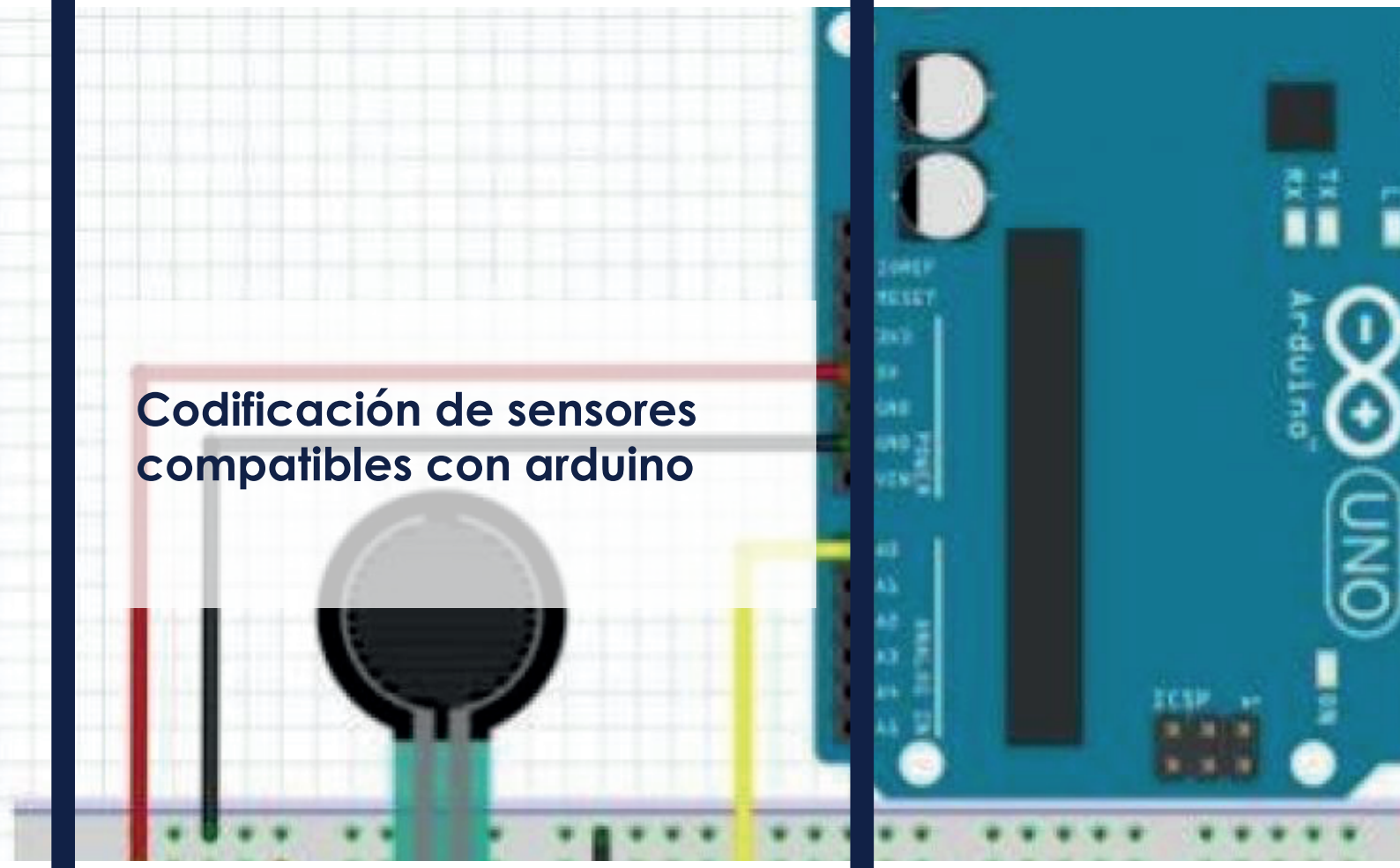
compAs

Grupo de capacitación e investigación pedagogica

Codificación de sensores compatibles con arduino



Ingrid Angélica García Torres
Rodolfo Antonio Parra López
Rosa Elizabeth Castillo León
Dennis Holger Zambrano Silva
Luis Javier Domínguez De la Torre
William Ricardo Navas Espín

Codificación de sensores compatibles con arduino

Ingrid Angélica García Torres
Rodolfo Antonio Parra López
Rosa Elizabeth Castillo León
Dennis Holger Zambrano Silva
Luis Javier Domínguez De la Torre
William Ricardo Navas Espín

Codificación de sensores compatibles con arduino

compAS
Grupo de capacitación e investigación pedagógica

Título original:
Codificación de sensores
compatibles con arduino

Primera edición: mayo 2020

Publicado por acuerdo con los autores.
© 2020, Editorial Grupo Compás
Guayaquil-Ecuador

Editado en Guayaquil - Ecuador

ISBN: 978-9942-33-296-7

compÁs
Grupo de capacitación e investigación pedagógica

Cita.
García. I, Parra. R, Castillo. R, Zambrano. D, Dominguez. L, Navas. W. (2020) Codificación de sensores compatibles con arduino, Editorial Grupo Compás, Guayaquil Ecuador, 95 pag

Índice

PLACA ARDUINO

1. Placa controladora Arduino

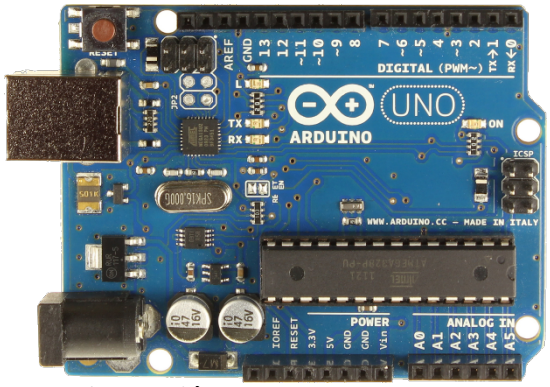


Figura 1 Placa Arduino UNO. Información tomada de https:/ /commons.wikimedia.org/wiki/File:ArduinoUNO.png

1.1. Características y descripción

El Arduino Uno R3 es una placa electrónica basada en el microprocesador ATmega328. Cuenta con 14 pines digitales de entrada / salida (de los cuales 6 pueden ser utilizados como salidas PWM), 6 entradas analógicas, un oscilador de 16MHz, una conexión USB, un conector de alimentación, un header ICSP, y un botón de reinicio. Contiene todo lo necesario para que funcione el microcontrolador; basta con conectarla a un ordenador con un cable USB, o alimentarla con un adaptador o la batería para empezar. El Uno se diferencia de todas las placas anteriores en que no utiliza el chip controlador de FTDI de USB a Serie. Las características adicionales que vienen con la versión R3 son:

ATmega16U2 en lugar del 8U2 como convertidor de USB a Serie. -1.0 Pinout: Se han añadido pines SDA y SCL para la comunicación TWI colocados cerca al pin AREF y otros dos nuevos pins colocados cerca del pin RESET, el IOREF que permiten a los shields adaptarse a la tensión proporcionada por la placa principal y el segundo es un pin que no está conectado y se reserva para usos futuros. -Circuito RESET más potente.

Microcontroller ATmega328 Operating Voltage 5V Input Voltage (recommended) 7-12V Input Voltage (limit) 6-20V Digital I/O Pins 14 PWM Digital I/O Pins 6 Analog Input Pins 6 DC Current per I/O Pin 40 mA DC Current for 3.3V Pin 50 mA Flash Memory 32 KB Flash Memory for Bootloader 0.5 KB SRAM 2 KB EEPROM 1 KB Clock Speed 16 MHz Lenght 68.6 mm Width 53.4 mm. Ilustración en el link http://arduino.cc/en/Main/arduinoBoardUno.

1.2. Ambiente de desarrollo

Figura 1. Inicialización del Ambiente de desarrollo. Información tomada por la investigación directa. Elaborado por autores.

La aplicación de Arduino o sea el ambiente de desarrollo nos mostrará las diferentes opciones que tenemos en la actualidad para desarrollar nuestros proyectos.

Como cualquier desarrollo de software, primero hay que elegir el lenguaje. En este caso para Arduino nos concentraremos en el lenguaje c/c++, dado que el propio programa nos facilita un Entorno de Desarrollo Integrado (su siglas en ingles son IDE) básico y muy práctico, el cual se puede descargar para los sistemas operativos principales existentes hoy en día (Windows, Mac OS y linux).

El Entorno de Arduino es el que se va a utilizar para la programación del funcionamiento de los diferentes sensores; nos proporciona herramientas

básicas que usaremos para subir, depurar y comunicarnos con nuestra placa.

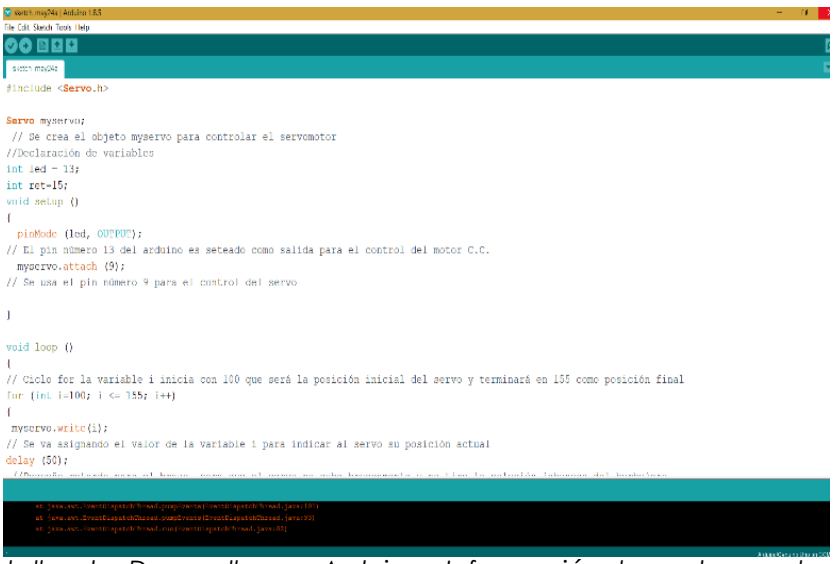


Figura 2:Pantalla de Desarrollo en Arduino. Información tomada por la Investigación directa. Elaborado por autores

SENSORES Y ACTUADORES

Cuando se construye un proyecto, en su gran mayoría requerirá de "medir" diferentes parámetros físicos de su entorno para poder tomar decisiones que se ejecuten, es para estas mediciones que se requieren del uso de elementos transductores que conviertan esas variables físicas en otra de tipo eléctrico que pueda ser interpretada por los procesadores que serán los que tomen las directrices de acuerdo con lo "medido" y en consecuencia realicen acciones por medio de actuadores.

Los transductores son equipos que pueden convertir señales físicas en señales eléctricas y los conocemos comercialmente como "sensores", estos elementos se clasifican de acuerdo con varias propiedades como: principio de funcionamiento, tipo de señal entregada, tipo de variable física medida, etc.

Los Actuadores son elementos o maquinarias que reciben señales eléctricas del procesador (Arduino) y ejecutan acciones diversas de forma que se manifieste los requerimientos hacia el ambiente exterior.

Tabla 1. Clasificación de sensores

Principio de funcionamiento	Tipo de señal generada	Tipo de variable física medida
Activos	Analógico	Acústicos
Pasivos	Digital	Eléctricos

	Temporal	Magnéticos Mecánicos Médicos Ópticos Químicos Radiación Térmicos

Información tomada de la investigación directa. Elaborada por autores

En el presente libro vamos a tratar en su gran mayoría acerca de una familia de sensores que son muy fáciles de conseguir en diferentes mercados y son conocidos como los sensores del KIT KEYES.

- KY001: Sensor de Temperatura DS18B20
- KY002: Sensor de Impactos (choques)
- KY003: Sensor de Campo Magnético por efecto Hall
- KY004: Sensor de Pulsación (Botón pulsador)
- KY005: Módulo Emisor de infrarrojo
- KY006: Módulo Parlante Pasivo (Diferentes Tonos)
- KY008: Módulo Diodo Laser
- KY009: Módulo LEDs RGB tipo SMD
- KY010: Sensor Foto Interruptor
- KY011: Módulo LEDs Bicolor
- KY012: Módulo Parlante Activo (Un solo Tono)
- KY013: Sensor de Temperatura Análogo
- KY015: Sensor de Temperatura y Humedad
- KY016: Módulo LEDs RGB tipo DIP
- KY017: Sensor de Inclinación por Interruptor de Mercurio
- KY018: Sensor Foto-Eléctrico
- KY019: Módulo Relay 5V DC
- KY020: Módulo de Inclinación por bola de acero
- KY021: Módulo de Interruptor de lámina Magnética Activo
- KY022: Módulo Receptor Infrarrojo
- KY023: Sensor JoyStick doble eje
- KY025: Módulo de Interruptor de lámina Magnética Graduable
- KY026: Sensor Detector de Fuego (Infrarrojo)
- KY027: Módulo LED con Interruptor de Mercurio
- KY028: Sensor Digital de Temperatura
- KY031: Sensor de Impactos por Sonido
- KY032: Sensor de Obstáculos (de proximidad)

- KY033: Sensor Seguidor de Línea
- KY034: Módulo LED de 7 Colores Parpadeantes
- KY035: Sensor Análogo Hall de Campo Magnético
- KY036: Sensor Interruptor Táctil
- KY037: Sensor de Sonido con micrófono de alta sensibilidad
- KY038: Sensor de Sonido con micrófono Electrek
- KY039: Sensor de Ritmo Cardiaco
- KY040: Sensor de Rotación codificada

Existen varios tipos de sensores que realizan la misma acción, pero basado en otro principio de medición, o con otra característica extra, por esta razón para un mejor análisis de estos haremos una separación entre sensores y actuadores clasificándolos según el tipo de variable física a medir.

2. Sensores

2.1. Sensores Acústicos

En este grupo de sensores se encuentran aquellos que tienen como finalidad detectar la presencia de ondas sonoras (comprendidas en el rango de audición humana - 20Hz a 20KHz) ya sea que se transmitan por el aire, agua u otro medio, así también se pueden encontrar los sensores que detecten ondas infrasónicas (menores a 20Hz) o Ultrasónicas (mayores a 20KHz) como aquellos usados en los sonares.

Sensor de Proximidad Ultrasónico

Éste detecta la distancia desde el sensor hasta los objetos en los cuales rebota la señal enviada por la parte emisora del sensor, trabaja libre de roces mecánicos y detectan objetos a distancias desde 1.7 cm hasta 425 cm. El sensor consta de dos componentes básicos uno emite un pulso (disparo) de 40 KHz (Ultrasónico) y el otro espera que este pulso rebote en el objeto deseado (eco) y mide el tiempo transcurrido entre estos dos estados. Dicho tiempo es medido en microsegundos y puede estar entre 100 us y 25000 us.

Con este tiempo es posible encontrar la distancia usando la siguiente formula:

Distancia = Tiempo * 0.0170

Aquí se asume que las condiciones en que se mide son normales para tener una velocidad del sonido de 340 m/s que transformadas es 0.034 cm/us y se divide para 2 ya que el tiempo es medido desde la emisión hasta el rebote lo que implica ida y vuelta, esto da el valor de 0.0170.

Figura 3: HC-SR04 - Sensor de Proximidad Ultrasónico. Información directa. Elaborado por autores.

Desarrollo

```
// Medir la distancia en centimetros desde la parte frontal del sensor
hasta el //objeto deseado, indicando esta medición en el monitor
serial configurado a //9600 baudios
double distancia;
long tiempo;
int Pulso=9;
int Eco=8;
void setup(){
Serial.begin(9600);
pinMode(Pulso, OUTPUT); /*activa la salida para el pulso ultrasónico*/
pinMode(Eco, INPUT); /*activa la entrada que medirá el tiempo del
rebote del pulso*/
}
void loop(){
digitalWrite(Pulso,LOW);
delay (30); /* Para asegurar que el sensor empiece sin pulsos*/
digitalWrite(Pulso, HIGH); /* envío del pulso ultrasónico*/
delayMicroseconds(10); /* espera el tiempo que el pulso dura */
tiempo=pulseIn(Eco, HIGH); /* lee el tiempo transcurrido en recibir el
rebote*/
distancia= 0.0170*tiempo; /* calcula la distancia */
/*Muestra la distancia medida en centímetros por el monitor serial*/
Serial.println(distancia ,1);
Serial.println(" cm");
}
```

Sensor de sonido con micrófono Electrek

En este grupo tenemos el Módulo KY037 y el KY038, ambos detectan el nivel de "Sonido", la diferencia es que en el modelo KY037 tiene un micrófono de mayor tamaño lo que en teoría le da mayor sensibilidad mientras que el KY038 usa un micrófono pequeño que es menos sensible, pero en la práctica no he apreciado la diferencia.

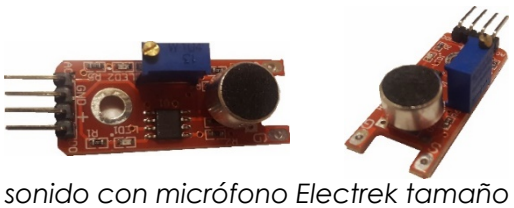


Figura 4: KY037 - Sensor de sonido con micrófono Electrek tamaño grande. Información directa. Elaborado por autores.

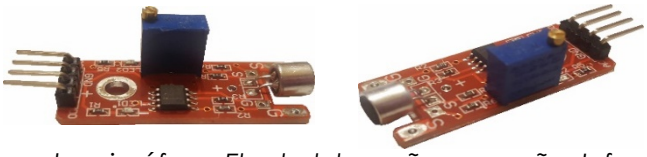


Figura 5: KY038 - Sensor de micrófono Electrek tamaño pequeño. Información directa. Elaborado por autores.

Estos sensores pueden captar las ondas sonoras por el micrófono, convertidas en señal eléctrica y dirigidas a un codificador que compara el nivel de la señal eléctrica (sonido convertido) contra el valor definido por el PreSet (potenciómetro multivuelta) y si se alcanza el valor seleccionado se codifica en el terminal de salida digital como un valor verdadero, caso contrario la salida digital se mantendrá en "FALSE".

Estos sensores tienen dos salidas, una analógica y otra digital, la analógica sirve para leer el valor de la posición del PreSet en una escala de 0 a 1023, si por ejemplo Ud. configura el PreSet a mitad de camino la salida en la terminal analógica leerá el valor 512 aproximadamente.

Mediante este sensor, si en un momento determinado se escucha un sonido que supere el nivel configurado, este lo detecta enviando a la salida digital el valor "TRUE", además gracias al valor leído del PreSet se puede tener una idea de que cantidad de sonido fue producido.

Este tipo de sensores sirve para detención de acciones que produzcan algún tipo de sonido como dar una palmada fuerte con lo que puede accionar un interruptor y encender las luces del cuarto, o en sitios donde se active alarmas en caso de algún tipo de ruido elevado, para buses urbanos o resguardo de bienes privados o públicos, etc.

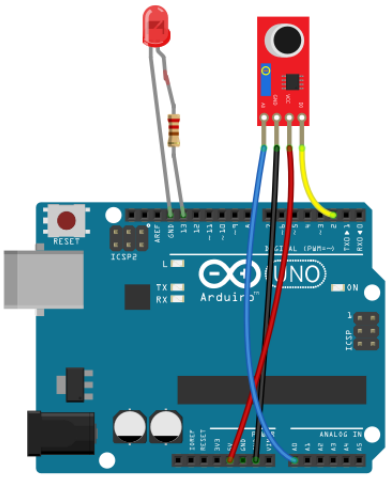


Figura 7. Esquema de conexión de Sensor de micrófono. Información tomada de la Investigación directa. Elaborada por los autores

Desarrollo

Ejemplo 01

```
//Sensor de sonido, cuando detecte un sonido encenderá un LED
//durante 1 segundo y luego apagara el LED hasta que detecte
//Nuevamente otro sonido
int LED = 13;
int SenIN_D0 = 2; // define el pin digital 2, para la señal de D0
bool SensorVal= false;
void setup(){
  pinMode( LED, OUTPUT); // inicializa el pin del LED como de SALIDA
  pinMode( SenIN_D0 , INPUT_PULLUP) ; // inicializa el pin SenIN_D0 como
entrada en modo PULLUP
  digitalWrite(LED , false) ;
}
void loop(){
  SensorVal =  digitalRead(SenIN_D0) ; // se lee el sensor
  if ( SensorVal) { // Si se obtuvo una lectura alta (1) se procede a hacer
los cambios
    digitalWrite(LED, true) ; // enciende el led
    delay (1000); // mantiene el led encendido 1 segundo
    digitalWrite(LED, false) ; // apaga el led
  }
}
```

Ejemplo 2

```
// Intercambiar el estado de un LED entre encendido y apagado según
// se vaya detectando sonidos fuertes como una palmada, el estado
// inicial del LED es apagado
```

```
int LED = 13;
int SenIN_D0 = 2; // define el pin digital 2, para la señal de D0
bool SensorVal= false, EstadoLED=false;
void setup(){
  pinMode( LED, OUTPUT); // inicializa el pin del LED como de SALIDA
  pinMode( SenIN_D0 , INPUT_PULLUP) ; // inicializa el pin SenIN_D0 como
entrada en modo PULLUP
  digitalWrite(LED , EstadoLED) ;
}
void loop(){
  SensorVal =  digitalRead(SenIN_D0) ; // se lee el sensor
  if ( SensorVal) { // Si se obtuvo una lectura alta (1) se procede a hacer
los cambios
    if (EstadoLED) { EstadoLED = false;} // Si el estado del LED era
encendido entonces se apaga
    else  { EstadoLED = true;} // Si el estado del LED era apagado
entonces se enciende
    digitalWrite(LED, EstadoLED); // se envía la orden de acción al pin del
LED
  }
}
```

2.2. Sensores Eléctricos

Este tipo de sensores miden variables físicas de tipo eléctrico que pueden ser niveles de tensión (Voltaje), de corriente (Amperaje), de detección de paso por cero, de potencia activa, o de cambios eléctricos como la capacitancia, también puede ser sensores que realicen la función de interruptor eléctrico, en la serie Keyes encontramos sensores al tacto por efecto capacitivo como es el modelo KY036 del cual trataremos más adelante.

Sensor de interrupción (Pulsador)

Consta de un interruptor de tipo pulsador normalmente abierto (N.O.) que al ser presionado cierra el circuito activando una señal de tipo digital de valor alto (5 V) en la salida, la corriente máxima que puede manejar es de 50 mA, y está diseñado para resistir aproximadamente 100 000 ciclos de interrupción antes que su mecanismo empiece a dar fallas bajo situaciones normales de uso.

Precisamente por esta ultima razón es que en la actualidad se a reemplazado en muchos proyectos por los interruptores de efecto capacitivo por tener este ultimo un mayor tiempo de la duración de este tipo de sensor está limitada según cada fabricante a un determinado número de pulsaciones ya que al ser de tipo mecánico el interruptor sufre desgaste físico y se va deteriorando de a poco.

Figura 8: KY004 - Sensor de interrupción (Pulsador). Información tomada de solo Arduino.

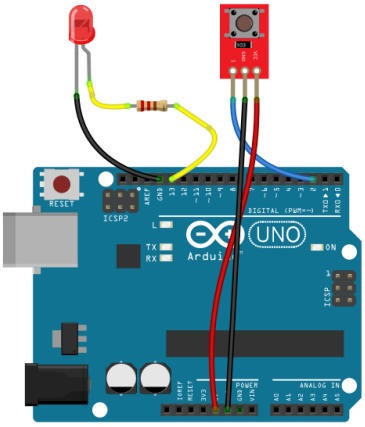


Figura 9. Esquema de conexión del Sensor de interrupción (Pulsador). Información tomada de la Investigación directa. Elaborada por los autores

Desarrollo

```
// Intercambiar el estado de un LED entre encendido y apagado según
// se vaya detectando el tacto del interruptor, el estado
// inicial del LED es apagado
int LED = 13;
int SenIN_D0 = 2; // define el pin digital 2, para la señal de D0
bool SensorVal= false, EstadoLED=false;
void setup(){
  pinMode( LED, OUTPUT); // inicializa el pin del LED como de SALIDA
  pinMode( SenIN_D0 , INPUT_PULLUP) ; // inicializa el pin SenIN_D0 como
entrada en modo PULLUP
  digitalWrite(LED , EstadoLED) ;
}
void loop(){
  SensorVal =  digitalRead(SenIN_D0) ; // se lee el sensor
```

```
  if ( SensorVal) { // Si se obtuvo una lectura alta (1) se procede a hacer los cambios
    if (EstadoLED) { EstadoLED = false;} // Si el estado del LED era encendido entonces se apaga
    else  { EstadoLED = true;} // Si el estado del LED era apagado entonces se enciende
    digitalWrite(LED, EstadoLED); // se envía la orden de acción al pin del LED
  }
}
```

Sensor de Interruptor Táctil

Este módulo consta de un sensor capacitivo sensible al tacto que cuando es "tocado" en la patita del sensor (flecha roja) que está expuesta compara el valor medido contra el valor definido por el PreSet (potenciómetro multivuelta) y si se alcanza el valor seleccionado se codifica en el terminal de salida digital como un valor verdadero (Voltaje 5V)., caso contrario la salida digital se mantendrá en falso (Voltaje 0V).

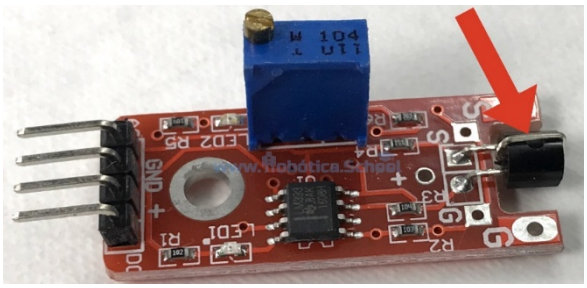


Figura 10: KY036 - Sensor de Interruptor Táctil. Información tomada de la Investigación directa. Elaborada por los autores

Este sensor tiene dos salidas, una analógica y otra digital, la analógica sirve para leer el valor de la posición del PreSet en una escala de 0 a 1023, si por ejemplo Ud configura el PreSet a mitad de camino la salida en la terminal analógica leerá el valor 512 aproximadamente.

Este tipo de sensor se usa como un interruptor táctil en el que, al no haber partes mecánicas involucradas, incrementa en mucho su uso ya que no sufre desgaste mecánico como los interruptores típicos como el KY004, a más de poder graduar el nivel de sensibilidad deseado mediante el potenciómetro multivuelta.

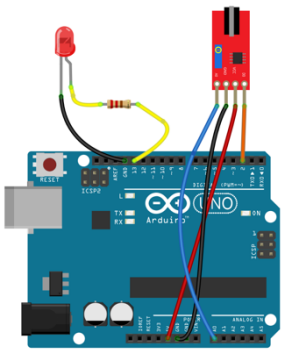


Figura 11. Esquema de conexión de Sensor de Interruptor Táctil. Información tomada de la Investigación directa. Elaborada por los autores

Desarrollo

```
// Intercambiar el estado de un LED entre encendido y apagado según
// se vaya detectando el tacto del interruptor, el estado
// inicial del LED es apagado
int LED = 13;
int SenIN_D0 = 2; // define el pin digital 2, para la señal de D0
bool SensorVal= false, EstadoLED=false;
void setup(){
  pinMode( LED, OUTPUT); // inicializa el pin del LED como de SALIDA
  pinMode( SenIN_D0 , INPUT_PULLUP) ; // inicializa el pin SenIN_D0 como
entrada en modo PULLUP
  digitalWrite(LED , EstadoLED) ;
}
void loop(){
  SensorVal =  digitalRead(SenIN_D0) ; // se lee el sensor
  if ( SensorVal) { // Si se obtuvo una lectura alta (1) se procede a hacer
los cambios
    if (EstadoLED) { EstadoLED = false;} // Si el estado del LED era
encendido entonces se apaga
```

```
    else  { EstadoLED = true;} // Si el estado del LED era apagado
entonces se enciende
    digitalWrite(LED, EstadoLED); // se envía la orden de acción al pin del
LED
  }
}
```

2.3. Sensores Magnéticos

Son aquellos que basan su trabajo en la detección de campos magnéticos, los que pueden ser generados tanto por imanes permanentes o por electroimanes, los hay aquellos que únicamente actúan como interruptor indicando la presencia o no de determinado nivel de campo magnético y también aquellos que nos permiten medir la magnitud de campo magnético, así como su dirección, presente en el punto donde está ubicado el sensor, esto último es útil también como una brújula dependiendo de la sensibilidad del sensor, este tipo de sensores se encuentran en teléfonos celulares modernos y pueden usarse con aplicaciones propias del equipo o de terceros.

En este grupo tenemos para el caso de la serie Keyes los módulos KY003, KY021 , KY024, KY025 y el KY035, que deben ser usados de acuerdo con las necesidades del proyecto a montar.

Sensor de campo magnético por efecto Hall

De estos el KY003 y el KY035 son prácticamente iguales en su funcionamiento, ambos están basados en alguno de estos sensores: 717bg , 44E311 , 3144-95301 , 3144 EUA-S o 3144 LUA-S dependiendo del fabricante del módulo, ambos detectan la presencia de un campo magnético que debe ser lo suficientemente alto para activar su salida digital, la diferencia es que el módulo KY035 no incluye el LED que indica cuando se activa la señal de salida mientras que en el KY003 si se incluye un led indicador.

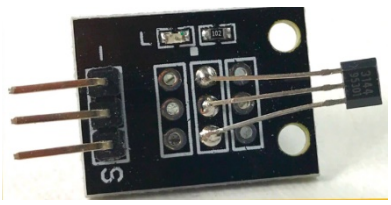


Figura 12. KY003 - Sensor de campo magnético por efecto Hall (con LED) – Sensor Bihor. Información tomada de la Investigación directa. Elaborada por los autores

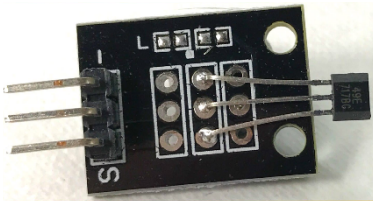


Figura 13. KY035 - Sensor de campo magnético por efecto Hall. (sin LED) – Sensor Bihor. Información tomada de la Investigación directa. Elaborada por los autores

Mediante estos sensores, si en un momento determinado se acerca una fuente de campo magnético que supere el nivel necesario de acuerdo con el dispositivo (sensor) usado, este lo detecta enviando a la salida digital el valor "TRUE".

Desarrollo

```
/*Sensor de Campo Magnetido de efecto HALL, cuando detecte un
determinado nivel de camp magnetico se accionará la salida digital
a un nivel alto, mientras tanto permanecerá en nivel bajo.
al recibir la señal del sensor se encendera un Led durante 1 segundo y
luego apagara el LED hasta que detecte nuevamente otro campo */
int LED = 13;
int SenIN_D0 = 2; // define el pin digital 2, para la señal de D0
bool SensorVal= false;
void setup(){
  pinMode( LED, OUTPUT); // inicializa el pin del LED como de SALIDA
  pinMode( SenIN_D0 , INPUT_PULLUP) ; // inicializa el pin SenIN_D0 como
entrada en modo PULLUP
  digitalWrite(LED , false) ;
}
void loop(){
  SensorVal =  digitalRead(SenIN_D0) ; // se lee el sensor
  if ( SensorVal) { // Si se obtuvo una lectura alta (1) se procede a hacer
los cambios
    digitalWrite(LED, true) ; // enciende el led
    delay (1000); // mantiene el led encendido 1000 milisegundos
    digitalWrite(LED, false) ; // apaga el led
  }
}
```

Sensor de Interruptor de lámina Magnética

Estos módulos KY021 y KY025 están conformados por un interruptor tipo REED (interruptor encapsulado en cristal) como se puede observar en las

imágenes, el modelo KY021 únicamente tiene tres terminales y una resistencia de 10KΩ que puede ser usada en configuración PUUL-DOWN o PULL-UP según la necesidad que tengamos, mientras que el modelo KY025 posee 4 pines, dos para alimentación y dos salidas, una analógica y otra digital, además tiene un potenciómetro para ser usado como PreSet para configurar la sensibilidad a la cual se acciona la salida digital.

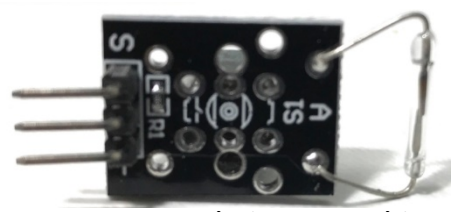


Figura 14. KY021 - Sensor de Interruptor de lámina Magnética. Información tomada de la Investigación directa. Elaborada por los autores

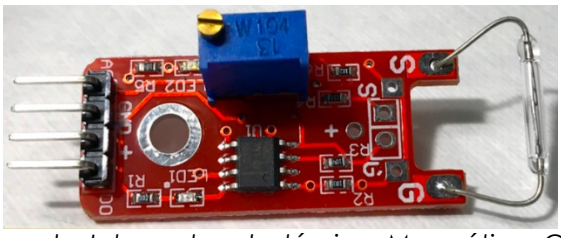


Figura 15. KY025 - Sensor de Interruptor de lámina Magnética Graduable. Información tomada de la Investigación directa. Elaborada por los autores

Descripción

Estos sensores son ampliamente usados en sitios donde hay un ambiente inflamable y se debe evitar cualquier tipo de chispas que pudiesen provocar incendios, ya que al estar encerrado al vacío en la capsula de cristal no hay chispas en el exterior y para accionarlos solo se acerca una fuente de campo magnético.

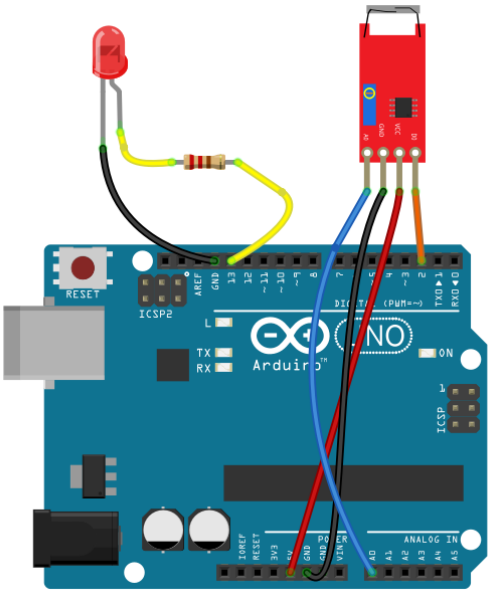


Figura 16. Esquema de conexión de Sensor de Interruptor de lámina Magnética Graduable. Información tomada de la Investigación directa. Elaborada por los autores

Desarrollo

```
/*Sensor de Campo Magnetido por lamina, cuando detecte un determinado nivel de campo magnetico se accionara la salida digital a un nivel alto, mientras tanto permanecera en nivel bajo.
al recibir la señal del sensor se encendera un Led durante 1 segundo y luego apagara el LED hasta que detecte nuevamente otro campo */
int LED = 13;
int SenIN_D0 = 2; // define el pin digital 2, para la señal de D0
bool SensorVal= false;
void setup(){
  pinMode( LED, OUTPUT); // inicializa el pin del LED como de SALIDA
  pinMode( SenIN_D0 , INPUT_PULLUP) ; // inicializa el pin SenIN_D0 como entrada en modo PULLUP
  digitalWrite(LED , false) ;
}
void loop(){
  SensorVal =  digitalRead(SenIN_D0) ; // se lee el sensor
  if ( SensorVal) { // Si se obtuvo una lectura alta (1) se procede a hacer los cambios
    digitalWrite(LED, true) ; // enciende el led
    delay (1000); // mantiene el led encendido 1000 milisegundos
```

```
    digitalWrite(LED, false) ; // apaga el led
  }
}
```

2.4. Sensores Mecánicos

Aquí encontramos aquellos sensores que miden o detectan variaciones de tipo mecánico como por ejemplo impactos, inclinación rotación, o alguna otra acción que implique una acción de algún mecanismo como interruptores de fin de carrera etc.

Sensor de interruptor de vibración

Figura 17. KY002 - Interruptor de vibración. Información tomada de la Investigación directa. Elaborada por los autores

Descripción

Este módulo funciona como un interruptor (Switch), al detectar una vibración (el nivel de sensibilidad puede ser configurado por medio del potenciómetro) cierra un circuito lo que genera que el usuario pueda detectar los movimientos, como parte de su mecanismo de lectura consta de un integrado LM393 que es un comparador basado en Amplificadores operacionales.

Puede ser usado para la detección de movimientos fuertes como sismos o terremotos, implementarlo en un motor para prevenir daños ante las vibraciones excesivas, etc. .

Desarrollo

```
int Led = 13;
int SenIN_D0 = 2; // define el pin digital 2, para la señal de D0
bool SensorVal= false;
void setup ()
{
  pinMode (Led, OUTPUT) ;
  pinMode (SenIN_D0, INPUT) ;
}
void loop ()
{
```

```
  SensorVal = digitalRead (SenIN_D0) ;
  if (SensorVal == HIGH)
  {
        digitalWrite (Led, LOW);
  }
  else
  {
        digitalWrite (Led, HIGH);
        delay (1000); /* Enciende el Led durante un segundo*/
  }
}
```

Sensor de Inclinación por Mercurio

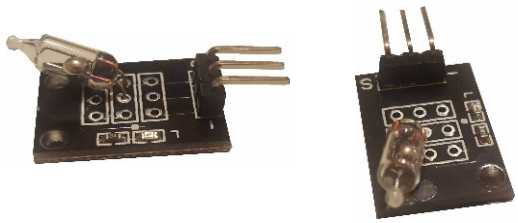


Figura 18. KY017 - Sensor de Inclinación por Mercurio. Información tomada de la Investigación directa. Elaborada por los autores

Descripción

Es un conmutador de inclinación que consta de una ampolla con dos terminales y una pequeña bola de mercurio (metal conductor) que le permite detectar la inclinación de su objeto para que pueda tomar la acción apropiada, para su uso se fija sólidamente al objeto del cual se pretende medir cuando se incline, se debe graduar mecánicamente para que al obtener la inclinación deseada el sensor este en posición de encendido y pueda ser útil.

Una vez ingresada la codificación de ejemplo se notará que se activa el sensor a espera del movimiento de inclinación, la cual de producirse tendrá como respuesta una acción.

Desarrollo

/*Sensor de nivel de inclinación, cuando detecte un nivel de inclinación configurado se accionará la salida digital a un nivel alto, mientras tanto permanecerá en nivel bajo.

```
al recibir la señal del sensor se encendera un Led durante 1 segundo y
luego apagara el LED hasta que detecte nuevamente otra inclinación
*/
int LED = 13;
int SenIN_D0 = 2; // define el pin digital 2, para la señal de D0
bool SensorVal= false;
void setup(){
  pinMode( LED, OUTPUT); // inicializa el pin del LED como de SALIDA
  pinMode( SenIN_D0 , INPUT_PULLUP) ; // inicializa el pin SenIN_D0
  digitalWrite(LED , false) ;
}
void loop(){
  SensorVal =  digitalRead(SenIN_D0) ; // se lee el sensor
  if ( SensorVal) { // Si se obtuvo una lectura alta (1) se procede a hacer
los cambios
    digitalWrite(LED, true) ; // enciende el led
    delay (1000); // mantiene el led encendido 1000 milisegundos
    digitalWrite(LED, false) ; // apaga el led
  }
}
```

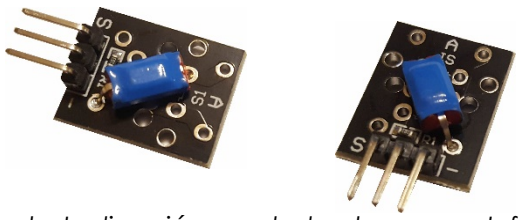


Figura 19. KY020 - Sensor de Inclinación por bola de acero. Información tomada de la Investigación directa. Elaborada por los autores

Descripción

Son sensores de bastante pequeños, usualmente de unos pocos milímetros de largo, que llevan en su interior una o dos pequeñas bolas de un material conductor, capaces de cerrar el circuito con los pines metálicos colocados al interior del cilindro.

Cuando hacen contacto permiten el paso de la corriente y cierran el contacto exactamente igual que si fueran un interruptor, pero que a partir de un cierto ángulo de inclinación dejan de hacer contacto y abren el circuito.

Este sensor es muy sencillo ya que nos indicará cuando se incline un determinado ángulo que dependerá de la posición relativa del objeto a

medir con la horizontal, esta diseñado para accionarse con una diferencia de más de 15º. Dando a su salida la indicación de que se superó dicha diferencia angular

Desarrollo

```
/*Sensor de nivel de inclinación, cuando detecte un nivel de
inclinación configurado se accionará la salida digital a un nivel alto,
mientras tanto permanecera en nivel bajo.
al recibir la señal del sensor se encendera un Led durante 1 segundo y
luego apagara el LED hasta que detecte nuevamente otra inclinación
*/
int LED = 13;
int SenIN_D0 = 2; // define el pin digital 2, para la señal de D0
bool SensorVal= false;
void setup(){
  pinMode( LED, OUTPUT); // inicializa el pin del LED como de SALIDA
  pinMode( SenIN_D0 , INPUT_PULLUP) ; // inicializa el pin SenIN_D0
  digitalWrite(LED , false) ;
}
void loop(){
  SensorVal =  digitalRead(SenIN_D0) ; // se lee el sensor
  if ( SensorVal) { // Si se obtuvo una lectura alta (1) se procede a hacer
los cambios
    digitalWrite(LED, true) ; // enciende el led
    delay (1000); // mantiene el led encendido 1000 milisegundos
    digitalWrite(LED, false) ; // apaga el led
  }
}
```

Sensor de dirección-Joystick de Ejes XY

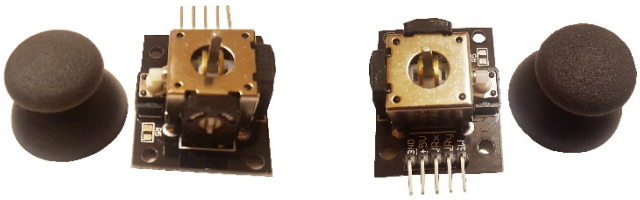


Figura 20. Joystick de Ejes XY. Información tomada de la Investigación directa. Elaborada por los autores

Descripción

Este Módulo protector, conecta los pines correspondientes a Arduino a través de cables de sensores de Arduino, con las siguientes especificaciones: movimientos direccionales son simplemente dos potenciómetros uno para cada eje; también es compatible con la interfaz. Sus características son las siguientes:

- Ejes independientes "X" y "Y"
- Botón (pushbutton) en el eje Z
- Experiencia similar al Joystick de un playstation
- Utiliza solamente 2 pines analógicos y un digital para su conexión
- Incluye palanca de plástico como se muestra en las fotos
- Perforaciones para montaje en el PCB
- Conexión mediante pines macho estándar
- Puede conectarse mediante arnés dupont

Su uso:

- Comúnmente utilizado en la creación de:
- Controles de mando de consolas
- Juguetes a control remoto

Desarrollo

```
int sensorPin = 5;
int value = 0;
void setup() {
  pinMode(3, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  value = analogRead(0);
```

```
Serial.print("X:");
Serial.print(value, DEC);
value = analogRead(1);
Serial.print(" | Y:");
Serial.print(value, DEC);
value = digitalRead(7);
Serial.print(" | Z: ");
Serial.println(value, DEC);
delay(100);}
```

Sensor de Impactos (choques)

Este sensor es de accionamiento muy similar a los sensores de inclinación de bola, pero está diseñado para que, cuando detecte un cambio brusco de movimiento (impacto) se accione su salida digital, al constar de un interruptor de accionamiento mecánico su uso debe ser controlado ya que las partes mecánicas tienden a sufrir desgastes típicos de su funcionamiento.

tiene muchas aplicaciones prácticas para la ayuda tanto humana (personas con deficiencia auditiva) como para los equipos electrónicos inteligentes e independientes, incluso se podría proyectar y aplicar en inteligencia artificial ya que su funcionamiento interactúa con factores físicos que por su composición consta de acción con reacción por medio de indicadores led programables.

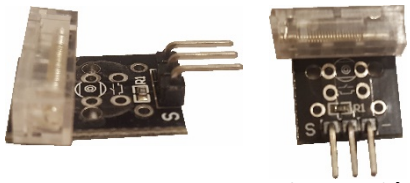


Figura 21. KY031 - Sensor de Impactos (choques). Información tomada de la Investigación directa. Elaborada por los autores

Desarrollo

```
/*Sensor de nivel de inclinación, cuando detecte un nivel de
inclinación configurado se accionara la salida digital a un nivel alto,
mientras tanto permanecera en nivel bajo.
al recibir la señal del sensor se encendera un Led durante 1 segundo y
luego apagara el LED hasta que detecte nuevamente otra inclinación
*/
int LED = 13;
int SenIN_D0 = 2; // define el pin digital 2, para la señal de D0
bool SensorVal= false;
```

```
void setup(){
  pinMode( LED, OUTPUT); // inicializa el pin del LED como de SALIDA
  pinMode( SenIN_D0 , INPUT_PULLUP) ; // inicializa el pin SenIN_D0
  digitalWrite(LED , false) ;
}
void loop(){
  SensorVal =  digitalRead(SenIN_D0) ; // se lee el sensor
  if ( SensorVal) { // Si se obtuvo una lectura alta (1) se procede a hacer
los cambios
    digitalWrite(LED, true) ; // enciende el led
    delay (200); // mantiene el led encendido 200 milisegundos
    digitalWrite(LED, false) ; // apaga el led
    delay (100); // apaga led 100 milisegundos para crear el efecto de
flash
    digitalWrite(LED, true) ; // enciende el led
    delay (200); // mantiene el led encendido otros 200 milisegundos
    digitalWrite(LED, false) ; // apaga el led
  }
}
```

Sensor de Rotación codificada ENCODER

Este sensor es un potenciómetro encoder de rotación y electromecánico que convierte la posición angular del potenciómetro a un valor digital o por pulsos. Digamos que es un potenciómetro sin fin con el cual se puede obtener una precisión muy alta. También funciona como botón pulsador el ejercer presión sobre su eje. Para saber a cuánto equivale un ángulo eléctrico con un ángulo mecánico de 360 grados.

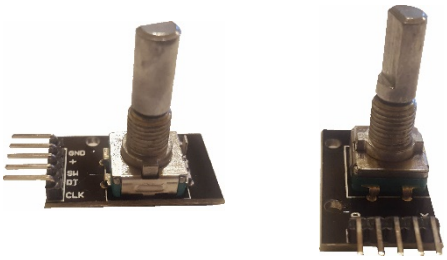


Figura 22. KY040 - Sensor de Rotación codificada ENCODER. Información tomada de la Investigación directa. Elaborada por los autores

Desarrollo

```
int redPin = 2;
int yellowPin = 3;
int greenPin = 4;
int aPin = 6;
int bPin = 7;
```

```
int buttonPin = 5;
int state = 0;
int longPeriod = 5000; // Time at green or red
int shortPeriod = 700; // Time period when changing
int targetCount = shortPeriod;
int count = 0;
void setup()
{
  pinMode(aPin, INPUT);
  pinMode(bPin, INPUT);
  pinMode(buttonPin, INPUT);
  pinMode(redPin, OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
}
void loop()
{
  count++;
  if (digitalRead(buttonPin))
  {
    setLights(HIGH, HIGH, HIGH);
  }
  else
  {
    int change = getEncoderTurn();
    int newPeriod = longPeriod + (change * 1000);
    if (newPeriod >= 1000 && newPeriod <= 10000)
    {
      longPeriod = newPeriod;
    }
    if (count > targetCount)
    {
      setState();
      count = 0;
    }
  }
  delay(1);
}
int getEncoderTurn()
{
```

```
  // return -1, 0, or +1
  static int oldA = LOW;
  static int oldB = LOW;
  int result = 0;
  int newA = digitalRead(aPin);
  int newB = digitalRead(bPin);
  if (newA != oldA || newB != oldB)
  {
    // something has changed
    if (oldA == LOW && newA == HIGH)
    {
      result = -(oldB * 2 - 1);
    }
  }
  oldA = newA;
  oldB = newB;
  return result;
}
  int setState()
  {
    if (state == 0)
    {
      setLights(HIGH, LOW, LOW);
      targetCount = longPeriod;
      state = 1;
    }
    else if (state == 1)
    {
      setLights(HIGH, HIGH, LOW);
      targetCount = shortPeriod;
      state = 2;
    }
    else if (state == 2)
    {
      setLights(LOW, LOW, HIGH);
      targetCount = longPeriod;
      state = 3;
    }
    else if (state == 3)
    {
```

```
      setLights(LOW, HIGH, LOW);
      targetCount = shortPeriod;
      state = 0;
    }
  }
  void setLights(int red, int yellow, int green)
{
  digitalWrite(redPin, red);
  digitalWrite(yellowPin, yellow);
  digitalWrite(greenPin, green);
}
```

2.5. Sensores médicos

Sensor de latidos de corazón

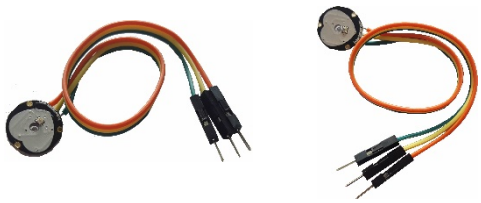


Figura 23. Sensor latido del corazón. Información tomada de la Investigación directa. Elaborada por los autores

Descripción

Funciona con un sensor de ritmo cardiaco óptico, una etapa de amplificación y un filtro para el ruido lo cual hace que su señal sea confiable y estable, este sensor por medio de un pequeño led IR y un transistor óptico puede detectar las pulsaciones en los dedos.

El sensor de latidos de corazón es un sensor para Arduino inmediatamente utilizable que permite medir el pulso de una persona. Está perfectamente adecuado para estudiantes, artistas, deportistas, desarrolladores, etc.

Sirve para poder medir el ritmo cardiaco y sirve para medir el pulso de una persona, si nos vamos para sensores de latidos más complejos se puede tomar la azúcar y también la presión, es un sensor muy eficaz para el campo de la medicina.

Desarrollo

```
int pulsi=0;
void setuo() { //configuración inicial de arduino
  Serial.begin(9600);
  pinMode(A5,INPUT);
  pinMode(11,OUTPUT);
```

```
}
void loop() { //repite infinitamente.
  pulso=analogRead(A5);
  if (pulso>=513){
     digitalWrite(11,HIGH);
     delay(30);
     digitalWrite(11,LOW);
  }
  delay(50);
  Serial.println(pulso);
}
```

2.6. Sensores ópticos

Sensor de infrarrojos o ultravioleta

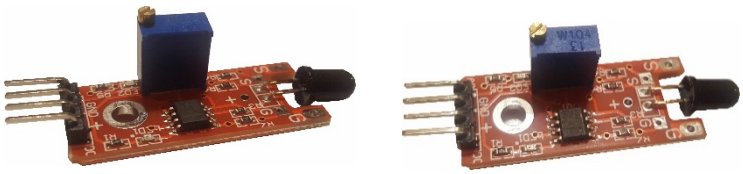


Figura 24 Sensor de infrarrojos. Información tomada de la Investigación directa. Elaborada por los autores

Descripción

Detectan las radiaciones infrarrojas o ultravioletas (según el tipo de sensor seleccionado) que acompañan a las llamas u otro fenómeno físico que genere emisiones infrarrojas o ultravioletas. Contienen filtros ópticos, célula captadora y equipo electrónico que amplifica las señales. Son de construcción muy complicada. Requieren mantenimiento similar a los sensores ópticos de humo.

El sensor infrarrojo para longitudes de onda entre 760 nm a 1100 nm es el más sensible por el momento y tiene un ángulo de detección de 60 grados.

Desarrollo

```
//Example for KY-026
//TkkrLab
int Led = 13 ;// define LED Interface
int buttonpin = 3; // define la flame sensor interface
int analoog = A3; // la flame sensor interface
 int val ;// define numeric variables val
float sensor; //read analoogo value
```

```
 void setup ()
{
  pinMode (Led, OUTPUT) ;// define LED as output interface
  pinMode (buttonpin, INPUT) ;// output interface defines the flame sensor
  pinMode (analoog, INPUT) ;// output interface defines the flame sensor
  Serial.begin(9600);
}
 void loop ()
{
  sensor = analogRead(analoog);
  Serial.println(sensor);  // display tempature
   val = digitalRead (buttonpin) ;// digital interface will be assigned a value of 3 to read val
    if (val == HIGH) // When the flame sensor detects a signal, LED flashes
  {
    digitalWrite (Led, HIGH);
  }
  else
  {
    digitalWrite (Led, LOW);
  }
  delay(1000);
}
```

Sensor receptor de infrarrojos

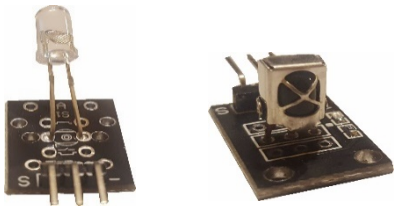


Figura 25. Emisor y Receptor Infrarrojo. Información tomada de la Investigación directa. Elaborada por los autores

Descripción

Este sensor de emisor de infrarrojos es un pequeño modulo que convierte la energía eléctrica en luz del infrarrojo cercano. También se conoce como

diodo emisor de infrarrojos. Su estructura es similar a la de la luz general del diodo emisor, pero hecha de un material semiconductor diferente. Está configurado para recibir, amplificar y demodular la luz del infrarrojo cercano en una señal digital. Muy útil cuando quieres controlar cualquier cosa a distancia.
Ya que usando un módulo transmisor y un receptor de infrarrojos podemos comunicar cualquier trama de información a un objetivo cercano, de hecho, ellos están ahora en nuestra vida diaria.
Juegan un papel importante en una gran cantidad de electrodomésticos y se utilizan en dispositivos tales como aire acondicionado, TV, DVD, etc.

Desarrollo

```
#include <IRremote.h>
int RECV_PIN = 11;
IRrecv irrecv(RECV_PIN);
decode_results results;
void setup()
{
 Serial.begin(9600);
 irrecv.enableIRIn();
}
void dump(decode_results *results)
{

int count = results->rawlen;
 if (results->decode_type == UNKNOWN)
 {
 Serial.print("Unknown encoding: ");
 }
 else if (results->decode_type == NEC)
 {
 Serial.print("Decoded NEC: ");
 }
 else if (results->decode_type == SONY)
 {
 Serial.print("Decoded SONY: ");
 }
 else if (results->decode_type == RC5)
 {
 Serial.print("Decoded RC5: ");
```

```
}
else if (results->decode_type == RC6)
{
Serial.print("Decoded RC6: ");
}
else if (results->decode_type == PANASONIC)
{
  Serial.print("Decoded PANASONIC - Address: ");
//Serial.print(results->panasonicAddress,HEX);
Serial.print(" Value: ");
}
else if (results->decode_type == LG)
{
Serial.print("Decoded LG: ");
}
else if (results->decode_type == JVC)
{
Serial.print("Decoded JVC: ");
}
else if (results->decode_type == AIWA_RC_T501)
{
Serial.print("Decoded AIWA RC T501: ");
}
else if (results->decode_type == WHYNTER)
{
Serial.print("Decoded Whynter: ");
}
Serial.print(results->value, HEX);
Serial.print(" (");
Serial.print(results->bits, DEC);
Serial.println(" bits)");
Serial.print("Raw (");
Serial.print(count, DEC);
Serial.print("): ");
for (int i = 0; i < count; i++)
{
if ((i % 2) == 1) {
Serial.print(results->rawbuf[i]*USECPERTICK, DEC);
}
else {
```

```
 Serial.print(-(int)results->rawbuf[i]*USECPERTICK, DEC);
 }
 Serial.print(" ");
 }
 Serial.println("");
}
void loop() {
 if (irrecv.decode(&results))
 {
 Serial.println(results.value, HEX);
 dump(&results);
 irrecv.resume();
 }
}
}
```

Sensor de proximidad (evasión de obstáculos)

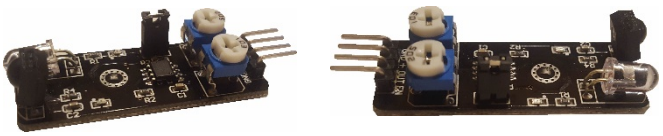


Figura 26. Proximidad (Evasión de obstáculo). Información tomada de la Investigación directa. Elaborada por los autores

Descripción

Sensor de evitación de obstáculos por infrarrojo, está diseñado para la elaboración robots de ruedas evasores de obstáculos. Este sensor de luz ambiental es adaptable, de alta precisión, que tiene un par de infrarrojos de transmisión y recepción, el infrarrojo transmisor emite una cierta frecuencia, al detectar la dirección de un obstáculo (reflector), él tuvo receptor de infrarrojos recibe el reflejo, cuando esto sucede el indicador LED enciende, a través del circuito, la señal digital de salida puede ser ajustada junto con la distancia de detección por medio de los potenciómetros, la distancia efectiva es de 2 ~ 40cm, voltaje de trabajo de 3.3V- 5V, el voltaje de funcionamiento es amplio.

El módulo es muy sencillo de programar, es usado en diversas aplicaciones como: parqueo automatizado de vehículos, construcción de roboots de batallas, detector para personas no videntes.

Desarrollo

```
const int buttonPin = 2;
```

```
const int ledPin = 13;
int buttonState = 0;
void setup() {
pinMode(ledPin, OUTPUT);
pinMode(buttonPin, INPUT);
}

void loop(){
buttonState = digitalRead(buttonPin);
if (buttonState == HIGH) {
digitalWrite(ledPin, HIGH);
}
else {
digitalWrite(ledPin, LOW);
}
}
```

Sensor de caza

Figura 27. Sensor de caza. Información tomada de la Investigación directa. Elaborada por los autores

Descripción

Un detector de obstáculos infrarrojo es un dispositivo que detecta la presencia de un objeto mediante la reflexión que produce en la luz. El uso de luz infrarroja (IR) es simplemente para que esta no sea visible para los humanos.

Constitutivamente son sensores sencillos. Se dispone de un LED emisor de luz infrarroja y de un fotodiodo (tipo BPV10NF o similar) que recibe la luz reflejada por un posible obstáculo.

Los detectores de obstáculo suelen proporcionarse con una placa de medición estándar con el comparador LM393, que permite obtener la lectura como un valor digital cuando se supera un cierto umbral, que se regula a través de un potenciómetro ubicado en la placa.

Este tipo de sensores actúan a distancias cortas, típicamente de 5 a 20mm. También son útiles en otro tipo de aplicaciones como, por ejemplo, detectar la presencia de un objeto en una determinada zona, determinar

una puerta está abierta o cerrada, o si una máquina ha alcanzado un cierto punto en su desplazamiento.

Desarrollo

```
const int sensorPin = 9;
 void setup() {
Serial.begin(9600);   //iniciar puerto serie
  pinMode(sensorPin , INPUT);  //definir pin como entrada
}
void loop(){
  int value = 0;
  value = digitalRead(sensorPin );  //lectura digital de pin
  if (value == HIGH) {
      Serial.println("Detectado obstaculo");
  }
  deLay(1000);
}
```

2.7. Sensores térmicos

Sensor de Temperatura y Humedad

Sensor de temperatura

Figura 28. Sensor de temperatura. Información tomada de la Investigación directa. Elaborada por los autores

Descripción

El sensor que corresponde al módulo de temperatura tiene la versión o ítem KY-013. Esta a su vez permite proporcionar al usuario de forma digital tanto la temperatura como la humedad, también son muy habituales en el mundo de los aficionados a la electrónica por su bajo precio.

Hay que tomar en cuenta que solamente el sensor que tiene la versión KY-013 puede detectar la humedad y la temperatura en Celsius de una habitación dada.

Desarrollo

```
float tempC;
int pinLM35 = 0;
void setup() {
  Serial.begin(9600);
}
void loop() {
  tempC = analogRead(pinLM35);
  tempC = (5.0 * tempC * 100.0)/1024.0;
  Serial.print(tempC);
  Serial.print("\n");
  delay(1000);
}
```

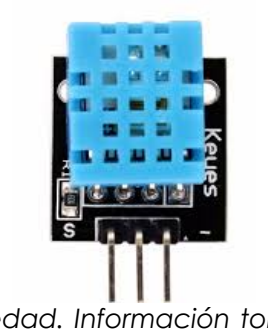


Figura 29. KY015 - Sensor de humedad. Información tomada de la Investigación directa. Elaborada por los autores

Es un sensor sencillo que mide la humedad del suelo por la variación de su conductividad. No tiene la precisión suficiente para realizar una medición absoluta de la humedad del suelo, pero tampoco es necesario para controlar un sistema de riego.

El sensor de humedad sirve para detectar la humedad del aire. Se usa cada vez más en el sector de la técnica de calefacción, ventilación y climatización, para poder experimentar poniéndole fuego o hielo cerca y así notar los cambios en el monitor serial.

Desarrollo

```
// Variable que almacenará la temperatura
float temperatura;
void setup () {
  Serial.begin(9600);
}
```

```
void loop() {
  // Leemos del pin analógico 0
  temperatura = analogRead(0);
  // Convertimos el valor leído de voltaje a ºC
  temperatura = 5.0*temperatura*100.0/1024.0;
  Serial.print(temperatura);
  Serial.println(" oC");
  delay(1000);
}
```

3. Actuadores

En este apartado trataremos acerca de los módulos que no detectan señales sino más bien generan señales que pueden ser de diversos tipos, a estos dispositivos los denominamos "actuadores" y son usados para realizar acciones en base a los requerimientos del proyecto como emitir sonidos al haber pensado una actividad extraña, emitir luz para indicadores o emitir rayos láser para que estos sean detectados y activen una u otra acción cuando dicho laser se "corte".

Trataremos los módulos actuadores que existen en la serie Keyes entre

3.1. Actuadores Ópticos

Láser rojo

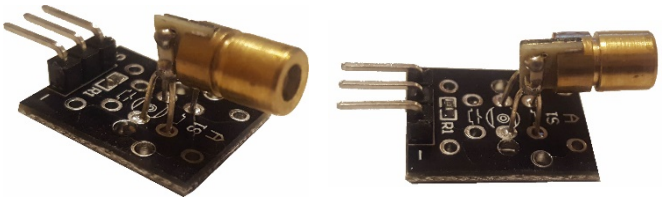


Figura 30. Sensor láser. Información tomada de la Investigación directa. Elaborada por los autores

Descripción

Esta a su vez contiene un diodo de emisor laser en la cual trabaja con 5 voltios y desata con una longitud de onda de 6650 nanómetro. Este módulo tiene integrado un transmisor laser y una interfaz digital, además cuenta con un LED incorporado.

Hay que tomar en cuenta que este tipo de sensores pueden usarse para el resguardo de manera extrema pertenencias privadas y que se pueden anexar a otros sensores como el sensor de sonido.

Desarrollo

```
int laser=9;
void setup() {
pinMode(laser, OUTPUT);
}
void loop() {
digitalWrite(laser, HIGH);
delay(3000);
digitalWrite(laser, LOW);
delay(1000);
}
```

LED RGB

Figura 31. LED RGB. Información tomada de la Investigación directa. Elaborada por los autores

Desarrollo

Son tres leds en un mimo empaque, estos leds están compuestos de leds de colores primarios: rojo (Red), verde (Green), y azul (Blue), al variar la intensidad de corriente de cada led se producen diferentes colores.
En este led se reconoce por consistir en tres luces (con sus chips) rojo, verde y azul (red, green, blue).

Desarrollo

```
//KY016 3-color LED module
int redpin = 11; // select the pin for the red LED
int bluepin = 10; // select the pin for the blue LED
int greenpin = 9 ;// select the pin for the green LED
int val;
void setup () {
 pinMode (redpin, OUTPUT);
 pinMode (bluepin, OUTPUT);
 pinMode (greenpin, OUTPUT);
 Serial.begin (9600);
}
void loop ()
{
 for (val = 255; val> 0; val --)
 {
 analogWrite (11, val);
 analogWrite (10, 255-val);
 analogWrite (9, 128-val);
 delay (10);
 Serial.println (val, DEC);
 }
 for (val = 0; val <255; val ++)
 {
 analogWrite (11, val);
```

```
 analogWrite (10, 255-val);
 analogWrite (9, 128-val);
 delay (10);
 Serial.println (val, DEC);
 }
}
```

LED de 2 colores

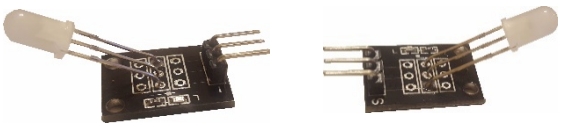


Figura 32. Módulo LED 2 colores. Información tomada de la Investigación directa. Elaborada por los autores

Descripción

El LED de dos colores funciona con frecuencia como luces indicadoras para una variedad de dispositivos, incluyendo televisores, cámaras digitales y controles remotos.

El módulo KY-011 puede alternar los colores del Led entre rojo y verde, es decir que no tendremos que hacer dos agujeros en la carcasa de nuestro proyecto y poner dos Led para mostrar encendido (verde) y apagado (rojo) por ejemplo, ya que el módulo Led de 2 colores hará el trabajo utilizando un solo Led.

Desarrollo

```
int redpin = 11; // select the pin for the red LED
int greenpin = 10; // select the pin for the green LED
int val;
void setup () {
pinMode (redpin, OUTPUT);
pinMode (greenpin, OUTPUT);
}
void loop () {
for (val = 255; val> 0; val--)
{
analogWrite (greenpin, val);
analogWrite (redpin, 255-val);
delay (15);
}
```

```
for (val = 0; val <255; val++)
{
analogWrite (greenpin, val);
analogWrite (redpin, 255-val);
delay (15);
}
}
```

LED 3 colores

Figura 33. Interruptor de inclinación. Información tomada de la Investigación directa. Elaborada por los autores

Descripción

Este LED utiliza el 5050LED de 3 colores con corriente máxima de 20mA
Voltaje: 1.80V Red (2,4 max), Verde, Azul 2.8V (3.6V)
RGB tricromático resistencia limitadora para prevenir el agotamiento a través de la PWM ajuste de tres colores primarios se pueden mezclar para obtener diferentes colores con una variedad de interfaz de un solo chip.
Voltaje de funcionamiento: 5V
Modo de avance del LED: conductor de cátodo común
Este ofrece cuando cuyos diodos están polarizados directamente, emite luz. Consiste en 3 colores primarios, rojo, verde y azul, con tres entradas y un cátodo común. Con las salidas PWM se puede controlar la intensidad de cada diodo LED.
Construcción de una alarma para casa. Con ayuda de detectores de movimiento y muchas luces y sonido si queremos, podremos construir la nuestra propia, no muy válida para proteger la casa, pero sí para estar alerta por si nuestra mascota decide invadir zona protegida de casa.

Desarrollo

```
//Parpadeo con el módulo LED SMD de 3 colores RGB
int led_verde_G = 8;
int led_rojo_R = 9;
int led_azul_B = 10;
void setup() {
```

```
    pinMode(led_verde_G, OUTPUT);
  pinMode(led_rojo_R, OUTPUT);
  pinMode(led_azul_B, OUTPUT);
}
void loop() {
  digitalWrite(led_verde_G, LOW);
  digitalWrite(led_rojo_R, LOW);
  digitalWrite(led_azul_B, HIGH);

  delay(1000);
  digitalWrite(led_verde_G, LOW);
  digitalWrite(led_rojo_R, HIGH);
  digitalWrite(led_azul_B, LOW);
  delay(1000);
  digitalWrite(led_verde_G, HIGH);
  digitalWrite(led_rojo_R, LOW);
  digitalWrite(led_azul_B, LOW);
  delay(1000);
}
```

Emisor de Infrarojos

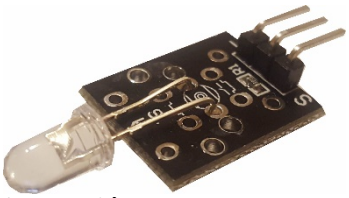


Figura 34. Emisor de Infrarojos. Información tomada de la Investigación directa. Elaborada por los autores

Descripción

Éste es un módulo transmisor que se utiliza para emitir señal de infrarrojos. Los sensores infrarrojos son muy cotidianos en nuestra vida, los vemos en muchos aparatos electrónicos como televisiones, DVD's y muchos otros. Este tipo de sensor en la actualidad se está utilizando en la detección de proximidad de un objeto y en base a eso evitar el impacto o colisión. La codificación mostrada a continuación es un ejemplo de programación para tal efecto.

Desarrollo

```
const int sensorPin = 9;
 void setup() {
  Serial.begin(9600);   //iniciar puerto serie
```

```
  pinMode(sensorPin , INPUT);  //definir pin como entrada
}
 void loop(){
  int value = 0;
  value = digitalRead(sensorPin );  //lectura digital de pin
   if (value == HIGH) {
      Serial.println("Detectado obstaculo");
  }
  delay(1000);
}
```

Magic Light

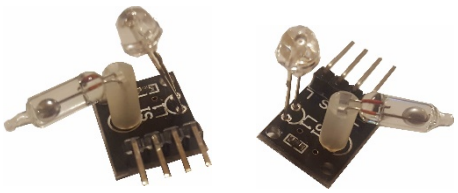


Figura 35. Módulo Magic Light. Información tomada de la Investigación directa. Elaborada por los autores

Descripción

Este sensor es un módulo tiene dos partes un LED y un interruptor de inclinación de mercurio.

Al realizar la conexión se alternará entre un pulso en alto y uno bajo al detectar una inclinación. Si se conecta una resistencia limitadora de corriente, se puede utilizar como un LED parpadeante.

Desarrollo

```
int LedPinA = 5;
int LedPinB = 6;
int ButtonPinA = 7;
int ButtonPinB = 4;
int buttonStateA = 0;
int buttonStateB = 0;
int brightness = 0;
void setup (){
pinMode (LedPinA, OUTPUT);
pinMode (LedPinB, OUTPUT);
pinMode (ButtonPinA, INPUT);
pinMode (ButtonPinB, INPUT);
}
```

```
void loop (){
buttonStateA = digitalRead (ButtonPinA);
if (buttonStateA == HIGH && brightness! = 255)
{
brightness + +;
}
buttonStateB = digitalRead (ButtonPinB);
if (buttonStateB == HIGH && brightness! = 0)
{
brightness -;
}
analogWrite (LedPinA, brightness);
analogWrite (LedPinB, 255 - brightness);
Delay (25);
}
```

3.2. Actuadores Acústicos

Zumbador

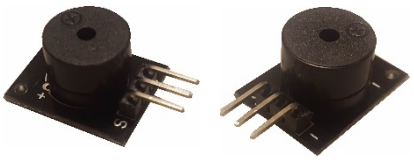


Figura 36. Módulo zumbador. Información tomada de la Investigación directa. Elaborada por los autores

Descripción

Este es un módulo de zumbador activo que puede hacer un sonido diferente para advertencias.

En los proyectos además de utilizar Leds para advertencias, viene bien el módulo zumbador el cual al llegar a un nivel peligroso emite una alarma sonora, la misma que puede ser variada en intensidad, duración e incluso el sonido puede ser personalizado.

Un ejemplo que todos habremos notado es el sonido agudo que se emite en caso de batería baja, dicho sonido le hace salir corriendo a buscar una fuente de energía.

Desarrollo

```
int buzzer = 8 ;// setting controls the digital IO foot buzzer
void setup ()
{
```

```
pinMode (buzzer, OUTPUT) ;// set the digital IO pin mode, OUTPUT out of
Wen
}
void loop ()
{
unsigned char i, j ;// define variables
while (1)
{
for (i = 0; i <80; i++) // Wen a frequency sound
{
digitalWrite (buzzer, HIGH) ;// send voice
delay (1) ;// Delay 1ms
digitalWrite (buzzer, LOW) ;// do not send voice
delay (1) ;// delay ms
}
for (i = 0; i <100; i++) // Wen Qie out another frequency sound
{
digitalWrite (buzzer, HIGH) ;// send voice
delay (2) ;// delay 2ms
digitalWrite (buzzer, LOW) ;// do not send voice
delay (2) ;// delay 2ms
}
}
}
```

3.3. Actuadores Eléctricos

Módulo 5V

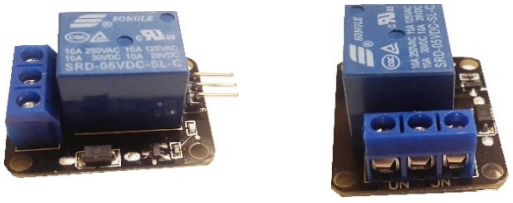


Figura 37. Módulo 5v. Información tomada de la Investigación directa. Elaborada por los autores

Descripción

Este módulo es compatible con cualquiera aplicación de Arduino. Ya que los microcontroladores como el Arduino funcionan con fuente de alimentación de 5V y no pueden controlar componentes de alto voltaje. Las lámparas, abanicos, y demás componentes electrónicos de casa funcionan con la corriente de 120V-240V. Un relé es un interruptor que podemos activar mediante una señal eléctrica. En su versión más simple es

un pequeño electroimán que cuando lo excitamos mueve la posición de un contacto eléctrico de conectado a desconectado o viceversa.
El símbolo del relé muestra la bobina y en este caso, un accionado que conmuta entre dos contactos, pero también existen relés de múltiples contactos. Mediante una señal de control de poca intensidad que excite la bobina podemos conmutar grandes tensiones o intensidades.
Se Caracteriza por:
Cada Relé de 5v requiere una corriente de 20mA.
Activación mediante señal de 5V que puede controlar directamente el microcontrolador.m
Tolerancia: 250v AC / 30v DC 10
Actualmente este sensor es usado para la creación de casas automatizadas ya que tienen buen control de artefactos de bajo voltaje como lo son un ventilador, luces, etc.

Desarrollo

```
int relay = 10; // el relé gira la señal de disparo - activo alto;
void setup ()
{
  pinMode (relay, OUTPUT); // Definir el atributo de puerto se emite;
}
void loop ()
{
  digitalWrite (relay, HIGH); // relé de conducción;
  delay (1000);
  digitalWrite (relay, LOW); // el interruptor de relé está apagado;
  delay (1000);}
```

4. Otros Sensores y Actuadores

EasyVR

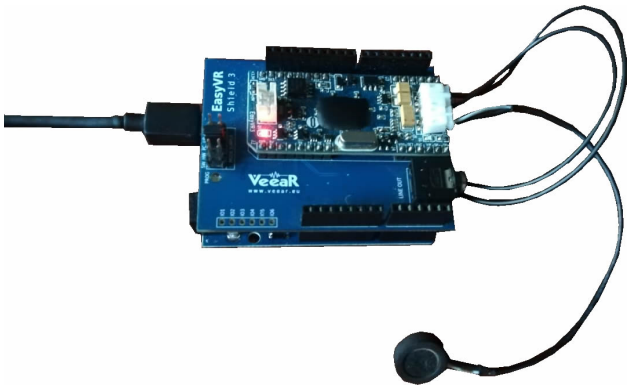


Figura 38. La Easy VR. Información tomada de la Investigación directa. Elaborada por los autores

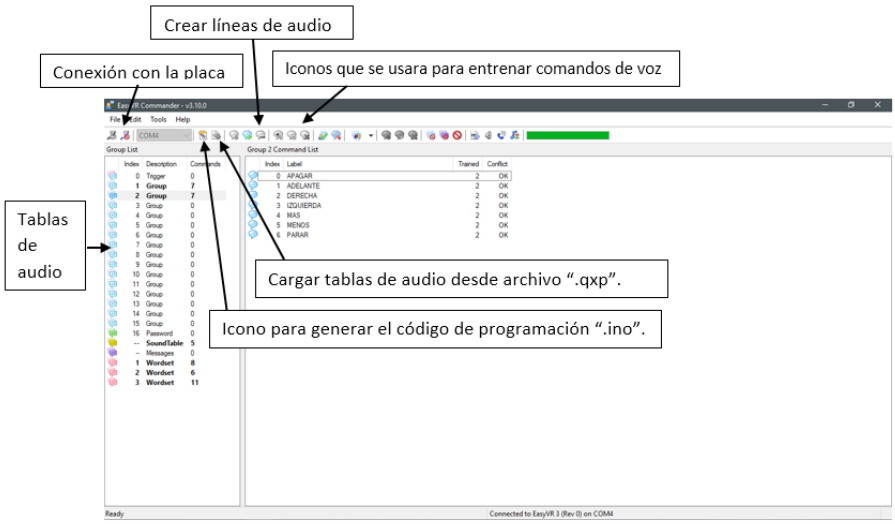


Figura 39. Easy Vr Commander. Información tomada de la Investigación directa. Elaborada por los autores

Descripción

Se usa el programa "EasyVr Commander" para cargar las tablas de audio en la placa EasyVr y generar el algoritmo de programación, en donde se corresponderá a hacer la configuración respectiva ya sea de acciones y otras funciones por medio del programa "Arduino.INO".

Éste tiene la capacidad de reconocer comandos de voz (por medio de un micrófono). Alguien dice una palabra y configurándola con los programas adecuados podemos hacer que ejecute una orden o una acción por medio de nuestra voz. Lo segundo es que también podemos hacer que esta nos responda creando una interacción entre nosotros y la tarjeta, usando

una bocina de 8ohms que se conecta en dos de sus terminales. O si lo preferimos también tiene un puerto para audífonos 3.5mm, por si en el momento no tienes la bocina o por si tu diseño así lo requiere.
Cuando se hace el reconocimiento de voz se utiliza una tarjeta compatible con Arduino para ejecutar el programa. Apoyado de otros dos softwares se puede crear el programa final. Cuando se usan sonidos de respuesta de la tarjeta (o sea que la tarjeta hable y responda) no se usara el Arduino IDE directamente ya que usamos dos programas llamados Easy Commander y QuickSythesis. Básicamente esas son las funciones principales, y aunque parecen muy simples son un arma poderosa a la imaginación de proyectos con reconocimiento de voz.

Desarrollo

```
#include "Arduino.h"
#if !defined(SERIAL_PORT_MONITOR)
  #error "Arduino version not supported. Please update your IDE to the latest version."
#endif
#if defined(SERIAL_PORT_USBVIRTUAL)
  #define port SERIAL_PORT_HARDWARE
  #define pcSerial SERIAL_PORT_USBVIRTUAL
#else
  #include "SoftwareSerial.h"
  SoftwareSerial port(12, 13);
  #define pcSerial SERIAL_PORT_MONITOR
#endif
#include "EasyVR.h"
EasyVR easyvr(port);
enum Groups
{
  GROUP_1  = 1,
  GROUP_2  = 2,
};
enum Group1
{
  G1_ENCENDER_LED_1 = 0,
  G1_ENCENDER_LED_2 = 1,
  G1_ADELANTE_SERVO = 2,
  G1_ATRAS_SERVO = 3,
  G1_APAGAR_LED1 = 4,
```

```
  G1_APAGAR_LED_2 = 5,
};
enum Group2
{
  G2_APAGAR = 0,
  G2_ADELANTE = 1,
  G2_DERECHA = 2,
  G2_IZQUIERDA = 3,
  G2_MAS = 4,
  G2_MENOS = 5,
  G2_PARAR = 6,
};
int8_t group, idx;
void setup()
{
  pcSerial.begin(9600);
  int mode = easyvr.bridgeRequested(pcSerial);
  switch (mode)
  {
  case EasyVR::BRIDGE_NONE:
    port.begin(9600);
    pcSerial.println(F("---"));
    pcSerial.println(F("Puente no iniciado!"));
    break;
     case EasyVR::BRIDGE_NORMAL:
    port.begin(9600);
    easyvr.bridgeLoop(pcSerial);
    pcSerial.println(F("---"));
    pcSerial.println(F("Conexión de puente interrumpida!"));
    break;

  case EasyVR::BRIDGE_BOOT:
    port.begin(115200);
    easyvr.bridgeLoop(pcSerial);
    pcSerial.println(F("---"));
    pcSerial.println(F("Conexión de puente interrumpida!"));
    break;
  }

  while (!easyvr.detect())
```

```
  {
    Serial.println("EasyVR no detectada!");
    delay(1000);
  }

  easyvr.setPinOutput(EasyVR::IO1, LOW);
  Serial.println("EasyVR detectada!");
  easyvr.setTimeout(5);
  easyvr.setLanguage(0);

  group = EasyVR::TRIGGER; //<-- start group (customize)
}
void action();
void loop()
{
  easyvr.setPinOutput(EasyVR::IO1, HIGH);
  Serial.print("Say a command in Group ");
  Serial.println(group);
  easyvr.recognizeCommand(group);

  do {
  }
  while (!easyvr.hasFinished());

  easyvr.setPinOutput(EasyVR::IO1, LOW);
  idx = easyvr.getWord();
  if (idx >= 0)
  {
    return;
  }
  idx = easyvr.getCommand();
  if (idx >= 0)
  {
    uint8_t train = 0;
    char name[32];
    Serial.print("Command: ");
    Serial.print(idx);
    if (easyvr.dumpCommand(group, idx, name, train))
    {
      Serial.print(" = ");
```

```
      Serial.println(name);
    }
    else
      Serial.println();
    easyvr.playSound(0, EasyVR::VOL_FULL);
    action();
  }
  else // errors or timeout
  {
    if (easyvr.isTimeout())
      Serial.println("Timed out, try again...");
    int16_t err = easyvr.getError();
    if (err >= 0)
    {
      Serial.print("Error ");
      Serial.println(err, HEX);
    }
  }
}
void action() {
    switch (group)
    {
   case GROUP_2:
      switch (idx)
      {
      case G2_APAGAR:
commands
        break;
      case G2_ADELANTE:
commands
        break;
      case G2_DERECHA:
commands
        break;
      case G2_IZQUIERDA:
commands
        break;
      case G2_MAS:
commands
        break;
```

```
      case G2_MENOS:
commands
        break;
      case G2_PARAR:
commands
        break;
      }
      break;
    }
}
```

5. Desarrollo de Proyectos

Las realizaciones de proyectos utilizando Arduino con diferentes sensores y módulos ayuda al usuario a experimentar otro universo en el ámbito tecnológico. Los siguientes ejemplos te mostrará la creación de proyectos tecnológicos que nunca imaginaste hacer por pensar es su dificultad.

5.1. Alcoholímetro

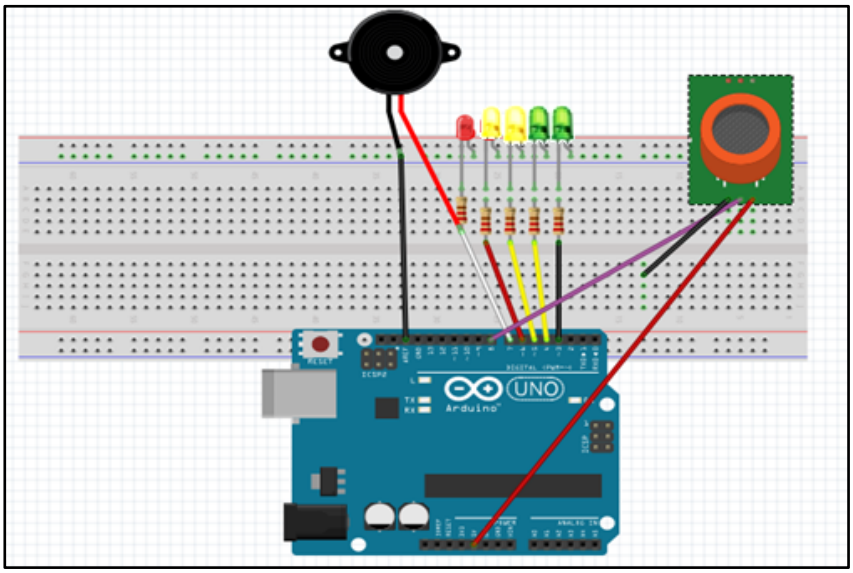


Figura 40. Diseño de conexión alcoholímetro. Información tomada de la Investigación directa. Elaborada por los autores

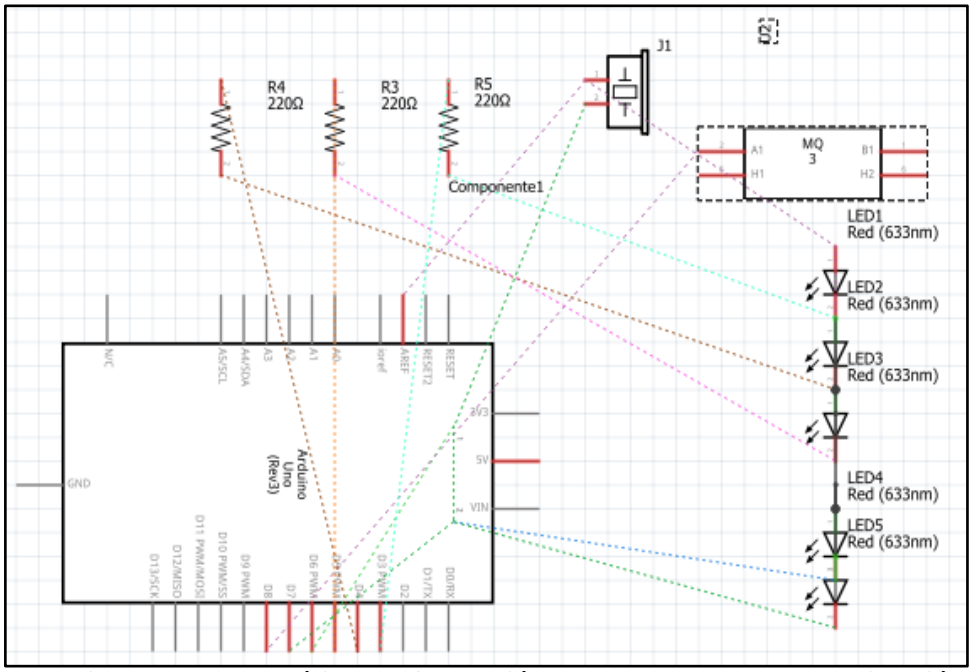


Figura 41. Sistema Esquemático. Información tomada de la Investigación directa. Elaborada por los autores

Materiales:

1. 2 led verdes.
2. 1 led naranja.
3. 1 led rojo.
4. 5 resistores de 1KΩ.
5. Arduino mini pro.
6. Cables Dupont (Macho-Hembra).
7. Sensor de Alcohol MQ3.
8. Buzzer.
9. Protoboard.

Desarrollo

```
int sensor=0;
void setup(){
Serial.begin(115200);
pinMode(2, OUTPUT);
 pinMode(3, OUTPUT);
 pinMode(4, OUTPUT);
 pinMode(5, OUTPUT);
 pinMode(6, OUTPUT);
 }
 void loop(){
sensor=analogRead(A0);
Serial.println(sensor);
if(sensor<99){
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
  digitalWrite(6, LOW);
   }
if(sensor>100){
  digitalWrite(2, HIGH);
  digitalWrite(3, LOW);
```

```
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
  digitalWrite(6, LOW);
   }
if(sensor>200){
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
  digitalWrite(6, LOW);
   }
if(sensor>300){
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(4, HIGH);
  digitalWrite(5, LOW);
  digitalWrite(6, LOW);
   }
if(sensor>350){
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(4, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(6, LOW);
   }
if(sensor>400){
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(4, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(6, HIGH);
```

```
 }
delay(100);
 }
```

Detalle del Proyecto

Es un instrumento usado para determinar el nivel de alcohol que se halla presente en un líquido o gas. Puede usarse, por tanto, para medir el porcentaje de alcohol en una bebida alcohólica o para determinar la presencia de alcohol en la sangre o en un gas.

Para este proyecto se usa el sensor MQ3, el cuál es muy sensible al alcohol y de menor sensibilidad a la bencina, también es sensible a gases como GLP, Hexano, CO, CH4 pero con sensibilidad muy baja, la cual se puede despreciar si hay poca concentración de estos.

Luego de que este percibe la presencia de alcohol o gas, previamente pasada la programación a la placa de ARDUINO, se encenderán los LED's de manera ascendente de verde a rojo.

5.2. Bastón para no videntes utilizando Sensores de Rango

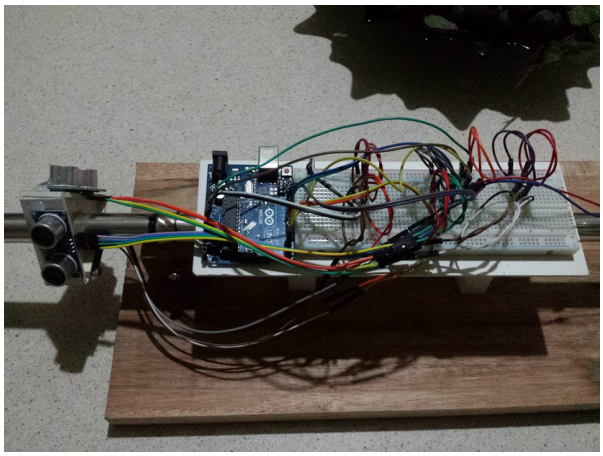


Figura 42. Estructura General del bastón para no videntes. Información tomada de la Investigación directa. Elaborada por los autores

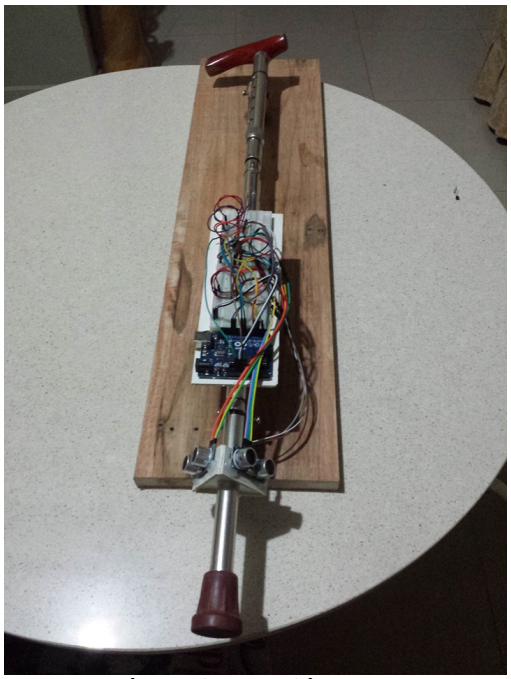


Figura 43. Vista superior del bastón. Información tomada de la Investigación directa. Elaborada por los autores

Materiales:

1. Protoboard o Breadbord
2. Cables puente
3. Base de madera
4. Sensores de rango
5. Placa de Arduino
6. Bastón
7. Vibrador de celular

Desarrollo

```
int trigerA =13;
int echoA =12;
int trigerB =9;
int echoB =8;
int trigerC =5;
int echoC =4;
int vibration=2;
void setup()
{
 pinMode(trigerA,OUTPUT);
pinMode(trigerB,OUTPUT);
pinMode(trigerC,OUTPUT);
pinMode(echoA,INPUT);
pinMode(echoB,INPUT);
pinMode(echoC,INPUT);
pinMode (2,OUTPUT);
Serial.begin(9600);
}
void loop()
{
long duracionA;
long distanciaA;
long duracionB;
long distanciaB;
long duracionC;
long distanciaC;
  //apagado y encendido sensor A
  digitalWrite(trigerA,LOW);
  delayMicroseconds(4);
  digitalWrite(trigerA,HIGH);
  delayMicroseconds(10);
digitalWrite(trigerA,LOW);
duracionA=pulseIn(echoA,HIGH);
duracionA=duracionA/2;
distanciaA=duracionA/29;
Serial.print(distanciaA);
Serial.println("cmA");
  //apagado y encendido sensor B
digitalWrite(trigerB,LOW);
```

```
delayMicroseconds(4);
digitalWrite(trigerB,HIGH);
delayMicroseconds(10);
digitalWrite(trigerB,LOW);
duracionB=pulseIn(echoB,HIGH);
duracionB=duracionB/2;
distanciaB=duracionB/29;
Serial.print(distanciaB);
Serial.println("cmB");
  //apagado y encedido sensor C
 digitalWrite(trigerC,LOW);
 delayMicroseconds(4);
 digitalWrite(trigerC,HIGH);
 delayMicroseconds(10);
 digitalWrite(trigerC,LOW);
duracionC=pulseIn(echoC,HIGH);
duracionC=duracionC/2;
distanciaC=duracionC/29;
Serial.print(distanciaC);
Serial.println("cmC");
if( distanciaA <=50 || distanciaB <=50 || distanciaC <=50)
  digitalWrite(2,0);
{
Serial.println("");
delay(500);
}
```

Detalle del Proyecto

El proyecto se basa en la utilización de sensores de rango (HC- SR04) los cuales al detectar interferencias en las señales de ondas que este emite empieza a detectar lo como un obstáculo además se puede determinar un rango y se alto o bajo para la detección de dicho objeto. Por esa razón se decidió incorporarlo en un bastón para personas con discapacidad visual, con el objetivo de que dichas personas con discapacidad puedan tener un ayuda extra al momento de caminar en zonas pobladas así los sensores en el bastón le avisaran si cerca de él se encuentran objetos con los cuales el sujeto pueda colisionar y tener algún tipo de accidente.

Además, se le incorporo un vibrador esto con el fin de que si el sujeto además de sufrir discapacidad visual también tenga discapacidad auditiva

pueda recibir la alerta de los sensores median te una vibración y que esta no cese hasta que ya el obstáculo no esté cerca de la persona.

5.3. Burbujero con Arduino

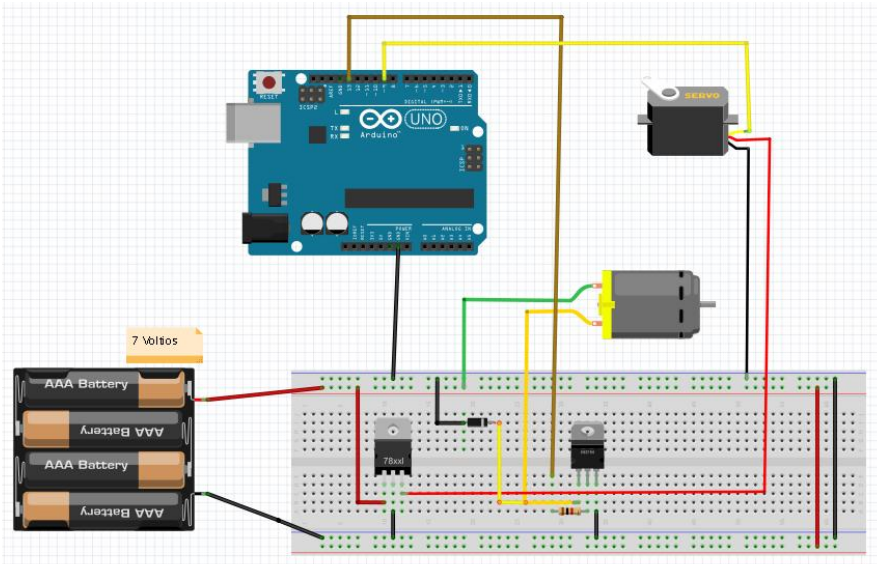


Figura 44. Circuito Burbujero. Información tomada de la Investigación directa. Elaborada por los autores

Materiales:

1. Transistor NPN
2. Regulador de tensión 7805
3. Resistor 1K al 5 porciento
4. Diodo 1N4004
5. Motor Corriente continua
6. Servo sg90
7. Arduino UNO
8. Protoboard
9. Madera, plástico o aluminio.
10. Tronillos varios
11. Cables

Desarrollo

```
#include <Servo.h>
 Servo myservo;
 // Se crea el objeto myservo para controlar el servomotor
//Declaración de variables
int led = 13;
```

```
int ret=15;
void setup ()
{
  pinMode (led, OUTPUT);
// El pin número 13 del arduino es seteado como salida para el control del
motor C.C.
  myservo.attach (9);
// Se usa el pin número 9 para el control del servo
}
 void loop ()
{
// Ciclo for la variable i inicia con 100 que será la posición inicial del servo y
terminará en 155 como posición final
for (int i=100; i <= 155; i++)
{
 myservo.write(i);
// Se va asignando el valor de la variable i para indicar al servo su posición
actual
delay (50);
 //Pequeño retardo para el brazo, para que el servo no suba bruscamente
y no tire la solución jabonosa del burbujero
 }
digitalWrite (led, HIGH);
//Activa el motor de corriente continua
 delay (3000);
// Por 3 segundos
 digitalWrite (led, LOW);
// Luego lo apaga
 myservo.write (100);
// Lleva al servo a la posición inicial donde está la solución jabonosa
delay (2000);
// Durante dos segundos
}
```

Detalle del Proyecto

Lo que realiza éste trabajo es que al momento de encenderlo el brazo de madera bajará al envase donde está el líquido recogiendo un poco de aquello, cuando suba el motor se encenderá dando movimiento a la hélice y soplara para producir burbujas.

5.4. Dispositivo de seguridad de una caja fuerte

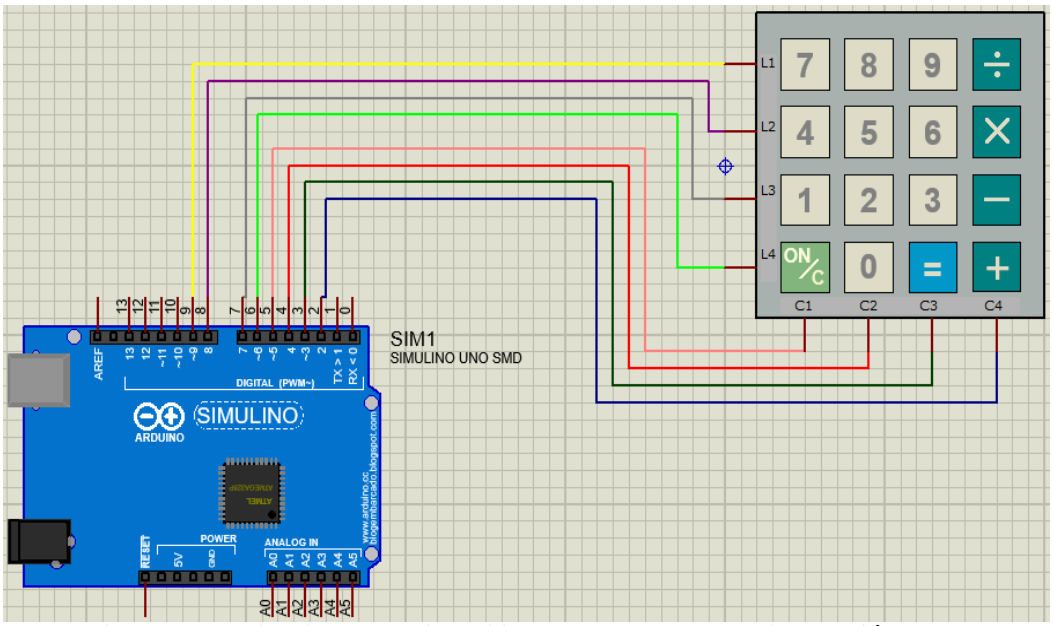


Figura 45. Diseño del circuito del dispositivo con teclado. Información tomada de la Investigación directa. Elaborada por los autores

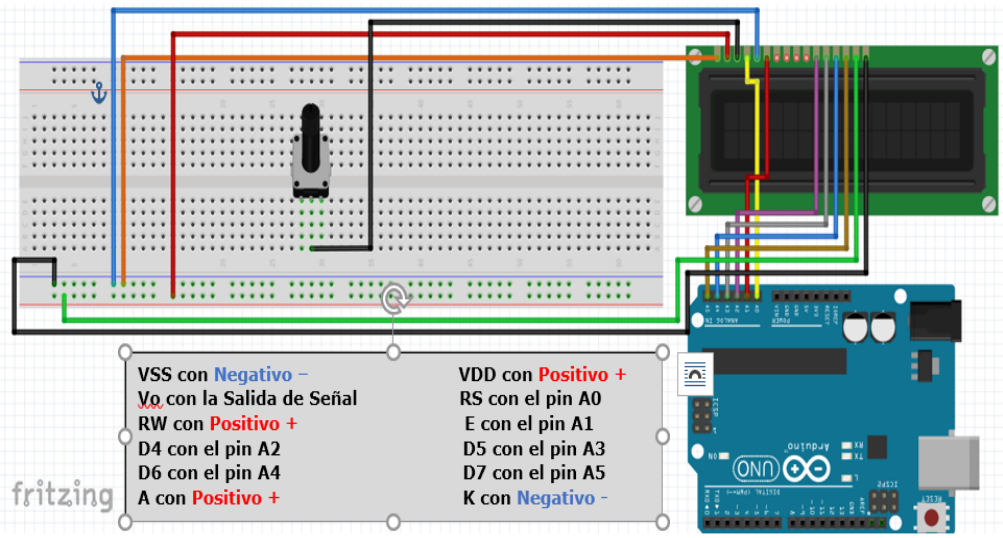


Figura 46. Diseño del circuito del dispositivo con Display. Información tomada de la Investigación directa. Elaborada por los autores

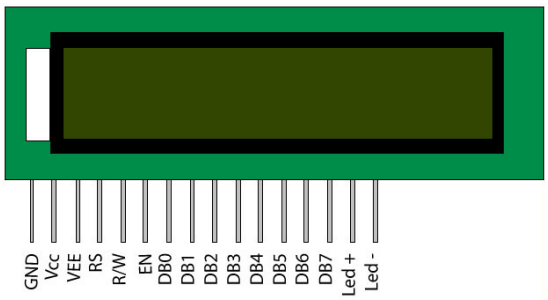


Figura 47. Display LCD, de 16×2. Información tomada de la Investigación directa. Elaborada por los autores

Tabla 2. Funciones del Display LCD, de 16×2

	Vss	Fuente de Alimentación	Tierra de señal para LCM.
	Vdd	Fuente de Alimentación	Fuente de alimentación para LCM lógica
	Vo	Fuente de Alimentación	Ajuste de contraste.
	RS	MPU	Registrar la señal de selección.
	R/W	MPU	Señal de selección de lectura / escritura.
	E	MPU	(Lectura / escritura de datos).
	DB0-DB3	MPU	Cuatro líneas bidireccionales bi direccionales de bus de datos de tres estados. Se utiliza para la transferencia de datos entre la MPU y la LCM. Estos cuatro no se utilizan durante la operación de 4 bits.
	DB4-DB7	MPU	Cuatro líneas bidireccionales bi direccionales de bus de datos de tres estados. Se utiliza para la transferencia de datos entre la MPU
	LED+	Fuente de Alimentación LED BKL	Fuente de alimentación para BKL
	LED-	Fuente de Alimentación LED BKL	Fuente de alimentación para BKL

Información tomada de la Investigación directa. Elaborada por los autores

Materiales:

1. Arduino UNO R3
2. Protoboard más cables macho macho
3. Teclado Matricial 4×4
4. Display LCD, DE 16×2
5. Potenciómetro
6. Servomotor

Desarrollo

```
#include <Keypad.h> // Controla el teclado
#include <LiquidCrystal.h> //controla el LCD
```

```
#include <Password.h> //Control de la Contraseña
#include <Servo.h> //Control del servomotor
#include <EEPROM.h> //Controla E/S EEPROM
#define CERRAR 90 //Calibrar a gusto la pos del servomotor cerrado
#define ABRIR 180 //Calibrar a gusto la pos del servomotor abierto
//****************************************
//*** Declaracion de variables locales*****
//****************************************
Servo seguro;  //servomotor
LiquidCrystal lcd (A0, A1, A2, A3, A4, A5); //display LCD
const byte filas = 4;
const byte columnas = 4;
byte pinsFilas[filas] = {9, 8, 7, 6};
byte pinsColumnas[columnas] = {5, 4, 3, 2};
char teclas[filas][columnas] = {  // Declaración del teclado
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'},
};
Keypad teclado = Keypad(makeKeymap(teclas), pinsFilas, pinsColumnas,
filas, columnas);
char password[5]; //Almacena la Contraseña en EEPROM
char ingreso; //Ingreso del Usuario
char passUser[4];
char confirmPass[4];  //Confirmacion de la Contraseña
//char ca[7] = {'1', '9', '9', '7', '2', 'A'}; //Clave Admin Establecida
//char cal[7];   // Clave admin leida
//int contador = 0; //Lleva la Posicion del Array
//int cursorr = 6; //Lleva la Posicion del Cursor
//int comp;  // Comparacion Entre 2 Arrays
int i = 0;
int a; //aux
void setup() {
  pinMode(12, OUTPUT); // LEDS QUE INDICAN ABIERTO(12) O CERRADO(11)
  pinMode(11, OUTPUT);
  digitalWrite(11, HIGH); //Enciende el Led de Cerrado
  seguro.attach(13);  // Pin del servomotor
  seguro.write(CERRAR);
  lcd.begin(16, 2);   //Configuracion lcd 16X2 (columnas,fila)
```

```
  seguro.write(90);  //Cerrar puerta
  lcd.setCursor(0, 0);
  lcd.print("  **PASSWORD**");
  lcd.setCursor(5, 1);
  lcd.print(" ____");
  //LEER CONTRASEÑA DE LA EEPROM
  //error if(password[4]!='Z'){
  //correción
  if (EEPROM.read(4) != 'Z') {
      EEPROM.write(0, '1');
      EEPROM.write(1, '2');
      EEPROM.write(2, '3');
      EEPROM.write(3, '4');
      EEPROM.write(4, 'Z');
  }
  for (int i = 0; i <= 4; i++) {
    password[i] = EEPROM.read(i);
  }
}//fin del setup
void loop() {
  leerIngreso(1);
}
////////////////////////////////////////
void leerIngreso(int a) {
  ingreso = teclado.getKey();
  if (ingreso != NO_KEY)
    switch (ingreso) {
      case 'A':    // es como el "enter" para introducir la password
        if (evaluar(1) == 1)
          correcto();
        else {
          msgError();
        }
        reset();
        break;
      case 'B':
        informacion();  //muestra en el lcd las opciones de la caja fuerte
        reset();
        break;
      case 'C':
```

```
        cambioPass();
        reset();
        break;
      case 'D':
        lcd.setCursor(0, 0);
        lcd.print("   CERRAR -_-  ");
        delay(1500);
        reset();
        digitalWrite(12, LOW);  //Apaga el Led de Verde
        digitalWrite(11, HIGH); //Enciende el Led de Rojo
        seguro.write(CERRAR);
        break;
      case '*':
        lcd.setCursor(0, 0);
        lcd.print("Presionaste *");
        delay(900);
        reset();
        break;
      case '#':
        lcd.setCursor(0, 0);
        lcd.print("Presionaste #");
        delay(900);
        reset();
        break;
      default: //Si es un Sumero debe Imprimirlo en el LCD y Ademas
Guardarlo en el Arreglo PassUser
        if (a == 1) {
          passUser[i] = ingreso;
          printPass(passUser[i], 5 + i, 1);
        }
        if (a == 2) {
          confirmPass[i] = ingreso;
          printPass(confirmPass[i], 5 + i, 1);
        }
        i++;
        if (i > 3)
          i = 0;

    }
}
```

```
void cleanlcd() {
  lcd.setCursor(0, 0);
  lcd.print("                ");
  lcd.setCursor(0, 1);
  lcd.print("                ");
}
void printPass(char a, int columna, int fila ) {
  lcd.setCursor(columna, fila);
  lcd.print(a);
  delay(100);
  lcd.setCursor(columna, fila);
  lcd.print("*");
}
int evaluar(int a) {
  int j = 0;
  if (a == 1) {
    for (int i = 0; i <= 3; i++) {
      if (password[i] == passUser[i]) {
        j++;
      }
    }
  }
  if (a == 2) {
    for (int i = 0; i <= 3; i++) {
      if (passUser[i] == confirmPass[i]) {
        j++;
      }
    }
  }
  if (j == 4) {
    return j = 1;
  }
  else {
    return j = 0;
  }
}
void reset() {
  lcd.setCursor(0, 0);
  lcd.print("  **PASSWORD**");
  lcd.setCursor(5, 1);
```

```
  lcd.print("____");
  for (int i = 0; i <= 3; i++) {
    passUser[i] = NO_KEY;
    confirmPass[i] = NO_KEY;
  }
  i = 0;
}
void msgError() {
  lcd.setCursor(0, 0);
  lcd.print("      ERROR      ");
  delay(500);
}
void correcto() {
  digitalWrite(12, HIGH);
  digitalWrite(11, LOW);
  lcd.setCursor(0, 0);
  lcd.print("   CORRECTO :)      ");
  //Abrir servomotor
  seguro.write(ABRIR);
  delay(1500);
}
void informacion() {
  lcd.setCursor(0, 0);
  lcd.print("'A' para introdu");
  delay(200);
  lcd.setCursor(0, 0);
  lcd.print("cir la pass        ");
  delay(200);
  lcd.setCursor(0, 0);
  lcd.print("'C' para cambiar");
  delay(200);
  lcd.setCursor(0, 0);
  lcd.print(" la pass             ");
  delay(200);
}
void cambioPass() {
  lcd.setCursor(0, 0);
  lcd.print("Cambio de pass    ");
  delay(200);
  lcd.setCursor(0, 0);
```

```
lcd.print("Introduce pass     ");
delay(200);
lcd.setCursor(0, 0);
lcd.print("anterior        ");
delay(200);
reset();
while (passUser[3] == NO_KEY) {
  leerIngreso(1);
}
if (evaluar(1) == 1) {
  lcd.setCursor(0, 0);
  lcd.print("Introduce           ");
  delay(200);
  lcd.setCursor(0, 0);
  lcd.print("la pass nueva           ");
  delay(200);
  reset();
  while (passUser[3] == NO_KEY) {
    leerIngreso(1);
  }
  lcd.setCursor(0, 0);
  lcd.print("Vuelve a intro     ");
  delay(200);
  lcd.setCursor(0, 0);
  lcd.print("cirla           ");
  delay(200);
  lcd.setCursor(0, 0);
  lcd.print("  **PASSWORD**");
  lcd.setCursor(5, 1);
  lcd.print("____");
  lcd.setCursor(0, 0);
  i = 0;
  while (confirmPass[3] == NO_KEY) {
    leerIngreso(2);
  }
  if (evaluar(2) == 1) {
    // función de EEPROM
    for (int i = 0; i <= 3; i++) {
      EEPROM.write(i, passUser[i]);
    }
```

```
      for (int i = 0; i <= 3; i++) {
        password[i] = EEPROM.read(i);
      }
      lcd.setCursor(0, 0);
      lcd.print("Contraseña cam  ");
      delay(200);
      lcd.setCursor(0, 0);
      lcd.print("biada          ");
      delay(200);
    }
    else {
      lcd.setCursor(0, 0);
      lcd.print("Error las pass   ");
      delay(200);
      lcd.setCursor(0, 0);
      lcd.print("no coinciden       ");
      delay(200);
    }
  }
  else {
    msgError();
  }
  reset();
}
void passChange() {
  for (int i = 0; i <= 3; i++) {
    password[i] = passUser[i];
  }
}
```

Detalle del Proyecto

Este Trabajo me permite abrir un dispositivo por medio de claves que se ingresa por medio de un teclado, para ello fue necesario utilizar lo que son las librerías Keypad que son del teclado matricial, LiquiCrystal para el LCD, Password para la contraseña, Servo para controlar el Servo motor.

5.5. Carro inalámbrico controlado por bluetooth, integrando Arduino

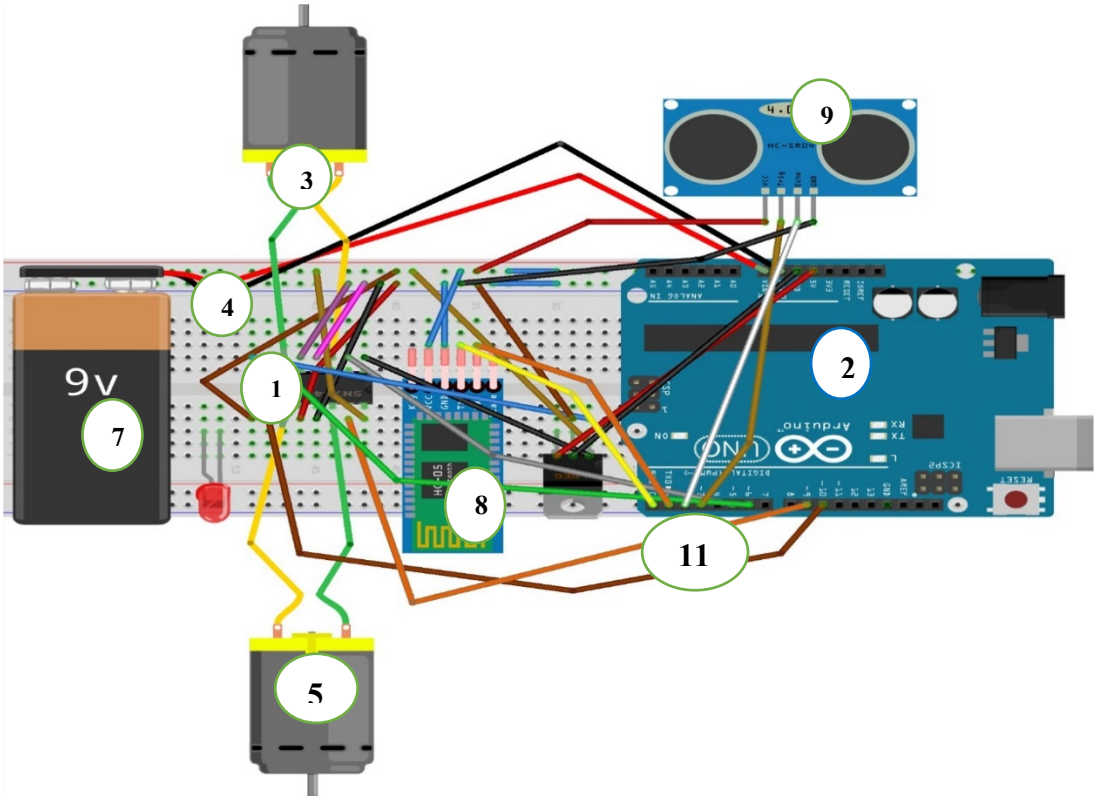


Figura 48. Estructura General. Información tomada de la Investigación directa. Elaborada por los autores

Figura 49. Diseño de App Ara control bluetooth. Información tomada de la Investigación directa. Elaborada por los autores

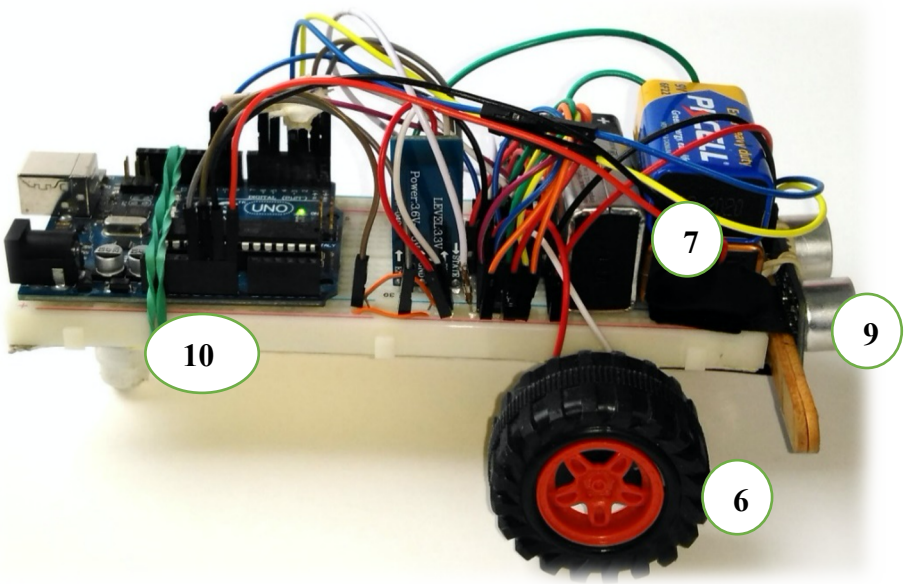


Figura 50. Vista lateral del prototipo. Información tomada de la Investigación directa. Elaborada por los autores

Materiales:

1. Puente H L293D. (controlador de los 2 Motorreductores)
2. Arduino UNO R3. (Se conectan los pines con los que se distribuirá los voltajes requeridos para el funcionamiento del carro)
3. Cables de conexión. (Usados para conectar desde la batería al Arduino)
4. Protoboard. (Placa en la que realizamos todas las conexiones requeridas)
5. 2 motores reductores 1,5Kg de fuerza, 100RPM 3-12Vdc. (Pequeños motores con los cuales controlamos el movimiento de las llantas)
6. 2 llantas para Motor reductores. (Unidas a los Motorreductores para que el carro se mueva)
7. Batería 9Vdc. (Envía la energía a todo el circuito)
8. Modulo Bluetooth HC06. (Utilizado para controlarlo con la aplicación de celular)
9. Sensor Ultrasónico. (Es el encargado de la detección de objetos que se acercan al carro)
10. Rueda Loca. (Usada para darle más movilidad al carro)
11. 20 cables de conexión jumper. (Son utilizados para realizar las conexiones de todo el circuito)

Desarrollo

```
int izqA = 5;
```

```
int izqB = 6;
Int derA = 9;
int derB = 10;
Int vel = 255;            // Velocidad de los motores (0-255)
Int estado = 'g';         // inicia detenido
Int pecho = 2;            // define el pin 2 como (pecho) para el Ultrasonido
Int ptrig = 3;            // define el pin 3 como (ptrig) para el Ultrasonido
Int duración, distancia;  // para Calcular distancia
Void setup () {
  Serial. Begin (9600);    // inicia el puerto serial para comunicación con el
Bluetooth
  PinMode (derA, OUTPUT); pinMode (derB, OUTPUT);
  PinMode (izqA, OUTPUT);  pinMode (izqB, OUTPUT);

  PinMode (pecho, INPUT);   // define el pin 2 como entrada (pecho)
  PinMode (ptrig, OUTPUT);   // define el pin 3 como salida (ptrig)
  PinMode (13, OUTPUT);
}
Void look () {
  If (Serial.available ()>0) {        estado = Serial.read ();
  }
  If (estado=='a') {          // Botón desplazar al Frente
    AnalogWrite (derB, 0);      analogWrite (izqB, 0);
    AnalogWrite (derA, vel);   analogWrite (izqA, vel);
  }
  If (estado=='b') {          // Botón IZQ
    AnalogWrite (derB, 0); analogWrite (izqB, 0);
    AnalogWrite (derA, 0); analogWrite (izqA, vel);
  }
  If (estado=='c') {        // Botón Parar
    AnalogWrite (derB, 0); analogWrite (izqB, 0);
    AnalogWrite (derA, 0); analogWrite (izqA, 0);
  }
  If (estado=='d') {          // Botón DER
    AnalogWrite (derB, 0); analogWrite (izqB, 0);
    AnalogWrite (izqA, 0); analogWrite (derA, vel);
  }
  If (estado=='e') {          // Botón Reversa
    AnalogWrite (derA, 0); analogWrite (izqA, 0);
    AnalogWrite (derB, vel); analogWrite (izqB, vel);
```

```
}
 If (estado =='f') {         // Botón ON, se mueve censando distancia
   DigitalWrite (ptrig, HIGH);       delay (0.01);
   DigitalWrite (ptrig, LOW);   duración = pulseIn (pecho, HIGH);
   distancia = (duración/2) / 29;              delay (10);
     If (distancia <= 15 && distancia >=2) {     DigitalWrite (13, HIGH);
     AnalogWrite (derB, 0); analogWrite (izqB, 0);
     AnalogWrite (derA, 0); analogWrite (izqA, 0);
     delay (200);
     AnalogWrite (derB, vel);   analogWrite (izqB, vel);
     delay (500);
     AnalogWrite (derB, 0);    analogWrite (izqB, 0);
     AnalogWrite (derA, 0);     analogWrite (izqA, vel);
     delay (1100);       digitalWrite (13, LOW);
   }
   Else {                // Si no hay obstáculos se desplaza al frente
      AnalogWrite (derB, 0);    analogWrite (izqB, 0);
      AnalogWrite (derA, vel);   analogWrite (izqA, vel);
   }
 }
 If (estado=='g') {         // Botón OFF, detiene los motores no hace nada
   AnalogWrite (derB, 0);   analogWrite (izqB, 0);
   AnalogWrite (derA, 0);   analogWrite (izqA, 0);
 }
}
```

Detalle del Proyecto

Éste trabajo está basado en el estudio de los sensores de Arduino, tal como es el caso del módulo ultrasónico el cual hemos implementado en nuestro prototipo con el fin de buscar que nuestro carro detecte los objetos que se acerquen a él y reaccione con un movimiento que se definirá en la codificación.

De igual forma cuenta con otra funcionalidad que ofrece el módulo bluetooth hc06 implementado en el mismo, el cual nos permite tener el control sobre los movimientos del carro sin necesidad de una conexión directa de cables, por medio de una aplicación que fue mostrado antes (FIGURA 40), de esta manera el usuario podría contar con dos modos de uso para el auto presentados.

5.6. Comprobador de pilas con Arduino

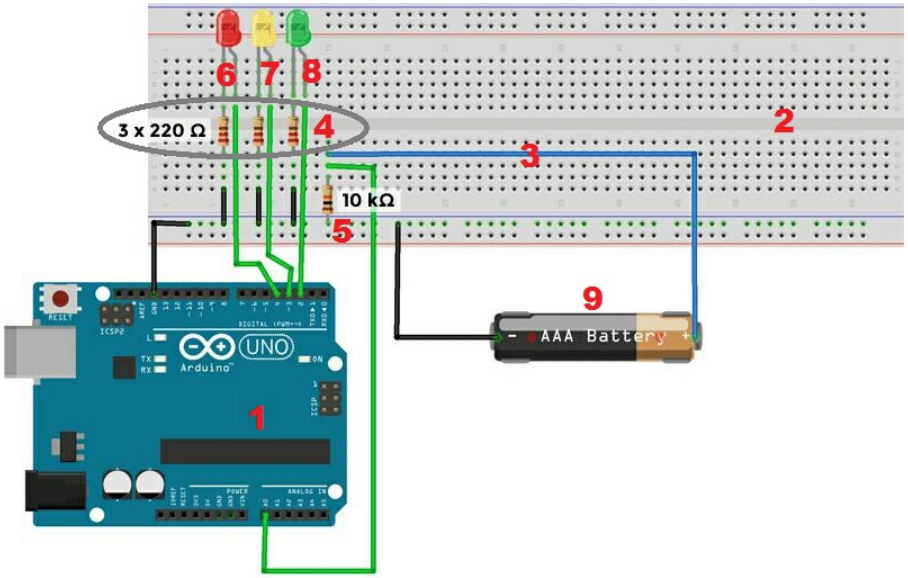


Figura 51. Diseño del comprobador Información tomada de la Investigación directa. Elaborada por los autores

Materiales:

1. Arduino UNO o cualquier placa de Arduino
2. Protoboard donde conectaremos los componentes
3. Cables macho -hembra
4. 3 resistencias de 220 Ω
5. 1 resistencia de 10 kΩ
6. 1 LED rojo de 5 mm
7. 1 LED amarillo de 5 mm
8. 1 LED verde de 5mm
9. Pilas AAA - AA

Desarrollo

```
// Pines para los LEDs
#define LEDVERDE 2
#define LEDAMARILLO 3
#define LEDROJO 4
#define ANALOGPILA 0
 // Variables
int analogValor = 0;
float voltaje = 0;
int ledDelay = 800;
```

```
 // Umbrales
float maximo = 1.6;
float medio = 1.4;
float minimo = 0.3;
 void setup() {
  // Iniciamos el monitor serie
  Serial.begin(9600);
   // Los pines de LED en modo salida
  pinMode(LEDVERDE, OUTPUT);
  pinMode(LEDAMARILLO, OUTPUT);
  pinMode(LEDROJO, OUTPUT);
 }
 void loop() {
  // Leemos valor de la entrada analógica
  analogValor = analogRead(ANALOGPILA);
   // Obtenemos el voltaje
  voltaje = 0.0048 * analogValor;
  Serial.print("Voltaje: ");
  Serial.println(voltaje);
   // Dependiendo del voltaje mostramos un LED u otro
  if (voltaje >= maximo)
  {
    digitalWrite(LEDVERDE, HIGH);
    delay(ledDelay);
    digitalWrite(LEDVERDE, LOW);
  }
  else if (voltaje < maximo && voltaje > medio)
  {
    digitalWrite(LEDAMARILLO, HIGH);
    delay(ledDelay);
    digitalWrite(LEDAMARILLO, LOW);
  }
  else if (voltaje < medio && voltaje > minimo)
  {
    digitalWrite(LEDROJO, HIGH);
    delay(ledDelay);
    digitalWrite(LEDROJO, LOW);
  }
   // Apagamos todos los LEDs
  digitalWrite(LEDVERDE, LOW);
```

```
    digitalWrite(LEDAMARILLO, LOW);
    digitalWrite(LEDROJO, LOW);
  }
```

Detalle del Proyecto

Se utiliza la Placa Arduino para leer a través de una entrada analógica el voltaje que suministra una pila. Dependiendo de este voltaje, encenderemos un LED de un color. Si la pila está nueva, se encenderá un LED verde. Si la pila no es nueva, pero se ha consumido parte de su energía encenderemos un LED amarillo. Por último, si la pila está gastada o no suministra el suficiente voltaje, encenderemos un LED rojo.

Debemos de tener mucho cuidado con el tipo de pila y de batería que vamos a medir. Es muy peligroso suministrar más de 5V a los pines analógicos de Arduino. Si lo que queremos es medir pilas, lo más típico, solo podremos hacerlo con pilas AA, AAA, C y D. Son las que se utilizan en los mandos de la televisión, en juguetes e incluso para alimentar Arduino. Hay que tener la precaución con una pila de 9V, que por lo general son cuadradas. Como bien he dicho antes, estas pilas superan con creces el límite de 5V. También debemos de llevar cuidado con las baterías ya que dependerá del voltaje que suministren. Comprueba antes de conectar que realmente es menor o igual que 5V.

5.7. Contador de objetos de entrada y salida

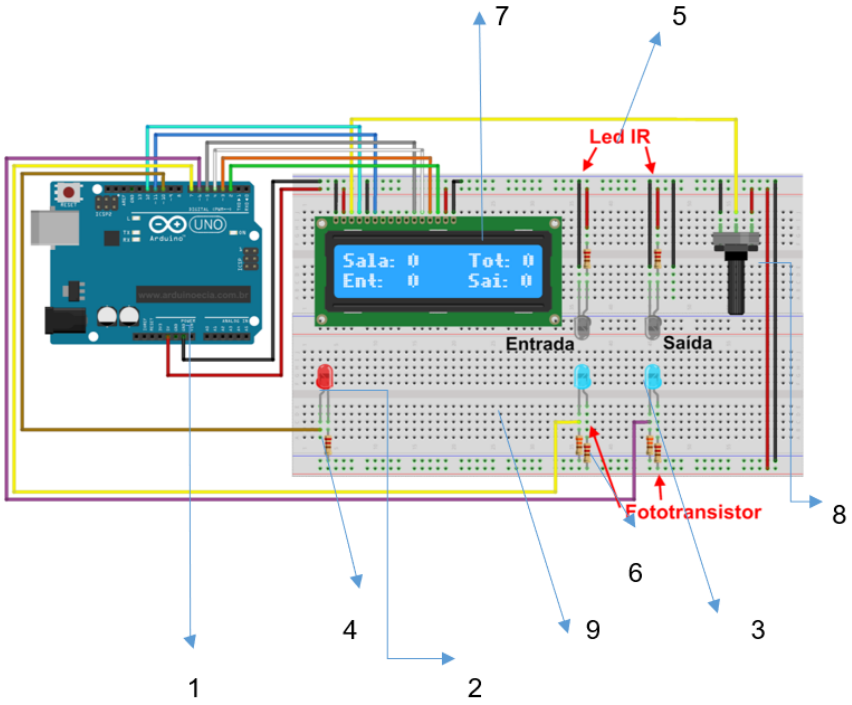


Figura 52. Diseño de Contador de objeto. Información tomada de la Investigación directa. Elaborada por los autores

Materiales:

1. Arduino uno.
2. Led común (para simular la luz de la habitación).
3. Fototransistores 5mm.
4. Resistor de 220R para el led común.
5. Resistencias de 220R para los leds IR.
6. Resistencias de 330R para el emisor del fototransistor.
7. Pantalla LCD 16x2 HD44780.
8. Potenciómetro de 10 K.
9. Protoboard.

Desarrollo

```
int pinoirent = 7;
int pinoirsaida = 6;
int valorirent = 0;
int valorirsai = 0;
int content = 0;
int contsai = 0;
int contsala = 0;
```

```
int contanterior = 0;
int pinoledsala = 10;

#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
 void setup()
{
  Serial.begin(9600);
  pinMode(pinoirent, INPUT);
  pinMode(pinoirsaida, INPUT);
  pinMode(pinoledsala, OUTPUT);
  lcd.begin(16, 2);
  lcd.setCursor(0,0);
  lcd.print("Sala:0 ");
  lcd.setCursor(9,0);
  lcd.print("Tot: 0");
  lcd.setCursor(0,1);
  lcd.print("Ent: 0");
  lcd.setCursor(9,1);
  lcd.print("Sai: 0");
}
void loop()
{
  valorirent = digitalRead(pinoirent);
  valorirsai = digitalRead(pinoirsaida);
  Serial.print("entrada : ");
  Serial.print(valorirent);
  Serial.print(" saida : ");
  Serial.println(valorirsai);
  if(valorirent!=1)
  {
    content=content+1;
    while(digitalRead(pinoirent)!=1)
     {
       delay(100);
       Serial.println("Sensor de entrada parado !");
     }
    lcd.setCursor(5,1);
    lcd.print(content);
    lcd.setCursor(14,0);
```

```
      lcd.print(content);
    }
    if(valorirsai!=1)
    {
      contsai=contsai+1;
      while(digitalRead(pinoirsaida)!=1)
        {
         delay(100);
         Serial.println("Sensor de saida parado !");
        }
      lcd.setCursor(14,1);
      lcd.print(contsai);
    }
    contsala=content-contsai;
    if (contsala != contanterior)
    {
      lcd.setCursor(5,0);
      lcd.print("  ");
      lcd.setCursor(5,0);
      lcd.print(contsala);
      contanterior = contsala;
      if (contsala > 0)
      {
        digitalWrite(pinoledsala,1);
      }
        else
      {
        digitalWrite(pinoledsala,0);
      }
    }
  }
```

Detalle del Proyecto

En el circuito anteriormente, cada fototransistor (en azul) tienen la pierna más corta (el colector) conectado al positivo junto con un resistor 220R y la pata más larga (el emisor) conectado a la puerta Arduino, con una resistencia de 330R decisiones la función de pull-down. El potenciómetro ajusta el contraste de la pantalla.

El programa comprueba el estado de las puertas 7 (sensor sala de entrada) y 6 (sensor de salida de la habitación) que están en un alto nivel

durante la recepción de iluminación de IR LED (que siempre están conectados). Cuando la iluminación IR es interrumpida por algún objeto, el estado de la puerta va a nivel bajo (0 / LOW), y el contador se actualiza.

En la parte superior del display, mostramos la cantidad de personas en la sala, y el total de visitantes. En la parte inferior del display, tenemos el contador de entrada y de salida, sólo para información y acompañamiento.

5.8. Control de Acceso mediante medios Magnéticos

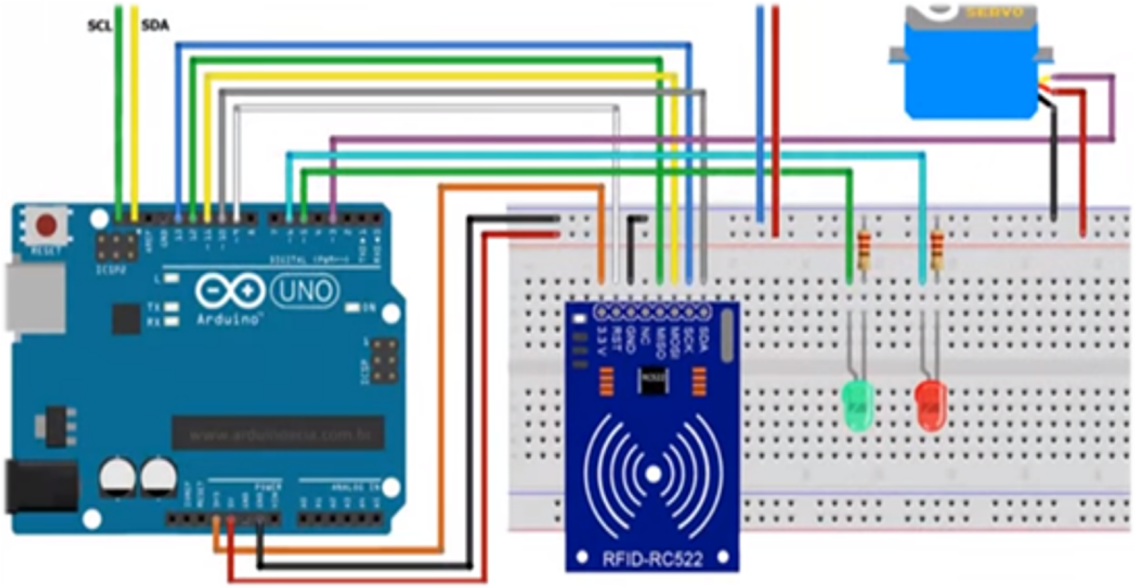


Figura 53. Diseño de control de Acceso mediante medios magnéticos. Información tomada de la Investigación directa. Elaborada por los autores

Materiales:

1. Arduino UNO R3 x 1
2. Módulo RFID RC -522 x 1
3. Tarjetas RFID x 1
4. Servomotor x 1
5. Zumbador x 1
6. Resistencia de 220 ohmios x 2
7. Potenciómetro de 10K ohm x 1
8. Protoboard x 1
9. Cables de puente macho - macho (M-M) x 20
10. Cables de puente Hombre - Fe-Masculino (M-F) x 20

Desarrollo

```
#include <pitches.h>
```

```
#include <RFID.h>
#include <SPI.h>
#include <Servo.h>
//GeekChickens
//http://geekchickens.blogspot.com.es/
/*
SDA es el pin 10
SCK es el pin 13
MOSI es el pin 11
MISO es el pin 12
RST es el pin 9
*/
 //------------------------------------------------variables, includes, funciones, etc----------
---------------------------------------
#include <rfid.h>
  #define SS_PIN 10
#define RST_PIN 9
RFID rfid(SS_PIN, RST_PIN);
Servo myservo;
char orden;
int led_verde = 8;
int led_rojo = 7;
int sensor_puerta = 6;
int contador = 0;//<============================esta variable sirve para
evitar bucles de impresión de " usuario correcto " o " usuario incorrecto"
int estado_puerta = 0;
int puerta_abierta = 0;
int contador_estado_puerta = 0;
int i = 0 ;
int numero_serie[5]; // aqui guardaremos el número que nuestro sensor
RFID detectará
int usuario[1][5]={{116,19,46,150,223}}; // lo he hecho asi para poder añadir
más en un futuro
 //------------------------------------------------variables, includes,funciones, etc-----------
---------------------------------------
void setup()
{
myservo.attach(3);
Serial.begin(9600);
pinMode(led_verde, OUTPUT);
```

```
pinMode(led_rojo, OUTPUT);
pinMode(sensor_puerta,INPUT);
SPI.begin();
rfid.init();
}
 void loop()
{
    myservo.write(180);
   comprobar_puerta();
  if(estado_puerta == 1)
  {
            for(int i = 0 ; i < 60; i++)  // aquest bucle fa que cada segon envii un
" Alerta, la puerta està abierta"
              {
            leer_usuario();
                delay(1);
              }

             // Serial.println("Alerta, la puerta esta abierta");
     }else
      {
         leer_usuario();
         }
}

void abrir_puerta()
{
    comprobar_puerta();

  if(estado_puerta == 0)
  {
 digitalWrite(led_verde,HIGH);
  myservo.write(90);
 Serial.println("PUERTA ABIERTA");
  delay(5000);
   digitalWrite(led_verde,LOW);
   Serial.println("PUERTA CERRADA");
       puerta_abierta = 1;
 contador = 1;
  }else{
```

```
        //Serial.println("Error, La puerta esta abierta");
        contador = 1;
       }
}
  void cerrar_puerta()
{
  comprobar_puerta();
    if(estado_puerta == 0)
  {
        digitalWrite(led_verde,LOW);
  //Serial.println("Puerta cerrada");
  contador = 1;
  }else{
        //Serial.println("Error, La puerta esta abierta");
        contador = 1;
       }
}
  void comprobar_puerta()
{
  estado_puerta = digitalRead(sensor_puerta);

  if((estado_puerta == 0) && (puerta_abierta == 1) &&
(contador_estado_puerta == 0) )  // puerta cerrada pero le hemos dado la
señal de abrir
  {
     delay(100);

         if((estado_puerta == 0 ) && (puerta_abierta == 1))
          {
            contador_estado_puerta = 2;
          }
   }
  if(estado_puerta == 1) // puerta abierta y le hemos dado la señal de abrir
  {
    delay(100);

   if ((estado_puerta == 1 ) && (puerta_abierta == 1) &&
(contador_estado_puerta == 2 ) )
          {
            contador_estado_puerta = 1;
```

```
        }
  }
    if((estado_puerta == 0) && (puerta_abierta == 1) &&
(contador_estado_puerta == 1) ) // si la puerta está cerrada y ha pasado
por los anteriores
  {
     delay(100);

        if((estado_puerta == 0 ) && (puerta_abierta == 1) &&
(contador_estado_puerta == 1))
        {
          contador_estado_puerta = 3;
          puerta_abierta = 0;
          delay(2000);
          cerrar_puerta();
        }
   }
}


void comprobar_usuario()  // con este bucle miramos si alguno de nuestros
usuarios coincide con el array rfid.serNUM ( el numero que está para
comprobar)
{
  i=0;
for(int j=0; j<=5; ) // numero de digitos de nuestro tag
 {
   if( (usuario[i][j]) == (numero_serie[j]))
     {
      j++;
      }else{
            i++;
            j=0;
          }

   if(i >= 2) // ese número es el número máximo de usuarios que pondremos
     {
      j=5;
     }
  switch(i)
```

```
{
  case 0:
  usuario_correcto();
  break;
  default:
  usuario_incorrecto();
  break;

}
 }
}

void leer_usuario()
{
 contador = 0;
   if (rfid.isCard())
   {
         if (rfid.readCardSerial())
          {
              Serial.print("Numero usuario: "); // guardamos el numero del
usuario
              for(int i=0; i<=4 ; i++)
              {
              numero_serie[i] = rfid.serNum[i];
              }
                for(int i=0; i<=4 ; i++) // y lo imprimimos
              {
              Serial.print(numero_serie[i]);
              }
             Serial.println(" "); // esto es para que quede más bonito y el "
usuario correcto " o " usuario incorrecto " aparezcan debajo y no pegados
              delay(500); // para que no se imprima constantemente el mismo
numero
              contador = 1;
             comprobar_usuario();
           }
   }
   rfid.halt();
}
void usuario_correcto()
```

```
{
 if(contador == 1)
  {
   Serial.println("Usuario correcto");
   abrir_puerta();
   contador = 0;
  }
}
 void usuario_incorrecto()
{
 if(contador == 1)
  {
   Serial.println("Usuario incorrecto");
   error();
   contador = 0;
  }
}
void error()
{
    digitalWrite(led_verde,LOW);
  for(int j = 0; j < 4 ; j++)
  {
   digitalWrite(led_rojo,HIGH);
   delay(500);
   digitalWrite(led_rojo,LOW);
   delay(500);
  }
leer_usuario();
}
```

Detalle del Proyecto

El objetivo del proyecto es mostrar cómo podemos construir sistemas reales aplicables al diario vivir con la placa de prototipo ARDUINO y así poder ser trasladados al hogar o implementarlos en equipos, autos, casas, departamentos, electrodomésticos, robots, etc… los mismos que ayudarían a facilitar su desenvolvimiento al sistematizarlos de una manera funcional y evolutiva futurista.

El módulo lector RFID-RC522 RF utiliza 3.3V como voltaje de alimentación y se controla a través del protocolo SPI. También puede ser controlada con un puerto UART. Entonces, podemos decir que es compatible con casi

cualquier microcontrolador, Arduino o tarjeta de desarrollo. El RC522 utiliza un sistema avanzado de modulación y demodulación para todo tipo de dispositivos pasivos de 13.56Mhz. Incluso, puesto que se hará una lectura y escritura de la tarjeta, es necesario conocer las características de los bloques de memoria una tarjeta:
La tarjeta que viene con el módulo RFID cuenta con 64 bloques de memoria (0-63) donde se hace lectura y/o escritura. Cada bloque de memoria tiene la capacidad de almacenar sobre todo hasta 16 Bytes. Finalmente, el número de serie consiste de 5 valores hexadecimales, se podría utilizar esto para hacer una operación dependiendo del número de serie.

Características del módulo lector Rfid-Rc522 Rf

- Modelo: MF522-ED.
- Corriente de operación: 13-26mA a 3.3V.
- Isb de stand by: 10-13mA a 3.3V.
- Ism de sleep-mode: <80uA.
- Im máxima: 30mA.
- Frecuencia de operación: 13.56Mhz.
- Distancia de lectura: 0 a 60mm.
- Protocolo de comunicación: SPI.
- Velocidad de datos máxima: 10Mbit/s.
- Dimensiones: 40 x 60 mm.
- Temperatura de operación: -20 a 80°C.
- Humedad de operación: 5%-95%.
- Máxima velocidad de SPI: 10Mbit/s.
- Incluye pines, llavero y tarjeta.

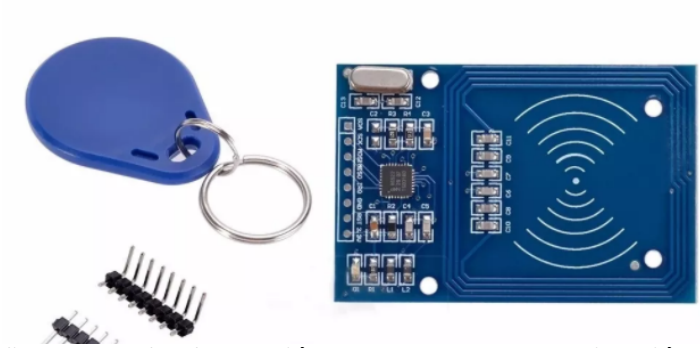


Figura 54. lector Rfid-Rc522 Rf. Información tomada de la Investigación directa. Elaborada por los autores

Descripción módulo lector Rfid-Rc522 Rf

En este tutorial haremos uso del Arduino UNO y el Módulo Lector RFID-RC522 RF para dar uso de algunas funciones de la librería RFID. Las operaciones que finalmente haremos en este tutorial serán:

- Lectura de bloque de memoria de un tarjeta (rfid.read)
- Escritura en un bloque de memoria (rfid.write)
- Lectura de número de serie de tarjeta (rfid.numSerie)

Características técnicas de un servomotor con Arduino

Hay varios modelos de servomotor con Arduino. En este caso vamos a utilizar un Micro Servo 9g SG90 de Tower Pro. Como siempre digo, hay que mirar la ficha técnica del producto. Todos tienen un funcionamiento muy parecido y la programación puede variar muy poco.
Cosas para tener en cuenta con este dispositivo. Lo primero, el ángulo de giro, en este caso nos permite hacer un barrido entre -90º y 90º. Lo que viene a ser un ángulo de giro de 180º.

Figura 55. Servomotor 9g SG90. Información tomada de la Investigación directa. Elaborada por los autores

Aunque el servo puede moverse con una resolución de más de 1 grado, este es el máximo de resolución que vamos a conseguir debido a la limitación de la señal PWM que es capaz de generar Arduino UNO.
Estos motores funcionan con una señal PWM, con un pulso de trabajo entre 1 ms y 2 ms y con un periodo de 20 ms (50 Hz). ¿Qué quiere decir todo esto? Este dato nos indica la velocidad máxima a la que podemos mover el servomotor con Arduino. Solo podremos cambiar de posición cada 20 ms. Esto dependerá del tipo y marca de nuestro servo.
El elegir una salida PWM u otra da lo mismo, todas las salidas de este tipo funcionan igual.

5.9. Réplica del sensor Cardiaco emitido por un tambor

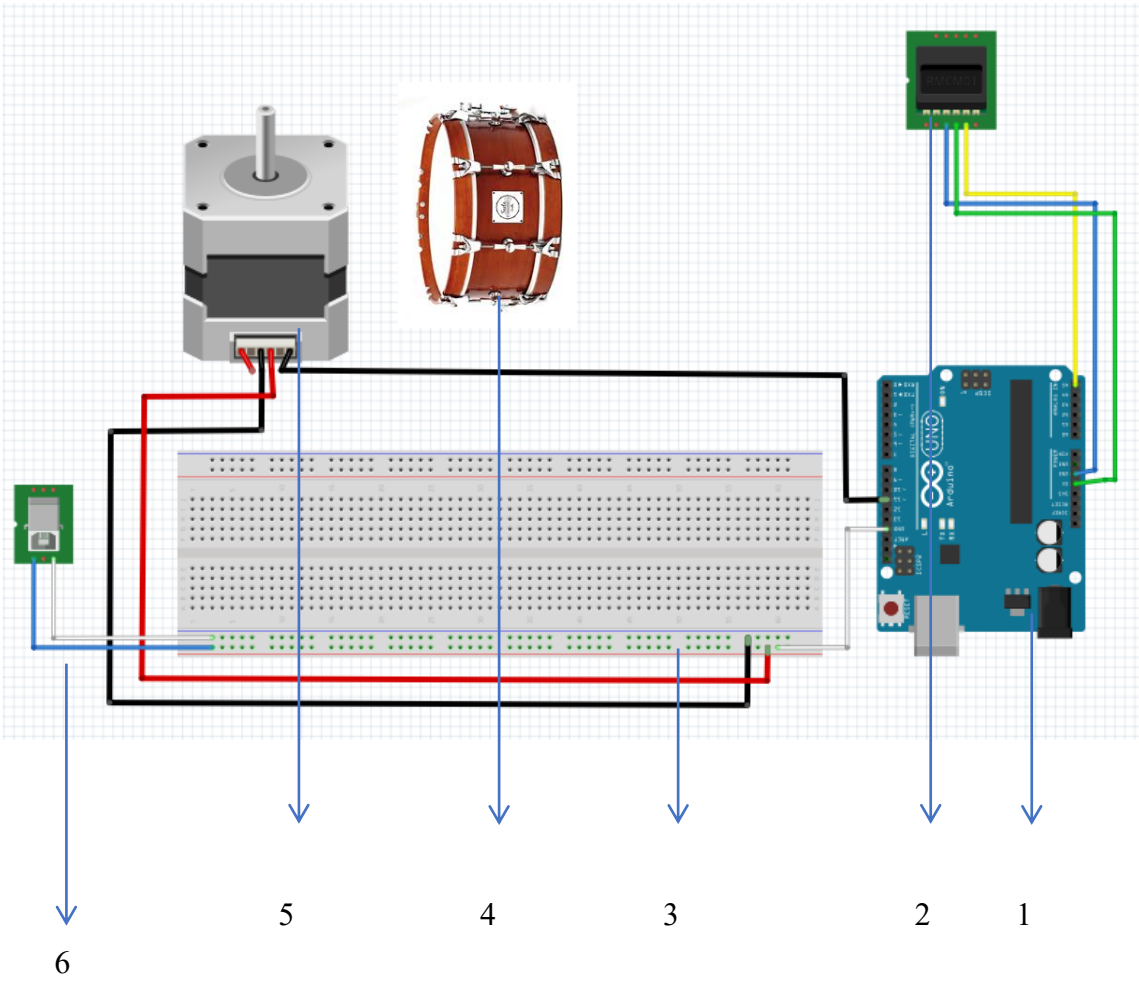


Figura 56. Diseño del sensor cardíaco. Información tomada de la Investigación directa. Elaborada por los autores.

Materiales:

1. Arduino UNO.
2. Sensor Cardiaco.
3. Protoboard.
4. Tambor.
5. Servomotor de 12Kg.
6. Puerto USB.

Desarrollo

```
#include <Servo.h>
```

```
Servo servoMotor;
int pulso=0;
void setup() {
  Serial.begin(9600);
  pinMode(A5,INPUT);
  pinMode(11,OUTPUT);
  servoMotor.attach(11);
}

void loop() {
 pulso=analogRead(A5);
 if(pulso>=513){
  digitalWrite(11,HIGH);
  delay(35);
  digitalWrite(11,LOW);
  servoMotor.write(25);
  delay(92);
  servoMotor.write(180);
  delay(92);  }
  delay(80);
  Serial.println(pulso);
 }
```

Detalle del Proyecto

En el circuito antes mencionado, cada latido del corazón es detectado por el sensor cardiaco que está conectado a la placa de Arduino Uno y a su vez el servomotor de 12 kg fuerza está conectado con el Protoboard para recibir 5 voltios que son alimentados gracias al puerto USB de color blanco. Gracias a eso, el servomotor mueve la baqueta de acuerdo con la programación dada por la placa Arduino y el sensor cardiaco antes mencionado.

El proyecto consiste en mostrar de una forma diferente los latidos del corazón de una persona, ya que existe la forma de mostrar con la presentación de un Led, bocina o incluso una pantalla LCD, por lo tanto, la iniciativa del proyecto es un tanto diferente a lo que se está acostumbrado normalmente. A continuación, el proyecto finalizado.

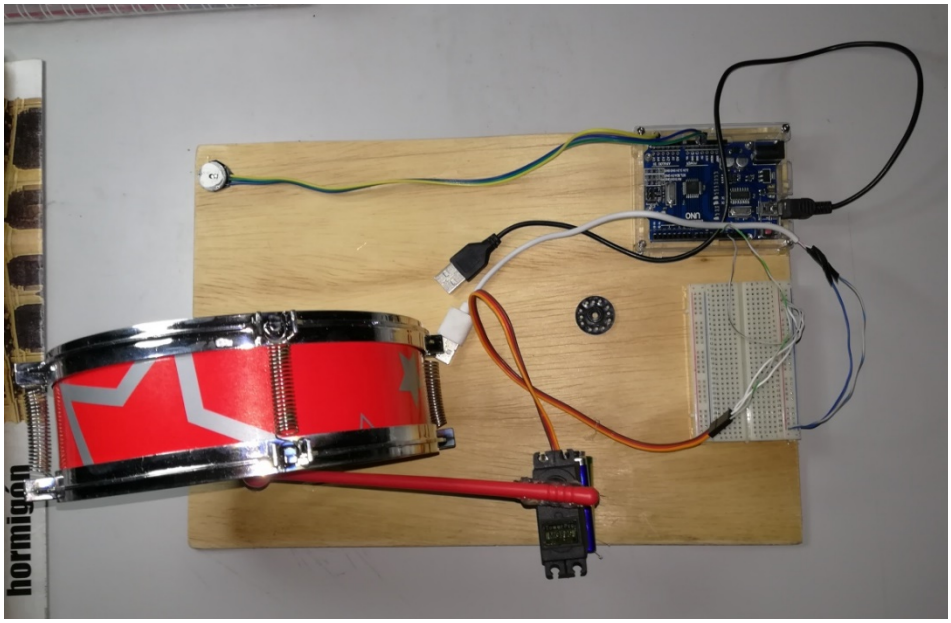


Figura 57. Proyecto finalizado del sensor cardíaco. Información tomada de la Investigación directa. Elaborada por los autores.

REFERENCIAS BIBLIOGRÁFICAS

C.V, P. D. (2016). *CursoArduino*. http://cursoarduino.proserquisa.com/2016/10/12/tutorial-16-kit-de-sensores-y-actuadores-compatibles-con-arduino/

Crespo, E. (24 de Noviembre de 2014). *Aprendiendo Arduino*. https://aprendiendoarduino.wordpress.com/2016/03/29/entorno-de-programacion-de-arduino-ide/

UploadWizard. (16 de Febrero de 2016). *Wikimedia Commons*. https://commons.wikimedia.org/wiki/File:ArduinoUNO.png

Descubre tu próxima lectura

Si quieres formar parte de nuestra comunidad,
regístrate en **https://www.grupocompas.org/suscribirse**
y recibirás recomendaciones y capacitación

compAs

Grupo de capacitación e investigación pedagógica

@grupocompas.ec
compasacademico@icloud.com

Ingrid Angélica García Torres, Ingeniera en Sistemas Informáticos, Universidad Técnica de Manabí, Portoviejo, Ecuador; Magister en Educación Informática, Universidad de Guayaquil. Cursando un Programa de Doctorado en la Universidad Nacional de la Plata (UNPL), La Plata, Argentina en la Facultad Informática. Investiga temas: Microcontroladores, Realidad Aumentada, Programación Estructurada, Imagen por computadora, Docente a nivel Secundario de Informática Unidad Educativa Pablo Zamora, Portoviejo, Manabí y Unidad Educativa Camilo Ponce Enríquez, Guayaquil, Guayas, Ecuador. Actualmente gestora de Titulación, directora de Trabajos de Titulación y docente de la Carrera de Ingeniería en Telemática de la Facultad de Ingeniería Industrial de la Universidad de Guayaquil. Docente investigador. Correo ingrid.garciat@ug.edu.ec

Rodolfo Antonio Parra López, Ingeniero en Electricidad especializado en Sistemas de Potencia en la Escuela Superior Politécnica del Litoral (ESPOL); Diplomado en Auditoria de Sistemas de Gestión en Universidad Técnica Particular de Loja; Magister en Sistemas Integrados de Gestión en Universidad Estatal de Guayaquil; Fiscalizador eléctrico principal en megaobra Malecón 2000; Director de proyectos en Promelec, Director de proyecto de automatización del catastro urbano en la ciudad de Ventanas por parte de Ecuacorporac s.a., Gerente Técnico en Rivalesa s.a.; Gerente Técnico en Bkpack s.a.; Docente de la carrera de Ingeniería en Telecomunicaciones de la Universidad Católica Santiago de Guayaquil, Actualmente docente de la Universidad Estatal de Guayaquil en la Facultad de Ingeniería Industrial. Correo rodolfo.parral@ug.edu.ec

Rosa Elizabeth Castillo León, Ingeniera en Computación especialización Sistemas Tecnológicos, Escuela Superior Politécnica del Litoral, Guayaquil, Ecuador; Magister en Gestión de Proyectos, Escuela de Negocios ESPAE, Guayaquil, Ecuador. Investiga temas: Gestión de proyectos de desarrollo de software. Ha trabajado como Jefe de Proyectos en i-Route Solutions, Guayaquil. Gestor de Calidad, Analista Programador y Administrador del equipo de pruebas en el departamento de Desarrollo en Corpacel, Guayaquil. Actualmente profesor de la Universidad de Guayaquil, Ecuador. Correo rosa.castillol@ug.edu.ec

Dennis Holger Zambrano Silva, Magister en Producción y Productividad Universidad Estatal de Guayaquil, Ingeniero Industrial Universidad Estatal de Guayaquil, Se desempeño por tres décadas del libre ejercicio de la profesión en diferentes cargos de Dirección en áreas de Proyectos Industriales, Ingeniería Diseño y Desarrollo, Planificación y Control en los Departamentos de Producción, Calidad, Logística, Almacenamiento y Mantenimiento. Proyecto de automatización y Control de planta por medio de Plataforma integrada a los procesos. Tutor de Tesis de Maestría de Producción y Productividad, Tutor de Tesis de Maestría de Sistemas Integrados, Tutor de Tesis de maestría de Seguridad Industrial. Docente en la carrera de Ingeniería Industrial, docente en carrera de Telemática de Universidad Estatal de Guayaquil. Correo dennis.zambranos@ug.edu.ec

Luis Javier Domínguez De la Torre, Magister en Educación e Ingeniero Industrial graduado en la Universidad Estatal de Guayaquil, se desempeñó por dos décadas en el libre ejercicio de la profesión en diferentes cargos: Asistente de Producción en la Empresa Municipal de Agua Potable de Guayaquil, Gerente de ventas de la compañía COPROALCA S.A. y Gerente de ventas de la compañía SUMITEC S.A. También como Tutor de tesis en la carrera de Teleinformática de la Facultad de Ingeniería Industrial de la Universidad de Guayaquil. Docente en la carrera de Teleinformática, y Telemática de la de la Facultad de Ingeniería Industrial de la Universidad de Guayaquil durante 13 años. Correo luis.dominguezt@ug.edu.ec

William Ricardo Navas Espin, es Máster en Dirección de Empresas y Recursos Humanos por la Universidad de Castilla – La Mancha (España), Contador Público Autorizado y Licenciado en Contaduría y Auditoría por la Universidad Laica Vicente Rocafuerte de Guayaquil. Ha desempeñado cargos alineados a su profesión en empresas privadas en la parte administrativa. Colabora en publicaciones universitarias y asiste continuamente a congresos nacionales e internacionales. Docente a tiempo completo de la Universidad de Guayaquil, Facultad de Ingeniería Industrial, carrera de Ingeniería en Telemática-Teleinformática. Correo william.navase@ug.edu.ec

compAS
Grupo de capacitación e investigación pedagógica

@grupocompas.ec
compasacademico@icloud.com

ISBN: 978-9942-33-296-7

compAS
Grupo de capacitación e investigación pedagógica

@grupocompas.ec
compasacademico@icloud.com