Actividad 1: Juego LEDs

Grupo 5

Nuno Del Pino Escalante, Antonio Rojas Vélez y Sonia Rus Morales.

1º Bachillerato IES Vicente Aleixandre.

24/11/19

1. Finalidad del sistema:

El proyecto que se pretende realizar consiste en que con 5 leds, dos pulsadores y un zumbador; los leds se vayan encendiendo en una secuencia uno de los pulsadores hace de interruptor del sistema y con el otro se intentaría acertar el led encendido, con esto el zumbador se activaría.

2. Búsqueda de Información:

Para hacer este proyecto nos hemos ayudado de las diapositivas del powerpoint del siguiente enlace: Robótica, el coche fantástico.

También hemos utilizado las siguientes páginas web para las distintas canciones:

Megalovania y La marcha imperial.

3. Hardware:

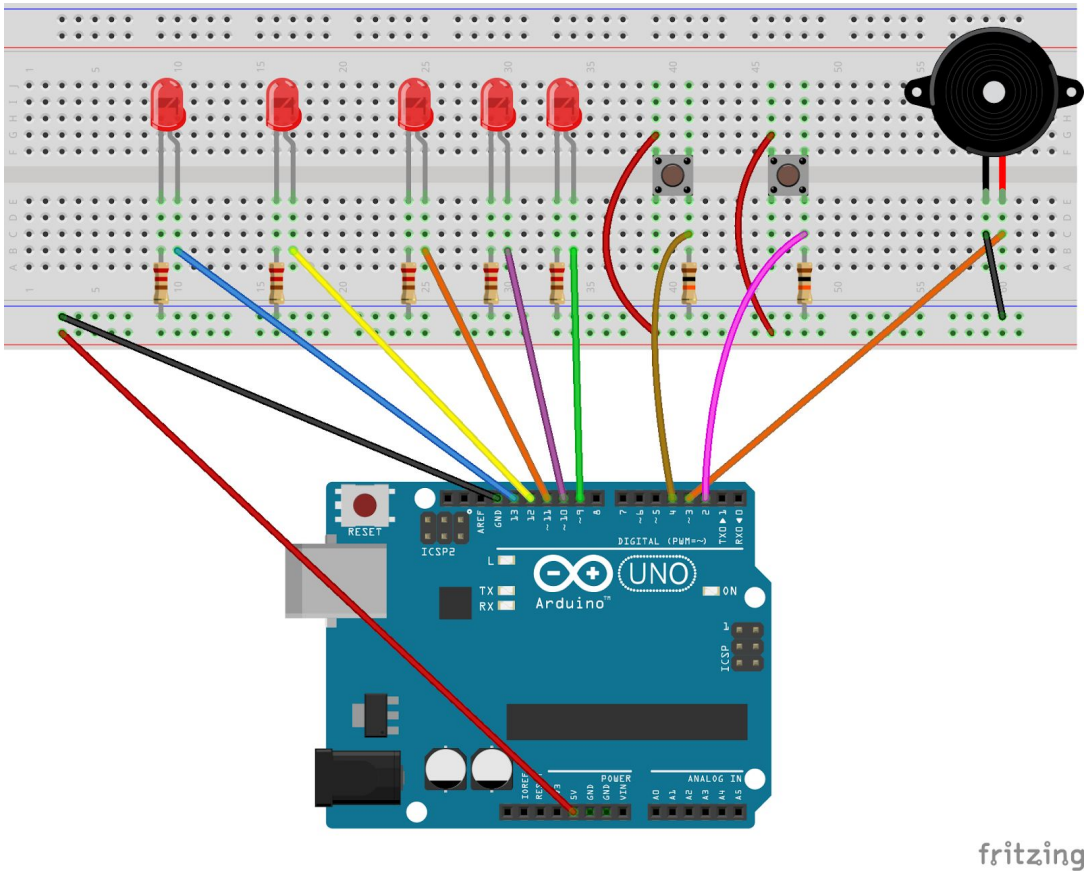
a. Esquema de entradas y salidas

Entradas: Pulsadores → Microcontrolador: Arduino → Salidas: Led y Buzzer

b. Lista de materiales: tabla con descripción y cantidad

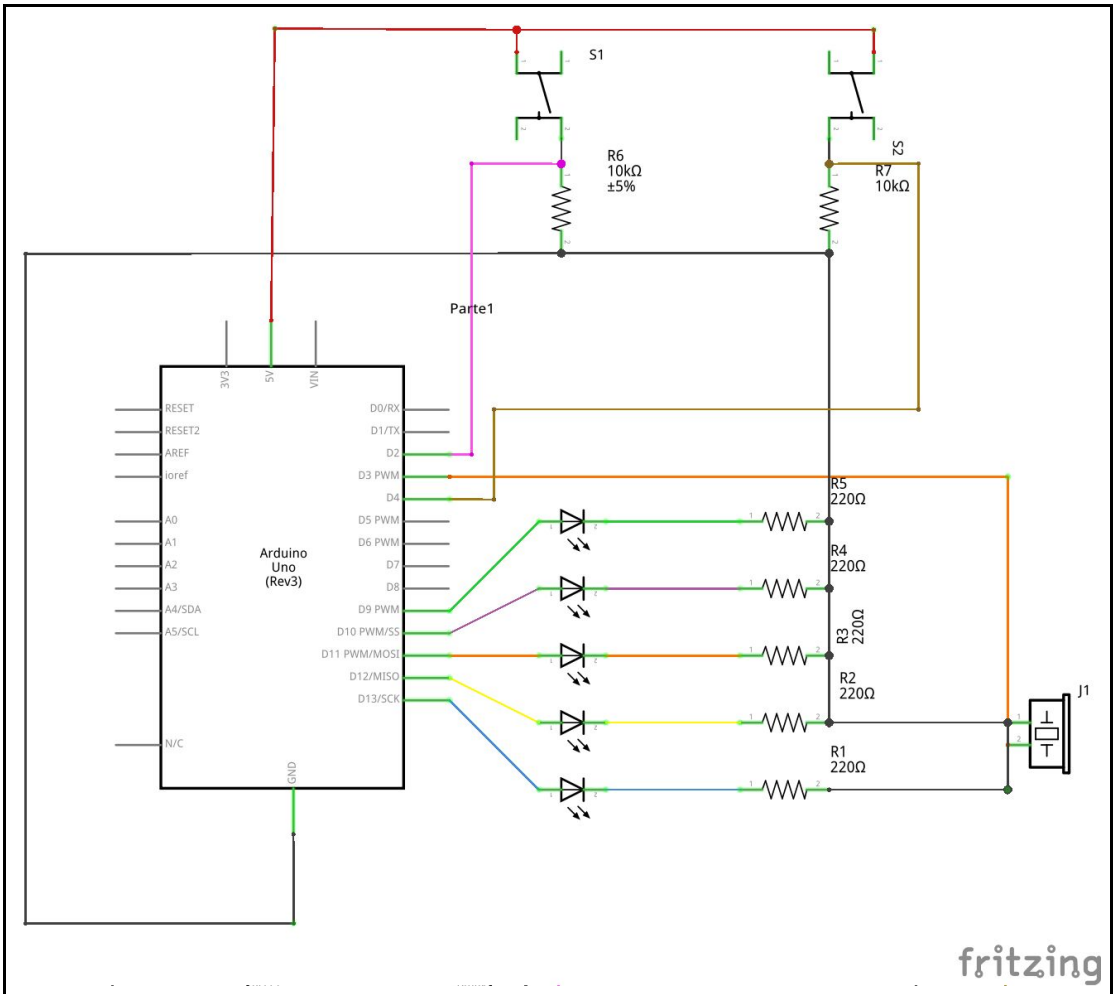
Nº	Descripción	Cantidad	Precio Unitario	Precio Total
1	Arduino	1	8,90 €	8,90 €
2	Diodo LED	5	0.06€	0.3€
3	Pack de Cables Macho-Macho 40 unidades	1	2€	2€
4	Protoboard 420 puntos	1	1,45€	1,45€
5	Resistencia 220Ω	5	0.013€	0.065€
	Resistencia 10kΩ	2	0.08€	0.16€
7	Pulsador	2	0.21€	0.42€
8	Buzzer	1	0.7€	0.7€
9	Potenciómetro 10kΩ	1	2€	2€
			Total	15.99€

c. Esquema de la protoboard



Esquema de la protoboard

d. Esquema electrónico



Esquema electrónico

4. Software:

Programación con comentarios:

```
#include "tonos.h"
// VARIABLES MUSICA
int const TEMPO = 1200;
int melody[] = {
  N_D3, N_D3, N_D4, N_A3, 0, N_GS3, N_G3, N_F3, N_D3, N_F3, N_G3, N_C3, N_C3,
N_D4, N_A3, 0, N_GS3, N_G3, N_F3, N_D3, N_F3, N_G3, N_B2, N_B2, N_D4, N_A3, 0,
N_GS3, N_G3, N_F3, N_D3, N_F3, N_G3, N_AS2, N_AS2, N_D4, N_A3, 0, N_GS3, N_G3,
N_F3, N_D3, N_F3, N_G3, N_D3, N_D3, N_D4, N_A3, 0, N_GS3, N_G3, N_F3, N_D3,
N_F3, N_G3, N_C3, N_C3, N_D4, N_A3, 0, N_GS3, N_G3, N_F3, N_D3, N_F3, N_G3,
N_B2, N_B2, N_D4, N_A3, 0, N_GS3, N_G3, N_F3, N_D3, N_F3, N_G3, N_AS2, N_AS2,
N_D4, N_A3, 0, N_GS3, N_G3, N_F3, N_D3, N_F3, N_G3, N_D4, N_D4, N_D5, N_A4, 0,
N_GS4, N_G4, N_F4, N_D4, N_F4, N_G4, N_C4, N_C4, N_D5, N_A4, 0, N_GS4, N_G4,
N_F4, N_D4, N_F4, N_G4, N_B3, N_B3, N_D5, N_A4, 0, N_GS4, N_G4, N_F4, N_D4,
N_F4, N_G4, N_AS3, N_AS3, N_D5, N_A4, 0, N_GS4, N_G4, N_F4, N_D4, N_F4, N_G4,
N_D4, N_D4, N_D5, N_A4, 0, N_GS4, N_G4, N_F4, N_D4, N_F4, N_G4, N_C4, N_C4,
N_D5, N_A4, 0, N_GS4, N_G4, N_F4, N_D4, N_F4, N_G4, N_B3, N_B3, N_D5, N_A4, 0,
N_GS4, N_G4, N_F4, N_D4, N_F4, N_G4, N_AS3, N_AS3, N_D5, N_A4, 0, N_GS4, N_G4,
N_F4, N_D4, N_F4, N_G4, N_F4, N_F4, N_F4, N_F4, N_F4, N_D4, N_D4, N_D4, N_F4,
N_F4, N_F4, N_G4, N_GS4, N_G4, N_F4, N_D4, N_F4, N_G4, 0, N_F4, N_F4, N_F4,
N_G4, N_GS4, N_A4, N_C5, N_A4, N_D5, N_D5, N_D5, N_A4, N_D5, N_C5, N_F4, N_F4,
N_F4, N_F4, N_F4, N_D4, N_D4, N_D4, N_F4, N_F4, N_F4, N_F4, N_D4, N_F4, N_E4,
N_D4, N_C4, 0, N_G4, N_E4, N_D4, N_D4, N_D4, N_D4, N_F3, N_G3, N_AS3, N_C4,
N_D4, N_F4, N_C5, 0, N_F4, N_D4, N_F4, N_G4, N_GS4, N_G4, N_F4, N_D4, N_GS4,
N_G4, N_F4, N_D4, N_F4, N_F4, N_F4, N_GS4, N_A4, N_C5, N_A4, N_GS4, N_G4, N_F4,
N_D4, N_E4, N_F4, N_G4, N_A4, N_C5, N_CS5, N_GS4, N_GS4, N_G4, N_F4, N_G4,
N_F3, N_G3, N_A3, N_F4, N_E4, N_D4, N_E4, N_F4, N_G4, N_E4, N_A4, N_A4, N_G4,
N_F4, N_DS4, N_CS4, N_DS4, 0, N_F4, N_D4, N_F4, N_G4, N_GS4, N_G4, N_F4, N_D4,
N_GS4, N_G4, N_F4, N_D4, N_F4, N_F4, N_F4, N_GS4, N_A4, N_C5, N_A4, N_GS4,
N_G4, N_F4, N_D4, N_E4, N_F4, N_G4, N_A4, N_C5, N_CS5, N_GS4, N_GS4, N_G4,
N_F4, N_G4, N_F3, N_G3, N_A3, N_F4, N_E4, N_D4, N_E4, N_F4, N_G4, N_E4, N_A4,
N_A4, N_G4, N_F4, N_DS4, N_CS4, N_DS4,
};
int noteDurations[] = {
  16, 16, 8, 6, 32, 8, 8, 8, 16, 16, 16, 16, 16, 8, 6, 32, 8, 8, 8, 16, 16, 16, 16, 16, 8, 6, 32, 8, 8,
8, 16, 16, 16, 16, 16, 8, 6, 32, 8, 8, 8, 16, 16, 16, 16, 16, 8, 6, 32, 8, 8, 8, 16, 16, 16, 16, 16,
8, 6, 32, 8, 8, 8, 16, 16, 16, 16, 16, 8, 6, 32, 8, 8, 8, 16, 16, 16, 16, 16, 8, 6, 32, 8, 8, 8, 16,
16, 16, 16, 16, 8, 6, 32, 8, 8, 8, 16, 16, 16, 16, 16, 8, 6, 32, 8, 8, 8, 16, 16, 16, 16, 16, 8, 6,
32, 8, 8, 8, 16, 16, 16, 16, 16, 8, 6, 32, 8, 8, 8, 16, 16, 16, 16, 16, 8, 6, 32, 8, 8, 8, 16, 16, 16,
16, 16, 8, 6, 32, 8, 8, 8, 16, 16, 16, 16, 16, 8, 6, 32, 8, 8, 8, 16, 16, 16, 16, 16, 8, 6, 32, 8, 8,
8, 16, 16, 16, 8, 16, 8, 8, 8, 8, 4, 16, 8, 16, 8, 8, 8, 16, 16, 16, 16, 16, 8, 8, 16, 8, 8, 8, 8, 8, 8,
8, 8, 16, 16, 16, 2, 8, 16, 8, 8, 8, 8, 4, 16, 8, 16, 8, 8, 8, 8, 8, 16, 8, 16, 8, 8, 8, 8, 8, 8, 8, 16,
```

```
8, 15, 8, 8, 2, 3, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 8, 2, 16, 8, 16, 8, 16, 16, 16,
16, 16, 16, 8, 8, 8, 8,  8, 8, 16, 16, 16, 2, 8, 8, 8, 8, 4, 4, 4, 4, 4, 2, 8, 8, 8, 8, 2, 2, 3, 16,
16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 8, 2, 16, 8, 16, 8, 16, 16, 16, 16, 16, 16, 8, 8, 8, 8,
8, 8, 16, 16, 16, 2, 8, 8, 8, 8, 4, 4, 4, 4, 4, 4, 2, 8, 8, 8, 8, 2, 1
};
int melody_len = sizeof(melody) / sizeof(melody[0]);
int notas = 0;

// VARIABLES JUEGO
int contador = 1;// Sirve para que los led vayan cambiando
int sentido = 0;//Es el sentido de la secuenzia de los leds(derecha-izquierda o
izquierda-derecha)
int acierto = 0;//Numero de aciertos
int vidas = 5;//Numero de vidas que según fallas disminuye
int juego = LOW;//declara si el juego está encendido o apagado

// VARIABLES LEDS
const int ledPin1 = 9;
const int ledPin2 = 10;
const int ledPin3 = 11;
const int ledPin4 = 12;
const int ledPin5 = 13;

// VARIABLES PULSADORES
const int buttonPin = 4;
const int buttonPin2 = 2;
int buttonState = LOW;
int buttonState2;
int lastButtonState2 = LOW;
//VARIABLE DEL ALTAVOZ
const int speakerPin =  3;

//VARIABLES PULSADOR CON MEMORIA
unsigned long lastDebounceTime = 0;  // the last time the output pin was toggled
unsigned long debounceDelay = 50;    // the debounce time; increase if the output flickers

//VARIABLES PARA PODER UTILIZAR LA FUNCIÓN MILLIS
unsigned long lastTime = 0;
unsigned long intervalo = 300;

void setup() {
  // Declaramos los LEDS
  pinMode (ledPin1, OUTPUT); //Se declara como salida
  pinMode (ledPin2, OUTPUT); //Se declara como salida
  pinMode (ledPin3, OUTPUT); //Se declara como salida
  pinMode (ledPin4, OUTPUT); //Se declara como salida
```

```
  pinMode (ledPin5, OUTPUT); //Se declara como salida
  // Declaramos los pulsadores
  pinMode(buttonPin, INPUT);
  pinMode(buttonPin2, INPUT);
  // abrimos el puerto serie
  Serial.begin(9600);
}

void loop() {

  // LECTURA PULSADOR MEMORIA
  leerpulsador();

  // JUEGO ON JUEGO OFF
  if (juego == HIGH) {
    on();
  }
  else if (juego == LOW) {
    off();
  }
}

//PULSADOR CON MEMORIA
void leerpulsador() {
  int reading = digitalRead(buttonPin2);

  if (reading != lastButtonState2) {
    // reset the debouncing timer
    lastDebounceTime = millis();
  }
  if ((millis() - lastDebounceTime) > debounceDelay) {
    // whatever the reading is at, it's been there for longer than the debounce
    // delay, so take it as the actual current state:

    // if the button state has changed:
    if (reading != buttonState2) {
      buttonState2 = reading;

      // only toggle the LED if the new button state is HIGH
      if (buttonState2 == HIGH) {
        juego = !juego;
      }
    }
  }
  lastButtonState2 = reading;
}
```

```
//JUEGO
void on() {
  // lectura pulsador de detección
  buttonState = digitalRead(buttonPin);

  // CONTADOR
  unsigned long  currentTime = millis();
  if (currentTime - lastTime >= intervalo) {
    lastTime = currentTime;
    // CONTADOR
    if (sentido == 0)
      contador = contador + 1;
    if (sentido == 1)
      contador = contador - 1;
    if (contador > 4)
      sentido = 1;
    if (contador < 2)
      sentido = 0;
  }

  // ACTIVACION LEDS
  if (contador == 1) led(HIGH, LOW, LOW, LOW, LOW);
  else if (contador == 2) led(LOW, HIGH, LOW, LOW, LOW);
  else if (contador == 3) led(LOW, LOW, HIGH, LOW, LOW);
  else if (contador == 4) led(LOW, LOW, LOW, HIGH, LOW);
  else if (contador == 5)led(LOW, LOW, LOW, LOW, HIGH);


  // DETECCION
  if ( buttonState == HIGH) {
    if (contador == 3) {
      acierto = acierto + 1;
      intervalo = intervalo - intervalo * 0.15;
      analogWrite(speakerPin, 100);
      delay(500);
      analogWrite(speakerPin, 0);
      lastTime = currentTime;

    }
    else {
      analogWrite(speakerPin, 25);
      led(HIGH, HIGH, LOW, HIGH, HIGH);
      delay(500);
      vidas = vidas - 1;
      analogWrite(speakerPin, 0);
      lastTime = currentTime;
```

```
    }
  }

  // FIN DE JUEGO
  if (acierto >= 7) {
    victoria();
  }
  else if (vidas <= 1) {
    derrota();
  }
}

void off() {
  led(LOW, LOW, LOW, LOW, LOW);
  intervalo = 300;
  acierto = 0;
  notas = 0;
  vidas = 5;
  contador = 1;
}

void led(int stateLed1, int stateLed2, int stateLed3, int stateLed4, int stateLed5) {
  digitalWrite(ledPin1, stateLed1);
  digitalWrite(ledPin2, stateLed2);
  digitalWrite(ledPin3, stateLed3);
  digitalWrite(ledPin4, stateLed4);
  digitalWrite(ledPin5, stateLed5);
}
//VICTORIA
void victoria() {
  led(HIGH, HIGH, HIGH, HIGH, HIGH);
  //MÚSICA VICTORIA
  for (int thisNote = 0; thisNote < melody_len; thisNote++) {
    if (notas <= 240) {
      int noteDuration = TEMPO / noteDurations[thisNote];
      tone(3, melody[thisNote], noteDuration);
      notas = notas + 1;
      int pauseBetweenNotes = noteDuration * 1.45;
      delay(pauseBetweenNotes);
      noTone(3);
    }
    else if (notas > 240) {
      juego = LOW;
    }
  }
```

```
}
//DERROTA
void derrota() {
  march();
  juego = LOW;
}
//MÚSICA DERROTA
void beep (unsigned char speakerPin, int frequencyInHertz, long timeInMilliseconds) {
  int x;
  long delayAmount = (long)(1000000 / frequencyInHertz);
  long loopTime = (long)((timeInMilliseconds * 1000) / (delayAmount * 2));
  for (x = 0; x < loopTime; x++) {
    analogWrite(speakerPin, 255);
    delayMicroseconds(delayAmount);
    analogWrite(speakerPin, 0);
    delayMicroseconds(delayAmount);
  }
  delay(20);
  //a little delay to make all notes sound separate
}


void march()
{
  beep(speakerPin, a, 500);
  beep(speakerPin, a, 500);
  beep(speakerPin, a, 500);
  beep(speakerPin, f, 350);
  beep(speakerPin, cH, 150);

  beep(speakerPin, a, 500);
  beep(speakerPin, f, 350);
  beep(speakerPin, cH, 150);
  beep(speakerPin, a, 1000);
  //first bit

  beep(speakerPin, eH, 500);
  beep(speakerPin, eH, 500);
  beep(speakerPin, eH, 500);
  beep(speakerPin, fH, 350);
  beep(speakerPin, cH, 150);

  beep(speakerPin, gS, 500);
  beep(speakerPin, f, 350);
  beep(speakerPin, cH, 150);
  beep(speakerPin, a, 1000);
```

```
  //second bit...

  beep(speakerPin, aH, 500);
  beep(speakerPin, a, 350);
  beep(speakerPin, a, 150);
  beep(speakerPin, aH, 500);
  beep(speakerPin, gSH, 250);
  beep(speakerPin, gH, 250);

  beep(speakerPin, fSH, 125);
  beep(speakerPin, fH, 125);
  beep(speakerPin, fSH, 250);
  delay(250);
  beep(speakerPin, aS, 250);
  beep(speakerPin, dSH, 500);
  beep(speakerPin, dH, 250);
  beep(speakerPin, cSH, 250);
  //start of the interesting bit

  beep(speakerPin, cH, 125);
  beep(speakerPin, b, 125);
  beep(speakerPin, cH, 250);
  delay(250);
  beep(speakerPin, f, 125);
  beep(speakerPin, gS, 500);
  beep(speakerPin, f, 375);
  beep(speakerPin, a, 125);

  beep(speakerPin, cH, 500);
  beep(speakerPin, a, 375);
  beep(speakerPin, cH, 125);
  beep(speakerPin, eH, 1000);
}
```

Y también en una pestaña diferente llamada "tonos.h" definimos los tonos para que no hubiese tanta programación en una misma hoja e hiciese todo más lioso.Dicha programación es:

```
#define c 261
#define d 294
#define e 329
#define f 349
#define g 391
#define gS 415
#define a 440
#define aS 455
#define b 466
#define cH 523
#define cSH 554
#define dH 587
#define dSH 622
#define eH 659
#define fH 698
#define fSH 740
#define gH 784
#define gSH 830
#define aH 880
#define N_B0  31
#define N_C1  33
#define N_CS1 35
#define N_D1  37
#define N_DS1 39
#define N_E1  41
#define N_F1  44
#define N_FS1 46
#define N_G1  49
#define N_GS1 52
#define N_A1  55
#define N_AS1 58
#define N_B1  62
#define N_C2  65
#define N_CS2 69
#define N_D2  73
#define N_DS2 78
#define N_E2  82
#define N_F2  87
#define N_FS2 93
#define N_G2  98
#define N_GS2 104
#define N_A2  110
#define N_AS2 117
```

```
#define N_B2  123
#define N_C3  131
#define N_CS3 139
#define N_D3  147
#define N_DS3 156
#define N_E3  165
#define N_F3  175
#define N_FS3 185
#define N_G3  196
#define N_GS3 208
#define N_A3  220
#define N_AS3 233
#define N_B3  247
#define N_C4  262
#define N_CS4 277
#define N_D4  294
#define N_DS4 311
#define N_E4  330
#define N_F4  349
#define N_FS4 370
#define N_G4  392
#define N_GS4 415
#define N_A4  440
#define N_AS4 466
#define N_B4  494
#define N_C5  523
#define N_CS5 554
#define N_D5  587
#define N_DS5 622
#define N_E5  659
#define N_F5  698
#define N_FS5 740
#define N_G5  784
#define N_GS5 831
#define N_A5  880
#define N_AS5 932
#define N_B5  988
#define N_C6  1047
#define N_CS6 1109
#define N_D6  1175
#define N_DS6 1245
#define N_E6  1319
#define N_F6  1397
#define N_FS6 1480
#define N_G6  1568
#define N_GS6 1661
```

```
#define N_A6  1760
#define N_AS6 1865
#define N_B6  1976
#define N_C7  2093
#define N_CS7 2217
#define N_D7  2349
#define N_DS7 2489
#define N_E7  2637
#define N_F7  2794
#define N_FS7 2960
#define N_G7  3136
#define N_GS7 3322
#define N_A7  3520
#define N_AS7 3729
#define N_B7  3951
#define N_C8  4186
#define N_CS8 4435
#define N_D8  4699
#define N_DS8 4978
```

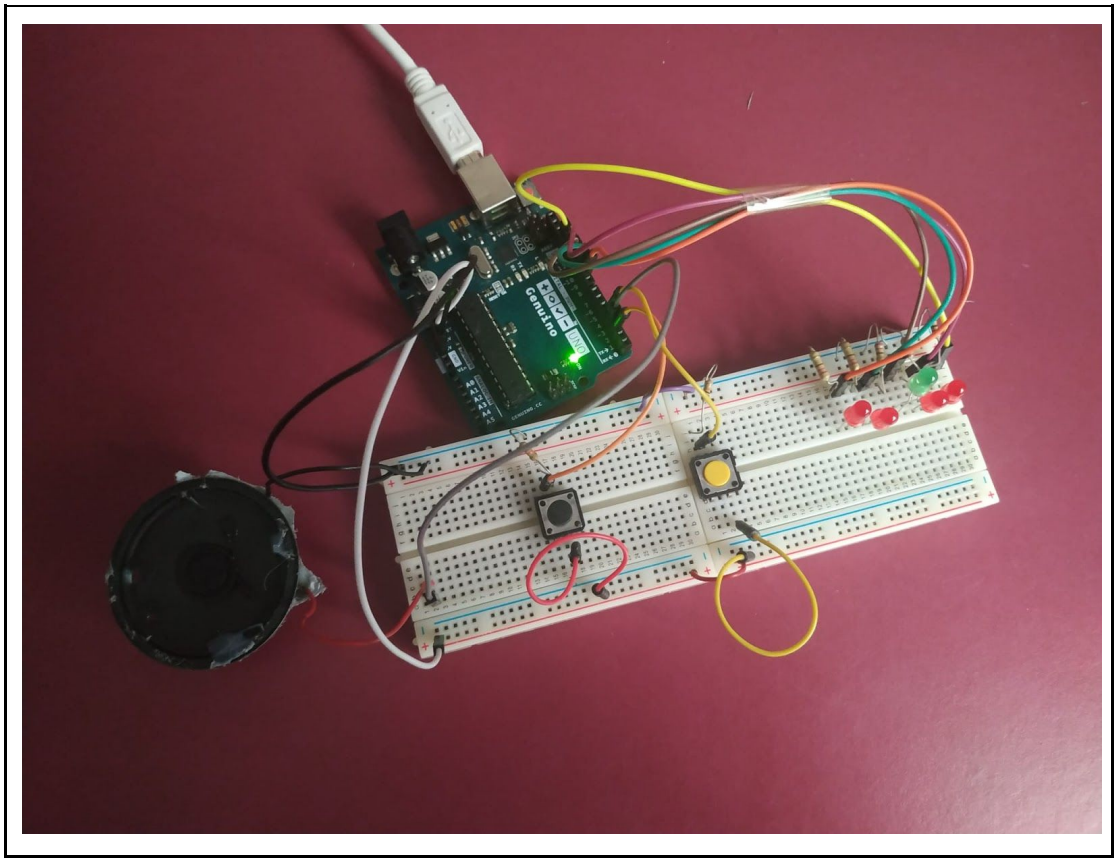
5. Funcionamiento:

a. Vídeos :

[Vídeo victoria]

[Vídeo derrota]

b. Fotos:



6. Evaluación:

a. Qué funciona bien y qué se puede mejorar.

Todo funciona correctamente.

b. Problemas que hemos tenido y soluciones.

Problemas de programación con el contador.

c. Propuestas de mejora y ampliación del proyecto

Mejorar la estética del trabajo:

- Que el color de los leds sea distinto para que visiblemente quede mejor.
- Añadir más leds.
- Crear una carcasa.