Por: Adrian Morales, Blanca Lara 4ºESO A 2019

Proyecto final

Gafas de Invidente

- Índice:

1. - Finalidad del proyecto:

- La finalidad del proyecto es crear un dispositivo para personas invidentes, que le advierta de la cercanía de un obstáculo a través de la frecuencia de sonido de un zumbador. De forma que mientras más cerca esté el objeto mayor sea la frecuencia de pitido.
- **Esquema de entradas y salidas**

2. - Búsqueda de información

- **Principios de funcionamiento**

  El sensor de ultrasonido manda un sonido (que los humanos no percibimos), el cual rebota en cualquier superficie plana, lo vuelve a captar y nos da una cifra en relación con el tiempo que ha tardado, este lo pasamos a distancia ya que sabemos la velocidad del sonido (343,2 m/s). La distancia la escalamos a una frecuencia de pitido, la cual es la que hace pitar el buzzer directamente proporcional a la distancia.

- **Proyectos documentados que nos sirvan como referencia**

  Anti collision glasses nos ha dado la idea de cómo hacerlo, junto a más proyectos.

- **Decisiones técnicas de cómo se van a resolver los subsistemas**

  El sensor de distancia ha sido elegido por su bajo valor económico y el buzzer por su fácil manejo.

- **Materiales**

  Diseño gafas realizado con PLA mediante impresora 3D

- **Hojas de características de componentes (datasheets)**

Sensor de ultrasonido info

Buzzer info

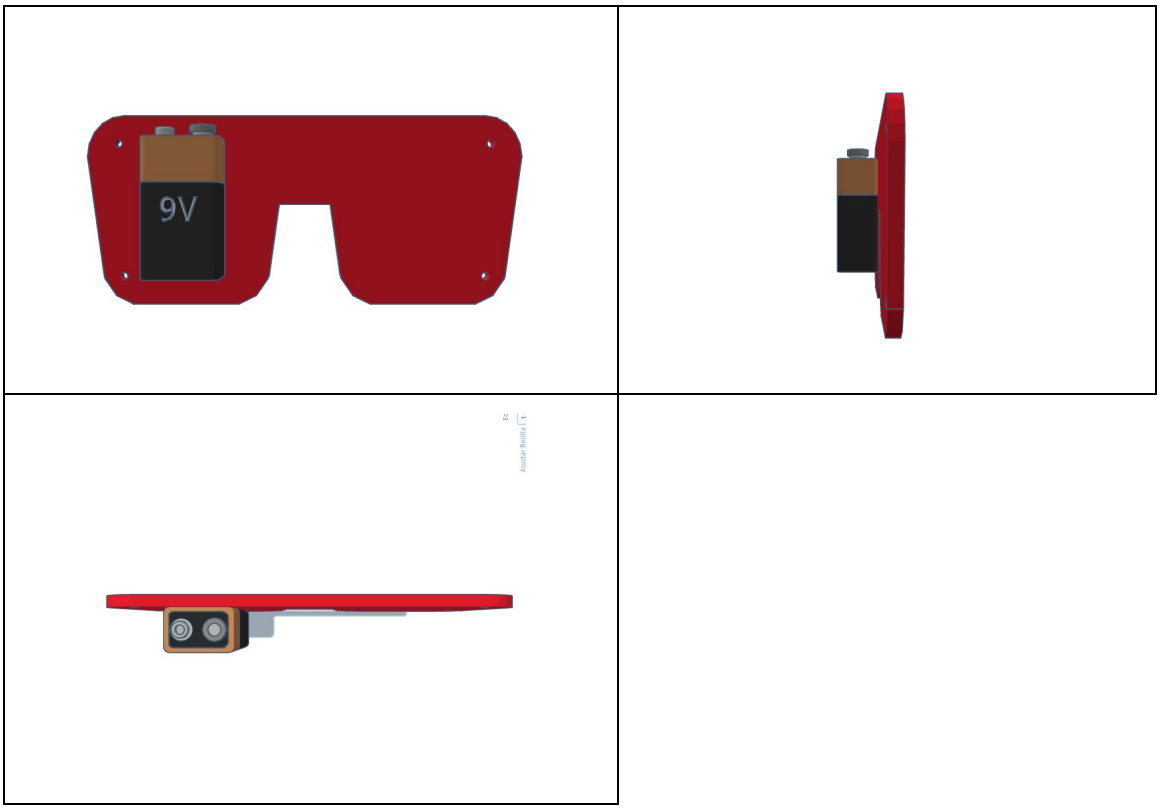
3. - Planificación

- **Lista de materiales**

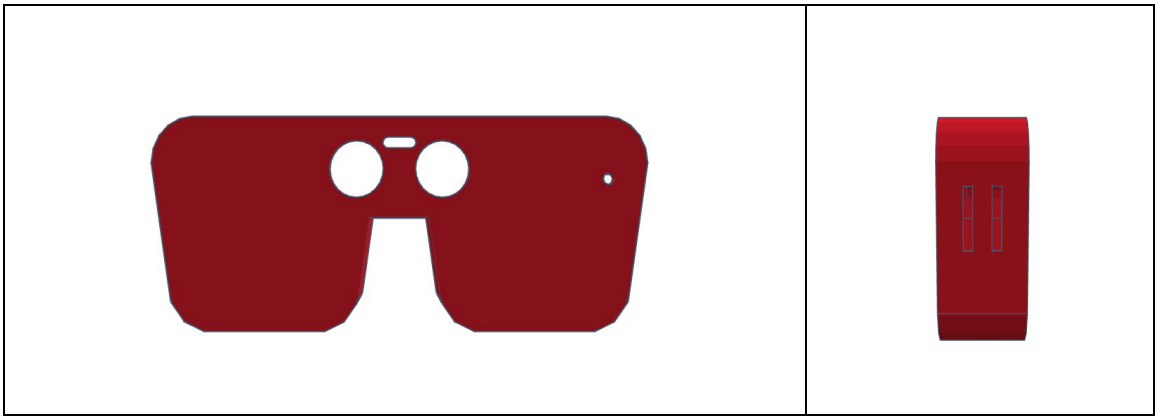
nº	COMPONENTES	CANTIDAD	PRECIO	TOTAL
1	Sensor de ultrasonido HCSR04	1	2,50 €	2,50 €
2	Buzzer	1	0,95 €	0,95 €
3	Batería 9v	1	2,00 €	2,00 €
4	Pulsador	1	0,95 €	0,95 €
5	LED de 3mm	1	0,95 €	0,95 €
6	Resistencia 10kΩ	1	0,99 €	0,99 €
7	Resistencia 220Ω	1	0,99 €	0,99 €
8	Tornillos y tuercas	8	0,05 €	0,40 €
9	Protoboard	1	6,00 €	6,00 €
10	Cinta elástica	1	0,50 €	0,50 €
11	Gafas 3D	1	6,30 €	6,30 €
12	Tapa gafas 3D	1	5,33 €	5,33 €

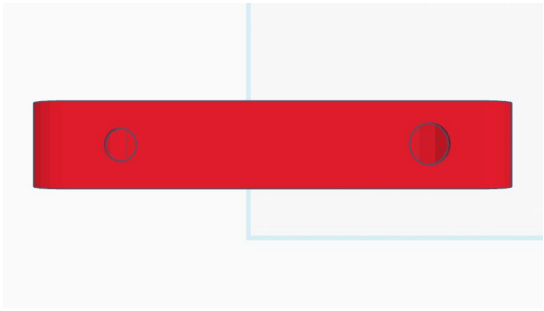
4. - Diseño de la maqueta

- **Tapa Gafas**

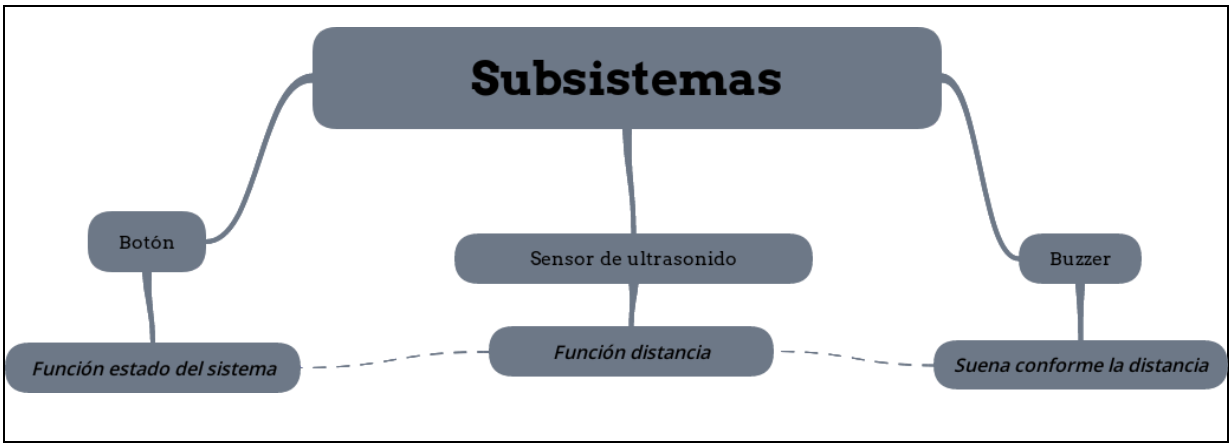


- **Gafas**





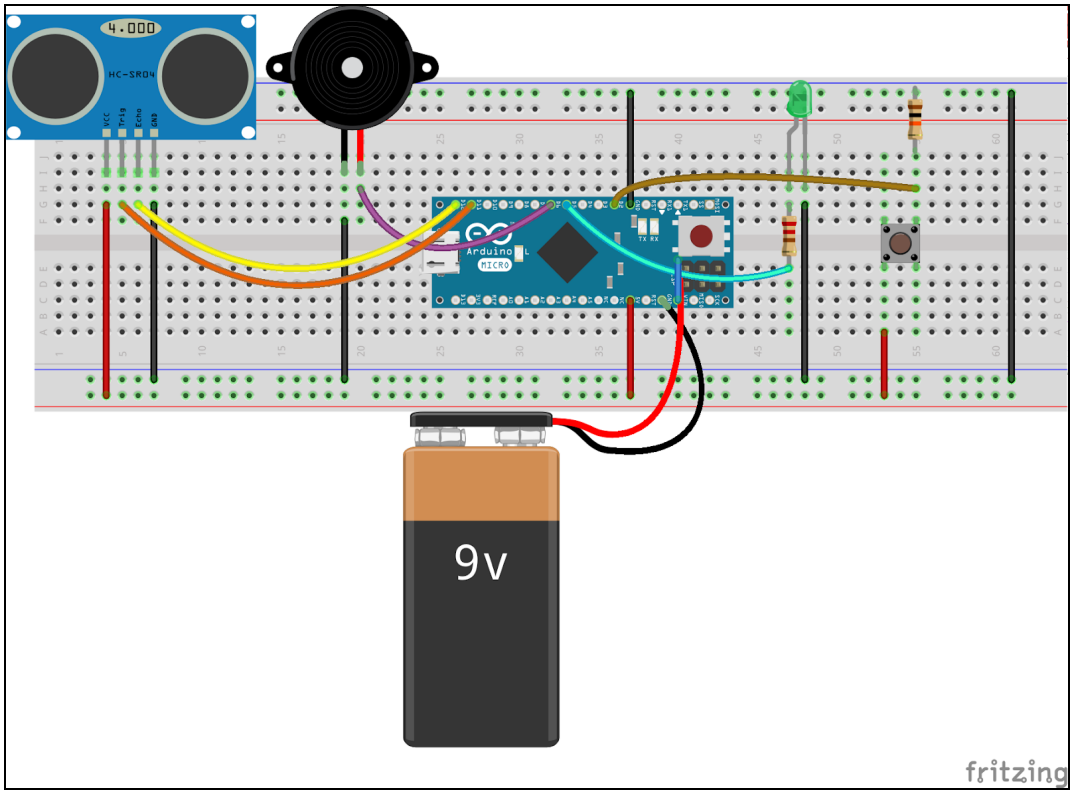
5. - Subsistemas

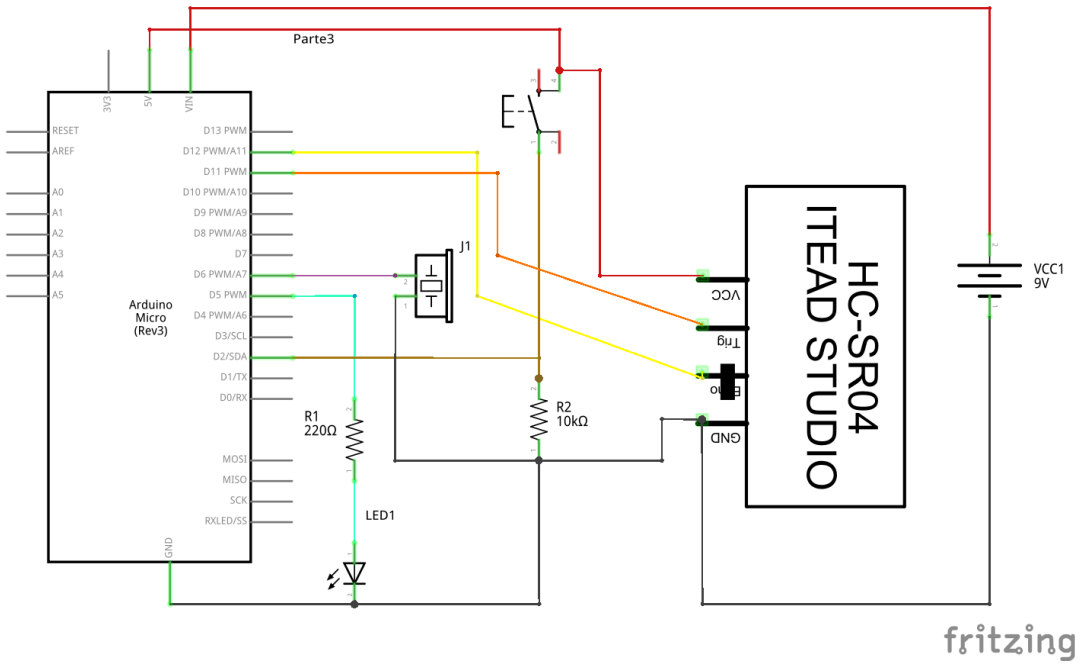


Este dispositivo tiene tres subsistemas principales, cada uno con una función en específico. El botón participa en el estado del sistema que hace que funcione o no el dispositivo con una función (void leerpulsador), haciendo que si está activo haga el resto de las órdenes. El sensor de ultrasonido nos da la información de la distancia que hay entre él y el obstáculo con una función al igual que el botón (float measuringdistance). Por último el buzzer suena con frecuencia escalada a la distancia, anteriormente obtenida por la función de medida de distancia.

6. - Hardware

- **Esquemas electrónicos**





7. - Software

Código final:

```
/*
 Gafas para invidente

 Este código permite con un sensor de ultrasonido saber
 si la persona que lo lleva se acerca a un obstáculo

 creado Marzo 2019
 Por Adrián M
*/

//Estado del sistema
int systemState = LOW;

//Variables botón
const int buttonPin = 2; //pin del pulsador
int buttonState = 0; // variable para almacenar el estado del pulsador
int lastButtonState = LOW; // variable para almacenar el último estado del pulsador
unsigned long lastDebounceTime = 0;  // la última vez que se cambió el pin de salida
```

```
unsigned long debounceDelay = 50;  // el tiempo debounce

//Variables del sensor de distancia
const int triggerPin = 11; // Pin donde conectamos el emisor
const int echoPin = 12; // Pin donde conectamos el receptor
const long interval = 100; // intervalo de tiempo de medición
unsigned long previous_time = 0; //Variable que almacena el tiempo calculado para calcular el siguiente (sensor de ultrasonido)

//Variables buzzer
const int buzzerPin =  6;   // el numero de pin del zumbador
int buzzerState = 0; // variable para almacenar el estado del buzzer
long intervalBuzzer = 0; //Variable frecuencia del sonido
unsigned long previous_timeBuzzer = 0; //Variable que almacena el tiempo calculado para calcular el siguiente (buzzer)

// variable para almacenar tiempo actual
unsigned long current_time = 0;

//Variable distancia
int cm = 0;

//Variable LED
const int ledPin = 5; //pin del LED
int ledState = LOW; //estado del LED



void setup() {
  Serial.begin(9600); // Abrimos el puerto serie
  pinMode(triggerPin, OUTPUT); // Configuramos trigger como salida digital
  pinMode(echoPin, INPUT); // Configuramos echo como entrada digital
  pinMode(buttonPin, INPUT); // Configuramos button como entrada digital
  pinMode(ledPin, OUTPUT); // Configuramos led como salida digital
}
void loop() {

  //función que da systemState
  leerpulsador();

  current_time = millis(); //tiempo desde que se inicia el programa
```

```
  //hace funcionar el sistema o no
  if (systemState == HIGH) {

    if (current_time - previous_time > interval) { //si el tiempo es mayor que el intervalo medimos
      previous_time = current_time;

      //retornamos el valor en cm de la función measuringdistance
      cm = measuringdistance ();
      Serial.println (cm); //imprimimos el valor en el monitor serie

      //Escalamos los cm a la frecuencia con la que el buzzer suena
      intervalBuzzer = map(cm, 5, 245, 0, 1000);
    }

    //frecuencia del pitido
    if (cm > 0) { //Si da valores, el buzzer da sonido
      if (current_time - previous_timeBuzzer > intervalBuzzer) { //se activa y desactiva el pitido con frecuencia de intervalBuzzer
        previous_timeBuzzer = current_time;

        if (buzzerState == 0) {
          buzzerState = 130; //Se activa el buzzer si está desactivado
        }
        else {
          buzzerState = 0; //Se desactiva el buzzer si está activado
        }
        analogWrite(buzzerPin, buzzerState); //Se cambia al estado del Buzzer
      }
    }
    else {
      analogWrite(buzzerPin, 0); //se desactiva para que no se quede encendido
    }
  }
  else {
    analogWrite(buzzerPin, 0); //se desactiva para que no se quede encendido
  }
}


// Función para medir la distancia
float  measuringdistance() {
```

```
  // Variable para almacenar el tiempo de la onda y la distancia
  float duration, distance;
  //Inicializamos el sensor
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(5);
  // Enviamos una señal activando la salida trigger durante 10 microsegundos
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  // Medimos el ancho del pulso, cuando la lectura sea HIGH
  // devuelve el tiempo transcurrido en microsegundos
  duration = pulseIn(echoPin, HIGH, 15000);
  // Calculamos la distancia en cm y la guardamos en distance
  distance = duration * 0.01715;
  // devolvemos la distancia
  return distance;
}

// Función para conocer systemState
void leerpulsador() {
  // lee el estado del interruptor (variable local)
  int reading = digitalRead(buttonPin);

  // Si el estado del botón cambia
  if (reading != lastButtonState) {
    // reinicia el contador de debounce
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime) > debounceDelay) {

    // Si el estado del botón ha cambiado
    if (reading != buttonState) {
      buttonState = reading;

      // Solo está encendido el LED si buttonState == HIGH
      if (buttonState == HIGH) {
        ledState = !ledState;
        systemState = !systemState;
      }
    }
  }
```

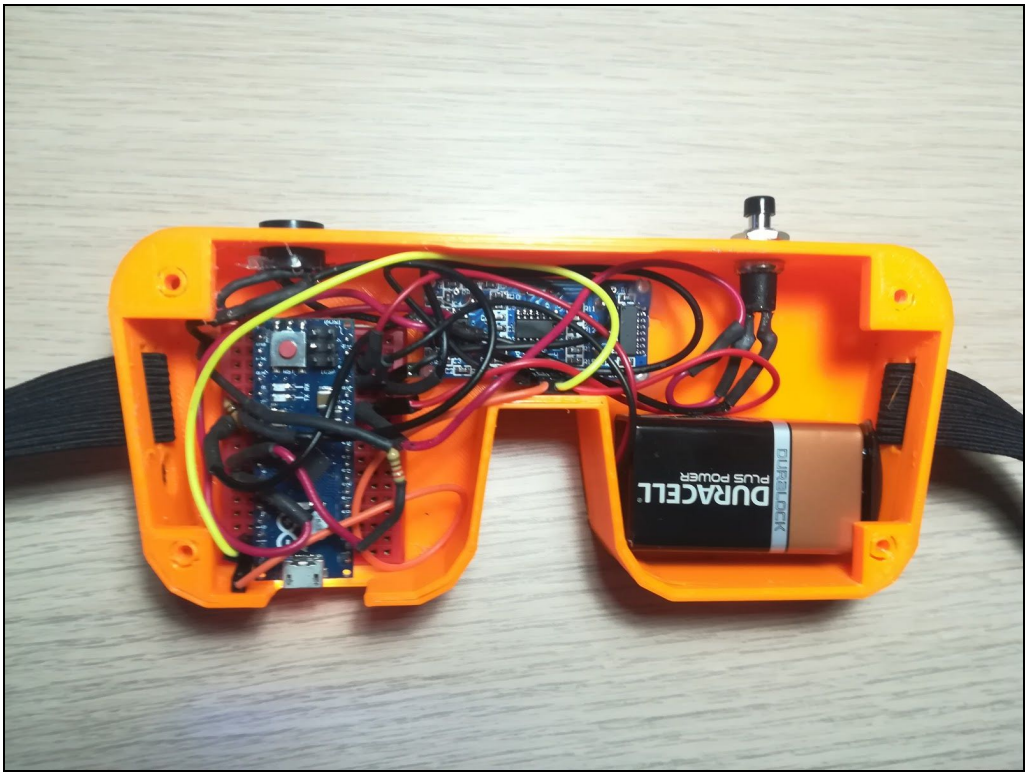
```
  //LED testigo del sistema on/off
  digitalWrite(ledPin, ledState);

  // Guarda la lectura. La próxima vez será lastButtonState
  lastButtonState = reading;
}
```

8. - Funcionamiento

- **Vídeo**

- **Fotos**



9. - Análisis del proyecto

- **Análisis del funcionamiento del proyecto**

  El proyecto funciona a la perfección teniendo el código muy bien estructurado, en cambio la comodidad a la hora de llevar las gafas es escasa.