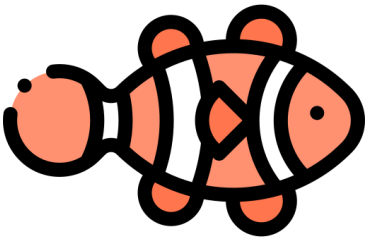
Universidad Rey Juan Carlos

Grado en Ingeniería de Computadores
Diseño de Sistemas Empotrados
Curso (2021 - 2022)

ALBANTA LA PEZ QUE CANTA:

MAQUETA DE PEZ QUE SINCRONIZA SONIDO Y MOVIMIENTO



Chris Andaya Bernales
Edson Víctor Fernández Espinoza
Lucía Fresno Olmeda

- Índice

1. Proyecto

El proyecto presentado consiste en una maqueta de un pez que se mueve al ritmo de la música que esté sonando. Esta maqueta puede realizar movimiento en tres puntos distintos: la cabeza del pez, que se mueve hacia el espectador y vuelve a su posición inicial; la cola, que realiza el mismo movimiento; y la boca, que se mueve de arriba a abajo para dar la sensación de que el pez estuviera cantando.

En cuanto al sonido, el pez podrá "cantar" dos canciones distintas: "Barbie Girl", del grupo Aqua y "Bajo el Mar", de la película "La Sirenita". El usuario podrá cambiar de canción pulsando un botón o pausar la canción que esté sonando pulsando otro.

La idea de este proyecto ha surgido de los famosos vídeos de Big Mouth Billy Bass, aunque la estética es ligeramente distinta: la maqueta simula un pez payaso y las canciones no son las que suelen aparecer en estos vídeos.

Aquí se puede ver un vídeo de ejemplo para que se pueda visualizar más claramente la idea del proyecto: https://www.youtube.com/watch?v=m3CjHPXmZbw

2. Implementación

La implementación y desarrollo del proyecto se puede separar en tres bloques: la creación del código, su funcionamiento con el circuito de arduino y la creación de la maqueta.

Para elaborar el código se ha utilizado el software Arduino IDE 1.8.16 y se han usado varias referencias, tanto encontradas en Internet como en las prácticas guiadas de la asignatura.

Por otra parte, para realizar el circuito se han usado dos placas Arduino Uno, ya que se encontraron problemas con el multitarea, los cuales se comentarán a fondo más adelante. Además de esto, se usaron las mismas referencias que para el código, ya que estos dos bloques van muy unidos.

Por último, se ha confeccionado una maqueta, usando goma eva y cosiéndola para conseguir la forma deseada. Esta maqueta se ha pintado para obtener la estética deseada y se ha montado en unas tablas de cartón pluma.

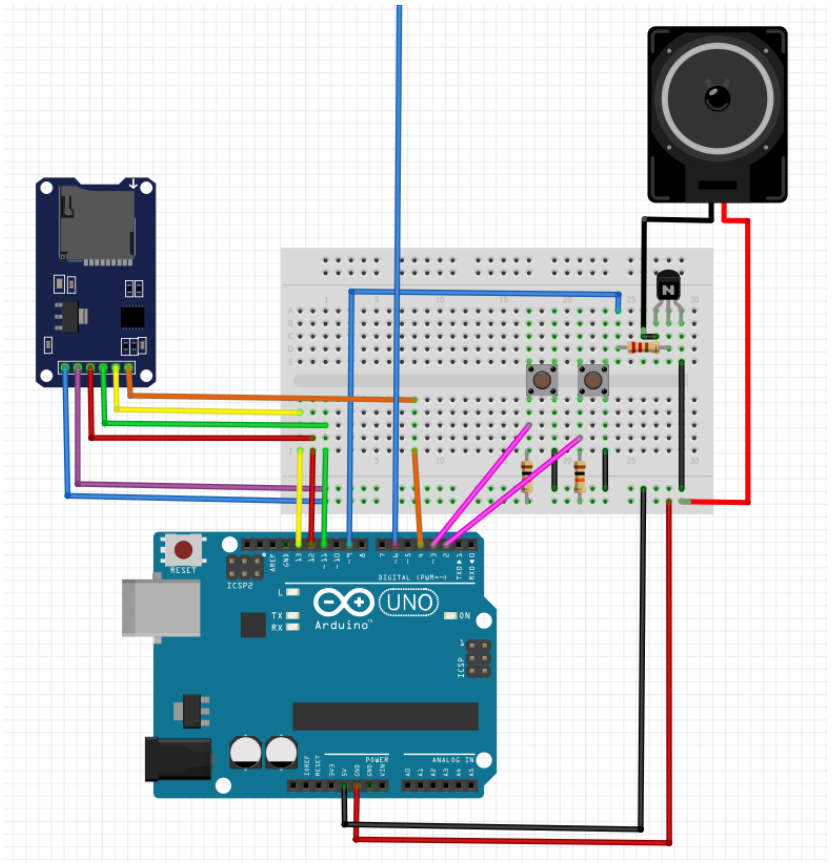
En los siguientes apartados se va a explicar más detalladamente el proceso seguido en el desarrollo del proyecto y las dificultades que se han ido encontrando.

3. Pasos dados

Durante las primeras semanas el trabajo se centró en la implementación del código y el circuito, ya que la maqueta no estaba aún preparada.

Lo primero que se quiso desarrollar fue el funcionamiento del sonido. Al principio se pensó en usar el buzzer proporcionado por la universidad, pero se concluyó en que la calidad de sonido producida por el buzzer era bastante peor de la que queríamos conseguir. Es por esto que durante la investigación para esta parte del proyecto se decidió que altavoces comprar y se decidió comprar un módulo de tarjeta SD, de forma que las canciones estuvieran ahí almacenadas. Además, se instalaron y utilizaron las librerías TMRpcm.h, SD.h y SPI.h, las cuales permiten leer el contenido de la SD, asociar la SD a un pin concreto y reproducir archivos .wav, ya que los audios se encuentran en este formato.

Una vez se consiguieron todos estos materiales, se comenzó a montar el circuito y comprobar que funcionaba correctamente. El código usado inicialmente (el cuál es muy parecido al código final en esta parte del proyecto) se basó en un tutorial encontrado en Youtube, en el cuál se explica cómo reproducir audio con arduino. El circuito desarrollado finalmente para esta parte es el siguiente:



Es importante tener en cuenta de este circuito el cable que sale del pin 6 y no está conectado a nada. La razón de esto es que este mismo estará conectado también a la placa Arduino del circuito realizado para el movimiento, de forma que se pueda coordinar el inicio y la pausa de este último con el de la música.

Cuando se consiguió que esto funcionase correctamente, el trabajo se centró en el movimiento del pez. Antes de comenzar a pensar en el código se hizo una pequeña prueba con una maqueta provisional. El objetivo de esta prueba era determinar si los servomotores tendrían fuerza suficiente como para mover las distintas partes del pez o si iba a ser necesario comprar motores más potentes.

Tras realizar esta prueba se comprobó que los servomotores serían suficientes para mover la maqueta y se decidió que la maqueta debía hacerse en 4 piezas separadas para que no tuviera problemas al moverse: una pieza para la cola, otra para el cuerpo, otra para la cabeza y una para la boca. Por último se concluyó que el pez iba a estar relleno de esponja (para que mantuviera su forma) y que los servos se iban a conectar con las distintas partes de la maqueta usando lápices y alambre.

Llegados a este punto se comenzaron a hacer tareas en paralelo: por un lado se creaba la maqueta final y por otro se desarrollaba el código para coordinar el movimiento con el sonido. La maqueta se confeccionó en casa por uno de los integrantes del grupo y se pintó por otro de ellos. Por otro lado, el código se fue desarrollando en las horas de clase en la universidad.

El código para coordinar el movimiento con el sonido ha dado bastantes problemas y ha requerido mucha prueba y error.

En un primer momento se priorizó evitar comprar un micrófono, por lo que se barajaron varias alternativas. La idea que en un principio parecía más lógica era implementar el movimiento del pez preprogramado. Para ello, se pensó en crear una función (llamada “motor”) para que esta se ejecutase en el momento en el que empezase a sonar una canción. Para que esto funcionase correctamente, había que crear una función “motor” para cada una de las canciones que queríamos que sonaran.

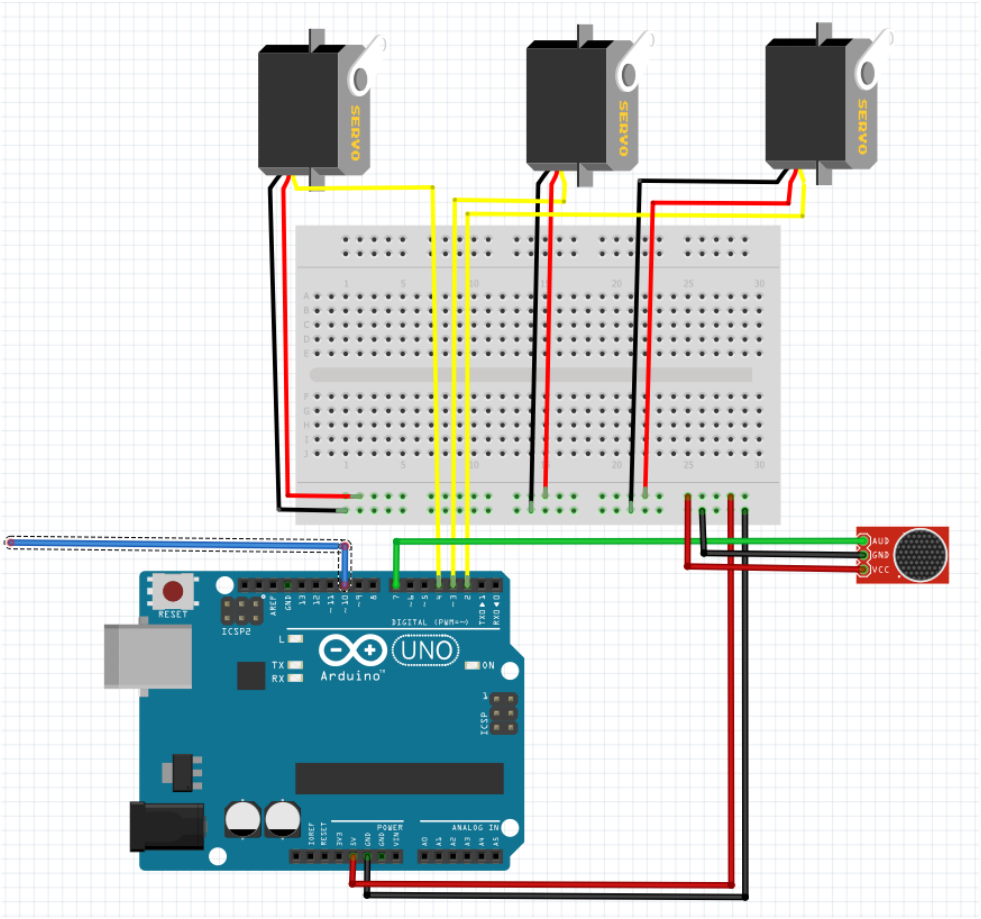
Partiendo de este planteamiento, se pensó en modificar el código elaborado para la práctica obligatoria (para la cuál usamos como referencias las prácticas guiadas analógicas 2 y 4). En esta entrega anterior, se añadieron las canciones con una función preprogramada que se generó a partir de la propia canción en formato MIDI. Esta función se obtuvo de un convertidor online y su funcionamiento se basaba en hacer sonar el buzzer con una nota y un tiempo concreto y añadir delays hasta la siguiente nota, de forma que se reprodujese la melodía deseada.

Con estos conocimientos, se quiso probar a generar esta misma función a partir de los audios MIDI y sustituir cada función tone() (que hace que se reproduzca el sonido de una nota) por la función servomotor.write() (que hace que se mueva el servomotor una cantidad concreta de grados). Tras programar el código correspondiente a la primera parte de una de las canciones se observó que esta opción no iba a ser óptima, ya que las funciones iban a ser excesivamente largas. Sin embargo, se probó a ejecutar el código y se pudo ver que este no funcionaba correctamente. Se hicieron varias pruebas y se buscó información hasta llegar a la conclusión de que Arduino no soporta el multitarea, por lo que no se podría producir sonido y movimiento al mismo tiempo.

Dadas estas circunstancias se llegó a una solución: usar dos Arduino, uno para reproducir el sonido y otro para recibirlo y mover los servomotores cuando tuviera la señal. Se compró un micrófono y se programó un código para el movimiento de los servos. Se decidió programar un movimiento fijo para los servomotores de la cabeza y la cola (que se mueven cada cierto número de segundos) y un movimiento que dependiera de la señal que reciba el micrófono para el movimiento de la boca. Para que esto funcionara correctamente se ajustó la sensibilidad del micrófono, de forma que este solo captase la señal de sonido de la melodía de las canciones (con un volumen más alto) y no la base musical.

Habiendo hecho estos ajustes se programó un código usando delays para controlar el movimiento de la cabeza y la cola, pero este dió problemas, ya que los delays hacían que toda la ejecución se parase unos segundos y que el movimiento se descoordinaba. Tras buscar otras alternativas por Internet se encontró un código basado en “tareas”, el cual se adaptó y funcionó correctamente, ya que este no usaba delays.

El código creado finalmente hace uso de la función millis(), la cual devuelve el tiempo actual. Gracias a esta función se ha podido medir el tiempo que pasaba entre un movimiento y el siguiente sin hacer uso de delays. Asociado a este código se elaboró el siguiente circuito:



Por último, se montó todo el circuito dentro de la maqueta ya terminada y se comprobó que todo funcionaba bien. Se escondieron los cables y se prepararon las baterías y se dió por terminado el proyecto.

5. Problemas y soluciones

Los mayores problemas que se han encontrado en el desarrollo del proyecto se han dado a la hora de coordinar el sonido y el movimiento. Relacionado con esto se han encontrado dos problemáticas principales: el uso del micrófono y la programación multitarea.

El primero de ellos, el micrófono, fue relativamente sencillo de solucionar. El principal contratiempo relacionado con este fue que no se pensaba obtener uno, ya que se confiaba en que Arduino fuese capaz de detectar y procesar la señal de sonido sin necesidad de usar un micrófono. Cuando se comprobó que esto no era posible, se optó por coordinar el movimiento y el sonido usando funciones preprogramadas (a partir de audios MIDI), como ya se ha comentado anteriormente en la memoria.

Tras estas pruebas se concluyó que esto no era posible y se decidió comprar un micrófono. Con la ayuda del micrófono todo resultó más sencillo, pero se encontraron algunos problemas más: no se detectaba bien el sonido de la música pero sí se detectaban los ruidos. Esto se resolvió fácilmente: se subió el volumen de los altavoces y se ajustó la sensibilidad del micrófono hasta lograr el efecto deseado.

El segundo y último problema fue la programación multitarea. Esta cuestión estaba causada por el desconocimiento a la hora de programar, dado que en un primer momento no se sabía que Arduino no es capaz de ejecutar varios procesos al mismo tiempo en hilos distintos. La solución para este inconveniente no fue muy compleja, pero requirió un tiempo de investigación. Gracias a esta misma se llegó a la conclusión de que no se debían usar delays en el código. Tras esta revelación se desarrolló un código secuencial que da la sensación de ser multitarea (aunque no lo es realmente), ya que las instrucciones se ejecutan seguidas y a gran velocidad.

6. Costes de materiales

Los materiales y costes que se han requerido para la realización del proyecto han sido los siguientes:

- 2 Arduino Uno: no representaron ningún coste, ya que una la proporcionó la universidad y otra pertenecía a uno de los integrantes del grupo.
- 2 placas de inserción: se dió la misma situación que con los Arduino Uno.
- 2 altavoces: costaron 7,99 euros los dos y se compraron por Amazon.
- 1 módulo de tarjeta SD: costó 4,99 euros y se compró por Amazon.
- 1 tarjeta microSD: no costó dinero, ya que pertenecía a uno de los integrantes del grupo.
- 1 micrófono: costó 1 euro y se compró por Wallapop.
- 3 servomotores: no representaron ningún coste, ya que algunos compañeros que no los iban a usar los prestaron.
- 2 botones: al igual que lo anterior, pertenecen a las cajas que nos proporcionó la universidad.
- 1 transistor: fue prestado por un compañero de otro grupo, por lo que fue gratis.
- 1 resistencia de 220 Ω y 2 resistencias de 10k Ω: no causaron ningún coste adicional. Se obtuvieron de la caja de materiales proporcionada por la universidad y de materiales que poseía uno de los integrantes del grupo.
- Cableado: se dió la misma situación que con las resistencias.
- 2 cables de alimentación de la batería: solo fue necesario comprar uno, ya que el otro lo proporcionó uno de los integrantes del grupo. El coste que supuso el cable comprado fue de 1 euro.
- 2 pilas de 9 voltios: costaron 2,50 cada una y se compraron en una tienda de tecnología.
- Goma eva, esponjas y alambre: costaron 4,10 euros en distintos bazares.
- Cartón pluma: no representaron costes, ya que eran materiales de un integrante del grupo.
- Pintura acrílica satinada: costó 3,80 euros y se compró en un bazar.

7. Código

Para este proyecto se han desarrollado dos códigos distintos: uno para la reproducción de la música y otro para el movimiento de los servomotores coordinado con esta. A continuación se muestran ambos:

Código desarrollado para la reproducción del sonido

```
#include <SD.h> // Se incluye la libreria SD
#include <SPI.h> // Se incluye la libreria para la comunicacion
SPI
#include <TMRpcm.h> //  Se incluye la libreria para reproducir
.WAV
#define SD_ChipSelectPin 4  // Selecciona la tarjeta SD. Puede ser
//cualquier pin que NO se esté utilizando ya.

TMRpcm Audio;   // Se crea un objeto para usar las funciones de la
//libreria TMRpcm

bool boton_A = 0; //Variable que guarda el estado del pulsador de
//cambiar de cancion
bool boton_B = 0; //Variable que guarda el estado del pulsador de
//pausar
int song = 0; //Se inician en 0 para que cuando se incremente la
//cancion al principio se reproduzca la cancion 1
int last_song = 0; //Almacena el número de la ultima cancion
/*  CONEXIONES DEL MÓDULO DE LA TARJETA SD

  Arduino Nano/Arduino UNO
  12 ------------------> MISO
  11 ------------------->MOSI
  13 ------------------->SCK
  4 ------------------->CS
  9 ------------------->Speaker
  GND ------------------->GND
  5V ------------------> VCC
*/

void setup() {
  Serial.begin(9600); //Se inicializa el puerto serie
  Serial.println("Inicializando");

  Audio.speakerPin = 9; //Selecciona la salida de audio: pin 9
  Audio.quality(1); // Mejoramos la calidad de sonido (puede ser 0
o  //1)
```

```
  Audio.setVolume(5); // Se selecciona el volumen: valor entre 0 y
7


  //Inicialización de la SD
  if (!SD.begin(SD_ChipSelectPin)) {  // Comprueba si la tarjeta
SD //se ha inicializado correctamente
    Serial.println("Fallo de la tarjeta SD, revisa las conexiones
papu :v");
    return;   // Si la tarjeta SD falla, el programa termina
  } else {
    Serial.println("SD inicializada correctamente!");
  }


  pinMode(3, INPUT); // Declara el boton de pasar de cancion como
//entrada
  pinMode(2, INPUT); // Declara el boton de pausar la cancion
  pinMode(6, OUTPUT); //Declara el pin 6 como salida, para poder
//pasar la señal al otro Arduino Uno y coordinar las acciones de
los //botones con el movimiento
// De esta forma nos aseguramos de que el pez no se mueve si no
está //sonando ninguna canción
}

void loop() {

  boton_A = digitalRead(3); //Se lee el boton de pasar de cancion
  boton_B = digitalRead(2); //Se lee el boton de pausa

  if (boton_A == 1) { // Si pulsas el pulsador se incrementa la
//cancion a reproducir
    if (song <= 1)
      song++; // Se incrementa la cancion a reproducir
    else
      song = 1; // Si se ha llegado a la ultima cancion, vuelve a
la //cancion numero 1
    while (boton_A == 1) {
      boton_A = digitalRead(3);  //Espera a que levantes el dedo
del //pulsador para continuar
    }
  }

  if (boton_B == 1) {
    digitalWrite(6, LOW);
    Audio.pause(); //Pausa la cancion que está sonando
    Serial.println("Cancion pausada o renaudada");
    while (boton_B == 1) {
```

```
      boton_B = digitalRead(2);  //Espera a que levantes el dedo
del //pulsador para continuar
    } }


  if (song != last_song) { //Reproduce el audio una unica vez
//siempre que se haya cambiado de cancion

    switch (song)
    {
      case 1:
        digitalWrite(6, HIGH); // Se envía una señal HIGH para que
//los servos conectados a la otra placa sepan que se tienen que
//mover
        Audio.play((char*)"barbie.wav");
        Audio.loop(1);
        while (!Audio.isPlaying()) {
          digitalWrite(6, LOW); // Cuando se para el audio los
//servos deben pararse también
        }
        Serial.println("Cancion 1");
        break;

      case 2:
        digitalWrite(6, HIGH);
        Audio.loop(1);
        Audio.play((char*)"sirenita.wav");
        while (!Audio.isPlaying()) {
          digitalWrite(6, LOW);
        }
        Serial.println("Cancion 2");
        break;

    }

    last_song = song;
  }

}
```

Código desarrollado para la coordinación del movimiento:

```
#include <Servo.h>
Servo boca, cola, cabeza;
int pinMicro = 7;
int micState;
int estadoCabeza = 0;
int estadoCola = 0;
unsigned long tAntCabeza = millis();
unsigned long tAntCola = millis();
long tCola = 1000;
long tCabeza = 3000;

bool boton = 0; //Variable que guarda el estado del pulsador de
//cambiar de cancion

void setup() {
  //Inicializamos el puerto serie
  Serial.begin(9600);

  //Pinneamos el micrófono como entrada
  pinMode(pinMicro, INPUT);

  //Attach los servos con sus pines
  cola.attach(2);
  cabeza.attach(4);
  boca.attach(3);

  //Inicializamos su posición inicial
  cola.write(0);
  cabeza.write(0);
  boca.write(70);


  //Partimos del sensor apagado
  digitalWrite(pinMicro, LOW);

  //botones
   pinMode(10, INPUT); // Declara el pin que recibe la señal del
//boton para saber si los servos se deben mover o no
}
```

```
void loop() {
  boton = digitalRead(10); //Se lee la señal del boton de entrada


   while (boton == HIGH) { //El movimiento ocurre siempre que la
//señal que se reciba del botón sea HIGH
    unsigned long currentTime = millis();

    micState  = digitalRead(pinMicro);

     //El estado del servo de la boca cambia según la señal que
//reciba del micrófono
    if (micState == HIGH) {
      boca.write(0);
    } else {
      boca.write(70);
    }

     //El movimiento de la cola y la cabeza ocurre cada cierto
//tiempo. El tiempo que ha pasado entre un el momento actual t el
//último movimiento se mide con la operación
     //currentTime - tAnt: si este tiempo es mayor al establecido
//para que ocurra el movimiento, el servo se mueve y la variable
//tAnt se actualiza
    if (currentTime - tAntCabeza > tCabeza) {
      tAntCabeza = currentTime;

      if (estadoCabeza == 0) {
        cabeza.write(180);
        estadoCabeza = 1;
      }
      else if (estadoCabeza == 1) {
        cabeza.write(0);
        estadoCabeza = 0;
      }
    }
    if (currentTime - tAntCola > tCola) {
      tAntCola = currentTime;

      if (estadoCola == 0) {
        cola.write(50);
        estadoCola = 1;
      } else if (estadoCola == 1)  {
        cola.write(0);
        estadoCola = 0;
      }
    }
    boton = digitalRead(10);
```

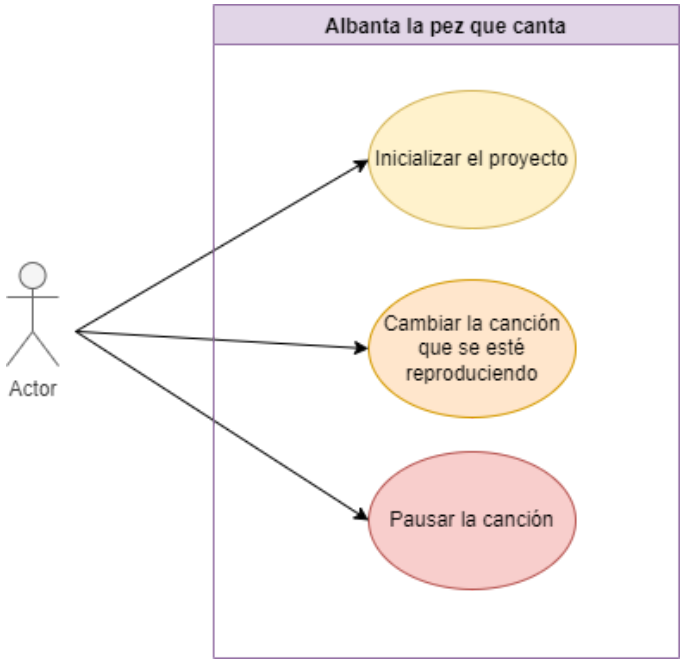
```
}

Serial.println(pinMicro, DEC);}
```

8. Casos de uso

Los casos de uso de este proyecto son bastante reducidos, para que este mismo sea fácilmente usable. Además, a nivel de usabilidad su único objetivo es el entretenimiento, por lo que la complejidad del proyecto se centra en su implementación y no tanto en aportar una amplia variedad de funcionalidades al usuario.

Por lo tanto las acciones que podrá realizar el usuario en el proyecto son las siguientes:

- **Inicializar el proyecto**: encenderlo para que este comience a funcionar. Para ello, el usuario debe pulsar el botón de "On/Next Song".
- **Cambiar la canción que se esté reproduciendo**: se reproducirá la siguiente canción desde el principio. Para que esto ocurra se debe volver a pulsar el botón "Play".
- **Pausar la canción**: la reproducción de la canción se parará al pulsar el botón "Play/Pause". Si se vuelve a pulsar este botón se reanudará la reproducción en el momento en el que paró.



9. Reparto de tareas

Todas las clases se han empleado para el desarrollo del proyecto, más concretamente en los apartados de programación y circuito. Por otro lado, como trabajo fuera de clase se ha realizado la creación y montaje de la maqueta y el desarrollo de la memoria y presentación. Por lo general el trabajo ha sido bastante equitativo.

10. Bibliografía

Enlaces utilizados en el desarrollo del código y la circuitería:

- Big Mouth Billy Bass: https://www.youtube.com/watch?v=m3CjHPXmZbw
- Implementación del sonido: https://www.youtube.com/watch?v=DIrd697688s
- Información sobre la librería TMR: https://github.com/TMRh20/TMRpcm/wiki
- Convertidor de MP3 a MIDI: https://evano.com/audio/mp3-to-midi-online-converter
- Convertidor de MP3 a .WAV: https://convertio.co/es/mp3-wav/
- Convertidor de MIDI a Arduino: https://extramaster.net/tools/midiToArduino/
- Código de referencia usado para los servomotores: https://descubrearduino.com/como-hacer-multitarea-con-arduino/
- Otras páginas que se consultaron durante el desarrollo del proyecto:
    - https://forum.arduino.cc/t/how-to-synchronize-servo-with-audio-mp3-files-like-talking-robot/618049
    - https://www.youtube.com/watch?v=RTwhn8-zvr8

Enlaces de los materiales comprados por Internet:

- Altavoces: https://www.amazon.es/dp/B0738NLFTG?tag=mialtavoz.com-21&linkCode=ogi&th=1&psc=1&keywords=altavoz%20arduino
- Módulo de tarjeta SD: https://www.amazon.es/azdelivery-Reader-Tarjeta-memoria-Arduino/dp/B06X1DX5WS/ref=mp_s_a_1_1_sspa?keywords=Arduino+Sd+Card+Shield&qid=1636715093&refinements=p_76%3A831314031&rps=1&s=computers&sr=1-1-spons&psc=1&smid=A1X7QLRQH87QA3&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEzTVJZUEU3UkxEV1RCJmVuY3J5cHRlZElkPUEwNDY1MjcwMjNPU1IxRVk0TDhRVSZlbmNyeXB0ZWRBZElkPUEwNzA1ODU3REpBU043Q1o0QU5KJndpZGdldE5hbWU9c3BfcGhvbmVfc2VhcmNoX2F0ZiZhY3Rpb249Y2xpY2tSZWRpcmVjdCZkb05vdExvZ0NsaWNrPXRydWU=