Proyecto final de CFGS:

Sincronización de brazo robótico con contenido audiovisual

IES Politécnico Jesús Marín

Streater Arranz, Eduardo

CFGS Mantenimiento electrónico

Tutor: Enrique Norro Gañán

Índice

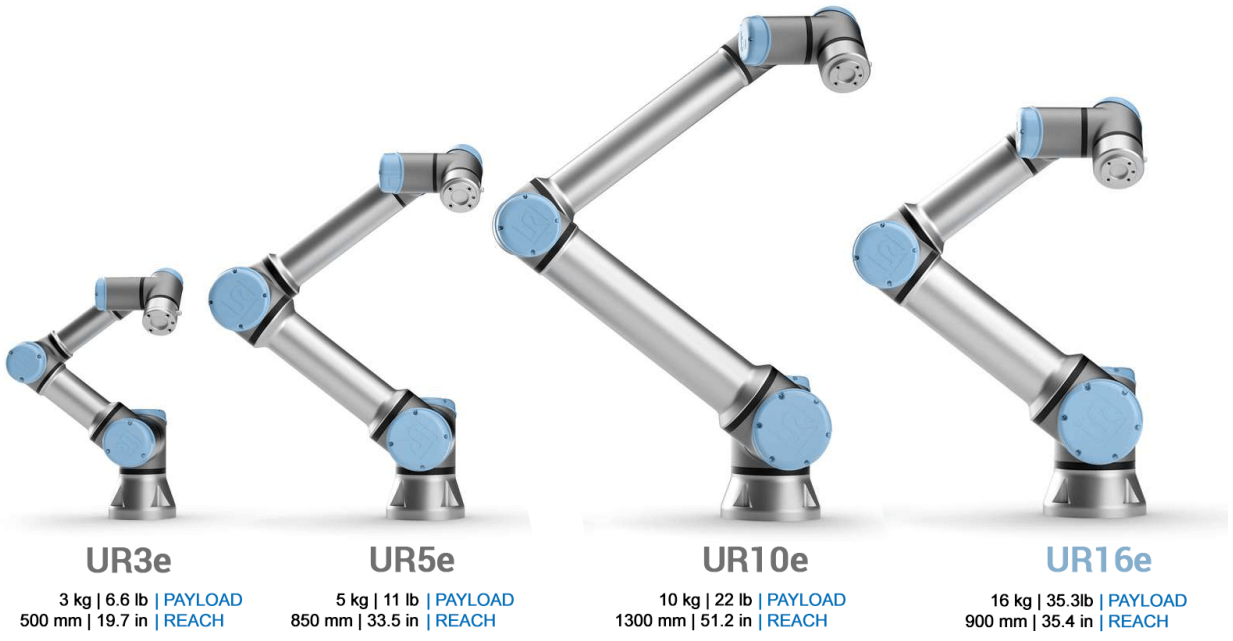
Sinopsis

Para el desarrollo de este proyecto, se ha tomado la decisión de emplear el brazo robótico UR3e, el cual fue adquirido este año por el centro, como una herramienta audiovisual destinada a su uso en entornos que requieren sincronización, como conciertos en vivo. Se han considerado diversos desafíos asociados a lograr dicha sincronización de manera adecuada.

En primer lugar, se ha abordado el problema de programar el robot para que sus movimientos se ajusten correctamente a la cuadrícula temporal de la música. Esto implica ajustar los movimientos del brazo robótico para que estén en consonancia con los tiempos y ritmos de las canciones, lo que garantizará una sincronización precisa.

Además, se han explorado diferentes métodos para activar los movimientos del brazo robótico. Se han utilizado materiales analógicos, como interruptores, que permiten controlar los movimientos de forma manual. Asimismo, se ha implementado una solución basada en microcontroladores, lo que posibilita la sincronización del brazo robótico con una claqueta. Esta opción contribuye a solucionar posibles problemas de latencia o sincronización que podrían surgir al utilizar únicamente métodos manuales.

Por otro lado, se ha trabajado en establecer una comunicación eficiente entre el ordenador encargado de enviar las señales MIDI y el brazo robótico. Esto implica desarrollar un sistema que permita transmitir los comandos y señales necesarios para coordinar los movimientos del brazo robótico con la música en tiempo real.



Familia de brazos robóticos URe

Justificación

El mundo audiovisual siempre ha ido muy de la mano con el mundo de la tecnología, por lo que es solo cuestión de tiempo que el uso de este tipo de herramientas robóticas empiecen a tener un lugar en las distintas producciones que engloba el mundillo audiovisual. Por ejemplo, se podría utilizar un brazo robótico para acoplar una cámara y poder grabar tomas y movimientos de plano que no serían posibles para una persona, como ya hizo en su día el artista Kendrick Lamar, o en una de las últimas modas que se han visto en las alfombras rojas del cine en donde se saca un video a cámara lenta de los actores posando (*glambot*).

También se podrían usar en los estudios de grabaciones, pudiendo acoplar a estos brazos micrófonos que puedan colocarlos de manera precisa en las distintas fuentes de sonido, convirtiendo al brazo en un asistente del ingeniero de sonido.

Asimismo, en el mundo de los espectáculos, en el que se basa este trabajo, es posible ver estos brazos robóticos en los escenarios con los artistas, ya sea realizando una labor funcional, o como una pieza más de atrezo, como es el caso de Merrit Moore, que usa también robots de Universal Robots, o como es el caso de Beyoncé (otro ejemplo), que ha usado brazos robóticos en su última gira.

El hecho de poder obtener ejemplos específicos como los linkeados en este apartado significa que realmente hay un interés en estas tecnologías dentro del sector audiovisual desde hace años, pero todavía necesitan tiempo para crecer, dado que están al alcance de pocas personas dado su elevado coste económico. Además, el hecho de que muchas de estas tecnologías sean adaptaciones de sus respectivas partes industriales indica que existe cierto vacío para un sistema que se adhiera a protocolos más normalizados dentro del mundo audiovisual como puede ser MIDI o DMX.

Merritt Moore con un UR10e

Objetivos

El objetivo de este proyecto va a consistir en la integración de la tecnología del brazo robótico en un ámbito audiovisual, específicamente como pieza de atrezzo para espectáculos.

Para ello necesitaremos por una parte, programar el robot con los movimientos que necesitamos realizar, en el que deberemos tener en cuenta los tiempos de actuación, tanto del propio robot, como en el plano de la obra o espectáculo. Estos tiempos tendrán que ser revisados con especial atención, dado que para la audiencia, es muy fácil percibir cuando el robot se desincroniza.

Por otra parte, debemos encontrar un aforma para realizar un *trigger* para el robot, dado que podemos controlar los tiempos de cada movimiento de forma relativamente precisa en el software de programación del robot, en principio bastaría un solo trigger, (ya sea analógico, como un interruptor, o digital, como un microcontrolador enviando una señal o un transistor) para que el robot inicie la secuencia.

Pero esto puede llevar a problemas de sincronización si este trigger no se realiza en el momento adecuado, por lo que lo ideal sería una comunicación con una claqueta global a todo el espectáculo, como la claqueta de la música que usan los intérpretes, y realizar un trigger por cada movimiento dentro de la secuencia, de esta forma, garantizamos una sincronización real.

Show en Tokio usando brazos robóticos.

Dado que este proyecto se va a presentar en *Malakabot*, podemos realizar un cronograma para organizarnos:

Cronograma proyecto de final de grado						
	Abril		Mayo		Junio	
Objetivo	1-15	16-30	1-15	16-31	1-15	16-22
Programación Brazo						
Implementar interruptor como trigger						
Ensayos Malakabot						
Implementar ESP32 como trigger inicial						
Implementar ESP32 como trigger secuencial						
Implementar ESP32 con ESP now						
Realización de la memoria						
Realización de la presentación en Power Point						

Desarrollo

Para desarrollar este proyecto, se ha realizado una investigación previa en el que hemos buscado proyectos similares, o proyectos con apartados comunes que podemos utilizar en nuestro proyecto ya sea como código “prestado” o como ideas o inspiraciones para solucionar posibles problemas.

Otros proyectos del mismo ámbito

Como se ha comentado antes, existen otros proyectos parecidos al que estamos planteando nosotros, como es el caso de Merritt Moore, que utiliza la misma línea de robots que nosotros utilizamos, la UR de Universal Robots. Sin embargo, no hay demasiada información en el aspecto técnico que utiliza. Podemos ver en algunos de sus videos que el control del robot lo realiza desde un software de ordenador, lo cual se aleja un poco de nuestra idea de integrar un microcontrolador.

Descartando el proyecto de Merritt Moore, se decidió a investigar otros proyectos, en nuestro caso, nuestro proyecto se puede subdividir en tres subproyectos:

- La programación del brazo.
- La comunicación entre el ordenador y el microcontrolador.
- La comunicación entre el microcontrolador y el brazo.

Una vez descompuesto el proyecto de esta forma, nos resulta más fácil buscar información. Hemos podido encontrar algunos proyectos que nos han servido de inspiración y nos han resuelto algunas dudas, aunque ninguno de estos proyectos tenga en mente el mismo objetivo que el nuestro, tienen varios apartados en común, en especial, la comunicación entre el ordenador y el microcontrolador.

Uno de los proyectos que encontramos fue el de *DrasikProductions*, sin embargo, la librería que usa el, promidi, ya no es soportada por los desarrolladores, por lo que además de que los links donde las puedes encontrar no funcionan, dificultando la descarga, nos deja con la duda de la compatibilidad con los nuevos sistemas que vamos a utilizar, dado que esta librería que está diseñada para el microcontrolador arduino, quizás no sea la adecuada para nuestro proyecto, en el que idealmente queremos utilizar un microcontrolador ESP32. Aunque este proyecto no vaya a ayudarnos de forma directa, sí que tiene un aspecto a tener en cuenta, realizar la comunicación midi a microcontrolador por un canal MIDI virtual, y a través del puerto serial.

Una vez encontrado un camino al que seguir, encontramos el proyecto de de *Yearnings*, el cual, de la misma forma que el proyecto anterior, el objetivo del mismo no es el de controlar un brazo robótico, pero hay apartados que son comunes entre los dos proyectos: la comunicación serial.

En cuanto a la programación del robot, los recursos de la propia Universal Robots han sido suficientes para programar el robot sin necesidad de buscar proyectos externos. De la misma forma, la comunicación entre el microcontrolador y el brazo se ha podido llevar a cabo sin problemas, aunque sí es cierto que el representante de Universal Robots que se acercó en el *Malakabot* dio la sugerencia de utilizar las entradas analógicas del brazo en lugar de las digitales, para que de esta forma el conexionado sea más sencillo dado que no tendremos que tener en cuenta los cambios de voltajes necesarios para la comunicación.

Materiales

Para este proyecto necesitamos de disponer de ciertos materiales, tanto hardware como software:

- Hardware:
    - Microcontrolador: en este caso se ha optado por un ESP32 gracias a sus capacidades wifi, el cual se explotará más adelante.
    - Brazo robótico UR3e: el corazón del proyecto, un brazo robótico con distintas articulaciones.
    - Ordenador.

- Software:
    - IDE de Arduino: se utilizará para escribir el código del microcontrolador, en nuestro caso un ESP32.
    - IDE de Processing 4: se utilizará un pequeño programa de Processing 4 (idioma similar a C++, pero más enfocado a la realización de programas rápidos y sketches), que se utilizará para realizar la comunicación serial entre el ordenador y el microcontrolador.
    - LoopMIDI: software que crea un canal virtual de MIDI.
    - Reaper: el DAW (Digital Audio Workstation) que utilizaremos para la claqueta y los mensajes MIDI.

Malakabot y Metrópolis

Antes de empezar a explicar el groso del trabajo, me gustaría resaltar que este proyecto ha formado parte de la feria de electrónica *Malakabot* que se realiza en el instituto I.E.S. Politécnico Jesús Marín, así como en la obra *Metrópolis* que llevaron a cabo los alumnos de Bachillerato de Artes. Desde el inicio este proyecto ha tenido fechas límite un poco complicadas, por lo que siempre se ha trabajado algo a contrarreloj, razón por la cual en estas primeras versiones del proyecto, se optó por una de las soluciones más sencillas. Aun con todo, el proyecto funcionó y dió un buen resultado.

Procedimiento

En primer lugar, se ha debido de realizar una programación previa del robot, para ello, dado que este proyecto se realizó en colaboración con el curso de Bachillerato de artes, dos alumnas realizaron una coreografía que formaría parte de una obra de teatro llamada Metrópolis. Esta escena se dividía en tres partes, una primera parte donde el robot “despierta”, una segunda parte donde el robot y una bailarina realizaban una coreografía, en donde los movimientos del robot mimetizan a los de la bailarina, y otra parte donde el robot “muere”. Teniendo una idea clara del papel del robot, y teniendo una referencia en video de los movimientos de la bailarina, se pudo crear un programa inicial donde se secuenciaron todos los movimientos.

Una vez teniendo el programa hecho, pasamos a la parte de sincronización. En esta escena se utilizará una canción original compuesta por Enrique Norro, por lo que el primer paso fue la grabación con claqueta de esta canción, la cual fue realizada en el aula de sonido del politécnico.

A la hora de programar el robot nos encontramos con el problema del tiempo de los movimientos, dado que aunque el robot puede realizar movimientos bastante rápidos, a la hora de especificar el tiempo, el software tiende a redondear los datos introducidos, por lo que para esta primera parte del proyecto, que se presentaba en *Malakabot*, se decidió grabar a un tempo en el que las subdivisiones fueran los

números más redondos posibles. Este tempo se situó en 100 BPM, o lo que es lo mismo, por cada negra pasarán 300 ms.

Una vez habiendo calculado los milisegundos por cada golpe de la claqueta, se procedió a hacer una subdivisión de cada redonda en la canción, dado que cada redonda equivale a cuatro negras. De esta manera, obtenemos una "jerarquización" de los movimientos del robot y una forma más sencilla de navegar entre todos los pasos de la pieza musical.

Una vez tenemos el programa y la sincronía resueltos, podemos dar paso a pensar cómo podemos accionar este programa sin necesidad de utilizar la tablet con la que viene el robot, y poder hacerlo de una forma más sencilla, para esto, se pensaron en varias soluciones:

- En primer lugar, la utilización de un interruptor analógico, que sirva como señal para que el robot empiece la secuencia.
- La segunda solución, algo más funcional, es la utilización de un microcontrolador que mande el trigger a través del ordenador que se encargue de reproducir la claqueta.

En los siguientes apartados abordaremos las distintas soluciones en detalle.

Posibles soluciones

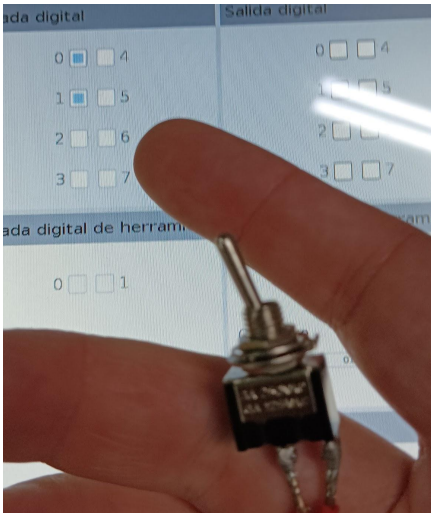
Ahora se procederá a explicar las posibles soluciones a la hora de accionar el brazo.

Solución 1: Trigger

Razonamiento

Esta primera solución es la más sencilla, utilizaremos un interruptor analógico y una función de espera en la programación del robot. Esta función se colocará al inicio del programa, y esperará hasta que reciba el voltaje establecido.

Como primer punto de contacto, se utilizó un interruptor de palanca conectado a la entrada digital del UR3e, de esta forma, al accionarlo, el robot recibiría 24V, también suministrados por el brazo, y accionaría la secuencia.

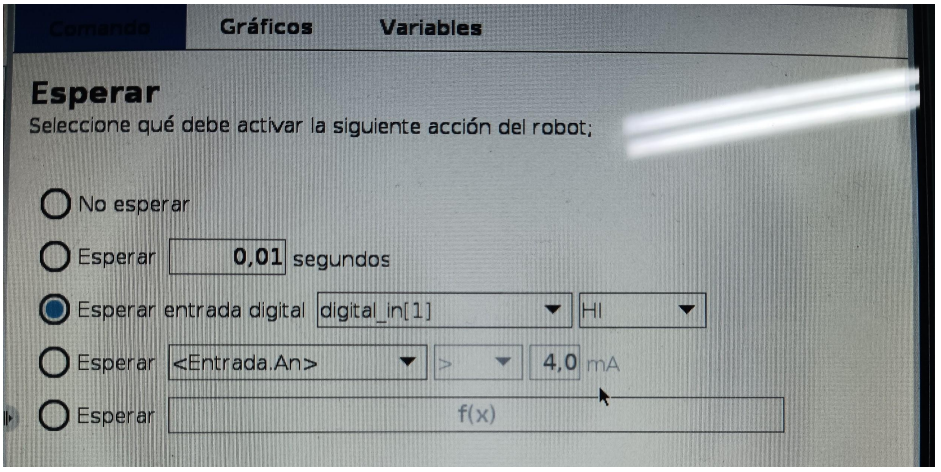


Interruptor de palanca utilizado con la pantalla de monitorización E/S detrás

A la hora de realizar la actuación de *Malakabot* y *Metrópolis* se decidió utilizar un interruptor de pie, en este caso se trata de los mismos interruptores que se utilizan en los amplificadores de guitarra o bajo, por lo que se cortó el jack de conexión que normalmente iría conectado al amplificador y se introdujo en el brazo, también en una entrada digital.

Pulsador de pie utilizado

Una vez tenemos estos métodos conectados al robot, debemos de monitorizar de alguna forma que está funcionando correctamente. Esto lo podemos comprobar fácilmente en la pestaña de entradas y salidas (E/S) en el software del robot, allí encontraremos toda la información necesaria para saber si en las distintas entradas llega un HIGH o un LOW. Esta comprobación resulta importante, dado que cuando utilizamos el interruptor de palanca, como cabe de esperar, cuando está accionado y deja pasar la corriente de 24V, el robot recibe un HIGH. Sin embargo, con el pedal de pie, es cuando este no está accionado que el robot recibe el HIGH en la entrada, estando este invertido.



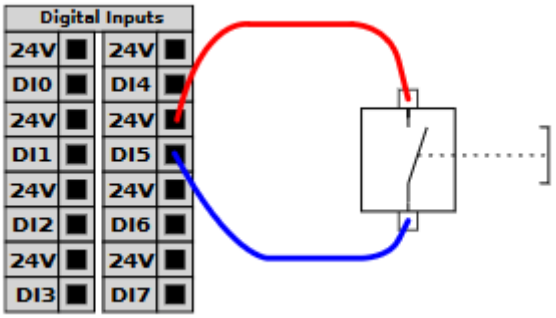
Pantalla de función de espera en la programación del brazo

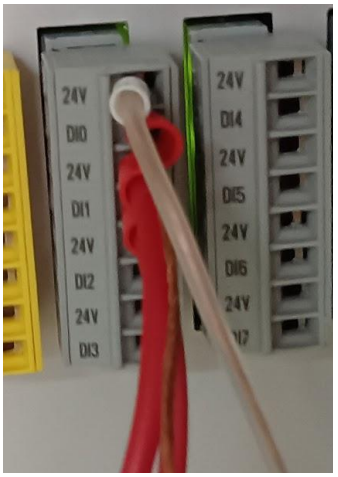
Una vez sabemos cómo actúan nuestros interruptores, podemos agregar una función de espera en el programa del robot, donde podremos especificar qué tipo de entrada queremos poner como condicionante

(en nuestro caso, la entrada digital), y en qué estado queremos que se acciona (en el caso del interruptor de palanca en HIGH, y en el caso del pedal de pie en LOW).

Conexionado

Como ya hemos comentado, estos interruptores están conectados a las entradas digitales del UR3e, un cable del interruptor va conectado a la salida de 24V a la que trabaja el robot, y el otro a la propia entrada (en nuestro caso DI0 y DI1).





Conclusiones

Como hemos podido ver, esta solución es la más simple para el proyecto, aporta una sincronización aceptable y una configuración de entradas y salidas muy sencilla, comparemos los pros y los contras:

Pros:

- Configuración sencilla.
- Al trabajar con HIGHs y LOWs que te proporciona el brazo no tienes que preocuparte de lecturas de voltajes externos.
- Al estar programado el brazo al tiempo de la canción, este sincroniza bien.

Contras:

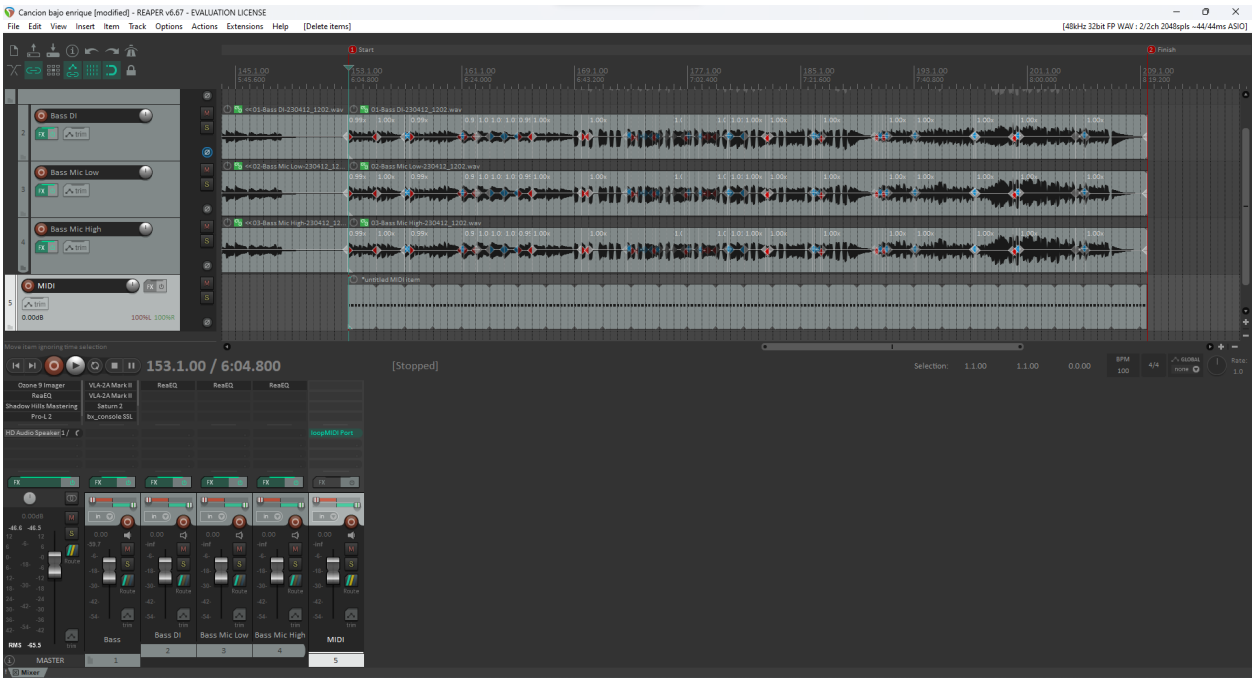
- No es del todo fiable, dado que deja mucho lugar al error humano. Si no se activa a tiempo, el robot irá desincronizado durante todo el tiempo.
- Al utilizar un cable físico conectado del interruptor al robot, nos veremos limitados a la hora de la distancia que podemos utilizar entre nosotros y la centralita para activarlo.

Solución 2: Comunicación entre *daw* y microcontrolador

En la siguiente solución, hemos decidido utilizar un microcontrolador en lugar de un interruptor analógico. Para utilizar este microcontrolador, hemos decidido comunicarle una señal MIDI que dé el aviso de *start* a la secuencia del robot. En el siguiente esquema podemos ver el camino que nos proponemos a recorrer.

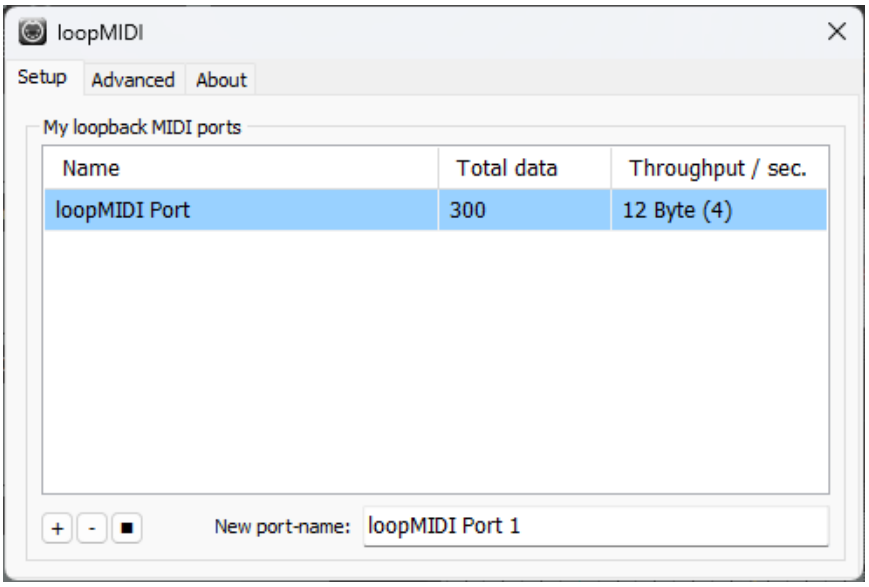
Como se puede observar, el primer paso será la comunicación entre DAW y ESP32. Esta primera parte será común para las dos soluciones aportadas con el uso de microcontroladores, por lo que se explicará en primer lugar su funcionamiento y más adelante, en detalle, cada una de las partes que hace uso de los microprocesadores.

Un DAW (*Digital Audio Workstation*) es una herramienta software de audio que nos servirá como centralita tanto como para la reproducción de pistas de audio, como para la reproducción de pistas MIDI, también podremos establecer distintos tempos a necesidad de la canción. En este proyecto se ha decidido trabajar con Reaper, aunque cualquier DAW puede realizar las funciones necesarias.



Captura de la sesión de Reaper utilizada para el proyecto, donde se grabó la canción y se programó el MIDI.

La idea principal al usar un DAW es la posibilidad de reproducir mensajes MIDI que podremos rutear de distintas maneras. En nuestro caso, este mensaje MIDI va a coincidir con la claqueta de la canción que vayamos a usar, en nuestro caso, 100 BPM, o una nota cada 300 ms.



Ventana de monitorización de LoopMIDI, donde podemos monitorizar los bytes/s que estamos transmitiendo.

Una vez hemos programado nuestro mensaje MIDI, tenemos que enviarlo de alguna manera al serial del ESP32. Para ello hemos decidido utilizar un canal virtual de MIDI dentro del propio ordenador, que hemos creado utilizando el software LoopMIDI. Este software permite crear canales virtuales para poder conectar varios programas que necesiten de la señal MIDI para trabajar, dado que a través de este software, no es necesario utilizar ninguna conexión externa, pues normalmente Windows no tiene ninguna herramienta incluida para esta función (al contrario iOs).

Una vez estamos mandando nuestra señal MIDI por un canal virtual de MIDI, utilizaremos un pequeño script realizado en Processing 4 para recoger esos datos y poder enviarlos al serial del microcontrolador. Processing 4 es un programa para realizar pequeños programas, similar a Python, pero, aunque Processing 4 tiene su propio lenguaje de programación, está muy influenciado por C++, motivo por el que se ha utilizado.

Código *Processing 4*

El siguiente código es el utilizado en el sketch de Processing 4 para recoger los datos MIDI.

```
//Importamos las librerias que usaremos
import themidibus.*;
import javax.sound.midi.MidiMessage;
import processing.serial.*;

//Configuramos las distintas variables
MidiBus midi;
IntList activeNotes = new IntList();
Serial arduino;
String valToSend;

//Configuramos los parametros de la librería midibus y processing.serial,
indicando el puerto de nuestro microcontrolador y el baud rate
void setup(){
```

```
  midi = new MidiBus(this, "loopMIDI Port", -1);
  String port = Serial.list()[0];
  arduino = new Serial(this, port, 9600);
}


void draw(){
//Cuando una nota MIDI sea accionada se envia al serial de arduino
  if(activeNotes.size() > 0){
      for(int n=0; n < activeNotes.size(); n++){
      arduino.write(activeNotes.get(n));
      }
  }
//si ninguna nota se acciona se envia un 0 al serial de arduino
  else{
      arduino.write(0);
  }
}

//Identificamos cuando una nota se acciona
void midiMessage(MidiMessage m){
  int status = m.getStatus();
  int msgData = m.getMessage()[1];
  if(status == 144) noteOn(msgData);
  if(status == 128) noteOff(msgData);
}

//recogemos que nota en especifico se ha accionado
void noteOn(int noteNumber){
  if(!activeNotes.hasValue(noteNumber)){
  activeNotes.push(noteNumber);
  }
}

//si se deja de accionar una nota, cambiamos eliminamos la ultima nota
accionada
void noteOff(int noteNumber){
  if(activeNotes.hasValue(noteNumber)){
      int noteIndex = indexOf(noteNumber, activeNotes);
      activeNotes.remove(noteIndex);
  }
      else{
      println("Turning off note that isnt on :" + noteNumber);
      }
}

int indexOf(int target, IntList source){
  for(int i = 0; i < source.size(); i++){
```

```
        if(source.get(i) == target) return i;
    }
    return -1;
}
```

Una vez tenemos estos datos enviados al serial del microcontrolador (se puede comprobar fácilmente que la comunicación es exitosa con un pequeño programa que encienda el LED interno cada vez que recibe un mensaje), podemos proseguir con la segunda solución.

Esta segunda solución se ha subdividido en dos partes, una de ellas con un solo controlador, y otra de ellas utilizando dos controladores.

Solución 2.1: Microcontrolador sincronizado con comunicación directa

Razonamiento

Nuestra primera solución que hace uso de un microcontrolador se basa en la idea de una comunicación directa entre nuestro PC al ESP32 a través del serial que podemos utilizar con el cable USB que utilizamos para programarlo, y otra comunicación directa entre este mismo controlador y el brazo, a través de dos cables, uno que actuará como el INPUT al robot, y otro como referencia a tierra.

Se ha de mencionar que se han realizado dos aproximamientos a esta solución, la primera, donde el microcontrolador hace la misma función que el interruptor de la solución inicial, donde actuará de trigger una sola vez para dar inicio a la secuencia, y otra donde actuará de trigger en cada movimiento que realice el brazo, realizando una sincronización *real*.

El problema de la latencia

Antes de comenzar, debemos de saber que los distintos equipos de procesado que utilizamos en nuestro proyecto producen pequeños tiempos de latencia a la hora de enviar los mensajes al robot. Por ello, deberemos desfasar la señal MIDI que sacamos de nuestro PC para compensar esta latencia.

Debemos de tener en cuenta la latencia que pueden producir los distintos equipos, en nuestro caso, tenemos la latencia de la tarjeta de sonido que utilizaremos, la latencia entre el PC y el microcontrolador, y la latencia entre el microcontrolador y el brazo. Al existir tantas fuentes de latencia, es complicado calcular de forma exacta la cantidad de ms que tenemos que desfasar la señal MIDI.

Sin embargo, dada la naturaleza del proyecto, y teniendo una claqueta sonora que marca los pasos de forma periódica, podemos ajustar esta latencia de manera manual, dado que podemos ver y escuchar cuando los movimientos del brazo robótico van adelantados o atrasados. Por ello, hemos contado con un desfase de aprox. 100 ms

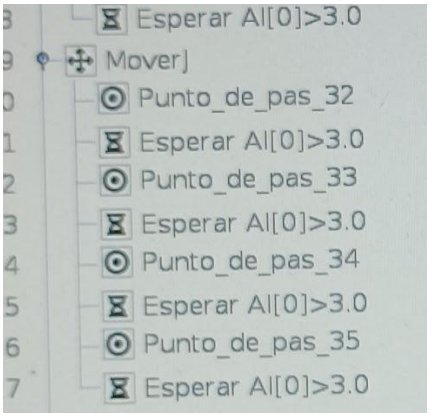
Microcontrolador como trigger inicial

Punto inicial de espera a trigger

Esta solución consiste en usar la señal enviada por el microcontrolador una sola vez, al inicio del programa del robot. Al estar el robot programado al tempo de la canción, bastaría con que el robot solo tuviera una función de espera al inicio de la secuencia, en nuestro caso, una entrada analógica de nuestro microcontrolador. Al ser un solo evento el que inicia, realmente no tenemos fiabilidad a la hora de que nuestro robot vaya al tempo adecuado, dado que uno de los problemas de nuestro brazo, es que en alguno de los movimientos donde el recorrido de las articulaciones es muy grande, es posible que tarde más tiempo del que se le especifica al robot, creando cierto retardo, lo que puede verse mal de cara a la audiencia.

Sin embargo, al poder utilizar solamente el trigger una vez, podremos encadenar los movimientos del robot de mejor forma, por lo que quedará un movimiento más suave, que dependiendo del tipo de acto que se requiera llevar a cabo, puede ser lo deseado.

Microcontrolador como trigger de sincronía



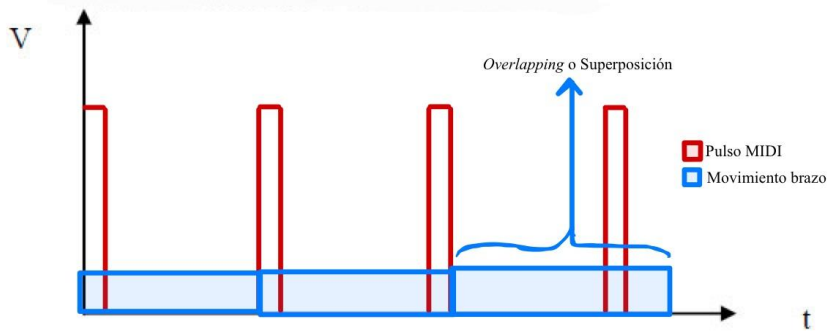
Sincronia de trigger en cada paso

Esta otra solución consiste en poner una función de espera en cada uno de los movimientos que realiza el robot, de esta manera, el movimiento que obtendremos podrá ser más preciso, dado que le indicamos exactamente donde iniciar y terminar los movimientos a través de nuestra señal MIDI.

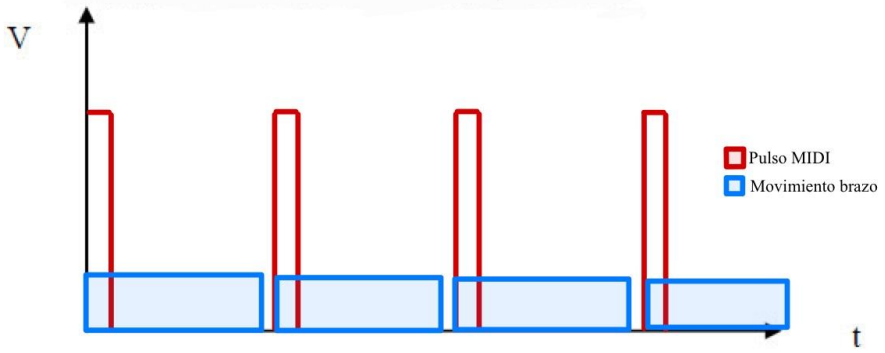
Aunque los movimientos sean más precisos, un contratiempo que podemos encontrar es que los movimientos de nuestro brazo sean más bruscos y “robóticos”. Esto se debe a la necesidad de terminar el movimiento antes de que llegue la siguiente señal MIDI.

El problema del overlapping o superposición

Este problema se, que se le ha decidido llamar *overlapping*, o superposición, ocurre cuando la señal MIDI que recibe el brazo robótico se pisa con el movimiento que está realizando, lo que puede provocar que, como se ha dicho antes, a la posibilidad de retrasar alguno de los movimientos, el brazo pueda saltarse algunas de las señales de claqueta que se le envían a través de MIDI, en el gráfico podemos observar en una línea de tiempo.



Una forma sencilla de solucionar este problema es simplemente acortando el tiempo de los movimientos, para que de esta forma, en vez de durar justamente lo que dura la barra, duren un poco menos (en este caso, se acortaron 10 ms cada movimiento).



Como se puede observar en el gráfico de arriba, acortando los movimientos para que terminen antes del trigger MIDI soluciona este problema, sin embargo, da un efecto más brusco a los movimientos del brazo, dado que este debe detenerse antes de empezar el próximo movimiento.

Conclusiones de los distintos métodos de trigger

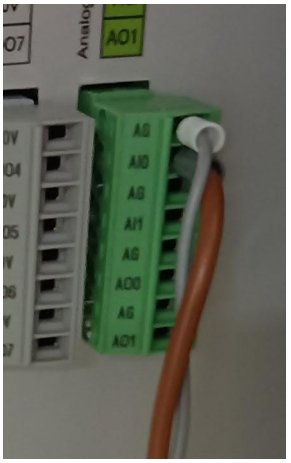
Como ya hemos mencionado, podemos elaborar una lista de las características de los dos tipos de soluciones:

- Trigger inicial:
  - Movimiento más suave.
  - Más simple a la hora de programar el robot (menos pasos).
  - Más vulnerable a errores de sincronía a largo plazo.
- Trigger de sincronía:
  - Movimiento más brusco o robótico.

- Más tedioso de programar (más pasos).
- Más preciso a la hora de controlar los movimientos dentro del tempo de la música.

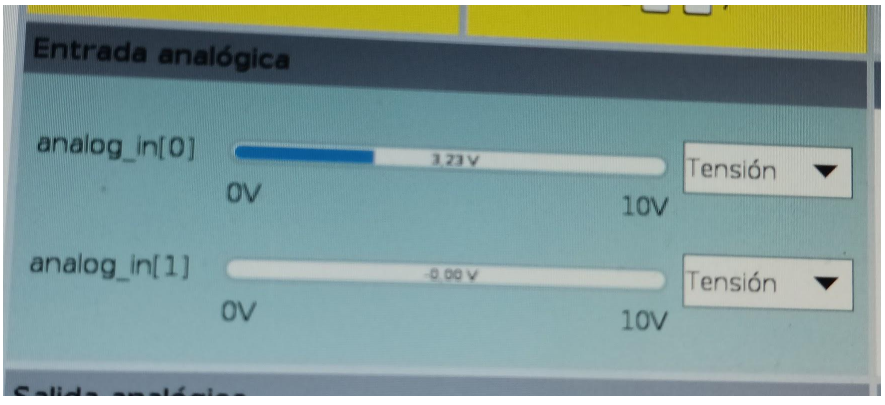
Conexionado

En este caso, la salida del microcontrolador irá a una de las entradas analógicas del brazo. El motivo de que se use una de las entradas analógicas es la facilidad a la hora de recibir voltajes menores de 24V por parte del brazo.



Conexión analógica al brazo

Estas entradas analógicas, permiten decidir sobre qué voltaje o corriente actuar, por lo que es perfecta para nuestro controlador. Para saber el voltaje que obtenemos de nuestro controlador al poner la salida en HIGH, podemos ir a las especificaciones del fabricante, pero si por cualquier motivo no disponemos de ellas, el brazo robótico monitoriza el voltaje que llega a la entrada, por lo que podemos usar de guía para configurar nuestra función de espera a la entrada digital, en nuestro caso, el ESP32 que utilizamos nos da 3,3V, por lo que podemos configurar la función de espera para que actué cuando recibe más de 3V.



Monitor de entradas y salidas donde se nos muestra el voltaje de nuestra entrada analógica

Código

El siguiente código fue utilizado para lanzar el trigger desde el microcontrolador. Como se puede observar, es un código sencillo que funciona en conjunción con el programa de Processing 4.

```
//definimos la salida que comunicará al robot
#define OUT 23

//inicializamos la variable que recibira los detos del serial
char serialData;

void setup() {
//especificamos las funciones de cada pin
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(OUT, OUTPUT);
//inicializamos el serial
  Serial.begin(9600);
}

void loop() {
//Leemos el serial
  if(Serial.available()){
      serialData = Serial.read();
      }

//Si nos envian 0 por serial, las salidas serán LOW
  if(serialData == 0){
      digitalWrite(LED_BUILTIN, LOW);
      digitalWrite(OUT, LOW);
      }
//si nos envian cualquier otra cosa, las salidas serán HIGH
  else{
      digitalWrite(LED_BUILTIN, HIGH);
      digitalWrite(OUT, HIGH);
      }
}
```

Conclusiones

Esta solución ha demostrado ser la más sencilla y efectiva por ahora, además de darnos dos soluciones que pueden dar varios “modos” de movimiento al robot (más suave o más brusco), nos asegura que nuestro programa quedará dentro de los parámetros de tiempo que hemos establecido, dando lugar a un *performance* decente.

Solución 2.2: Microcontrolador sincronizado con comunicación inalámbrica

Razonamiento

Aunque la solución anterior ha sido exitosa, dentro de un ambiente profesional podemos encontrarnos con varios contratiempos para nuestro proyecto. Por ejemplo, muchas veces, en los espectáculos, el control donde se colocan todos los técnicos e ingenieros se encuentra bastante alejado del escenario. Esto significa que muchas veces tendremos que utilizar cables de bastante longitud, lo que puede afectar al voltaje recibido en el robot, pudiendo afectar a la respuesta a estos pulsos MIDI. Por ello, se ha decidido utilizar dos microcontroladores, que conectados por WiFi, transmiten entre ellos datos de manera digital. De esta forma, un microcontrolador actuará como transmisor, y el otro como recibidor (que estará a su vez conectado al brazo).

Procedimiento

Para esta solución, se ha decido utilizar el protocolo ESP now, que permite conectar dos o más dispositivos de la línea ESP de Espressif a través de WiFi de baja potencia, similar a la tecnología que utilizan los ratones o teclados inalámbricos. Aunque esta solución a día de hoy aún está en desarrollo, dado que se decidió realizar como un extra añadido al proyecto, se ha conseguido escribir un código base, que se explicará más adelante.

Para ello, nos hemos ayudado de la explicación tanto de Espressif (desarrolladores de este protocolo), como de *Programing Electronics Academy*, que hace una explicación a fondo de todos los parámetros necesarios para poner a funcionar este protocolo.

Código del transmisor

El código del transmisor lee los datos enviados al serial a través del sketch de Processing 4, y los envía a través del protocolo ESP now.

```
//incluimos las librerías que utilizaremos
#include <esp_now.h>
#include <WiFi.h>

//definimos el canal que utilizaremos dentro del protocolo esp now
#define CHANNEL 1

//definimos la variable de esclavo dentro de esp now
esp_now_peer_info_t slave;

//declaramos la variable donde guardaremos nuestros datos
uint8_t data = 0;

void setup(){
  //iniciamos el serial
  Serial.begin(115200);
  //configuramos la libreria wifi e iniciamos el espnow
  WiFi.mode(WIFI_STA);
```

```
  esp_now_init();
  //escaneamos la red por un esclavo
  esp_now_register_send_cb(OnDataSent);
  ScanForSlave();
  esp_now_add_peer(&slave);

  //configuramos salidas
  pinMode(LED_BUILTIN, OUTPUT);
  }

void loop(){
  //enviamos los datos a traves de esp now
  esp_now_send(slave.peer_addr, &data, sizeof(data));
  //si tenemos datos en el serial, los leemos
  if(Serial.available()){
      data = Serial.read();
      }

  //si lo que leemos en el serial es 0, salida es LOW
  if(data == 0){
      digitalWrite(LED_BUILTIN, LOW);
      }
  //si la salida es cualquier otra cosa, salida es HIGH
  else{
      digitalWrite(LED_BUILTIN, HIGH);
      }
  }

//en esta funcion escaneamos la red para buscar el mac de nuestra otra
esp32, especificando el nombre RX
void ScanForSlave(){
  int8_t scanResults = WiFi.scanNetworks();

  for(int i = 0; i < scanResults; ++i){
  String SSID = WiFi.SSID(i);
  String BSSIDstr = WiFi.BSSIDstr(i);

  if(SSID.indexOf("RX") == 0){
      int mac[6];
      if(6 == sscanf(BSSIDstr.c_str(), "%x:%x:%x:%x:%x:%x", &mac[0],
&mac[1], &mac[2], &mac[3], &mac[4], &mac[5])){
      for (int ii = 0; ii < 6; ++ii){
      slave.peer_addr[ii] = (uint8_t) mac[ii];
      }
      }
      slave.channel = CHANNEL;
      slave.encrypt = 0;
      break;
```

```
      }
  }
}

//monitorizamos los datos que enviamos
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status){
  Serial.print("I sent my data -> ");
  Serial.println(data);
  }
```

Código del recibidor

En el código del transmisor, el microcontrolador recibe los datos de nuestro transmisor y actúa en base a estos.

```
//incluimos las librerias
#include <esp_now.h>
#include <WiFi.h>

//indicamos el canal y definimos el pin de salido
#define CHANNEL 1
#define OUT 23

void setup() {
  //iniciamos serial y esp now, y especificamos los paramitros de WiFi.h
  Serial.begin(115200);
  WiFi.mode(WIFI_AP);
  WiFi.softAP("RX_1", "RX_1_Password", CHANNEL, 0);
  esp_now_init();
  esp_now_register_recv_cb(OnDataRecv);

  //configuramos salidas
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(OUT, OUTPUT);
}

//si los datos que recibimos es 0, salida LOW, si los datos es !=0,
entonces la salida HIGH
void loop() {
  if(data == 0){
      digitalWrite(LED_BUILTIN, LOW);
      digitalWrite(OUT, LOW);
      }
  else{
      digitalWrite(LED_BUILTIN, HIGH);
      digitalWrite(OUT, HIGH);
      }
```

```
}

//monitorizamos los datos recibidos
void OnDataRecv(const uint8_t *mac_addr, const uint8_t *data, int
data_len){
  Serial.print("I just received -> ");
  Serial.println(*data);
  }
```

Conexionado

El conexionado en el brazo seguirá siendo a través de la entrada analógica, pero la diferencia será que un microcontrolador se situará en la zona de control conectado al ordenador responsable de las claquetas y backing tracks, y el otro se situará en el lugar donde se encuentre el brazo, que, mediante una batería externa o una toma de corriente, llevará los pulsos MIDI al brazo.

Conclusiones

Dado que este código se añadió al final del proyecto, no ha dado tiempo a tenerlo funcionando antes de la fecha límite para la presentación. El problema reside en el código y el bus serial, dado que tanto la comunicación de ESP now como la de la señal MIDI utilizan el bus serial, uno de ellos siempre dará error, dado que estará ocupado por el otro programa, más adelante se hablará de posibles soluciones.

El código del transmisor y el recibidor ha funcionado correctamente cuando se trabaja con datos que no se obtengan a través del puerto serial.

Resultados

Como hemos podido ver, hemos propuesto dos posibles soluciones, una de ellas, la más sencilla, que utiliza un interruptor analógico, pero puede llegar a dar lugar al error humano, dado que es una persona la que tiene que accionar el interruptor a tiempo.

Como segunda opción, se ha utilizado un microcontrolador de varias formas:

- En primer lugar, un microcontrolador utilizado para una señal inicial de inicio de secuencia, al accionar de esta forma el programa, eliminamos el posible error humano que podíamos sufrir con el interruptor analógico, dado que este trigger irá sincronizado a la música a través de MIDI.
- Como segunda opción, hemos insertado una función de espera hasta que se reciba un pulso de MIDI para que el brazo continúe con cada uno de los pasos dentro de la secuencia, de esta forma, podemos garantizar una sincronía del brazo bastante más exacta que la que obtendremos con un único trigger inicial, aunque encontramos el problema de la superposición de las señales enviadas con el movimiento del robot. Para solucionarlo podemos acortar un poco los movimientos para garantizar que la señal llega cuando el robot está esperando, lo que puede suponer que los movimientos sean más bruscos o robóticos.
- Como tercera opción, para solucionar posibles errores en las distancias entre el técnico encargado del movimiento del robot y el escenario, se ha decidido implementar el protocolo ESP now que permite comunicar dos microcontroladores de la familia ESP mediante WiFi de poca potencia. De esta forma, obtenemos un transmisor, que se encargará de enviar la información de manera digital, y un receptor, que recibirá esta información y la convertirá en el voltaje que actuará como trigger.

También hemos tenido que tener en cuenta el posible retardo que puede sufrir la señal al pasar por los distintos equipos necesarios, que se ha reducido a un adelantamiento de esta señal MIDI de aprox. 30ms.

Como hemos podido ver, a excepción de la solución que implica el uso del protocolo ESP now, tocas las soluciones han resultado exitosas, poniéndose incluso a prueba con un público real en la feria *Malakabot* y en la obra de teatro *Metrópolis.*

Conclusiones

Este proyecto ha supuesto un reto a la hora de su ejecución, dado que ha sido un desafío de investigación y desarrollo de las habilidades obtenidas durante el grado superior que se ha cursado durante estos dos años. Ha sido una oportunidad para recapitular sobre los conocimientos obtenidos y expandir sobre ellos.

Dado que este proyecto abarcaba muchos procesos distintos, podemos dividir el proyecto en 3 partes:

- En primer lugar, la programación del robot, donde realmente empezó el proyecto, dado que por la naturaleza del proyecto, el programa que se ejecuta ha resultado bastante extenso y con muchos pequeños detalles. Para programar el robot hemos trabajado tanto por libre como con referencias aportadas por el bachillerato de artes escénicas para lograr que el robot se moviese. Seguido de la importancia de la sincronización con la música incluso antes de utilizar un microcontrolador. En esta parte también se ha de tener en cuenta la fuente musical, dado que es importante que el ritmo se mantenga y que la calidad del audio sea la mejor posible para evitar posibles errores.
- En segundo lugar, tenemos la parte digital de los microcontroladores. Aquí es donde más tiempo del proyecto se ha puesto, dado que cada una de las tres soluciones precisaban de un programa distinto, además del programa adicional para intercambiar la señal MIDI. A parte de lo que ya hemos estudiado en el curso, se ha dado uso a los pointers, se ha aprendido a usar el protocolo ESP now, y más importante, a cómo utilizar el intercambio de información entre un PC y un microcontrolador a través del bus serial. Estos tres procesos han sido imprescindibles para el desarrollo del proyecto, y como han sido llevado a cabo desde el desconocimiento, ha resultado quizás la parte más complicada de este proyecto, aunque han resultado ser una muy buena herramienta.
- En tercer lugar, tenemos toda la parte física del proyecto, es decir, todas las conexiones y las formas de interacción de los distintos dispositivos. Para nuestro proyecto, hemos aprendido una forma sencilla de conectar dispositivos de bajo voltaje a nuestro brazo robótico UR3e, así como la conexión que podemos realizar entre dos tipos de microcontroladores distintos. Aunque en este proyecto se haya optado por el uso de ESP now, también se barajó momentáneamente trabajar con uno de los microcontroladores actuando como un servidor TCP, pero finalmente se descartó dado que suponía un programa más compleja y difícil de cuadrar en el tiempo limitado en el que se llevó a cabo el proyecto.

Como ya se ha dicho antes, la opción de utilizar dos microcontroladores de forma inalámbrica no se ha podido implementar correctamente, sin embargo, los siguientes pasos adelante están bastante claros, por lo que sólo es cuestión de dedicarle algo más de tiempo para poder ponerlo en funcionamiento.

También cabe resaltar que al presentarse este proyecto en la feria *Malakabot*, se ha trabajado desde el principio con una fecha límite para la primera parte del proyecto, lo que ha requerido un aumento de las horas dedicadas al trabajo durante este periodo. Dicho esto, esta fecha límite también ha ayudado a una mejor organización durante las diferentes fases del proyecto.

Futuro del proyecto

El siguiente paso adelante de este proyecto debería ser la correcta implementación del protocolo ESP now, si recordamos, esta función no pudo funcionar correctamente dado que tanto el intercambio de información entre los dos microcontroladores como el recibimiento de los datos de la señal MIDI se realizan a través del puerto serial, por lo que recibiremos un error de que el serial esta ocupado en alguno de los dos programas, limitando así nuestro envío de datos.

Sin embargo, se deben estudiar las soluciones para poder rectificar este error, estas soluciones pueden incluir:

- El uso de un tercer microcontrolador, que se encargue de recibir el MIDI y enviarlo a través de una conexión física al transmisor del ESP now y así liberar el uso del puerto serial.
- En lugar de transmitir los datos MIDI por el puerto serial, utilizar un cable MIDI que salga del ordenador y aplicar un conector MIDI (Din) a nuestro microcontrolador, de esta forma mantenemos dos placas y liberamos el puerto serial para la utilización del protocolo ESP now.

Otra opción mirando al futuro es la simplificación del software, podemos intentar implementar el programa de Processing 4 en la propia placa del microcontrolador, para ello, podemos hacer uso de la librería USBMIDI de Arduino, o la equivalente en los microcontroladores ESP32 en caso de que esta no sea compatible. El motivo por el que no se uso esta librería desde el principio es la poca información que hay para la utilización de la misma, dado que como se ha comentado anteriormente, trabajar con el puerto serial ha sido una de las cosas que se han tenido que aprender desde cero para este proyecto, por lo que nuestra pericia a la hora de usar este tipo de funciones ha resultado limitada, y ha sido más sencillo realizar esta tarea a través de un programa externo.

Otro de los aspectos que se podrían mejorar es la transmisión inalámbrica, el protocolo ESP now funciona muy bien en bajas distancias, pero si este proyecto llega a utilizarse en actuaciones grandes, lo cierto es que al estar utilizando las frecuencias WiFi (2.4GHz), es muy probable que en una actuación que esté llena de gente con teléfonos móviles utilizando la misma frecuencia que nuestro WiFi de lugar a un problema muy grande de interferencias. Para solucionar esto, se han pensado en dos posibles soluciones:

- Mandar los mensajes MIDI a través de una red local cableada, utilizando un Switch con cables de ethernet, de esta forma, uno de los microcontroladores podrá realizar el papel de servidor, y el otro podrá recibir los datos, eliminando la necesidad de utilizar el WiFi.
- Utilizar un modulador FM para el envío de nuestra señal. De esta forma, podemos utilizar antenas amplificadoras que garanticen que nuestra señal llegará de manera fiable.

Estos dos métodos ya se usan mucho en el ámbito profesional del mundo audiovisual, dado que su implementación sería sencilla para el resto del equipo y no supondría un coste extra de equipamiento.

Por último , en nuestro proyecto hemos decidido trabajar con MIDI, pero también sería interesante realizar una versión del proyecto que trabajase con DMX, que es un protocolo para el control de luces, de esta forma, podremos utilizar un programa externo (de Python, por ej), para controlar nuestro robot sin necesidad de la centralita de Univrsal Robots, por lo que podemos tener una integración completa con el equipo y el protocolo de luces, donde cada uno de los canales asignados a este DMX podrían mover cada una de las diferentes articulaciones del robot.

Bibliografía

- Protocolo musical MIDI (Musical Instrument Digital Interface) - Wikipedia
- Protocolo DMX (Digital Multiplex) - Wikipedia
- Robot UR3e - Universal Robots
- Manual del robot UR3e - Universal Robots
- Proyecto de *DrasikProduction* - Youtube
- Proyecto de *Yearnings* - Youtube
- Librería *themidibus* - Github
- Librería *Ethernet* - Arduino
- *Programming Electronics Academy* - Youtube
- ESPnow - Espressif
- Control de UR3e a través de Python por *Malau Robotics* - Youtube