una sola dirección.



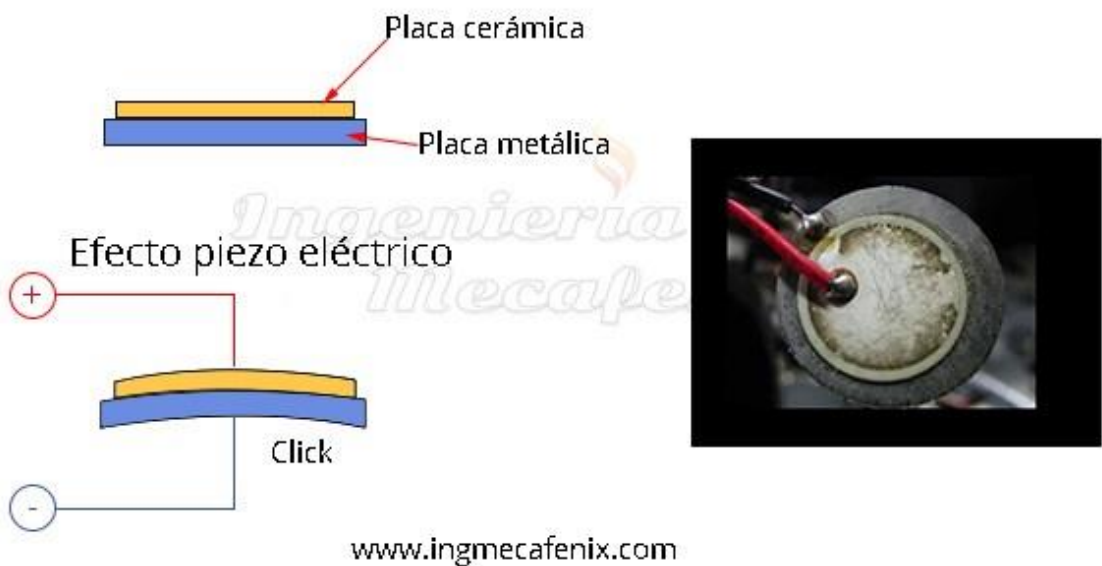
Fuente: <http://www.geekbotelectronics.com/producto/puente-rectificador-kbu610-linea>

Figura 0-8 Diodo rectificador

2.4 BUZZER

El funcionamiento se basa en el efecto piezoeléctrico de los materiales, Este efecto funciona de tal manera que cuando se aplica un voltaje el volumen del material cambia ligeramente. Los zumbadores están contruidos con dos pequeñas placas una metálica y una cerámica, las cuales aprovechan este efecto, pero solo generan un leve golpe seco, ya que los materiales cambiaron de forma, pero no regresan a su estado natural hasta que se les quita el voltaje.

Para que se pueda emitir un sonido continuo las placas necesitan vibrar constantemente, para ello se instala un oscilador que hace que los materiales cambien de estado una y otra vez, y así puedan cambiar su frecuencia para poder alcanzar un audio perceptible.



Fuente: <https://www.ingmecafenix.com/electronica/el-buzzer/>

Figura 0-9 Piezo eléctrico

2.5 SENSOR DE TEMPERATURA SUMERGIBLE

Un sensor de temperatura sumergible es un instrumento electrónico capaz de realizar una lectura precisa de una temperatura en un ambiente determinado.

Una de las técnicas utilizadas se denomina RTD (Resistance temperature detector), consiste en detectar los cambios de resistencia de un resistor interno debido a la temperatura (es una medición indirecta).

En la siguiente figura 2-10 se puede apreciar el sensor de temperatura que se utilizará en la maqueta con sus terminales de conexión.



Fuente: <https://moviltronics.com.co/sensores/434-sensor-de-temperatura-sumergible-ds18b20.html>

Figura 0-10 Sensor de temperatura sumergible

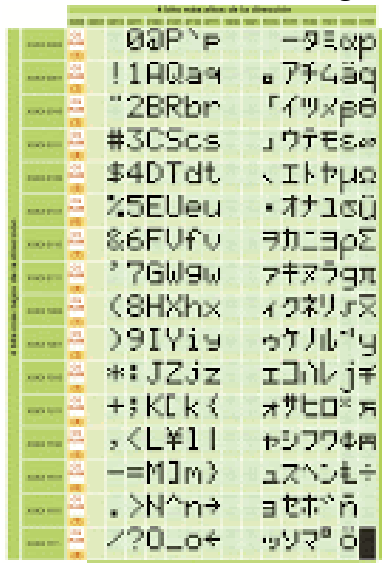
Este dispositivo que se comunica de forma digital. Cuenta con tres terminales, dos de alimentación y el pin “data”. Con Arduino se puede leer la temperatura que registra este sensor que posee una característica muy peculiar. Utiliza la comunicación OneWire, la cual es un poco complicada para los que no tienen mucha experiencia en el área de electrónica digital. Básicamente se trata de un protocolo especial que permite enviar y recibir datos utilizando un solo cable, a diferencia de la mayoría de los protocolos que requiere dos vías.

2.6 DISPLAY LCD 16X2

Pantalla LCD (Liquid Crystal Display) es un módulo de visualización electrónica que consta de 3 memorias: DDRAM, CGROM y CGRAM.

DDRAM (Data Display Ram): memoria en donde se almacena los datos que se mostrarán en el display.

CGROM: es una memoria interna en la cual se guardan los caracteres que se pueden visualizar en el display lcd como se muestra en la figura 2-11.



Hex	Char	Hex	Char
00	P	01	^
02	a	03	q
04	r	05	0
06	s	07	0
08	t	09	0
0A	u	0B	0
0C	v	0D	0
0E	w	0F	0
10	x	11	0
12	y	13	0
14	z	15	0
16	{	17	0
18		19	0
1A	}	1B	0
1C	~	1D	0
1E	0	1F	0
20	0	21	0
22	0	23	0
24	0	25	0
26	0	27	0
28	0	29	0
2A	0	2B	0
2C	0	2D	0
2E	0	2F	0
30	0	31	0
32	0	33	0
34	0	35	0
36	0	37	0
38	0	39	0
3A	0	3B	0
3C	0	3D	0
3E	0	3F	0

Fuente: <http://todoelectrodo.blogspot.com/2013/02/lcd-16x2.html>

Figura 0-11 Posibles caracteres a utilizar

CGRAM (Character Generator Ram): memoria en la cual se almacenan los caracteres especiales (creados por el programador)

En la programación se decide utilizar la CGRAM y/o la CGROM dependiendo lo que se quiere mostrar en pantalla. En el caso de la maqueta solo se utilizará la CGROM, ya que no se utilizarán caracteres especiales, sino los datos entregados por el sensor de temperatura sumergible DS18B20.

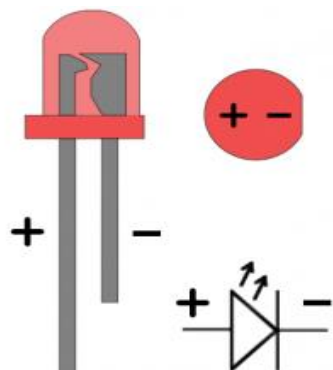
La pantalla LCD dispone de un microcontrolador propio para que cuando reciba la información del Arduino la interprete y de esta manera mostrar los caracteres que se necesitan. Por ejemplo, se quiere escribir el carácter “A”, para realizar esto Arduino enviará una señal en forma de voltaje al microcontrolador de la pantalla LCD para que procese esta información y gracias a la electricidad rotar los cristales en el interior del líquido y dejar pasar la luz hacia el pixel que sea necesario. En la parte trasera del LCD tiene un sistema de iluminación a base de leds, frente a ellos hay una placa de cristal líquido la que gracias a la electricidad puede modificar la posición de los cristales y por tanto permitir el paso de luz en los pixeles seleccionados y obstruir la iluminación a los demás

pixeles por el ángulo que toman los cristales, estos harán rebotar la luz o dejará el paso libre.

Para reducir la cantidad de conexiones entre el Display LCD y el Arduino se utiliza un módulo llamado I2C, el cual se conecta entre el Arduino y la pantalla, estableciendo una comunicación entre estos y de esta manera tener solo tener 2 pines de datos y 2 de alimentación, disminuyendo considerablemente los pines que se requerían para conectar la pantalla LCD.

2.7 DIODO LED

El funcionamiento es muy sencillo. Cuando se conecta con polarización directa el diodo led, el semiconductor en el interior del led permite el paso de la corriente que circulará por las patillas (cátodo y ánodo) y al pasar por el semiconductor, este emite luz. El diodo led puede ser conectado de una sola manera porque, al ser un diodo, solo deja pasar corriente en una dirección. Este posee un positivo (ánodo) y un negativo (cátodo), como se muestra en la figura 2-12. El positivo se identifica como la patilla más larga y el negativo como la patilla más corta. La otra forma de reconocer el polo negativo es por la forma de la base, en donde se puede apreciar un corte en la circunferencia del led.



Fuente: [https://www.thereminos.com/es/hardware/outputs/actuators](https://www.thereмино.com/es/hardware/outputs/actuators)

Figura 0-12 Partes de un diodo led

2.8 SENSOR DE GOLPE (VIBRACION) KY-031

El principio de funcionamiento es muy sencillo. El dispositivo dispone de un cilindro, con dos contactos. Uno de los contactos está unido a una varilla metálica ubicada en el centro del cilindro. A su alrededor, el otro contacto se arrolla en forma de muelle, como se muestra en la figura 2-13.

En caso de una vibración, el muelle se deforma por efecto de la inercia, estableciendo contacto en varios puntos con el contacto fijo. De esta forma, se establece una conexión eléctrica entre ambos, que puede ser leída con un microprocesador.



Fuente: <https://www.luisllamas.es/medir-vibracion-con-arduino-y-sensor-sw-18020p/>

Figura 0-13 Ilustración de contactos internos del sensor de vibración

2.9 SENSOR DE LUZ POR FOTORRESISTENCIA

La base del funcionamiento de una fotorresistencia está en su componente principal, el sulfuro de cadmio (CdS). Este componente químico es un semiconductor que tiene la capacidad de variar su resistencia según la cantidad de luz que él capte.

Cuanto mayor es la intensidad de la luz que incide sobre el sulfuro de cadmio, más baja es la resistencia, es decir mayor facilidad de los electrones para moverse. Es importante saber que la variación de la resistencia cuando hay cambios de luminosidad repentinos no sigue la misma velocidad, sino que tiene retardo.

En la siguiente figura 2-13 se puede apreciar el sensor de luz por fotorresistencia utilizado en la maqueta.



Fuente: <https://como-funciona.co/una-fotorresistencia/>

Figura 0-14 Sensor de luz

2.10 SENSOR DE LLUVIA

El sensor está constituido de unas placas metálicas recorridas por una corriente como se muestra en la figura 2-14. Cuando las placas metálicas entran en contacto con el agua estas producen una conducción de corriente gracias a que se unen mediante el agua. De esta manera se detecta la presencia de un líquido.

Luego, según la conducción que hay se puede saber si hay mucha o poca cantidad de agua. Es por eso que uno de los pines ofrece una señal analógica. Al leer esta señal analógica a través de los pines de Arduino se puede obtener una señal de 0 (la placa está seca y no hay conducción de corriente) a 1023 (la placa está empapada y ofrece el máximo de corriente que puede dar).



Fuente: <http://www.bytheway.com.co/sensor-de-humedad/107-sensor-detector-de-lluvia-agua-fc-37-fc37-arduino.html>

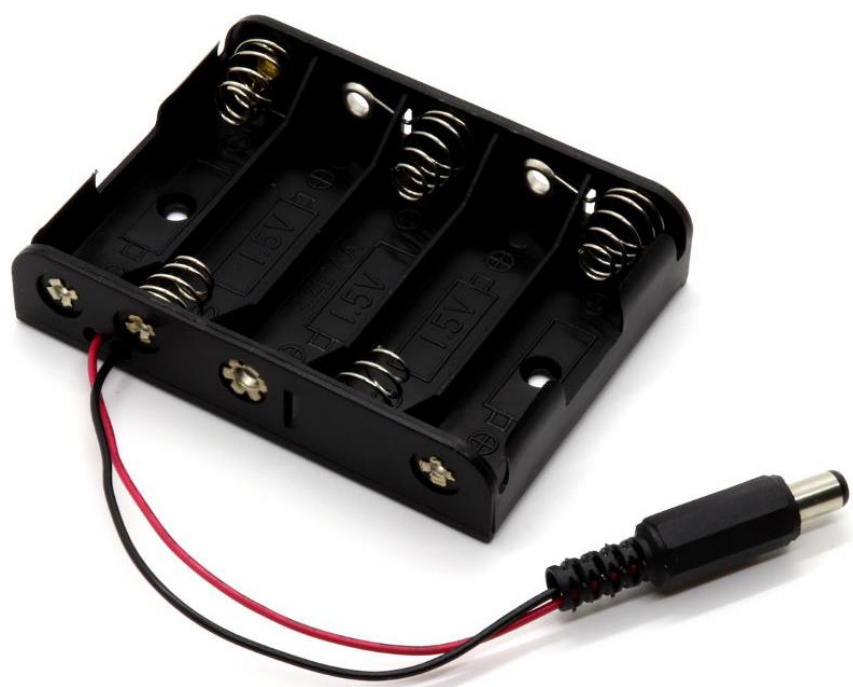
Figura 0-15 Sensor de lluvia

Capítulo 3: “Conexionado de la maqueta”

3.1 INTRODUCCION

Una vez entendidos los conceptos anteriores se procederá a explicar el armado de la maqueta, lo cual consistirá en una breve reseña del motivo de conectarlos así y detallar los pines que se utilizarán junto a su respectiva imagen de conexiones.

Con respecto a la alimentación de Arduino y los sensores cabe señalar que, para poder aplicarlo en cualquier vehículo, es necesario cambiar la alimentación del adaptador de corriente alterna por un porta pilas con conector Jack, el cual consta de 5 pilas AA de 1,5 [V], lo cual proporciona 7,5 [V] aproximadamente, suficientes para la alimentación del circuito completo. En la siguiente imagen se muestra el porta pilas mencionado.



Fuente: <https://shop.complubot.com/portapilas-5xaa-con-jack-compluino--arduino-p-1-50-224/>

Figura 0-1 Porta pilas con conector Jack

3.2 CONEXIONES DEL SISTEMA DE ASISTENCIA AL ESTACIONAMIENTO

El sistema de asistencia al estacionamiento consiste básicamente en un sensor encargado de medir la distancia y de un actuador que emita un sonido para alertar de la distancia a la que se encuentra el vehículo de un obstáculo. En la maqueta se decidió utilizar el sensor de ultrasonido como una herramienta de medición de distancia en conjunto con el buzzer, para que emita el sonido. Para llevar a cabo este sistema se debe conocer las conexiones que necesita cada componente, las cuales son:

Sensor de ultrasonido:

Pin VCC: Este pin es utilizado como la alimentación del sensor (positivo).

Pin TRIG: Este pin es utilizado como señal del Arduino hacia el sensor en forma de pulso, lo que provoca que el transmisor emita un sonido. Debe ser utilizado obligatoriamente un Pin PWM, ya que la señal que envía Arduino debe ser analógica.

Pin ECHO: Este pin es utilizado como señal del sensor al Arduino enviando la información que recibe cuando detecta el regreso del sonido.

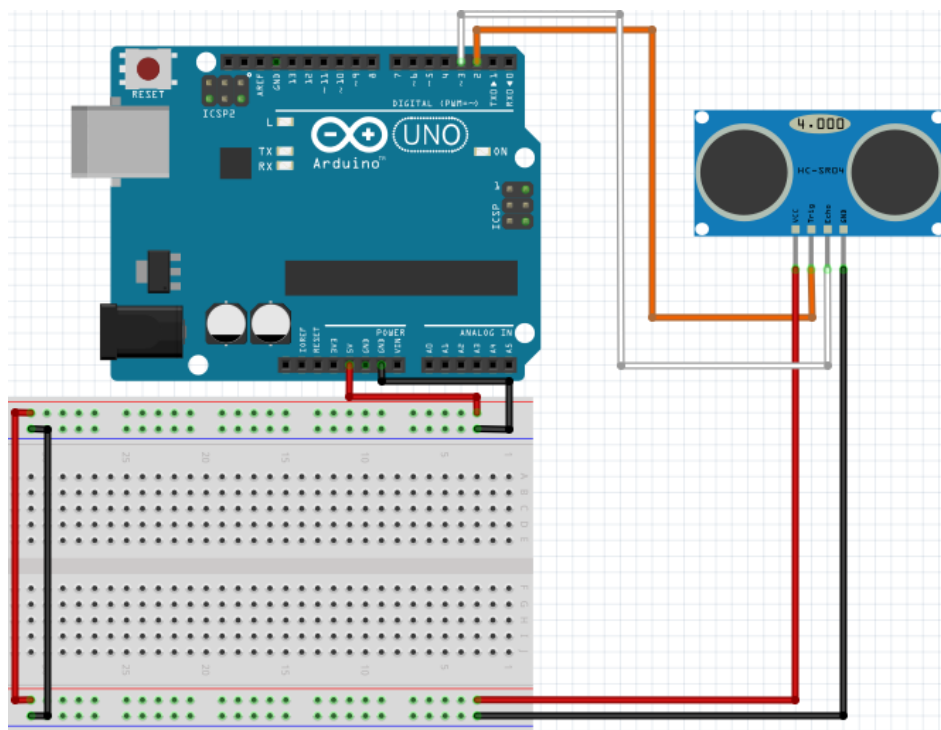
Pin GND: Este pin es utilizado como la tierra del sensor (negativo).

Buzzer

Pin “-”: Este pin es utilizado como tierra del buzzer (negativo).

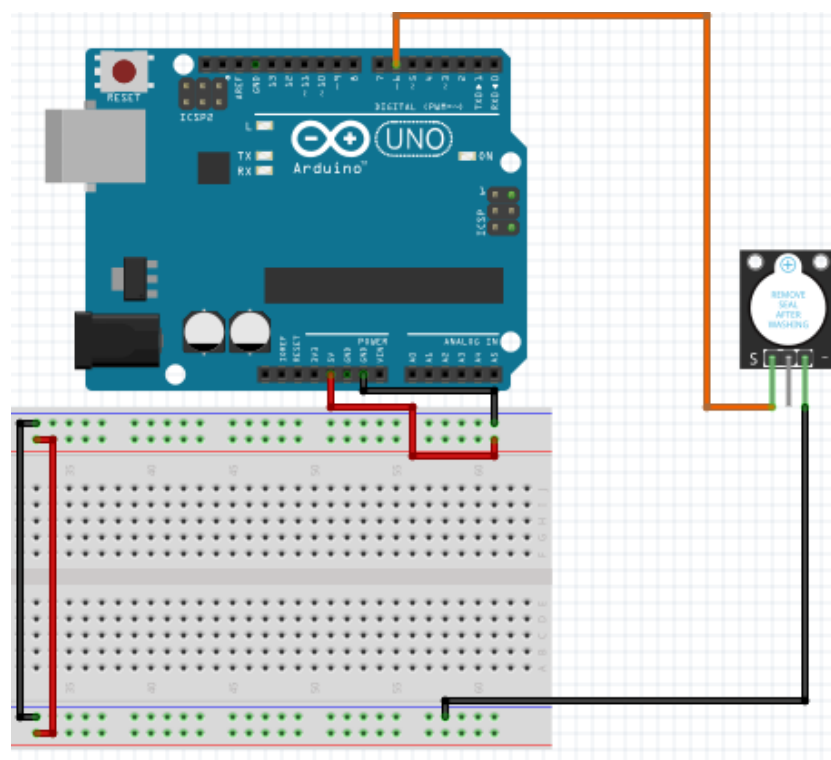
Pin S: Este pin es utilizado como señal proveniente de Arduino como positivo para así cerrar el circuito.

A continuación, se muestran las imágenes de conexión del sensor de ultrasonido y del buzzer en la maqueta:



Fuente: propia

Figura 0-2 Conexión sensor ultrasónico



Fuente: Propia

Figura 0-3 Conexiones Buzzer

3.3 CONEXIONES SISTEMA DE MONITOREO DE TEMPERATURA

Este sistema consta principalmente de un sensor de temperatura y de un display LCD para visualizar la medición obtenida del sensor.

En la maqueta se utilizó el sensor de temperatura DS18B20, el cual es un dispositivo que se comunica de forma digital. Cuenta con tres terminales: Vcc, GND y el pin Data. Este sensor utiliza comunicación OneWire, este protocolo permite enviar y recibir datos utilizando un solo cable, a diferencia de la mayoría de los protocolos que requieren dos cables.

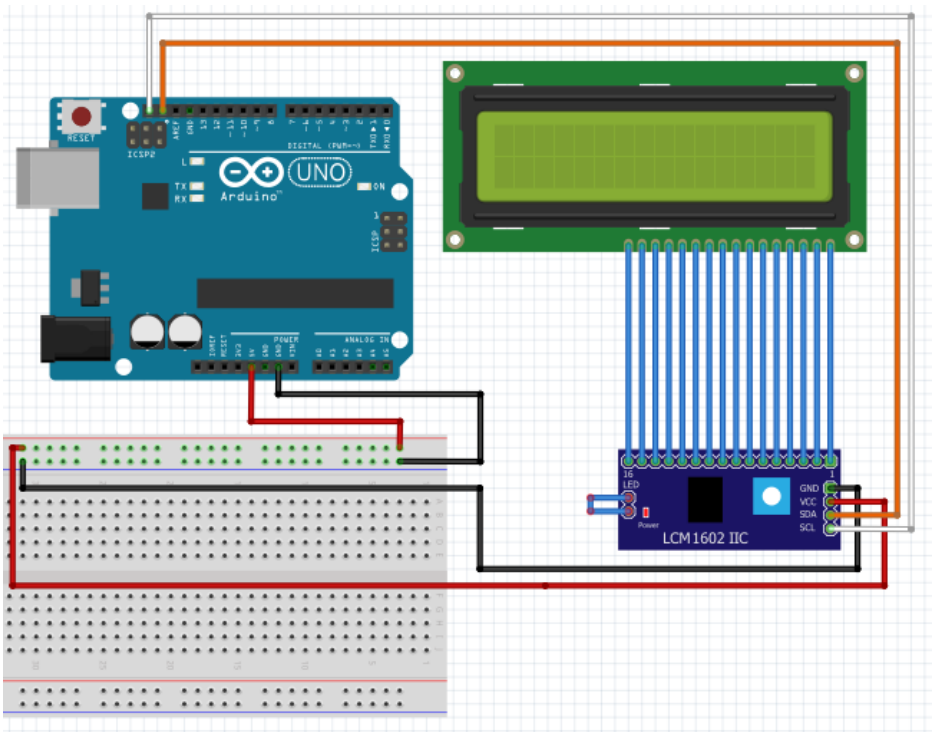
Pines del sensor de temperatura:

- Alimentación de 5 volts. (Cable rojo)
- Tierra o común del circuito. (Cable negro)
- Señal digital. (Cable amarillo)

Por otra parte, se encuentra la pantalla LCD, la cual cuenta con un total de 16 pines que mediante el módulo I2C esto se ve reducido a tan solo 4 pines. Esta pantalla se utilizó para mostrar la temperatura obtenida en grados Celsius y Fahrenheit.

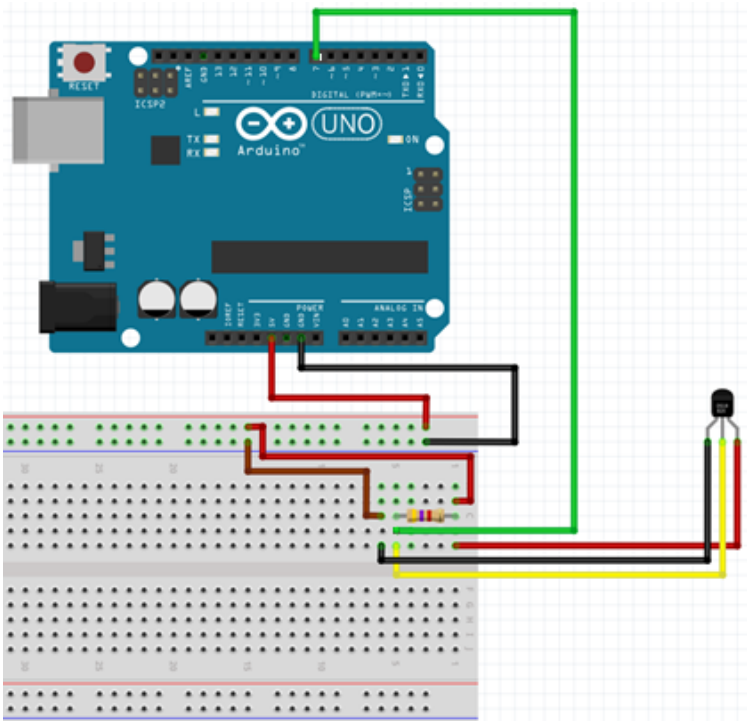
Para el correcto funcionamiento de este actuador son necesarios 4 pines:

- Alimentación de 5 volts (VCC). (Cable rojo)
- Tierra o común del circuito (GND). (Cable negro)
- Señal SDA (datos). (Cable naranja)
- Señal SCL (reloj). (Cable blanco)



Fuente:propia

Figura 0-4 Conexión I2C y LCD con Arduino



Fuente:propia

Figura 0-5 Conexión sensor de temperatura

3.4 CONEXIONES DEL SISTEMA LIMPIA PARABRISAS AUTOMATICO

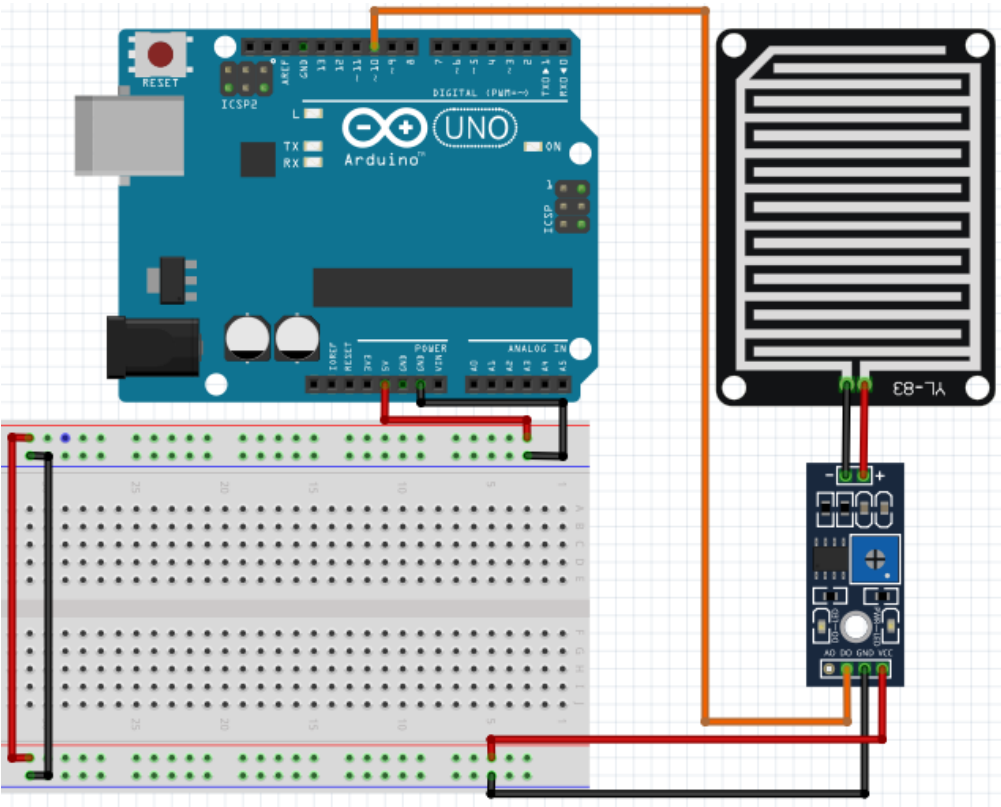
Este sistema se compone de un sensor de lluvia junto a el limpia parabrisas el cual será controlado mediante un relé, al detectar agua se activará el limpia parabrisa para retirar el agua y que el conductor pueda ver. En la maqueta solo se emulará su funcionamiento con un sensor de lluvia y unos leds.

El sensor de lluvia necesita tanto una señal analógica (A0) como una digital (D0) para su funcionamiento, la señal analógica, porque puede tomar un valor entre 0 y 1023 dando a conocer que tan mojado está el sensor y la señal digital simplemente informa si el sensor esta mojado o no sin saber si es mucho o poco.

En el caso de la maqueta solo se utilizó la señal de tipo digital debido a que el objetivo propuesto para este sensor es demostrar que puede detectar la presencia de lluvia, y de esta manera encender un led emulando el accionamiento de un motor limpia parabrisa.

Para el correcto funcionamiento de este sensor son necesarios 4 pines:

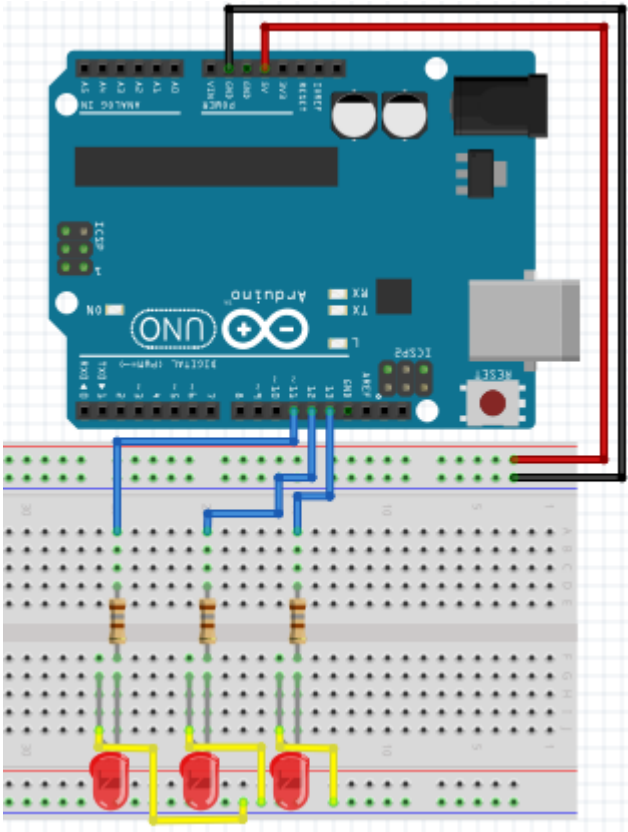
- Alimentación de 5 volts (VCC). (Cable rojo)
- Tierra o común del circuito (GND). (Cable negro)
- Señal analógica (A0). (Cable blanco)
- Señal digital (D0). (Cable naranja)



Fuente: propia

Figura 0-6 Conexiones sensor de lluvia

A continuación, se mostrarán los leds que se utilizarán para emular las 3 velocidades del limpia parabrisas (baja, media y alta velocidad).



Fuente: propia

Figura 0-7 Conexiones LEDS

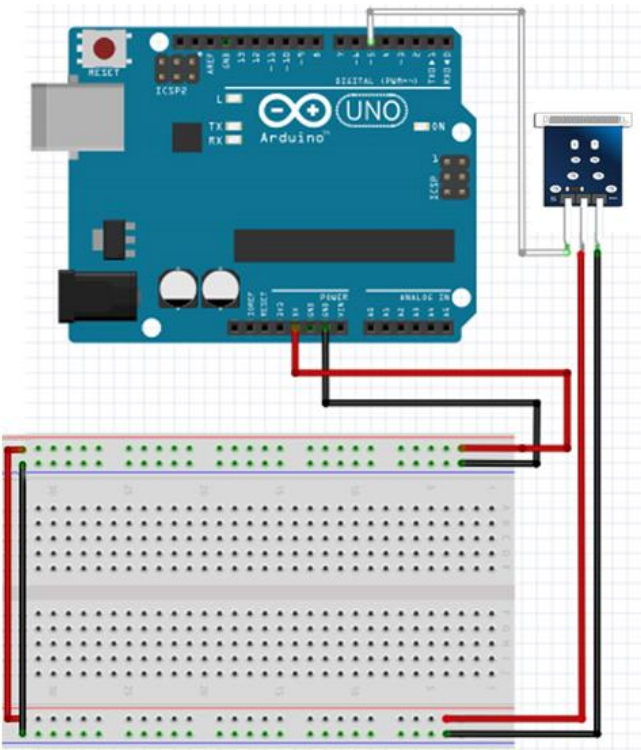
3.5 CONEXIONES DEL SISTEMA DE ALARMA

Para lograr el correcto funcionamiento del sistema de alarma se debe tener un sensor de vibración y un buzzer para que cuando se altere la inercia del sensor se modifique el muelle cierre el circuito y envíe la señal para que así el buzzer suene.

Para el correcto funcionamiento de este sensor son necesarios 3 pines:

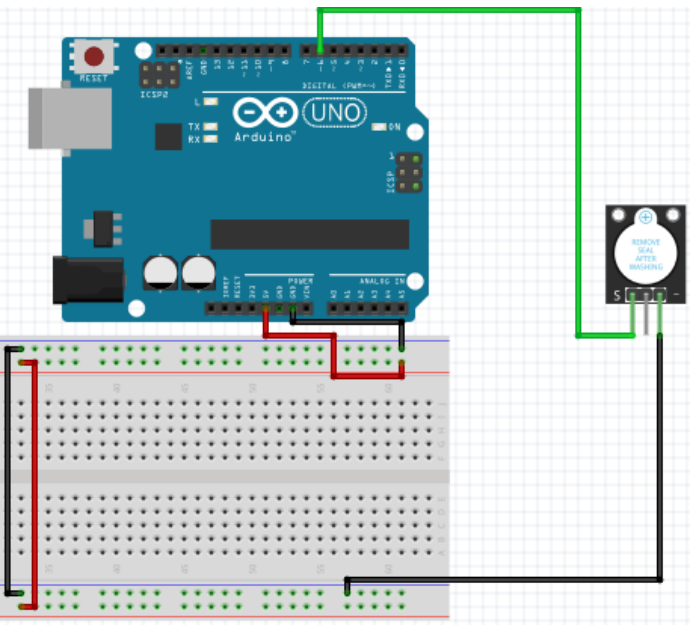
- Alimentación de 5 volts. (Cable rojo)
- Tierra o común del circuito (-). (Cable negro)
- Señal(S). (Cable blanco)

En la siguiente imagen están las conexiones del sensor de vibración y del buzzer del sistema de alarma.



Fuente: propia

Figura 0-8 Conexión sensor de vibración



Fuente: propia
Figura 0-9 Buzzer para alarma

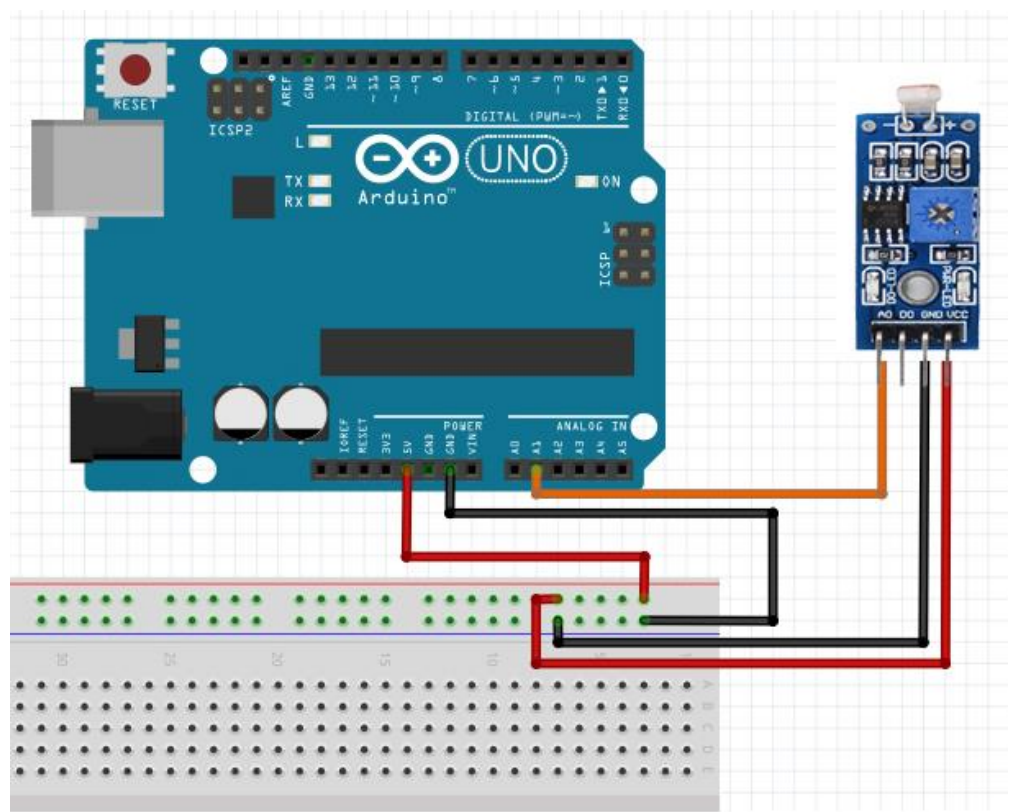
3.6 CONEXIONES DEL SISTEMA DE ILUMINACION AUTOMATICA

Los componentes necesarios para realizar este sistema son el sensor de luz y los focos del vehículo (luces altas y bajas), cuando detecta cierto nivel de luminosidad estos encenderán las luces bajas o altas dependiendo de la cantidad de luz

El sensor de luz por fotoresistencia tiene la capacidad de variar la resistencia según la cantidad de luz que este recibe. Este sensor consta de 4 pines, análogo, tierra, positivo y digital, de los cuales se utilizarán solo 3 de estos, dejando fuera el pin digital. Esto debido a que el pin análogo entrega un gran rango de voltajes, el cual permite medir con mayor exactitud la luz ambiental. En cambio, el pin digital solo entrega o no voltaje (tiene dos estados, HIGH y LOW) lo cual solo permitiría saber si hay presencia de luz o no.

Este sensor se compone de los siguientes pines con sus respectivos colores:

- Alimentación de 5 volts (VCC). (Cable rojo)
- Tierra o común del circuito (GND). (Cable negro)
- Señal analógica(A0). (Cable naranja)



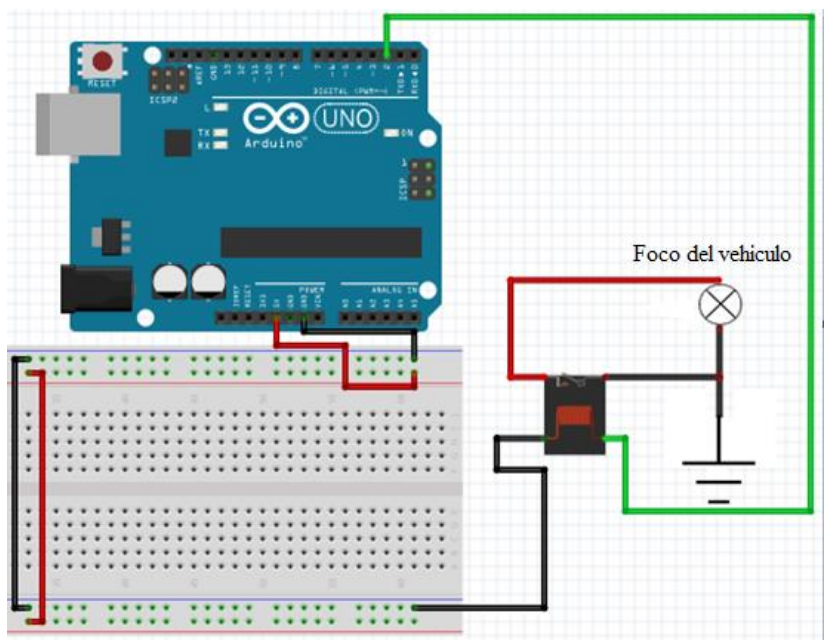
Fuente: propia

Figura 0-10 Conexión sensor de luz

3.7 APLICACIONES SOBRE 5[V]

Para realizar trabajos que necesiten un voltaje superior al que Arduino puede manejar, se tendrá que utilizar un relé para que de esta manera Arduino solo se encargue de controlar el relé y no energizar otros componentes. Para activar la bobina al interior del relé, la placa Arduino alimenta los puntos 85 y 86, y de esta manera crear un campo magnético, lo que hace que el switch baje y se pongan en contacto los puntos 87 y 30. De esta manera Arduino sólo controlará la alimentación del dispositivo a través del relé.

Por ejemplo las luces automaticas, el cual opera sobre los 5[V] deberá ser controlado por el relé, el que es comandado por Arduino. Se tendrá que utilizar la fuente que ya posee (bateria) y conectarla a los puntos 87 y 30 del relé, mientras que Arduino se debe conectar con una salida digital cualquiera hacia el relé (punto 86), y un negativo a la última conexión del relé (85) que puede ser de la placa Arduino.



Fuente: propia

Figura 0-11 Circuito de control del relé

Capítulo 4: “Programación”

4.1 INTRODUCCION

La programación se puede encontrar en prácticamente todo el entorno, desde un celular hasta un vehículo, pero ¿qué significa programar?

Programar es, mediante códigos, dar órdenes a un actuador dependiendo de la señal que reciba el microcontrolador.

En el área automotriz esto se puede ver en:

- Sistema de frenos ABS: ya que, al detectar el sensor que una rueda está derrapando este mandará una señal que la ECU detectará y a su vez dará la orden para que el sistema libere a esta rueda del efecto de frenado solo por un instante.
- Sistema de asistencia de estacionamiento: ya que, dependiendo de la distancia a la que se encuentre un objeto la ECU determinará la frecuencia en que el buzzer sonará alertando de un peligro.

Todos estos sistemas trabajan bajo ciertos parámetros que pueden ser modificados, también gracias a la programación se pueden realizar distintos tipos de cambios a los vehículos, ya sea potenciarlos o disminuir su consumo de combustible, entre otros.

A continuación, en la figura 4-1 se muestra el instrumento para realizar los cambios mencionados anteriormente (reprogramación)



Fuente: <https://www.alientech-tools.com/es/k-tag/>

Figura 0-1 Reprogramador de ECU

Cabe señalar que, en el caso de la maqueta a realizar, se decidió hacer diferentes experiencias con cada sensor por separado, esto porque la programación necesaria para llevar a cabo una actividad con todos los sensores en conjunto es posible, pero aumentaría considerablemente la dificultad de comprensión del código necesario para llevar a cabo dicha acción. Por lo dicho anteriormente, se tomó la decisión de realizar códigos específicos para cada experiencia que es posible desempeñar en la maqueta, y de este modo hacer que la comprensión de cada código sea más fácil de entender, además de aprender la comunicación entre la placa Arduino y el programa utilizado en el computador para programarlo.

Por la complejidad que presenta el Capítulo 4, se sugiere que se tome como un material de estudio, el cual necesita para su comprensión total realizar más de una lectura. El estudio en conjunto con la maqueta de ARDUINO facilitará el entendimiento de la programación a realizar.

4.2 PROGRAMACION

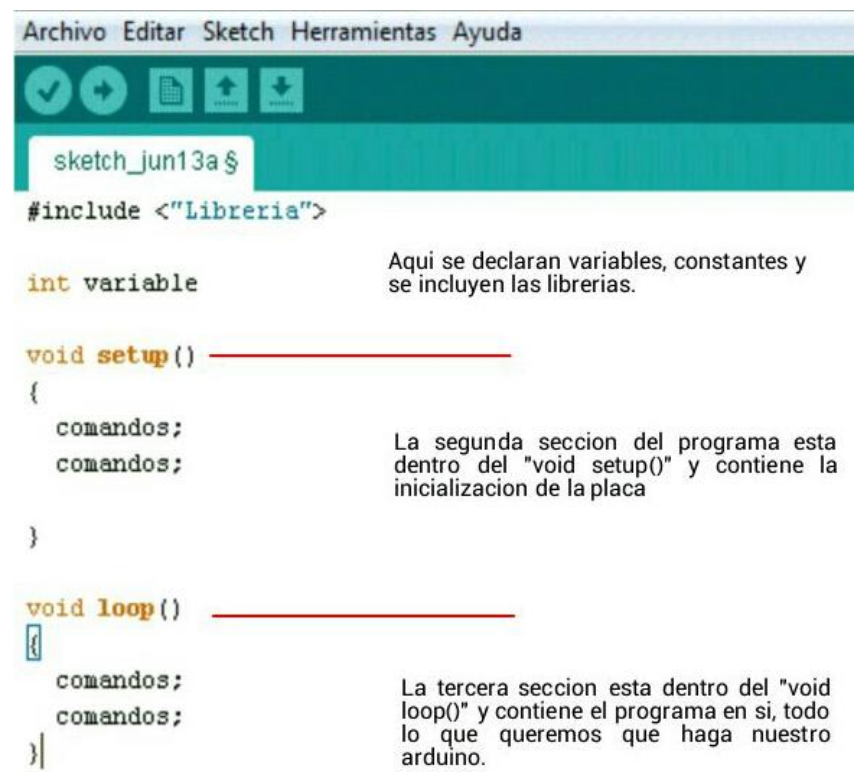
Dentro de la programación se pueden encontrar 3 grandes sectores:

El primer sector visto en la Figura 4-2 que se muestra a continuación, es utilizado para declarar pines, valores y librerías, esto quiere decir informar a Arduino que pines se utilizarán y que valores tendrá dependiendo de las señales.

Las librerías son un conjunto de códigos que facilitan la conexión entre sensores, pantallas, módulos electrónicos, etc. El entorno de Arduino ya incluye algunas librerías de manera que facilita, por ejemplo, mostrar texto en pantallas LCD.

El segundo sector es llamado “void setup” el cual solamente se utiliza cuando se enciende el Arduino ya con un archivo en su memoria o cuando se envía el archivo y lo lee por primera vez, esto quiere decir que solo se escribirá en void setup criterios de ejecución única. Por ejemplo: encender la pantalla LCD porque esta solo se enciende una vez al momento de conectar Arduino.

El tercer sector es llamado “void loop” este sector siempre se lee después de void setup y tiene una duración infinita, como su nombre lo indica es un bucle de comandos que se repite hasta que se apague el Arduino. Por ejemplo: la siguiente secuencia escrita en un LCD se determinó que aparezca el 1 luego el 2 y por último el 3, al finalizar esta secuencia volverá al inicio y empezará desde el 1 nuevamente repitiendo el ciclo infinitas veces.



Fuente: <http://ardurobotics.blogspot.com/2013/06/estructura-de-un-programa-arduino.html>

Figura 0-2 Sectores de programación Arduino

4.3 CODIGOS

La manera de realizar las acciones anteriormente mencionadas en los ejemplos es mediante comandos, existen infinidad de comandos algunos de estos se utilizan en prácticamente todas las programaciones y otros son solamente para componentes muy específicos. Memorizar cada uno de estos códigos es prácticamente imposible, pero gracias a la gran comunidad de Arduino y a su software libre es fácil encontrar la información sobre los códigos que se deben utilizar en el proyecto que se quiera llevar a cabo.

Se dará comienzo por el primer sector aprendiendo a declarar pines, el nombre que se le dará a cada pin y valores (solo se darán a conocer los códigos utilizados en la maqueta).

- El primer código utilizado es “int” o integer que significa entero este puede tomar un valor entero entre -32.768 y 32.767. Luego se deberá introducir el nombre con el que se quiera llamar a este valor, por último, se tendrá que utilizar un signo igual (=) y escribir el pin que se esté utilizando, como se muestra a continuación:

```
int SENSOR = 5;
```

Fuente: propia

Figura 0-3 Ejemplo uso int

Aquí se puede ver que se utilizó el comando int dándole el nombre de SENSOR y se puede apreciar el uso del pin 5.

- El segundo código utilizado es “bool”, el cual solo puede tener 2 valores: true (verdadero,1) o false (falso,0). Normalmente este valor dependerá del valor de medida que entregue el sensor. A continuación, se muestra un ejemplo del código bool:

```
bool digitalValue;
```

Fuente: propia

Figura 0-4 Ejemplo codigo bool

Aquí se puede ver el comando bool dándole el nombre digitalValue, ya que este tomará un valor digital, es decir, 0 o 1 que se lo dará la lectura del sensor.

- El tercer código utilizado será el “float”, traducido al español como “números de coma flotante”, o sea números decimales. Es utilizado para trabajar con este tipo de números, porque int no permite este uso.

```
float TempActualCB=0.00;
```

Fuente propia

Figura 0-5 Ejemplo codigo float

En el ejemplo anterior se puede ver el uso del código float, en el cual se puede apreciar el nombre que se le dio a la temperatura en grados Celsius con 2 decimales.

El segundo sector void setup utiliza los siguientes códigos:

- El primer código que se dará a conocer es pinMode, el cual determinará el modo de uso que se le dará al pin, para esto se debe utilizar el código pinMode seguido de un paréntesis que en su interior contiene primeramente el nombre o el pin que se quiere utilizar seguido de una coma (,) y el método que se utilizará, ya sea entrada o salida (INPUT O OUTPUT) seguido del cierre de paréntesis para finalizar con un punto y coma (;) para hacerle saber al programa que se finalizó el código pinMode. En la siguiente figura se muestra el uso del comando pinMode en un led.

```
pinMode(LED, OUTPUT);
```

Fuente: propia

Figura 0-6 Ejemplo código pinMode

- El segundo código es Serial.begin lo que significa inicio de comunicación serial, se le llama así porque es una comunicación entre 2 componentes controladores. Por ejemplo, la comunicación que tiene Arduino con un computador. Este código se puede utilizar para 2 cosas, la primera es para iniciar la comunicación entre Arduino y una pantalla LCD y la segunda es para comunicar las mediciones que está realizando Arduino mediante un sensor con la computadora y verlas en la pantalla de este. Se debe determinar la velocidad la cual en este caso debe ser 9600 [bps], como se muestra en la figura 4-7.

```
Serial.begin(9600);
```

Fuente: propia

Figura 0-7 Ejemplo código Serial.begin

El tercer sector es void loop en el cual está la mayor cantidad de códigos y se iniciara con:

- digitalRead, el cual como lo indica su nombre se utiliza para hacer lecturas digitales, luego de introducir este código se debe escribir dentro de un paréntesis que se quiere leer, como en la siguiente figura.

```
ESTADO = digitalRead(SENSOR);
```

Fuente: propia

Figura 0-8 Ejemplo codigo digitalRead

Como se puede apreciar ESTADO tomará el valor de la lectura digital del sensor.

- digitalWrite, este código es utilizado en conjunto con digitalRead, porque este último lee las señales y digitalWrite las ejecuta en un actuador.

```
digitalWrite(LED, HIGH);
```

Fuente propia

Figura 0-9 Ejemplo codigo digitalWrite

Como se puede observar en el ejemplo se está utilizando un led que está encendido (HIGH), gracias a que digitalRead interpretó los datos del sensor y digitalWrite procedió a encender el led.

- delay: este código es utilizado para retrasar alguna lectura o ejecución de una acción. Este trabaja en milisegundos.

```
delay(500);
```

Fuente propia

Figura 0-10 Ejemplo código delay

En el ejemplo anterior se puede ver un delay de 500 milisegundos.

- if: este comando es utilizado si y solo si una acción ocurre, si esto no llega a ocurrir simplemente no pasará nada.

```
if (DISTANCIA <= 100 && DISTANCIA >= 0) {  
    digitalWrite(BUZZER, HIGH);  
    delay(DISTANCIA * 10);  
    digitalWrite(BUZZER, LOW);  
}
```

Fuente propia

Figura 0-11 Ejemplo codigo if

Como se puede observar en el ejemplo anterior, el if solo actuará cuando la distancia sea menor o igual a 100 y mayor o igual a 0, de esta manera se activará el buzzer de manera proporcional a la distancia gracias al delay existente en el código, es decir mientras más cerca está el objeto del sensor más rápida será la frecuencia con la que suena el buzzer.

- else: este comando siempre que se utiliza debe estar escrito antes el comando “if”, ya que else quiere decir “de otra manera”, entonces su relación con el comando if es que si sucede X el comando if funcionara y si no sucede X se utiliza else para realizar otra acción distinta.

```
if (digitalValue_LLUVIA == LOW)
    digitalWrite(LED_LLUVIA, HIGH);
else
    digitalWrite(LED_LLUVIA, LOW);
```

Fuente propia

Figura 0-12 Ejemplo código else

Como se ve en el ejemplo, si el valor digital de lluvia es 0 el led se enciende (LED_LLUVIA, HIGH) y sino esta con valor 0 el led se mantiene apagado (LED_LLUVIA, LOW)

- pulseIn: este comando es utilizado como un cronometro debido a que inicia su funcionamiento cuando el pin al que está ligado tiene un valor HIGH y termina su funcionamiento cuando este valor cambia a LOW. Al finalizar su trabajo este toma un valor el cual es igual al tiempo que tarde el pin en cambiar de HIGH a LOW.

```
DURACION = pulseIn(ECO, HIGH);
```

Fuente propia

Figura 0-13 Ejemplo código pulseIn

En el ejemplo se puede ver que DURACION tomara el valor de pulseIn el cual está ligado al pin nombrado como ECO, HIGH quiere decir que inicia el funcionamiento en este valor.

- `analogRead`: como lo indica su nombre, `analogRead` significa “lectura analógica”. Esta es utilizada para realizar una lectura de las señales enviadas por los sensores del tipo analógico, tales como el sensor de temperatura y el sensor de luz por fotoresistencia utilizado en la maqueta.

```
SENSOR_LDR = analogRead(LDR)
```

Fuente propia

Figura 0-14 Ejemplo código `analogRead`

En la imagen anterior se puede apreciar que la placa Arduino, mediante el código `analogRead`, realizará una lectura analógica proveniente del sensor LDR (sensor de luz).

4.4 LIBRERIAS

Las librerías que se utilizan en la programación de la placa Arduino son para ayudar al programador al realizar un proyecto, con la finalidad de facilitar el entendimiento y programación al acotar el código agilizando su lectura.

Cuando se instala el programa Arduino, este incluye algunas librerías por defecto que fueron previamente seleccionadas por el equipo de desarrollo de Arduino. Estas librerías están listas para ser cargadas al código manualmente y debido a que la placa Arduino cuenta con una memoria de almacenamiento limitada para los códigos, se seleccionan solo las librerías necesarias para llevar a cabo el proyecto. Para incluir una librería simplemente se debe abrir el programa de Arduino e ir a la opción “programa”, luego al subíndice “incluir librería” y se selecciona la que se necesite, al añadirse esta se verá como en la siguiente imagen.

```
#include <LiquidCrystal_I2C.h>
```

Fuente propia

Figura 0-15 Ejemplo de librería incluida

En el supuesto caso de que la librería no esté incluida en el programa, se descarga desde GitHub como archivo .ZIP y para añadirla al programa de Arduino se debe ir a la opción “programa”, luego al subíndice “incluir librería” y posteriormente en “añadir biblioteca .ZIP” se selecciona el archivo descargado. Una vez realizado estos pasos, la librería estará disponible en el subíndice “incluir librería”.

4.5 CODIGOS ESPECIFICOS UTILIZADOS EN LA MAQUETA

A continuación, se detallarán los códigos de manera más específica para lograr de la mejor manera su comprensión, explicando línea a línea la acción a realizar por el componente determinado.

4.5.1 SISTEMA DE MONITOREO DE TEMPERATURA

En el siguiente código correspondiente al sistema de monitoreo de temperatura, está incluida la programación para el funcionamiento de la pantalla LCD y el sensor de temperatura.

- LCD

Sector 1

```
1    #include <LiquidCrystal_I2C.h>
2    LiquidCrystal_I2C lcd(0X3F,16,2);
```

1 Se incluye la librería del LCD de cristal líquido junto al I2C necesaria para la programación.

2 En esta línea se establece la comunicación entre el I2C y el Arduino, la cual puede cambiar dependiendo del LCD adquirido. Dentro del paréntesis se encuentra la dirección para establecer la comunicación y la resolución (16x2 en este caso). Para encontrar la dirección de comunicación que utiliza el LCD, se usa un código en el programa para adquirir esta dirección. Primero se abre el archivo “I2C_scan” con el programa “Arduino”. Después se sube la programación con el botón “Subir” y se abre la opción Monitor Serie, en este apartado el programa escanea y muestra la dirección de comunicación que utiliza el LCD. A continuación, se muestra el resultado del escaneo de la pantalla utilizada en la maqueta.

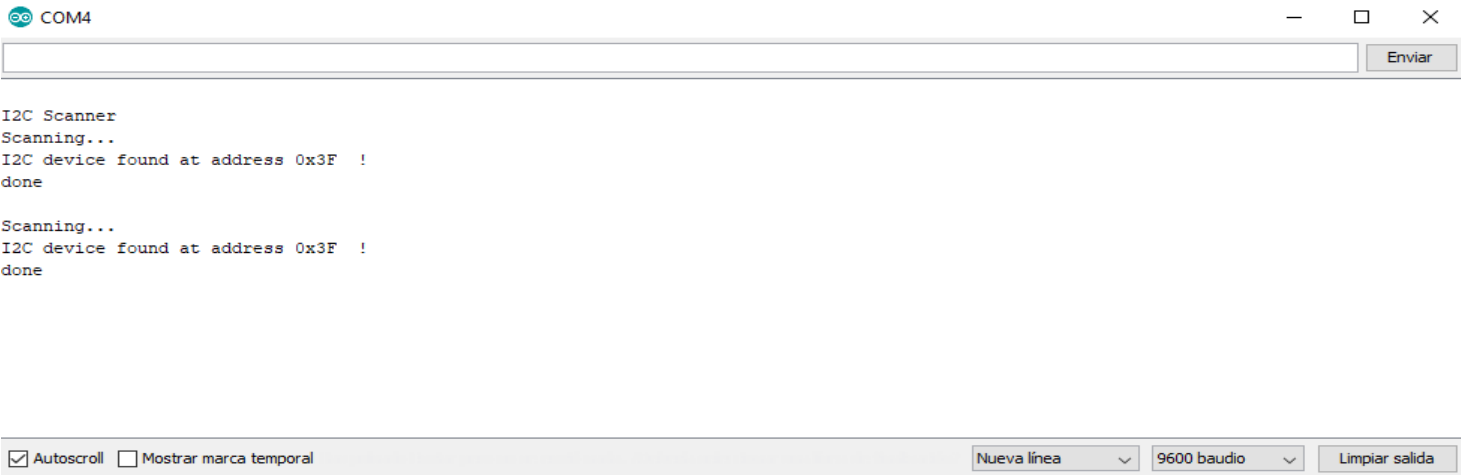


Figura 0-16 Escaneo de la dirección de comunicación I2C

Sector 2

void setup()

```
1    lcd.init();
2    lcd.clear();
3    lcd.backlight();
4    lcd.setCursor(0,0);
5    lcd.print("MEC.AUTOMOTRIZ ");
6    lcd.setCursor(0,1);
7    lcd.print("UTFSM ");
8    delay(3000);
9    lcd.clear();
```

- 1 Se pone en funcionamiento el LCD.
- 2 En esta línea se limpia la pantalla de cualquier carácter encontrado.
- 3 Se enciende la luz trasera del LCD.
- 4 Se selecciona la columna y fila, respectivamente en donde se digitan los caracteres.
- 5 Se escribe entre comillas lo que se desea mostrar en el LCD.
- 6 En este punto, se seleccionó la fila 1 (la segunda fila).
- 7 Se mostrarán los caracteres seleccionados entre las comillas.
- 8 Se da un tiempo de espera de 3000 milisegundos.
- 9 Se resetea la pantalla de los caracteres que contenga.

Sector 3

void loop()

```
1    lcd.setCursor(0,0);
2    lcd.print("Temp C ");
3    lcd.print(TempActualCB);
4    lcd.setCursor(0,1);
5    lcd.print("Temp F ");
6    lcd.print(TempActualFB);
```

7 delay(1000);

1 Se selecciona la columna y fila, respectivamente en donde se digitan los caracteres.

2 Se digitan los caracteres seleccionados entre las comillas.

3 Se digitan los valores medidos por el sensor en grados Celsius.

4 Se selecciona la columna y fila, respectivamente en donde se digitan los caracteres.

5 Se digitan los caracteres seleccionados entre las comillas.

6 Se digitan los valores medidos por el sensor en grados Fahrenheit.

7 Se da un tiempo de espera de 1000 milisegundos.

- Sensor de temperatura

1 #include <Wire.h>

2 #include <OneWire.h>

3 #include <DallasTemperature.h>

4 int pinDatosDQ = 7;

5 float TempAnterior=0.00

6 float TempActualCB=0.00;

7 float TempActualFB=0.00;

8 OneWire oneWireObjeto(pinDatosDQ);

9 DallasTemperature sensorDS18B20(&oneWireObjeto);

1 En las líneas 2 y 3 se incluyen las librerías necesarias para la operación del sensor.

4 Se indica que el pinDatosDQ está ubicado en el pin número 7.

5 Se almacena la última temperatura medida.

6 Mide la temperatura actual en grados Celcius.

7 Mide la temperatura actual en grados Fahrenheit.

8 Mediante la librería OneWire se obtiene oneWireObjeto, al cual se le llamará pinDatosDQ.

9 Mediante la librería DallasTemperature se obtiene sensorDS18B20, el cual tomará los valores de oneWireObjeto.

No es necesaria la programación en el sector 2

Sector 3:

void loop()

- 1 sensorDS18B20.requestTemperatures();
 - 2 TempActualCB = sensorDS18B20.getTempCByIndex(0);
 - 3 TempActualFB = sensorDS18B20.getTempFByIndex(0);
- 1 Solicita la temperature al sensor.
 - 2 Se obtiene la temperatura en grados Celcius del sensor.
 - 3 Se obtiene la temperatura en grados Fahrenheit.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <OneWire.h>
#include <DallasTemperature.h>
int pinDatosDQ = 7;
float TempActualCB=0.00;
float TempActualFB=0.00;
LiquidCrystal_I2C lcd(0X3F,16,2);
OneWire oneWireObjeto(pinDatosDQ);
DallasTemperature sensorDS18B20(&oneWireObjeto);
void setup()
{
  lcd.init();
  lcd.clear();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("MEC.AUTOMOTRIZ ");
  lcd.setCursor(0,1);
  lcd.print("UTFSM ");
  delay(3000);
  lcd.clear();
}
void loop()
{
  sensorDS18B20.requestTemperatures();
  TempActualCB = sensorDS18B20.getTempCByIndex(0);
  TempActualFB = sensorDS18B20.getTempFByIndex(0);
  lcd.setCursor(0,0);
  lcd.print("Temp C ");
  lcd.print(TempActualCB);
  lcd.setCursor(0,1);
  lcd.print("Temp F ");
  lcd.print(TempActualFB);
  //
  delay(1000);
}
```

Fuente propia

Figura 0-17 Código sistema de monitoreo de temperatura

4.5.2 SISTEMA DE ASISTENCIA AL ESTACIONAMIENTO

En el siguiente código correspondiente al sistema de asistencia al estacionamiento, está incluida la programación para el funcionamiento del buzzer y el sensor de ultrasonido.

- Buzzer

Sector 1

```
1    int BUZZER = 6;
```

1 Se da a conocer que el buzzer está conectado al pin 6

Sector 2

```
void setup()
```

```
1    pinMode(BUZZER, OUTPUT); // LED como salida
```

1 Se selecciona el modo de uso del pin en donde va ubicado el buzzer.

Sector 3

```
void loop()
```

```
1    if (DISTANCIA <= 30 && DISTANCIA >= 0)
```

```
2    digitalWrite(BUZZER, HIGH);
```

```
3    delay(DISTANCIA * 10);
```

```
4    digitalWrite(BUZZER, LOW);
```

1 Si la distancia esta entre 0 y 30 [cm].

2 Se enciende el buzzer.

3 Se da un tiempo de espera igual a la distancia multiplicado por 10.

4 Deja de sonar el buzzer.

- Sensor Ultrasonido.

Sector 1

```
1    int TRIG = 2;    // trigger en pin 10
```

```
2    int ECO = 3;     // echo en pin 9
```

```
3    int DURACION_PULSO;
```

- 4 int DISTANCIA;
- 1 Se indica que TRIG está en el pin 2, el cual se encarga de enviar el sonido ultrasónico.
 - 2 Se indica que ECO está en el pin 3, el cual recibe el rebote del sonido.
 - 3 Se indica la variable DURACION_PULSO.
 - 4 Se indica la variable DISTANCIA.

Sector 2:

void setup()

- 1 pinMode(TRIG, OUTPUT); // trigger como salida
 - 2 pinMode(ECO, INPUT); // echo como entrada
 - 3 Serial.begin(9600); // inicializacion de comunicacion serial a 9600 bps
-
- 1 Se indica que el pin TRIG será de salida.
 - 2 Se indica que el pin ECO será de entrada.
 - 3 Se inicia la comunicación a 9600 bps.

void loop()

- 1 digitalWrite(TRIG, HIGH);
 - 2 delay(1);
 - 3 digitalWrite(TRIG, LOW);
 - 4 DURACION_PULSO = pulseIn(ECO, HIGH
 - 5 DISTANCIA = DURACION_PULSO / 58.2
 - 6 delay(100);
-
- 1 Se inicia el envio del ultrasonido.
 - 2 Se deja pasar un milisegundo.
 - 3 Se termina el envio del ultrasonido.

- 4 DURACION_PULSO tomara el valor recibido por pulseIn que es entrada de pulso.
- 5 La distancia medida será DURACION_PULSO dividido por los microsegundos (calculado explicado en cap 2.2).
- 6 Se da un tiempo de 100 microsegundos para realizar otra medición.

```
int TRIG = 2;
int ECO = 3;
int BUZZER = 6;
int DURACION_PULSO;
int DISTANCIA;

void setup()
{
  pinMode(TRIG, OUTPUT);
  pinMode(ECO, INPUT);
  pinMode(BUZZER, OUTPUT);
  Serial.begin(9600);
}
void loop()
{
  digitalWrite(TRIG, HIGH);
  delay(1);
  digitalWrite(TRIG, LOW);
  DURACION_PULSO = pulseIn(ECO, HIGH);
  DISTANCIA = DURACION_PULSO / 58.2;
  delay(100);

  if (DISTANCIA <= 30 && DISTANCIA >= 0){
    digitalWrite(BUZZER, HIGH);
    delay(DISTANCIA * 10);
    digitalWrite(BUZZER, LOW);
  }
}
```

Fuente propia

Figura 0-18 Código del sistema de asistencia al estacionamiento

4.5.3 SISTEMA DE ALARMA

En el siguiente código correspondiente al sistema de asistencia al estacionamiento, está incluida la programación para el funcionamiento del buzzer, del led y el sensor de vibración.

- Buzzer

Sector 1

- 1 int BUZZER = 6;
- 1 Se determina que el Buzzer está en el pin 6

Sector 2

```
void setup(){  
1   pinMode(BUZZER, OUTPUT);
```

1 Se determina que el buzzer será una salida.

Sector 3

```
void loop(){  
1   if (ESTADO_VIBRACION == LOW){ // si se detecta vibracion  
2       digitalWrite(BUZZER, HIGH);  
3       delay(400); // demora de medio segundo para visualizar bien  
4       digitalWrite(BUZZER, LOW);  
5       delay(50);  
6       digitalWrite(BUZZER, HIGH);  
7       delay(200);  
8       digitalWrite(BUZZER, LOW);  
9       delay(50);  
10      digitalWrite(BUZZER, HIGH);  
11      delay(400);  
12      digitalWrite(BUZZER, LOW);  
13      delay(50);  
14      digitalWrite(BUZZER, HIGH);  
15      delay(400);  
16      digitalWrite(BUZZER, LOW);  
17      delay(50);  
18      digitalWrite(BUZZER, HIGH);  
19      delay(200);  
20      digitalWrite(BUZZER, LOW);
```

1 Si se detecta una vibración ocurrirá una acción.

2-20 la acción será una alerta sonora en forma de una breve melodía la cual consta de activar y desactivar el buzzer variando sus tiempos de funcionamiento y de esta forma emular el sonido de una alarma automotriz.

- LED

Sector 1

```
1 int LED_VIBRACION = 13; // anodo de LED a pin 3
```

1 El led está ubicado en el pin 13.

Sector 2

```
void setup(){
```

```
1 pinMode(LED_VIBRACION, OUTPUT); // pin 3 como salida
```

```
}
```

1 Se determina que el led será una salida.

Sector 3

```
void loop(){
```

```
1 if (ESTADO_VIBRACION == LOW){ // si se detecta vibracion
```

```
2 digitalWrite(LED_VIBRACION, HIGH); // enciende LED
```

```
1 Si se detecta una vibración
```

```
2 Se enciende el led.
```

- Sensor de vibración

Sector 1

```
1 int SENSOR_VIBRACION = 5;
```

1 Se determina que el sensor de vibración estará en el pin 5.

Sector 2

```
void setup(){
```

```
1 pinMode(SENSOR_VIBRACION, INPUT);
```

1 Se determina que el sensor de vibración es una salida.

Sector 3

```
void loop()
```

```
1 ESTADO_VIBRACION = digitalRead(SENSOR_VIBRACION);
```

```
2 if (ESTADO_VIBRACION == LOW)
```

- 1 Se ve el estado de vibración (si el sensor detecta vibración o no).
- 2 Si el estado es vibración, entonces será low y se ejecutará una acción.

```
int BUZZER = 6;
int SENSOR_VIBRACION = 5;
int LED_VIBRACION = 13;
int ESTADO_VIBRACION;
void setup() {
  pinMode(SENSOR_VIBRACION, INPUT);
  pinMode(LED_VIBRACION, OUTPUT);
  pinMode(BUZZER, OUTPUT);
}
void loop() {
  ESTADO_VIBRACION = digitalRead(SENSOR_VIBRACION);
  if (ESTADO_VIBRACION == LOW) {
    digitalWrite(LED_VIBRACION, HIGH);
    digitalWrite(BUZZER, HIGH);
    delay(400);
    digitalWrite(LED_VIBRACION, LOW);
    digitalWrite(BUZZER, LOW);
    delay(50);
    digitalWrite(BUZZER, HIGH);
    delay(200);
    digitalWrite(BUZZER, LOW);
    delay(50);
    digitalWrite(BUZZER, HIGH);
    delay(400);
    digitalWrite(BUZZER, LOW);
    delay(50);
    digitalWrite(BUZZER, HIGH);
    delay(400);
    digitalWrite(BUZZER, LOW);
    delay(50);
    digitalWrite(BUZZER, HIGH);
    delay(200);
    digitalWrite(BUZZER, LOW);
  }
}
```

Fuente propia

Figura 0-19 Código sistema de alarma

4.5.4 SISTEMA DE ILUMINACION AUTOMATICA

En el siguiente código correspondiente al sistema de asistencia al estacionamiento, está incluida la programación para el funcionamiento del led y el sensor de luz.

- LED

Sector 1

```
1 int LED_SENSOR_LUZ=11;
```

1 Se determina que el led está en el pin 11.

Sector 2

```
void setup()
```

```
1 pinMode(LED_SENSOR_LUZ,OUTPUT);
```

1 Se determina que LED_SENSOR_LUZ es salida.

Sector 3

```
void loop () //Configuración de los valores del LDR
```

```
{
```

```
1 if (luz <= valor_limite)
```

```
2 digitalWrite (LED_SENSOR_LUZ, LOW)
```

```
3 if (luz > valor_limite)
```

```
3 digitalWrite (LED_SENSOR_LUZ, HIGH);
```

1 Si el valor de luz es menor al valor limite.

2 El led se apagará

3 Si el valor de luz es mayor al valor limite

4 El led se prenderá

- Sensor de luz

Sector 1

```
1 int LDR=A1;
```

```
2 int luz = 0;
```

```
3 int valor_sensor = 0;
```

- 4 int valor_limite = 200;
- 1 Señal del sensor va al pin analógico A1.
- 2 Se determina la variable luz con un valor de 0.
- 3 Valor mínimo registrado por el sensor.
- 4 Valor limite propuesto por el programador.

Sector 2

```
void setup() {
```

- 1 Serial.begin(9600);
- 2 pinMode(LDR,INPUT);
- }

- 1 Se inicia la comunicación serial a 9600 bps.
- 4 Se determina que LDR es una entrada.

Sector 3

```
void loop () //Configuracion de los valores del LDR
```

- ```
{
```
- 1     valor\_sensor = analogRead(LDR);
  - 2     luz = (5.0 \* valor\_sensor \* 100.0)/1024.0;
  - 3     delay(300);

- 1 Se determina que el valor del sensor es igual a la lectura analógica de LDR
- 2 Mediante la fórmula escrita en esta línea y usando la variable valor sensor, se puede determinar el valor de luz
- 3 Se da tiempo de 300 milisegundos.

```

int LED_SENSOR_LUZ=11;
int LDR=A1;
int luz = 0;
int valor_sensor = 0;
int valor_limite = 200;

void setup() {
 Serial.begin(9600);
 pinMode(LED_SENSOR_LUZ,OUTPUT);
 pinMode(LDR,INPUT);
}

void loop ()
{
 valor_sensor = analogRead(LDR);
 luz = (5.0 * valor_sensor * 100.0)/1024.0;
 delay(300);

 if (luz <= valor_limite)
 {
 digitalWrite (LED_SENSOR_LUZ, LOW);
 }
 if (luz > valor_limite)
 {
 digitalWrite (LED_SENSOR_LUZ, HIGH);
 }}

```

Fuente propia

Figura 0-20 Código sistema de iluminación automática

#### 4.5.5 SISTEMA LIMPIA PARABRISAS AUTOMATICO

En el siguiente código correspondiente al sistema de asistencia al estacionamiento, está incluida la programación para el funcionamiento del led y el sensor de lluvia.

- LED

Sector 1

1      int LED\_LLUVIA = 12;

1 Se determina que el led esta en el pin 12

Sector 2

void setup()

1      pinMode(LED\_LLUVIA, OUTPUT); }

1 Se determina que el led será una salida

### Sector 3

```
void loop(){
```

```
1 if (digitalValue_LLUVIA == LOW)
2 digitalWrite(LED_LLUVIA,HIGH);
3 else
4 digitalWrite(LED_LLUVIA, LOW);
5 delay(500);}
```

1 Si el valor digital de lluvia es bajo

2 Se enciende el led

3 Si no ocurre lo anterior

4 El led se mantendrá apagado

5 Se da una pausa de 500 milisegundos para realizar una lectura nueva

- Sensor de lluvia

### Sector 1

```
1 bool digitalValue_LLUVIA;
```

1 Se determina el valor digital para saber si el sensor esta mojado o no

### Sector 2.

```
void setup()
```

```
1 Serial.begin(9600);
```

```
2 pinMode(digitalValue_LLUVIA, INPUT); // pin 2 como entrada}
```

1 Se inicia la comunicación serial a 9600 bps.

2 Se determina el modo de uso de digitalValue\_lluvia como entrada.

### Sector 3

```
void loop()
```

```
1 digitalValue_LLUVIA = digitalRead(10);
```

1 El valor digital es igual a la lectura digital del pin 10.

```
bool digitalValue_LLUVIA;
int LED_LLUVIA = 12;
void setup()
{
 Serial.begin(9600);
 pinMode(digitalValue_LLUVIA, INPUT);
 pinMode(LED_LLUVIA, OUTPUT);
}

void loop() {

 digitalValue_LLUVIA = digitalRead(10);
 if (digitalValue_LLUVIA == LOW)
 digitalWrite(LED_LLUVIA, HIGH);
 else
 digitalWrite(LED_LLUVIA, LOW);

 //retardo
 delay(500);
}
```

Fuente propia

Figura 0-21 Código sistema limpia parabrisas automático

## **CONCLUSIONES Y RECOMENDACIONES**

Uno de los objetivos de este trabajo fue dar a conocer la placa ARDUINO junto a los componentes que forman parte de la maqueta. Para ello se realizó una maqueta didáctica, con la cual el alumno podrá realizar diversas interacciones para entender el principio de funcionamiento de cada componente y la forma en que interactúan entre sí. Para poder llevar a cabo el proyecto elegido, se tuvo que acudir a la ayuda del profesor de la carrera Telecomunicaciones y Redes Sr. Piero Montenegro, ya que el conocimiento que se requería para la realización de la maqueta no está presente en la carrera TECNICO UNIVERSITARIO EN MECANICA AUTOMOTRIZ. Dicha ayuda consistió en diversas consultas que se le realizaron en su oficina, estas consistieron principalmente sobre la programación necesaria para que funcione correctamente cada elemento y el principio de funcionamiento de algunos componentes que contaban con varios tecnicismos complejos de entender en el acotado tiempo que se disponía para la realización del trabajo de título.

Se llevaron a cabo todos los objetivos que se propusieron al inicio de este escrito, con lo cual se creó un material complementario para la asignatura Control Electrónico Automotriz, lo cual consta de una maqueta didáctica de la placa ARDUINO y del presente informe que detalla cada elemento que lo conforma, tanto su funcionamiento como su conexión y la programación que necesita. Los alumnos contarán con un total de cinco experiencias disponibles para interactuar con el sistema, las que deberán programar previamente con los códigos que estarán disponibles en el pendrive que se entregará junto con la maqueta. Esto complementará de forma importante la carrera, iniciando a los estudiantes en el área de la programación, con lo cual podrán realizar diversos sistemas adicionales que podrán incorporar en reparaciones automotrices o en proyectos personales a un bajo costo.

## **BIBLIOGRAFIA**

[https://en.wikipedia.org/wiki/Arduino\\_Uno#Technical\\_specifications](https://en.wikipedia.org/wiki/Arduino_Uno#Technical_specifications)

<http://arduino.cl/que-es-arduino/>

<https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno>

<https://www.geekfactory.mx/tutoriales/tutoriales-arduino/sensor-ultrasonico-hc-sr04-y-arduino/>

<https://www.puomotores.com/13121256/como-funciona-un-adaptador-ca>

<https://www.electronicoscaldas.com/displays-lcd-alfanumericos-y-graficos/131-display-lcd-16x2-el-1602a.html>

<https://www.slideshare.net/Globant/arduino-workshop-arduino-fundamentals-globant-crdoaba>

<https://soloarduino.blogspot.com/2013/07/cablecitos.html>

<https://www.luisllamas.es/medir-distancia-con-arduino-y-sensor-de-ultrasonidos-hc-sr04/>

<https://aprendiendoarduino.wordpress.com/category/microcontrolador/>

<http://panamahitek.com/entradas-y-salidas-analogicas-y-digitales/>

[https://depaquete.cl/index.php?route=product/product&product\\_id=207](https://depaquete.cl/index.php?route=product/product&product_id=207)

[https://depaquete.cl/index.php?route=product/product&product\\_id=231](https://depaquete.cl/index.php?route=product/product&product_id=231)

<https://saber.patagoniatec.com/2014/06/ds18b20-sensor-temperatura-ds18b20-sumergible-agua-liquido-arduino-argentina-ptec/>

<https://www.luisllamas.es/arduino-lluvia/>

<https://tectronix.cl/producto/sensordelluviaarduino/>

<https://solectroshop.com/product-spa-1506-Sensor-Digital-de-Intensidad-de-Luz-con-Foto-Resistencia-LDR-con-Controlador-LM393.html>

<https://electronicastore.net/producto/modulo-ky-031-sensor-de-impacto-arduino-o-micro/>

<https://www.lozurytech.biz/product-page/sensor-magn%C3%A9tico-lineal-hall-ky-024>

<https://tuelectronica.es/que-es-la-protoboard/>

[https://www.ecured.cu/Diodo\\_led](https://www.ecured.cu/Diodo_led)

<https://arduino-modules.info/ky-012-active-buzzer-module/>

[https://www.ecured.cu/Sensor\\_hall](https://www.ecured.cu/Sensor_hall)

<https://www.luisllamas.es/medir-vibracion-con-arduino-y-sensor-sw-18020p/>

<https://www.areatecnologia.com/electronica/como-es-un-led.html>

<https://educarparaelcambio.com/arduino/reto-9-el-sensor-de-luz-ldr-otro-input-analogico/>

<https://como-funciona.co/una-fotorresistencia/>

<http://rufianenlared.com/sensor-agua/>

<https://www.infootec.net/sensor-campos-magneticos-sensor-hall/>

<https://aprendiendoarduino.wordpress.com/2017/07/09/i2c/>

<https://www.comunidadelectronicos.com/articulos/i2c.htm>

<https://aprendiendoarduino.wordpress.com/2017/01/23/programacion-arduino-5/>

<https://www.prometec.net/intro-programacion/>

<https://www.guioteca.com/mecanica-automotriz/la-temperatura-del-motor-como-funciona-el-sistema/>