INFORME PROYECTO ARDUINO

Presentado por:

Milton Sebastian Polanco Escobar

Jesus Arbey Muñoz Cabrera

Presentado a:

Carlos Hernán Tobar Arteaga

Facultad de Ingeniería Electrónica Y Telecomunicaciones

Programa de Ingeniería Electrónica

Popayán, Cauca

Introducción

El arduino es una plataforma de desarrollo basada en una placa electrónica de hardware y software libre, la cual es flexible y fácil de utilizar para los creadores y desarrolladores esta placa tiene incorporado un microcontrolador re-programable y una serie de pines.

Arduino ofrece las bases para que cualquier otra persona o empresa pueda crear sus propias placas, pudiendo ser diferentes entre ellas pero igualmente funcionales a partir de la misma base. Su software libre ofrece programas informáticos cuyo código es accesible por cualquiera para que quien quiera pueda utilizarlo y modificarlo. Además cuenta con la plataforma Arduino IDE (Entorno de Desarrollo Integrado), que es un entorno de programación con el que cualquiera puede crear aplicaciones para las placas Arduino, de manera que se les puede dar todo tipo de utilidades.

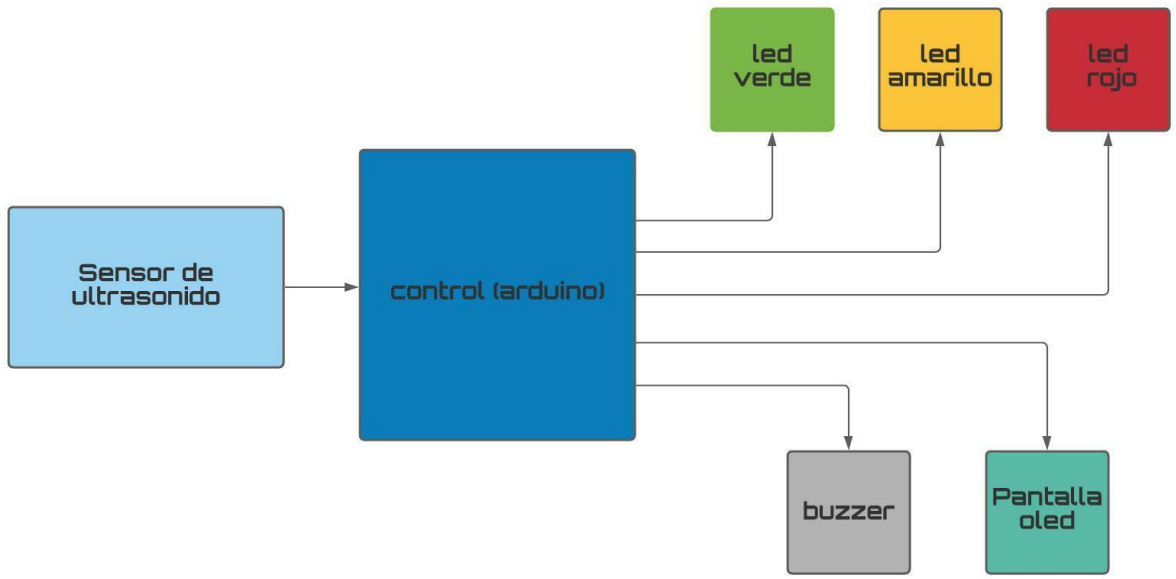
La enorme flexibilidad y el carácter libre y abierto de Arduino hacen que se pueda prácticamente para cualquier cosa como relojes, robots, persianas controladas por voz, entre otros.

Planteamiento del problema

En la actualidad el uso de sensores es algo que puede beneficiar, mejorar la calidad de vida e incluso facilitar tareas las cuales si no se tiene práctica pueden llegar a ser muy tediosas. Dado esto se encuentra la necesidad de suministrar a un conductor una herramienta con la cual pueda hacer un aparcamiento de su vehículo lo más rápido, económico y seguro posible.

Diseño de la solución:

Diagramas de bloques



Configuraciones realizadas

El sistema de aparcamiento consiste en detectar un objeto a través del sensor ultrasónico, luego utilizando una pantalla oled se mostrará la distancia y se avisará con señales de luz y sonido.

Por lo tanto, se requiere hacer un algoritmo el cual cuente con la detección de obstáculos y para ello se divide en zonas las cuales serán zona roja, zona amarilla y zona verde, cada una de estas zonas va a contar con un sonido el cual tendrá un tono diferente para cada una.

Se decide trabajar sobre los siguientes espacios de detección:

- Zona verde de 150 cm a 80 cm
- Zona amarilla de 80 cm a 50 cm
- Zona roja de menos 50 cm

Módulos empleados

Descripción del código inicialmente se declara la biblioteca la cual es necesaria para la pantalla lcd 16x2 y se asigna cada uno de los pines a utilizar en este caso se les da un nombre con el cual es fácil de identificar.

```
#include <LiquidCrystal.h>

//LCD 16x2 pines
LiquidCrystal lcd (7,8,10,11,12,13);

// Pines utilizados
#define LEDVERDE 2
#define LEDAMARILLO 3
#define LEDROJO 4
#define TRIGGER 5
#define ECHO 6
#define BUZZER 9

// Constantes
const float sonido = 34300.0; // Velocidad del sonido en cm/s
const float umbral1 = 150.0;
const float umbral2 = 80.0;
const float umbral3 = 50.0;
```

Se procede a iniciar la pantalla con el comando .begin y poder imprimir la distancia que proporciona el módulo de ultrasonido más adelante.

```
void setup() {

  // Iniciamos la pantalla lcd
  lcd.begin (16,2);

  pinMode(LEDVERDE, OUTPUT);
  pinMode(LEDAMARILLO, OUTPUT);
  pinMode(LEDROJO, OUTPUT);
  pinMode(ECHO, INPUT);
  pinMode(TRIGGER, OUTPUT);
  pinMode(BUZZER, OUTPUT);

  apagarLEDs();

}
```

Esta función es la principal la cual va a ejecutar cada uno de los métodos los cuales ayudan a calcular la distancia a la cual está el objeto y también emite una alerta si es necesario.

```
void loop() {
  lcd.setCursor (0,0);

  // Preparamos el sensor de ultrasonidos
  iniciarTrigger();

  // Obtenemos la distancia
  float distancia = calcularDistancia();

  // Apagamos todos los LEDs
  apagarLEDs();

  //  si estamos dentro del rango de peligro
  if (distancia < umbral1)
  {
    alertas(distancia);
  }

}
```

Se crea una función para apagar todos los leds, entonces con las variables declaradas al principio se asignan como 'LOW'. Esta función será usada para cuando el sensor no detecte o esté entre las distancias estipuladas.

```
// Apaga todos los LEDs
void apagarLEDs()
{
  digitalWrite(LEDVERDE, LOW);
  digitalWrite(LEDAMARILLO, LOW);
  digitalWrite(LEDROJO, LOW);
}
```

La función 'alertas' permite tomar la distancia a la que se encuentra el sensor del obstáculo el cual la calcula el método 'void loop' y según los parámetros encender el Led respectivo y emitir un tono con el buzzer.

```
void alertas(float distancia)
{
  if (distancia < umbral1 && distancia >= umbral2)
  {
    // Encendemos el LED verde
    digitalWrite(LEDVERDE, HIGH);
    tone(BUZZER, 2000, 200);
  }
  else if (distancia < umbral2 && distancia > umbral3)
  {
    // Encendemos el LED amarillo
    digitalWrite(LEDAMARILLO, HIGH);
    tone(BUZZER, 2500, 200);
  }
  else if (distancia <= umbral3)
  {
    // Encendemos el LED rojo
    digitalWrite(LEDROJO, HIGH);
    tone(BUZZER, 3000, 200);
  }
}
```

La función "calcular distancia" retorna una variable la cual es la distancia a la cual está el objeto esta se entrega en 'cm' por lo cual es necesario multiplicar por *0.000001* y esto lo hace mediante una operación:

$$Distancia = tiempo * 0.000001 * sonido/2$$

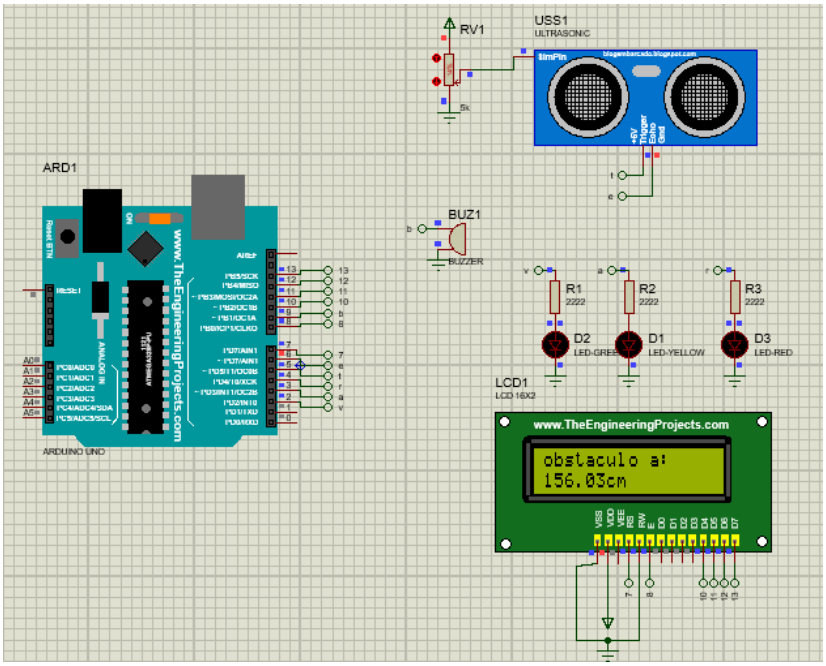
```
// Método que calcula la distancia a la que se encuentra un objeto.
// Devuelve una variable tipo float que contiene la distancia

float calcularDistancia()
{
  // La función pulseIn obtiene el tiempo que tarda en cambiar entre estados, en este caso a HIGH
  unsigned long tiempo = pulseIn(ECHO, HIGH);

  // Obtenemos la distancia en cm, hay que convertir el tiempo en segudos ya que está en microsegundos
  // por eso se multiplica por 0.000001
  float distancia = tiempo * 0.000001 * sonido / 2.0;

  //imprime en el LCD
  lcd.setCursor (0,0);
  lcd.print("obstaculo a:");
  lcd.setCursor (0,1);
  lcd.print(distancia );
  lcd.print("cm");
  lcd.println();

  return distancia;
}
```

El terminal Trigger del sensor de ultrasonido es un pin de salida que permite la medición de la distancia al obstáculo. Primero se pone el terminal en estado bajo a 2ms, después se cambia a estado alto donde manda la onda y se espera 10ms y se termina nuevamente cambiando a estado bajo para concluir la medición.

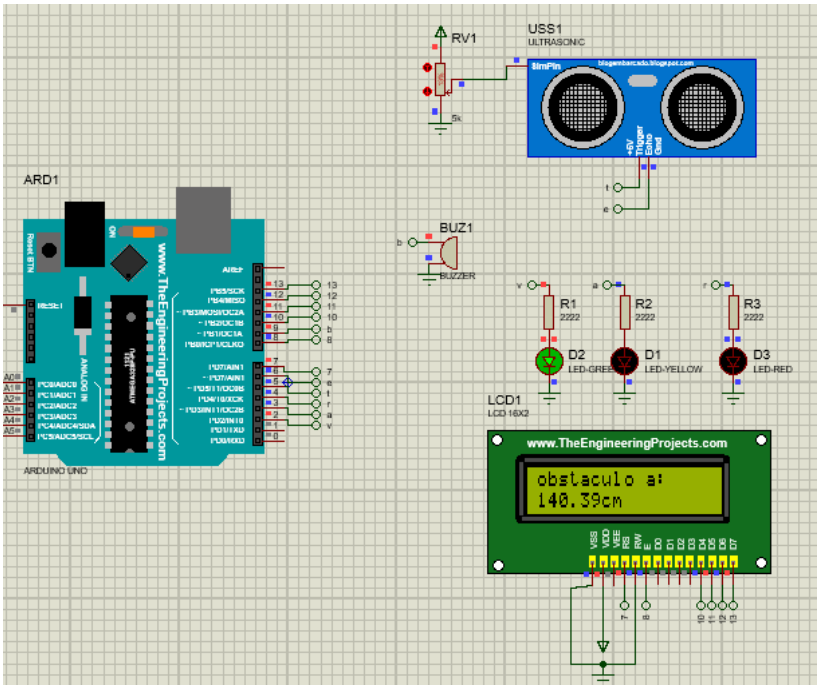
```
// Método que inicia la secuencia del Trigger para comenzar a medir
void iniciarTrigger()
{
  // Ponemos el Triiger en estado bajo y esperamos 2 ms
  digitalWrite(TRIGGER, LOW);
  delayMicroseconds(2);

  // Ponemos el pin Trigger a estado alto y esperamos 10 ms
  digitalWrite(TRIGGER, HIGH);
  delayMicroseconds(10);

  // Comenzamos poniendo el pin Trigger en estado bajo
  digitalWrite(TRIGGER, LOW);
}
```

- Pruebas funcionales

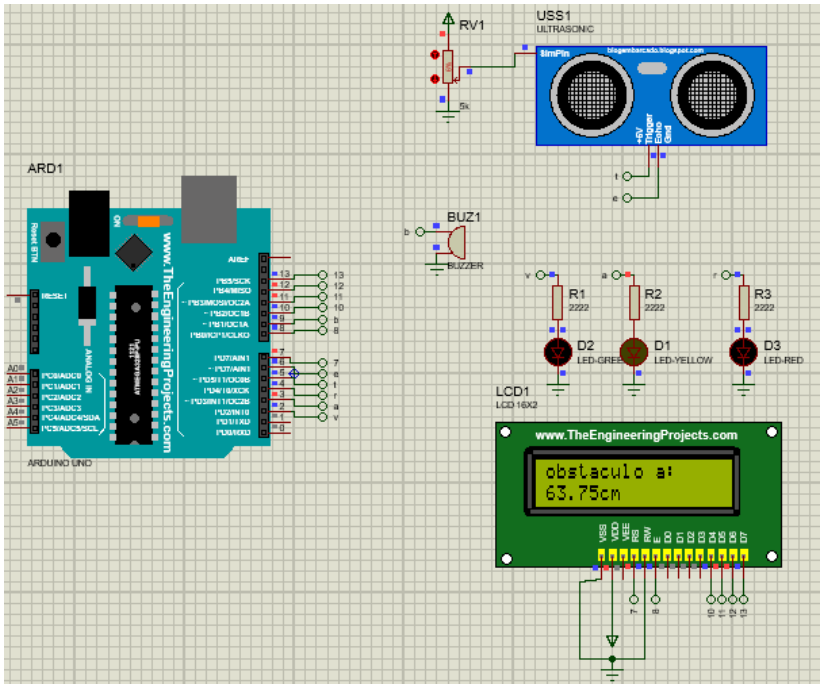
El primer estado es cuando el sensor está por encima de los rangos especificados y no acciona ningún led, la pantalla LCD muestra la distancia a la que se encuentra, si estuviera por encima de la capacidad del sensor no marca la LCD.



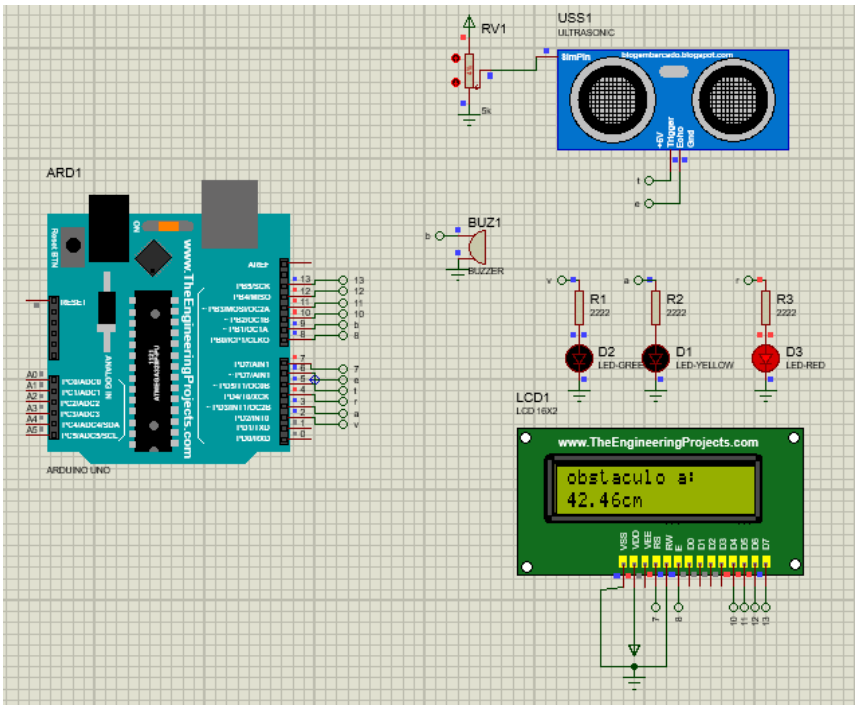
El segundo estado es cuando entra al primer rango de distancia que es prudencial por lo que enciende el Led verde, a su vez el buzzer también estará emitiendo sonido a una frecuencia baja y el Lcd permite saber a qué distancia se encuentra.



El tercer estado es cuando entra al rango intermedio por lo cual se enciende el Led amarrillo, el buzzer emite un sonido el cual es un poco más fuerte al del anterior caso dado que sube la frecuencia y por último en el lcd nos muestra la distancia a la cual está el objeto.(el led amarrillo en proteus tiene una baja intensidad).



Por último el cuarto estado es cuando la distancia está muy próxima al obstáculo y enciende el Led rojo, a su vez el buzzer emite un sonido con frecuencia alta para alertar al conductor y tendrá ayuda de la pantalla para poder tomar una decisión más acertada.



- Análisis de resultados y conclusiones

La plataforma Proteus permite realizar una simulación ideal de proyecto. El sistema puede ayudar al conductor en algunos lugares específicos en los que pueda operar el sensor del ultrasonido, tales como parqueaderos, garajes y supermercados o centros comerciales; esto por que la onda utilizada por el ultrasonido es unidireccional y la onda tiene poca ampliación.

Para un sistema más completo y por tanto más complejo sería implementar más de un sensor y tener una mayor cobertura del entorno, pero con el modelo realizado se puede usar en lugares con obstáculos contundentes como una pared y a un muy bajo coste y de fácil instalación.

- Anexo: enlace al repositorio GitHub

https://github.com/sebastianpolanco/SistemaAparcamientoProyecto