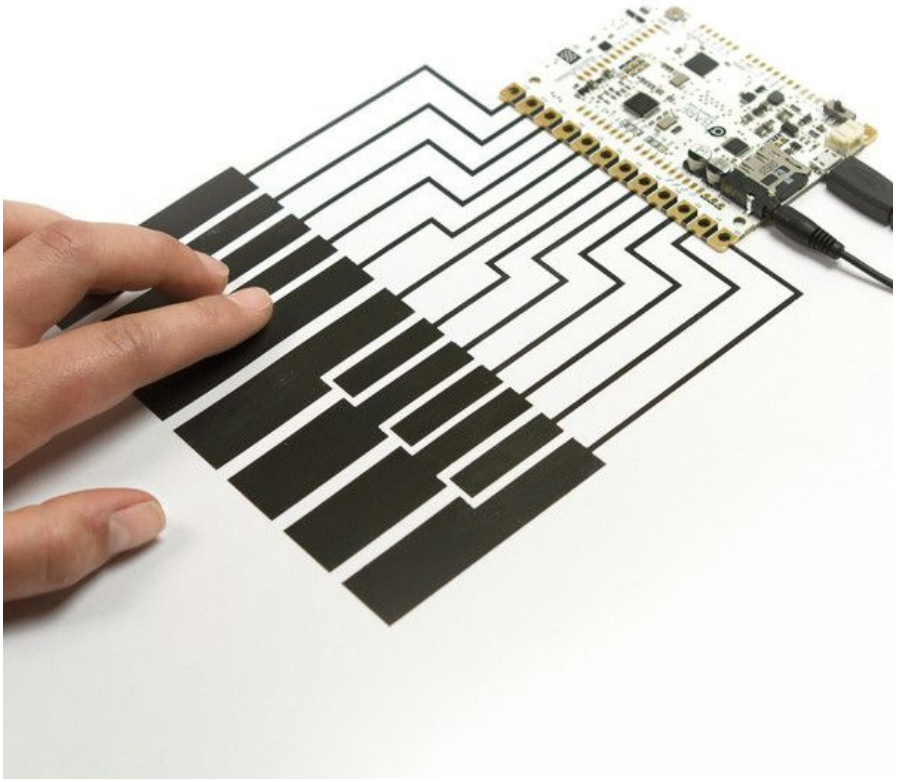
Institut Puig Castellar
Santa Coloma de Gramenet



Conduct the music at your touch

CFGS Desenvolupament d'Aplicacions Multiplataforma

Mario Gordillo Ortiz
Jonathan López Simón

2n DAM B

01/06/17

Resumen del proyecto:

Proyecto orientado a la sintetización de audio mediante una arquitectura cliente/servidor, en la que múltiples clientes pueden establecer conexión con el servidor, el cual hará peticiones de búsqueda, ya sea de clientes o de placas disponibles. Cada cliente puede tener conectada una o más placas (como puede ser BareConductive). Los clientes enviarán al servidor, mediante una conexión UDP, contínua información de los valores que recibe por el puerto serie de las placas conectadas, en las cuales hemos personalizado nuestro propio programa en código arduino, que nos devuelve un valor entre 0 y 1024, dependiendo de la cantidad de electrones que reciben los pines de ésta.

En nuestro caso hemos utilizado la placa llamada BareConductive, basada en Arduino, que contiene un total de doce pines sensibles a la electricidad estática, haciendo posible que detecte si está siendo tocado o no. Juntando estas posibilidades con teclas dibujadas con tinta conductiva sobre cartulina, simularemos un teclado, el cual podremos configurar a nuestro antojo y tocar nuestras canciones favoritas.

Abstract (in English, 250 words or less):

Project oriented towards audio synthesis through a client server architecture. Multiple clients can connect with the server, which will make search requests to find available boards or clients. Each client can have one or more boards connected, for example a BareConductive board. Through an UDP connection the client sends to server information from the serial port that boards create. The server will parse the data and will find which key is being pressed. That's possible thanks to the Arduino program that we wrote for the BareConductive. The program returns a value between 0 and 1024 depending on what detects in the pins.

In our case we have used a board named BareConductive, is based in Arduino and has 12 pins that are sensitive to static electricity, making possible the touch detection. Then we can draw a keyboard with conductive ink in a cardboard so we can "touch" our favourite songs.

Palabras clave:

Música, sintetización, BareConductive, Java, TCP, UDP, Cliente/Servidor

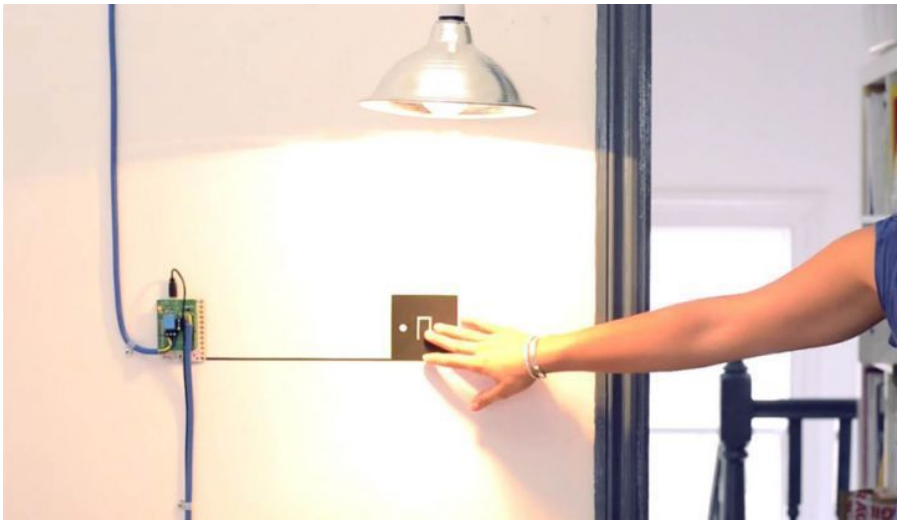
Índex

1. Introducción

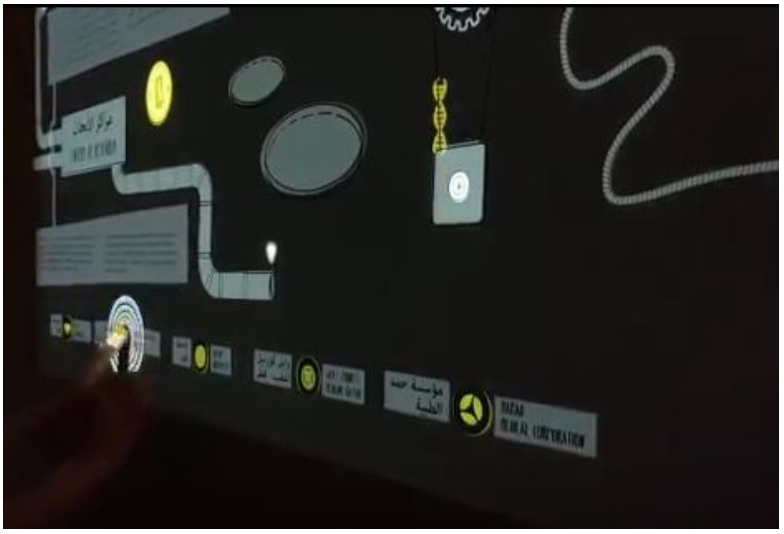
1.1 Contexto y justificación del trabajo

El objetivo de este trabajo es tratar con datos por serial, hilos, procesado de datos y su persistencia y sonido.
A raíz de la experimentación se espera obtener un sintetizador de audio, que puede ser controlado a partir de las entradas obtenidas a través de serial, previamente generadas por BareConductive.

Podemos utilizar la funcionalidad de BareConductive para hacer miles de proyectos. Por ejemplo, en esta imágen, se puede observar cómo la han utilizado como un interruptor capacitivo:



Otro ejemplo, podría ser el transformar un panel de madera, PVC, o cualquier material común (no conductor), en un panel interactivo, simulando una funcionalidad “táctil” en este.



El trabajo se distribuirá en las siguientes fases:

- Aprender a programar Arduino con la librería de Bare Conductive.
- Diseñar la interfaz a utilizar con JavaFX.
- Programar todo lo que pasa desde que se pulsa una tecla hasta que se reproduce el sonido, ya sea en local o en red.
- Diseñar físicamente el teclado.

1.2 Objetivos del trabajo

Sintetizar audio mediante entradas de datos recibidas de los clientes junto con la configuración establecida en el servidor.

- Arquitectura Cliente-Servidor
- Escaneo de puertos serie en busca de componentes específicos
- Escaneo de red en busca de clientes
- Sonidos configurables
- Sistema de módulos
- Identificación por radiofrecuencia
- Ajuste mediante sensor de proximidad

1.3 Enfoque y método seguido

Se decidió desde primera instancia comenzar el desarrollo del producto desde cero, en primer punto, por obtener una experiencia más enriquecedora a la hora de afrontar problemas en la planificación y organización de un proyecto nuevo, sin base alguna.
Esto además también nos suponía el ahorro de tiempo al intentar estudiar y conocer el funcionamiento de un código que no ha sido desarrollado por nosotros, lo cual nos podría provocar ciertos problemas a la hora de querer ampliar ciertas partes de un proyecto, las cuales no estaban contempladas en el inicio de éste.

1.4 Breve sumario de productos obtenidos

Hemos obtenido un software de sintetización de audio básico, el cual trabaja bajo una arquitectura cliente/servidor.

- ✓ Arquitectura Cliente-Servidor
- ✓ Escaneo de puertos serie en busca de componentes específicos
- ✓ Escaneo de red en busca de clientes
- ✓ Sonidos configurables
- ✓ Sistema de módulos
- × Identificación por radiofrecuencia
- × Ajuste mediante sensor de proximidad

2. Planificación del proyecto

Tarea	**Precedentes**
1. Definición: definición del proyecto	
1.1. Descargar documentación: Descargar del enlace proporcionado por el tutor el plan de trabajo de ejemplo.	
1.2. Leer documentación: Lectura detallada del plan de trabajo de ejemplo, así como toda la información adicional aportada por el tutor.	1.1
1.3. Reunión de grupo: Nos reunimos con el tutor con el fin de resolver dudas.	
2. Documentación: Creación de la memoria, diagramas, plan de trabajo	
2.1. Plan de trabajo: Definición de nuestro proyecto a través de un documento que explique cómo vamos a trabajar.	1
2.2. Memoria: Durante todo el proyecto se irá describiendo todo lo que se va probando, descartando o implementando	2.1
2.3. Diagramas: Necesitaremos hacer diagramas de clase, casos de uso, secuencia.	3.3
3. Estructuración de trabajo: Planificar cómo se va a estructurar el proyecto	1
3.1. Buscar información: Búsqueda de patrones de diseño de software, estudio de metodologías de organización de código. Intentar tener una base aún más consistente de trabajo en grupo.	
3.2. Estudio de utilización de repositorios Git: Conocer como trabajar en grupo de manera sincronizada y organizada de forma paralela.	
3.3. Sentar las bases de proyecto: Establecer unas directivas de trabajo para poder trabajar de forma organizada y modular	3.1
4. Pruebas de integración con el software: Integrar el hardware disponible con pequeños programas de prueba para ver las capacidades del proyecto	
4.1. Entradas de serial: Desarrollar un pequeño programa en Java que sea capaz de leer entradas desde Serial. Luego este código es totalmente aprovechable para el proyecto	
5. Montaje del hardware: Integración de módulos físicos en un	

componente arduino para llevar a cabo el dominio de todos los parámetros posibles.	
5.1. Montaje de hardware	
5.2. Pruebas con el hardware recientemente montado: Realizaremos comprobaciones para ver que se integra todo adecuadamente.	5.1
6. Programación: Estructuración de lo que se va a programar y programación en sí.	
6.1. Implantación de la estructura: Desde el principio del proyecto ya creamos los paquetes y las primeras clases para poder empezar a programar desde una base escalable.	3.3
6.2. Diseño de interfaz: Diseñar la parte gráfica que el usuario verá. Todas las ventanas, menús, desplegables, etc...	6.1
6.3. Programación del backend: Programar como se va a comportar internamente nuestro software.	6.1
6.4. Integrar frontend con el backend: Hacer que la interfaz gráfica se integre perfectamente con el backend.	6.2,6.3
7. Presets: Creación de sistema de presets y diseño de algunos de ellos	
7.1. Creación de sistema de presets: Diseñar cómo podríamos cargar, guardar, editar presets.	6.4
7.2. Diseño de presets: Creación de presets de prueba para ofrecer al usuario una idea de lo que se puede lograr.	7.1
8. Optimización: Revisión y optimización del código	
8.1. Revisión de fallos: Revisar el código y ponerlo a prueba para ver los diferentes fallos	7.1
8.2. Optimización: Ver que partes de código se pueden optimizar.	8.1
9. Presentación del proyecto: Preparación de la presentación del proyecto, y exposición de éste.	
9.1. Planificar cómo llevar a cabo la presentación: Marcar que rumbo se llevará a cabo en la presentación, orden de exposición y forma de explicación del proyecto.	8.2
9.2. Exponer frente al tribunal evaluador: Explicar al tribunal que es lo que hemos estado gestando durante todo este tiempo. Exponerlo de forma práctica.	9.1, 2

1. Definición: definición del proyecto	15/02/17	28/02/17
1.1 Descargar documentación	15/02/17	17/02/17
1.2 Leer documentación	18/02/17	27/02/17
1.3 Reunión de grupo	28/02/17	28/02/17
2. Documentación	1/03/17	1/06/17
2.1 Plan de trabajo	1/03/17	9/03/17
2.2 Memoria	10/03/17	1/06/17
2.3 Diagramas	20/03/17	9/04/17
3. Estructuración de trabajo	1/03/17	19/03/17
3.1 Buscar información	1/03/17	6/03/17
3.2 Estudio de utilización de repositorios Git	1/03/17	13/03/17
3.3 Sentar las bases de proyecto	7/03/17	19/03/17
4. Pruebas de integración con el software	15/02/17	26/02/17
4.1 Entradas de serial	15/02/17	26/02/17
5. Montaje del hardware	11/04/17	24/04/17
5.1 Montaje de hardware	11/04/17	13/04/17
5.2 Pruebas con el hardware recientemente montado	14/04/17	24/04/17
6. Programación	20/03/17	13/05/17
6.1 Implantación de la estructura	20/03/17	2/04/17
6.2 Diseño de interfaz	3/04/17	27/04/17
6.3 Programación del backend	3/04/17	30/04/17
6.4 Integrar frontend con el backend	1/05/17	13/05/17
7. Optimización	14/05/17	25/05/17
7.1 Revisión de fallos	14/05/17	20/05/17
7.2 Optimización	21/05/17	25/05/17
8. Presentación del proyecto	26/05/17	2/06/17
8.1 Planificar cómo llevar a cabo la presentación	26/05/17	31/05/17
8.2 Exponer frente al tribunal evaluador	2/06/17	2/06/17

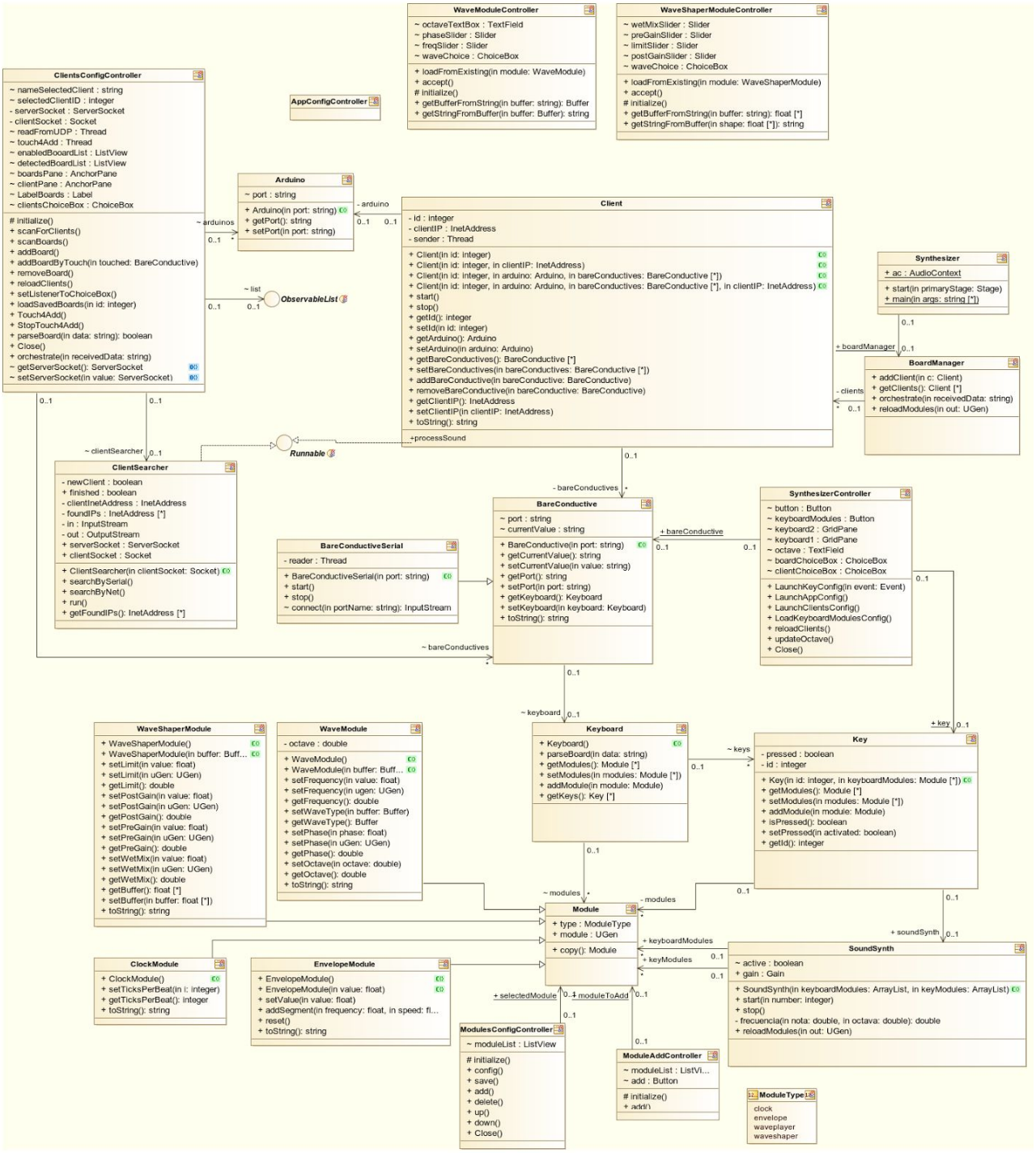


Diagrama de clases del servidor

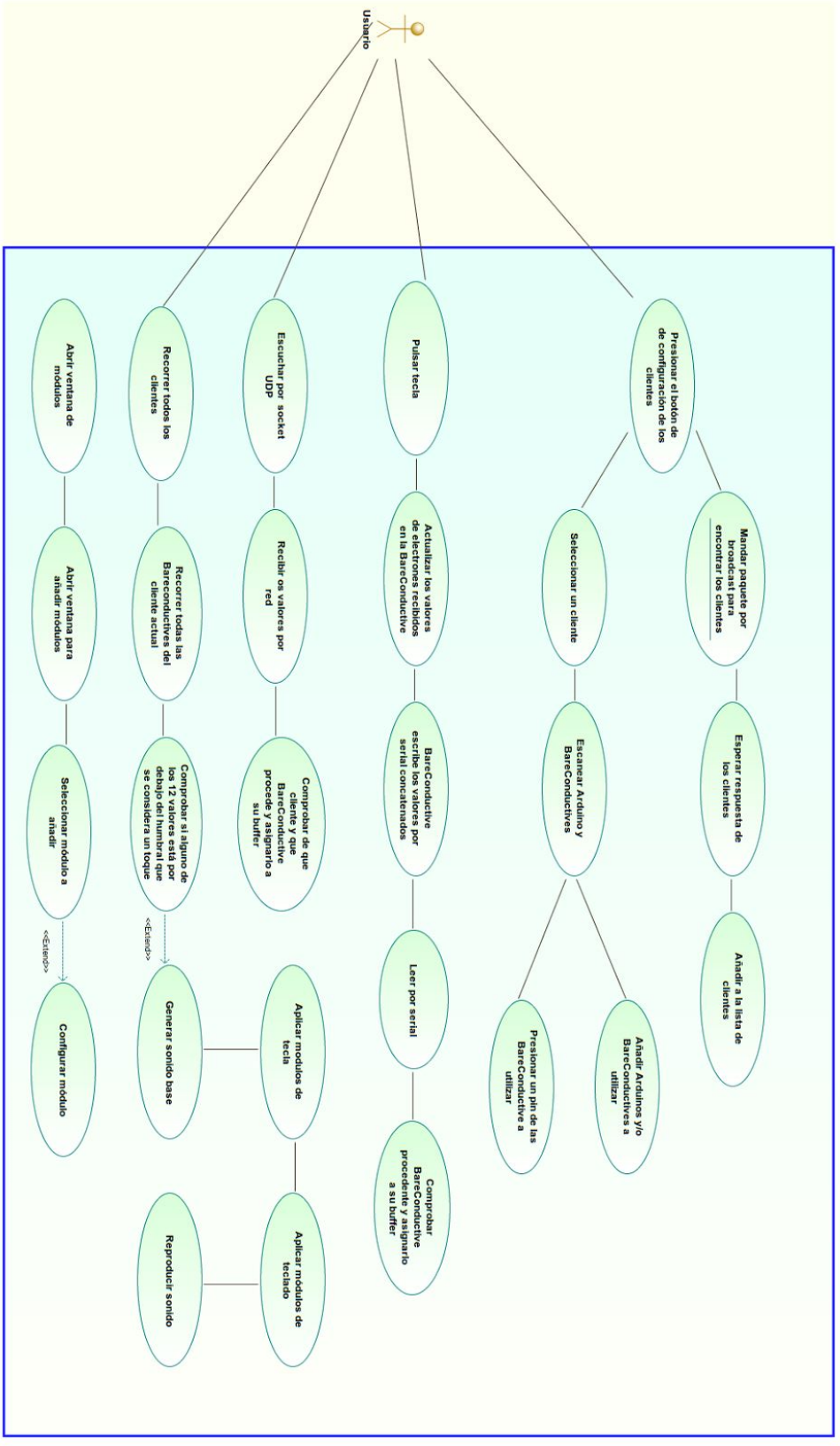


Diagrama de casos de uso del servidor

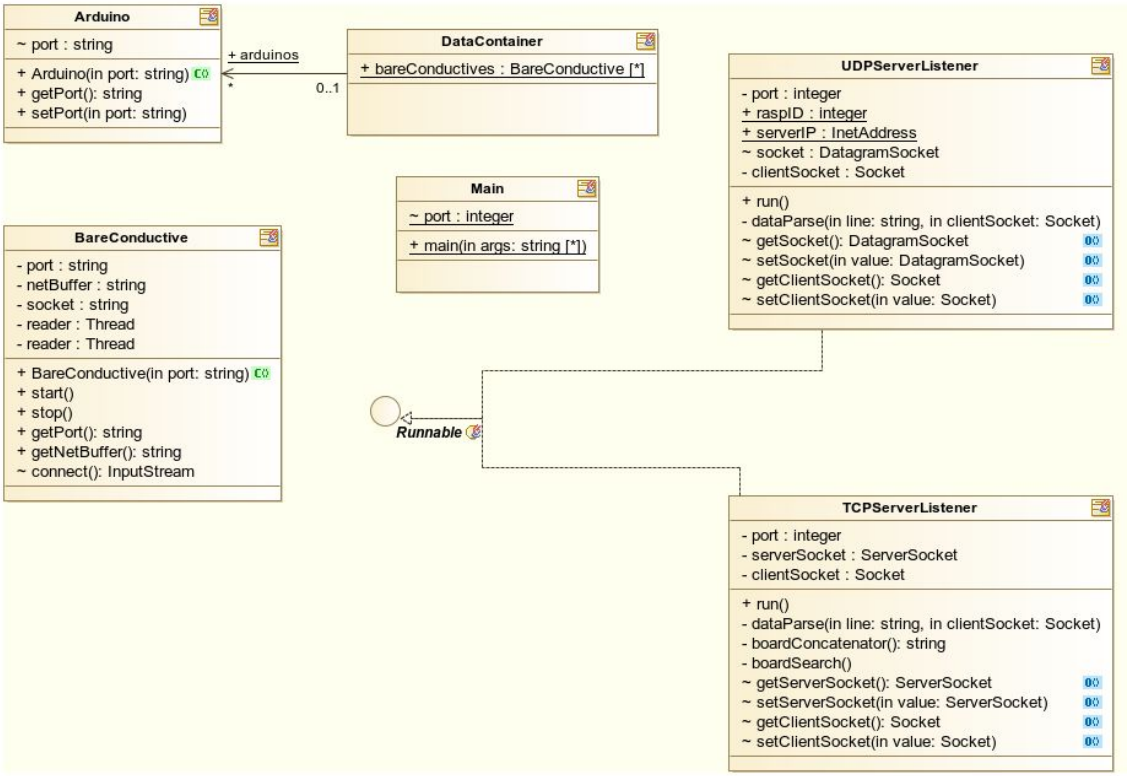


Diagrama de clases del cliente

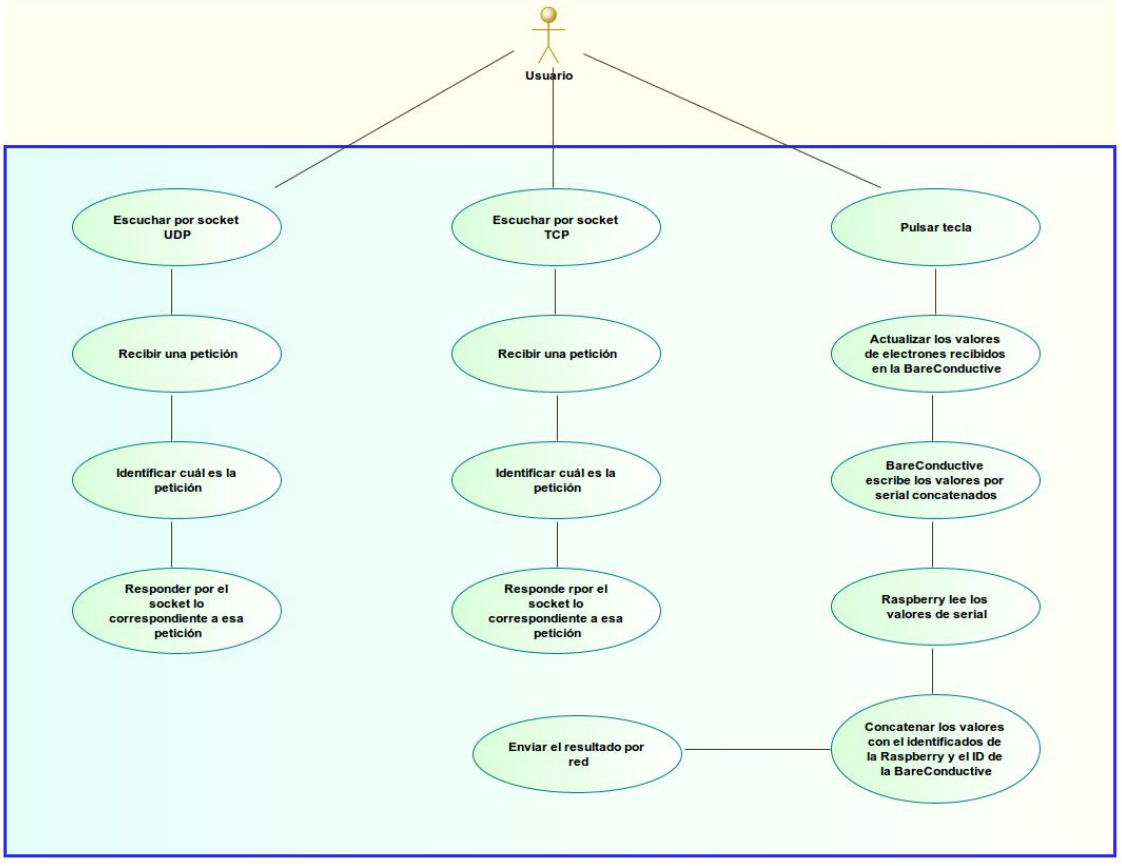


Diagrama de casos de uso del cliente

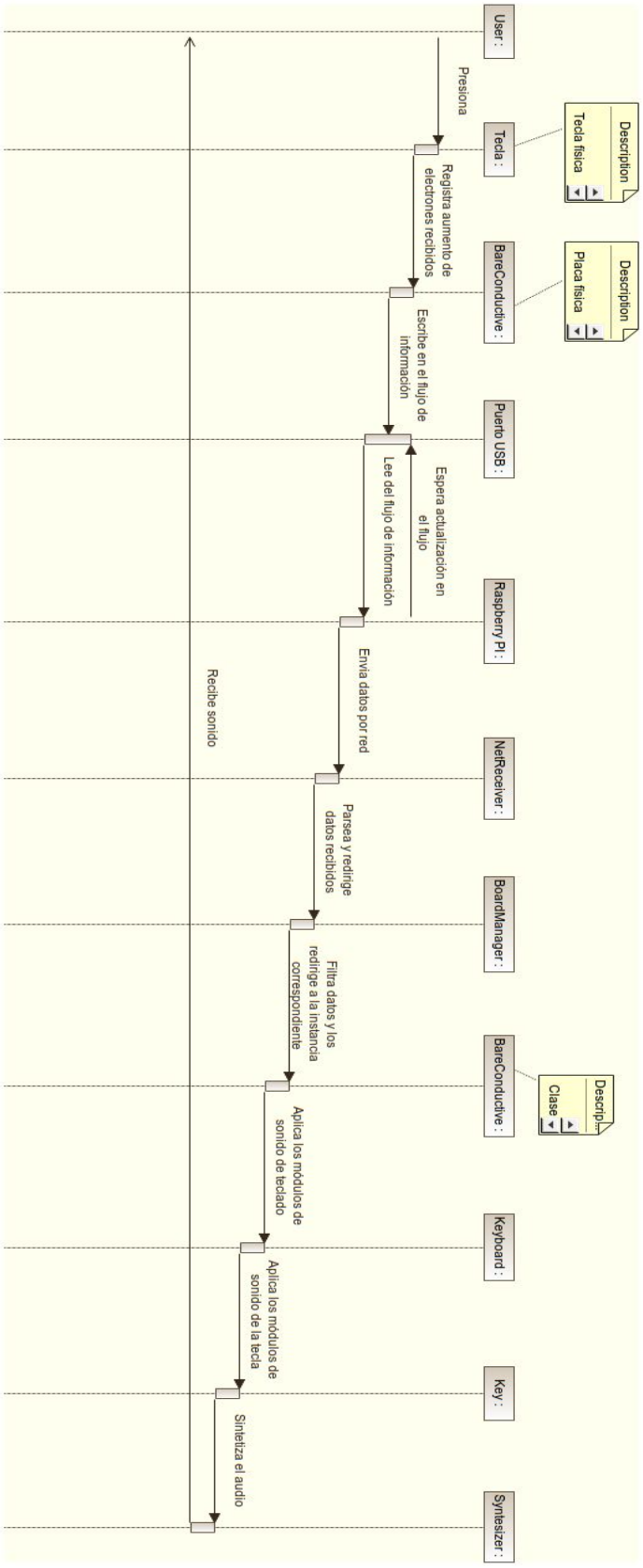


Diagrama de secuencia desde que se pulsa una tecla hasta que se reproduce el sonido

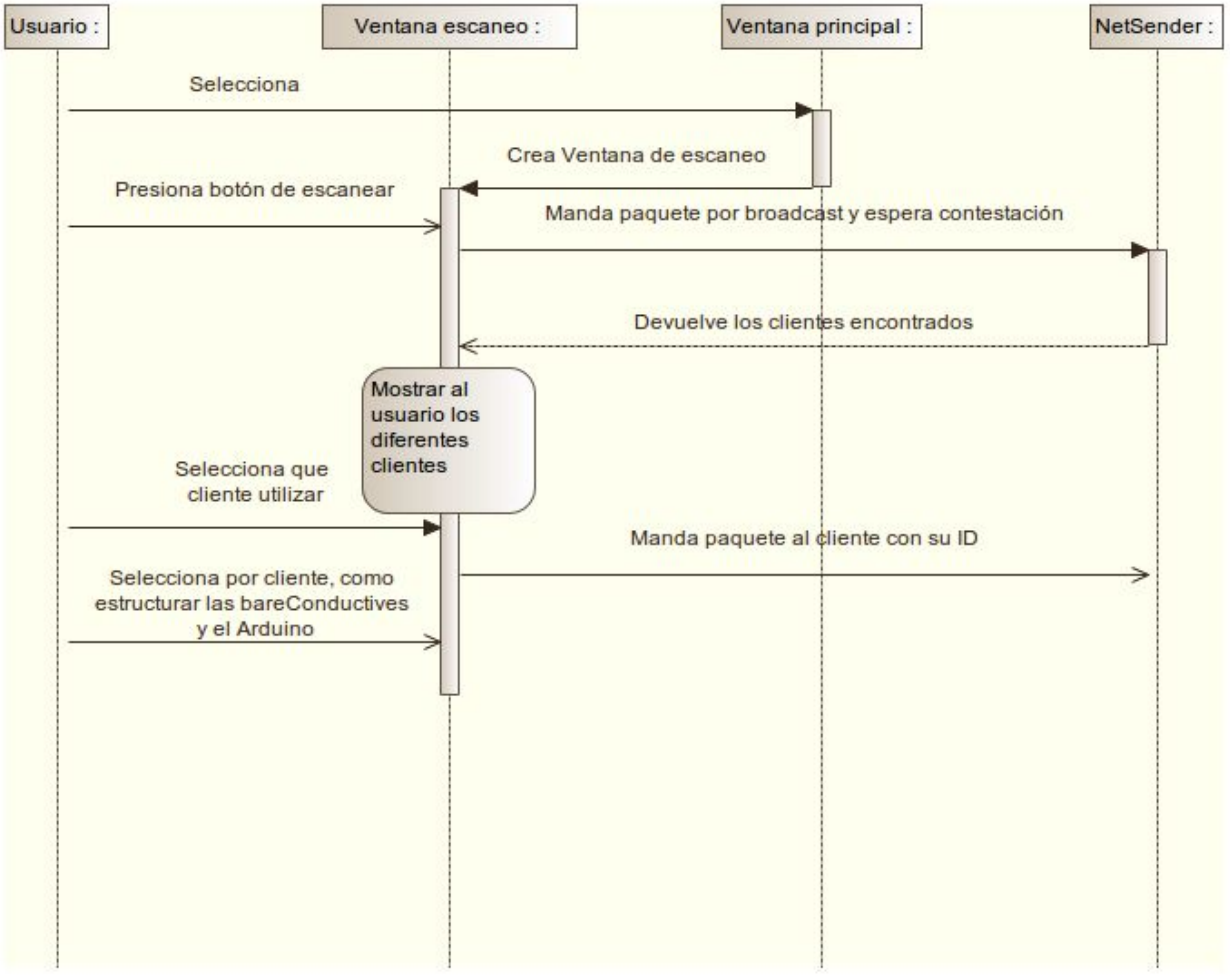


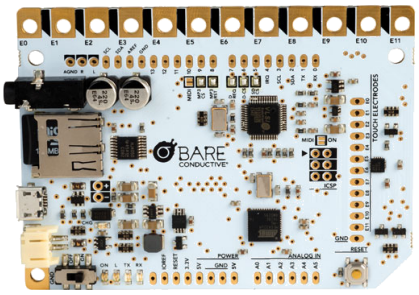
Diagrama de secuencia de búsqueda de clientes y placas

3. Requisitos

- Dinero
  - ✓ Placa Bare Conductive (85 €)
  - ✓ Placa Arduino (15 €)
  - ✓ Raspberry Pi Zero W (15 €)
  - ✓ Tinta conductiva (20 €)
  - ✓ DinA 2 (50 c)
  - ✓ Pincel para dibujar el teclado (2 €)
  - ✓ Cable Micro USB - USB (6 €)
- Conocimientos
  - Java
    - ➢ Hilos
    - ➢ Procesos
    - ➢ Callbacks
  - Arduino
    - ➢ Escribir en serial
    - ➢ Configurar la placa acorde a las necesidades exigidas
  - Serial
    - ➢ Leer entradas de serial desde cualquier S.O.
  - Git
    - ➢ Funciones básicas
    - ➢ Ramificación y unión de ramas

4. Hardware reducido

4.1 Placas: Explicación a fondo y datos que devuelve



BareConductive es una placa similar a Arduino que posee la característica de detectar cambios con la electricidad estática a través de sus doce pines.
El lenguaje de programación de la placa es Arduino, de hecho, para programar con BareConductive, podemos descargarnos desde su página web oficial los plugins, que posteriormente cargaremos en el IDE de Arduino.
Al igual que Arduino, la placa nos permite mandar información a un puerto serie virtual que se crea automáticamente al conectarla por USB, por lo tanto, podemos programar aplicaciones que apunten contra el puerto serie de nuestra placa. En nuestro caso, hemos utilizado el lenguaje de programación Java y para comunicarnos con el puerto serie hemos utilizado la librería RXTX.
RXTX es una librería de código abierto que nos permite acceder al sistema de puertos serie tanto como para leer y escribir. Está disponible en diferentes repositorios de distribuciones linux, la podemos instalar desde los repositorios oficiales de la distribución Archlinux bajo el nombre de paquete “java-rxtx” o en Ubuntu como “librxtx-java”. Una vez instalado el paquete, faltará incluir el JAR en nuestro proyecto como librería.

El programa que está en nuestras BareConductive simplemente recorre cada pin recogiendo el número de electrones que recibe representado en un número que oscila desde 0 hasta 1024, siendo 1024 en caso de que no reciba ningún electrón.
Una salida de ejemplo de este programa sería

```
860,983,903,950,840,845,859,945,864,824,864,945
```

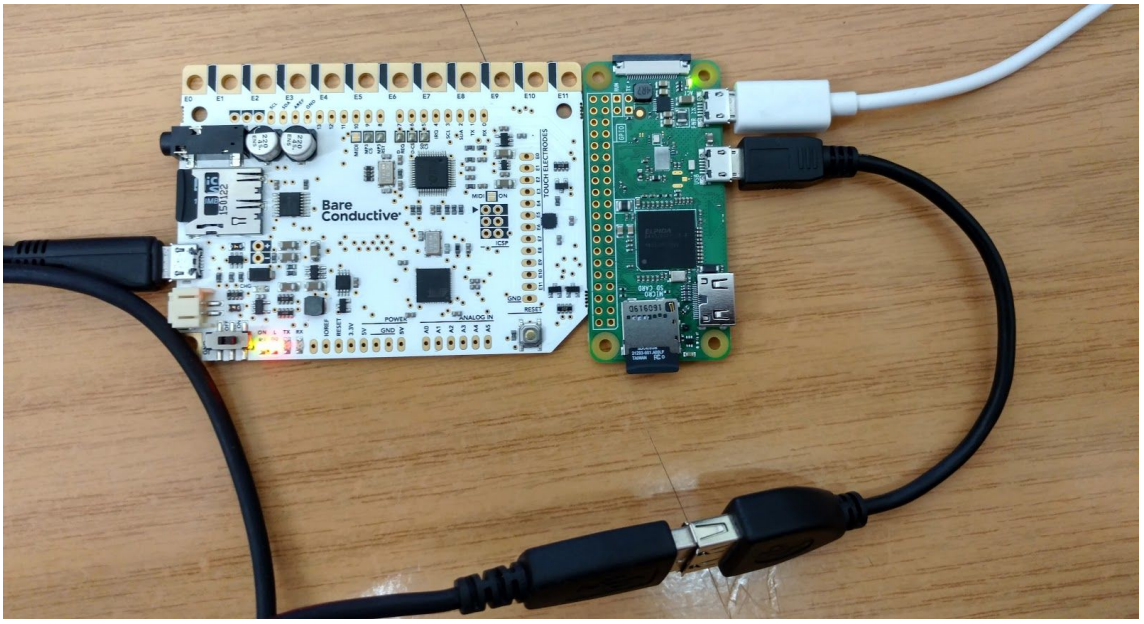
Como podemos ver, son doce valores, uno por cada pin, separados por comas. En este ejemplo, no estaríamos interactuando con la placa, pero si por ejemplo estuviésemos tocando el pin número 12 con la yema del dedo sería esta:

```
860,983,903,950,840,845,859,945,864,824,864,102
```

Como podemos ver el número ha decrecido porque estamos mandando electrones.
Gracias a esto, podemos tomar una decisión sabiendo que el pin número 12 está siendo tocado.

4.2 Raspberry Pi: Montaje y conexiones WiFi

Gracias a la arquitectura cliente/servidor podemos hacer que cualquier dispositivo capaz de correr Java y tener acceso a los puertos serie pueda ser un cliente. Cuándo Raspberry Pi Zero W fue lanzada, se nos ocurrió que podía ser un cliente perfecto.



Raspberry Pi Zero W se caracteriza por ser una Raspberry solo que de tamaño muy reducido. Además en su última versión a día de hoy incorpora una antena WiFi, lo que facilita mucho las cosas.

Solo nos tenemos que preocupar de que reciba corriente y tener conectadas nuestras BareConductive.

Raspberry puede correr distribuciones Linux compiladas para ARM. En su página oficial recomiendan Raspbian, que no deja de ser una adaptación a ARM de la famosa distribución Debian incluyendo algunas mejoras pensadas directamente para Raspberry.

El método de instalación es muy sencillo. Nuestra Raspberry tiene como disco duro una tarjeta SD, en esa tarjeta SD volcamos el sistema operativo que queramos.

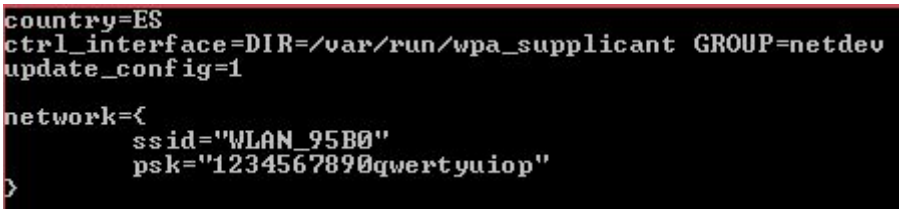
Raspbian provee de “img”s que podemos volcar directamente con un dd. El comando sería tal que así:

```
# dd if=/sitio/donde/tenemos/la/imágen.img of=/dev/SD
```

Una vez volcada la imágen, podemos insertar la tarjeta SD en nuestra Raspberry y encenderla para que se configure automáticamente.

Raspbian tiene un servicio de SSH preinstalado, pero todavía no tenemos internet. Para configurar nuestra red WiFi, tendremos que conectar un teclado y una pantalla a nuestra Raspberry y editar el archivo que hay en

```
/etc/wpa_supplicant/wpa_supplicant.conf
```



Para configurar nuestra WiFi podemos seguir el ejemplo de la imágen de arriba. Creamos un apartado llamado “network” y entre las llaves introducimos en “ssid” el nombre de nuestra red y en “psk” nuestra contraseña.

Una vez configurado el WiFi tendremos que iniciar y habilitar el servicio sshd, para ello utilizaremos los siguientes comandos:

```
# systemctl start ssh
# systemctl enable ssh
```

El primer comando inicia el servicio SSH y el segundo lo habilita para iniciarlo automáticamente en cada arranque de la Raspberry.

Si no tuviéramos acceso a un teclado ni a una pantalla, Raspberry nos ofrece la posibilidad de pre configurarla antes del arranque. Lo único que tenemos que hacer es crear un fichero “wpa_suppllicant.conf” tal y como lo configuraríamos desde la propia Raspberry y depositarlo en la primera partición de nuestra SD. Si también quisiéramos habilitar el servicio de SSH por defecto crearemos un fichero vacío con el nombre de “ssh” en la misma localización. De esta manera al arrancar sustituirá el fichero /etc/wpa_supplicant/wpa_supplicant.conf por el nuestro y habilitará ssh.

A la hora de ejecutar nuestro cliente necesitaremos tener instalado RXTX, para ello ejecutaremos este comando:

```
# apt-get update && apt-get install librxtx-java
```

Una vez hecho esto podremos conectarnos por SSH para copiar y ejecutar nuestro cliente. Podemos hacerlo fácilmente con estos comandos

```
$ scp CTMAYT-Client.jar pi@[IP de nuestra Raspberry]
$ ssh pi@[IP de nuestra Raspberry] \
    java -jar CTMAYT-Client.jar
```

Con el primer comando copiamos nuestro cliente en formato Jar a la home del usuario pi de nuestra Raspberry y con el segundo ejecutamos directamente nuestro cliente.

Y ya estaremos listos para ser encontrados por un servidor.

5. Librerías

5.1 Librerías gráficas

Desde los inicios del proyecto teníamos que tener clara qué librería gráfica íbamos a utilizar. Al principio empezamos a diseñar alguna pantalla con QtJambi. QtJambi es una adaptación de la librería Qt para Java que incluye su propio editor de interfaces.
Pero a la hora de poner en marcha las interfaces en nuestro programa, nos dimos cuenta de que lo que hacía era “Compilar” el diseño en un fichero Java. Tras una reunión de grupo decidimos que si queríamos modificar una ventana nos iba a tocar recompilar el fichero y volver a implementar las funciones asociadas de una manera poco cómoda, por eso la acabamos descartando y utilizando en su lugar JavaFX.
Los fxml de JavaFX se nos acomodaba más a nuestras necesidades para poder modificar interfaces libremente y de manera dinámica, sin tener que volver a implementar métodos que estuviesen asociados a la clase.

5.2 Librerías de audio

Durante el desarrollo del proyecto hemos ido probando y descartando algunas librerías de audio ya puede ser porque no nos acababan de sonar bien, no tenían un sistema el cual se pudiese adaptar a nuestra idea de los módulos o simplemente tenían un delay inaceptable.

JSyn: Tras varias pruebas de integración fue descartada por no ser del todo compatible con nuestra idea de los módulos.

TarsosDSP: Se adaptaba bien a nuestra idea de módulos pero tenía un retraso de casi un segundo que a la hora de tocar música, es un inconveniente.

Minim: Tras varias pruebas intentando hacer algo mínimamente complejo la acabamos descartando ya que prácticamente toda la documentación está enfocada a Processing.
Beads: Es la librería actualmente en uso, nos permite implementar módulos básicos ya que la entrada de un UGen(Objeto que genera o procesa sonido) es la entrada de otro.

6. Arquitectura cliente/servidor y comandos (peticiones)

Decidimos usar este tipo de arquitectura para permitirnos simular una banda real, situados cada uno en diferentes partes de una sala, sin tener que estar todos apretados en un mismo ordenador debido a que no se tienen cables USB suficientemente largos como para permitir usar a cinco personas un mismo ordenador sin que sea incómodo.
Nos decantamos también por éste, ya que nos da la facilidad de tener que montar sólo una máquina con la configuración adecuada (servidor), y a partir de éste punto, lo único que tienen que hacer los clientes es:
Iniciar el software del cliente, esperar que el servidor le haga una petición, y ponerse a tocar y disfrutar tocando sus canciones favoritas, ya sea con una o múltiples placas conectadas en cada uno de los clientes.

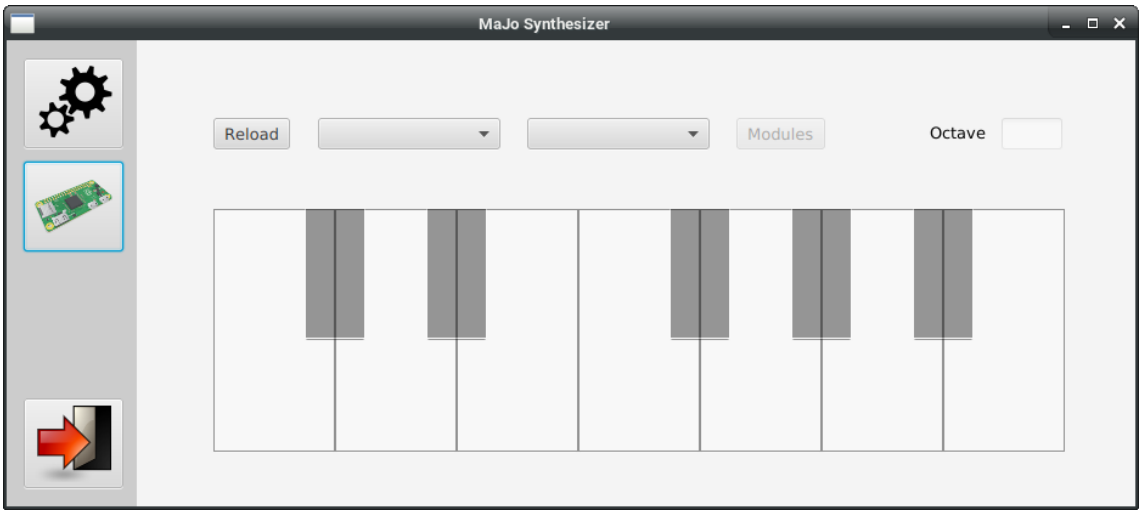
Actualmente contamos con dos instrucciones que puede hacer un servidor a los clientes, estas son:
Search: Es una solicitud para el reconocimiento de clientes en red, cada uno de los clientes deberá seguir el protocolo y contestar con un “+1” al servidor que le haga la petición. Por supuesto ésto está automatizado en la parte del cliente, y a nivel de usuario, no se debe realizar ninguna acción. Ésta es una petición que se hace siguiendo el protocolo UDP, permitiéndonos así enviar paquetes en broadcast para buscar cada uno de los clientes conectados a una red en concreto.
Boards: Es una solicitud que se hace a un cliente en concreto seleccionado por el servidor, con tal de poder reconocer cuales son las placas conectadas en determinado cliente, con el fin de poder habilitarlas/deshabilitarlas y aplicarle módulos de sonido sobre éstas. Esta petición se realiza mediante el protocolo TCP, estableciendo así una conexión a prueba de pérdida de datos, con tal de poder asegurar que cuando hagamos una petición a cierto cliente, obtengamos respuesta.

Este sistema está diseñado para ser ampliable en un futuro, añadiendo cualquier petición necesaria, ya sea mediante el protocolo TCP o UDP, ya que los clientes siempre están a la escucha por dos sockets diferentes, uno de cada tipo de conexión, permitiendo así seleccionar el más adecuado según las necesidades requeridas para cada una de las peticiones.

7. Interfaz de usuario

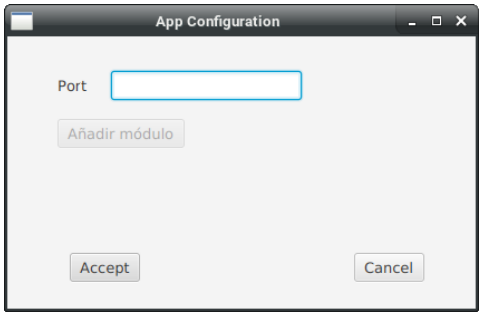
7.1. Diseño de la aplicación

Pantalla principal



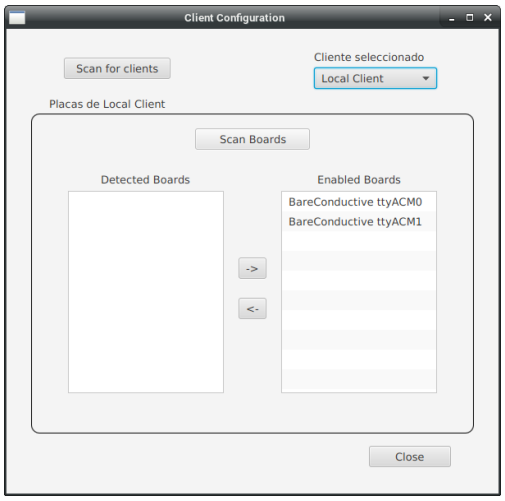
En la pantalla principal tenemos tres botones en la barra lateral.

Éste es el botón de configuración de la aplicación, de momento la ventana que abre no tiene utilidad, pero en un futuro tenemos planeadas funciones como por ejemplo la importación de módulos.



Esta es la ventana de configuración de la aplicación

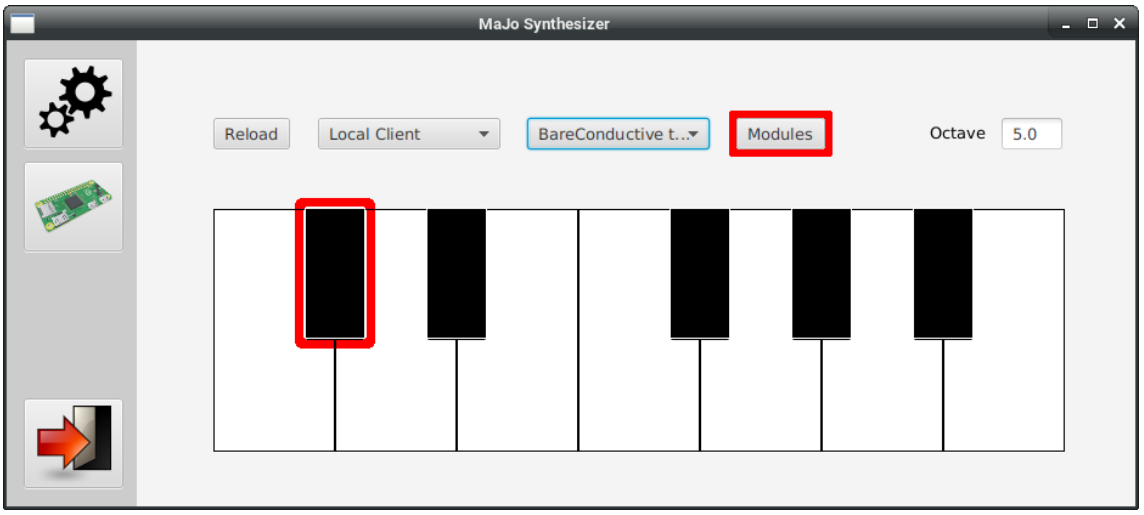
Éste es el botón de configuración de los clientes, desde aquí podemos encontrar clientes y placas remotas y locales

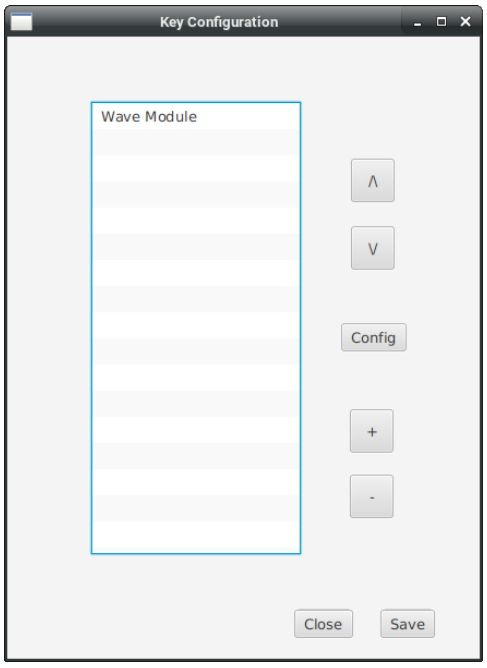


Esta es la ventana que abre el botón de configuración del proyecto.
Si le damos al botón de “Scan for clients”, nos rellenará el selector de la derecha con los clientes que encuentre.
Una vez seleccionado el cliente, podremos pulsar el botón de Scan Boards rellenará la lista de la izquierda con las placas encontradas. Para habilitar una placa, podemos tocarla físicamente o podemos seleccionar la placa y pulsar el botón “>>”

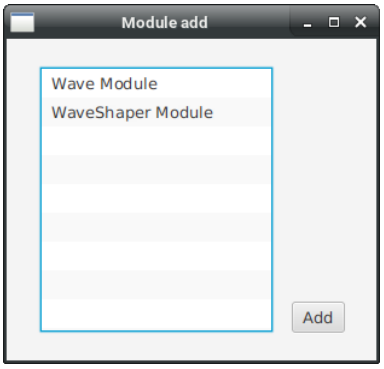
Éste es el icono de salir, si lo pulsas sales de la aplicación.

Una vez configurados los clientes, en la ventana principal del sintetizador podemos pulsar el botón de Reload para que se nos rellenen los desplegables con los clientes y placas. una vez seleccionada una placa de un cliente tendremos acceso al sistema de módulos.
Para configurar los módulos tenemos dos opciones, o configurarlos para toda la placa pulsando sobre el botón “Modules” o configurar módulos por tecla pulsando sobre la tecla que queramos configurar. A demás, podemos modificar la octava de todo el teclado si cambiamos el valor de “Octave”.

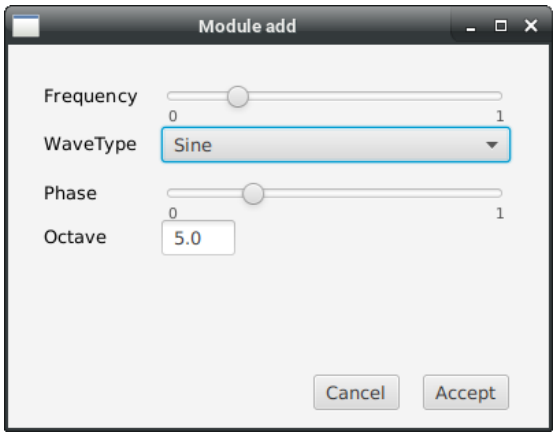


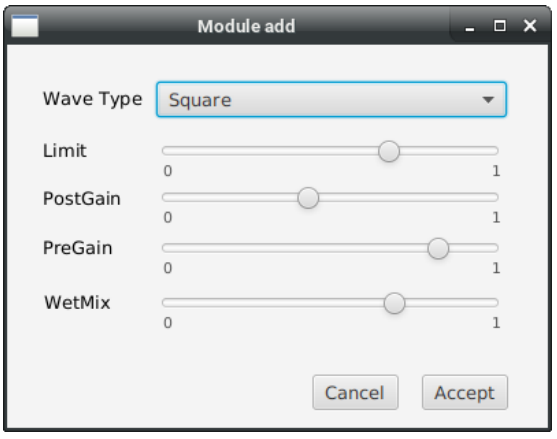


Esta es la ventana de configuración de módulos, tenemos cinco botones para manipularlos.
Los dos superiores “/\” y “\/”, nos permiten subir o bajar el módulo seleccionado.
Los dos botones inferiores “+” y “-” nos permiten agregar o quitar módulos.
En botón de “Config” central nos permite configurar el módulo.



Esta es la ventana que nos permite añadir un módulo, simplemente seleccionamos el que nos interese y pulsamos el botón “Add”

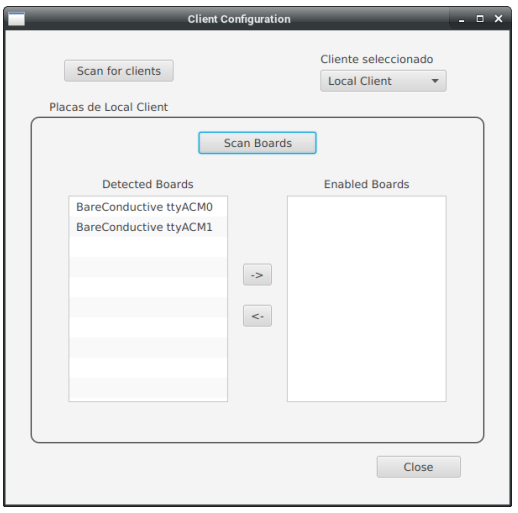




Estas son las ventanas de configuración de los módulos.
La ventana de la izquierda es la configuración de WavePlayer y la de la izquierda es la configuración de WaveShaper.
Como podemos observar, podemos modificar valores como la frecuencia, la octava, el tipo de onda, etc…

7.2. Touch4Add

Éste es un sistema pensado y llevado a cabo enteramente por nosotros mismos, llegados a cierto punto del proyecto, nos dimos cuenta que podíamos encontrarnos con la casuística de tener un cliente con varias placas conectadas, cuyo único método de identificarlo es su ID de puerto (por ejemplo: /dev/ttyACM2), lo cual nos llevó a la conclusión que era algo muy tedioso para tratar, y esto nos hizo pensar en cuál podía llegar a ser la mejor alternativa para poder seleccionar de forma fácil y sencilla, la placa que quieres añadir.



Tras mucho meditarlo, llegamos a la conclusión, que la mejor forma de seleccionar una placa de manera simple, era simplemente tocarla. Y así lo hicimos, implementamos un sistema que nos permitía, una vez teníamos las placas reconocidas y en la lista de selección, se añadiese a la lista de placas habilitadas para sonar con solo un toque en la misma, haciendo de éste un sistema sencillísimo para poder seleccionar las placas de una manera simple e interactiva.

7.3. Sistema de módulos

Nuestro sistema de módulos permite redirigir el sonido a través de diferentes módulos.
Para nosotros un módulo es un objeto que tiene de entrada un Módulo y de salida otro. De esta manera podemos hacer circuitos de sonido.
Un ejemplo sería el siguiente. Contamos con que cada elemento de la cadena es un módulo distinto.
WavePlayer (Genera una onda senoidal) >
WaveShaper (Altera la onda adaptándola a una triangular) >
Out (Salida de audio del sistema)
Además cada módulo tiene sus propias configuraciones, podemos hacer que el tipo de onda del WavePlayer sea: senoidal, triangular, cuadrada, dentada y de ruido. Variar su frecuencia o su phase.
WaveShaper tiene también sus propias configuraciones, como por ejemplo el WetMix que nos sirve para dejar parte de la onda proveniente mezclandola con la nueva.
Todo esto viene presentado por una interfaz de configuración por cada módulo.

8. Conclusiones

Lecciones

A lo largo de éste proyecto, hemos conseguido consolidar las bases ya aprendidas a lo largo del curso, como podrían ser:

M03 - Programación Orientada a Objetos

- Desde lo más básico, hasta sólidos conocimientos de la programación orientada a objetos. Así como creación de clases, clases abstractas, herencia de éstas, interfaces, sobreescritura y sobrecarga de métodos, tratamiento de excepciones, lambda expressions etc.
- Utilización de la librería de JavaFX e integración en un IDE.

M05 - Entornos de desarrollo

- Creación de diagramas, tanto de clase, como de secuencia, casos de uso, Gantt, etc…
- Uso de entornos de desarrollo integrados, como podrían ser Eclipse o IntelliJ IDEA.
- Utilización de Git para el control de versiones y la facilitación del trabajo en equipo sobre un mismo código.

M06 - Acceso a datos

- Gestión de de sistemas de ficheros con Java
- Creación y utilización de “streams” en las conexiones.
- Tratamiento de la persistencia de datos.

M09 - Programación de procesos e hilos

- Creación de hilos y “executors”, pudiendo así ejecutar tareas en segundo plano, y controlar estos hilos y la información que tratan de manera correcta.
- Establecer comunicaciones TCP y UDP entre diferentes ordenadores, con tal de poder comunicarnos y transmitir información de manera bidireccional.

También hemos tenido que aprender cosas por nuestra cuenta, como todo lo referente al sonido.

- Hemos aprendido a tratar con ondas de una forma básica. Aunque la librería de audio nos ha ahorrado el entender la creación y manipulación de ondas a bajo nivel, hemos tenido que aprender a base de experimentación con la cómo afecta la frecuencia al sonido de la onda o como afecta la propia forma al sonido.

Objetivos

Hemos conseguido asumir gran parte de los objetivos que nos propusimos al inicio del planteamiento de este proyecto, aunque nos hubiese gustado enormemente poder cumplir aún más objetivos, pudiendo profundizar y mejorar los existentes también.

Dos de los objetivos que no hemos podido cumplir son la creación de “presets”, y la posibilidad de configurar las opciones en los módulos desde el propio cliente mediante un arduino, instalándose en éste dos “hats”, uno de RFID (identificación por radiofrecuencia), pudiendo elegir así que quieres configurar, y un sensor de proximidad, pudiendo así manipular los valores de la configuración seleccionada con el movimiento de la mano sobre el sensor.
Es cierto que estos objetivos incumplidos no irrumpen en el correcto funcionamiento de la aplicación, pero sí sería una facilidad a la hora de ecualizar un cliente, sin tener que acceder físicamente al servidor.

Planificación y metodología

Al inicio del proyecto, definimos unas bases de cómo íbamos a estructurar las clases. Diseñamos los diagramas de clase pensando en poder implementar fácilmente cualquier cambio.
Por ejemplo, desde el principio sabíamos que queríamos hacer un sistema modular y cuando implementamos la librería de sonido, fué fácil adaptar el sistema a lo que queríamos porque estaba bien estructurado.
Constantemente íbamos aplicando cambios en la estructura del código con tal de mantenerlo organizado.

Posibles ampliaciones del proyecto en un futuro

Posibles ampliaciones podrían ser en un inicio, para comenzar, los dos objetivos del proyecto que no hemos podido alcanzar.
Además también nos gustaría aumentar el número de módulos disponibles de los que disponer a la hora de configurar el sonido de los clientes.
Mejora en el diseño de las interfaces para hacer más intuitiva la utilización del programa.
Y sin duda, nos gustaría poder mejorar el software que hemos implementado con tal de conseguir una más positiva experiencia de usuario en la utilización del mismo, haciendo de este algo divertido y didáctico a su misma vez con el que tratar, pudiéndose llevar también en un futuro a un software con más seriedad, como podría ser una herramienta para una banda que está comenzando y tiene poco presupuesto, el poder practicar e introducirse en el mundo de la música con sus compañeros.

Conclusiones personales

Al inicio de este proyecto, lo primero que hicimos fue sentarnos en una misma mesa, papel y lápiz en mano, y planificar.
¿A dónde podía llevar este proyecto?
¿Que conocimientos aprendidos en clase podríamos implementar?
¿Que puede tener de especial?
¿Conseguiremos llamar la atención?
¿Tiene posibilidad de crecer y de ser un software del cual nos podamos sentir orgullosos al finalizarlo?
Y la más importante de todas: ¿Disfrutaremos haciendo este proyecto?

Tras este tipo de preguntas y después de debatirlo mucho, entre los tres proyectos que teníamos planificados, este era sin duda el que cumplía más requerimientos para ser el que llevásemos a cabo, nos ofrecía posibilidades de hacerlo crecer cuánto quisiéramos, a la vez de ser el más llamativo, y sin duda, era un reto enorme el poder llegar a plasmar nuestras ambiciones y la infinidad de ideas que pasaban por nuestra mente a la hora de pensar en lo moldeable que podía llegar a ser.
Pero, como ya se ha comentado antes, para dos amantes de la música como nosotros, sobre todo, tenía perspectiva de ser el proyecto con el que más disfrutaríamos a la hora de implementarlo, pudiendo decir en la actualidad, una vez lo hemos finalizado, que considero que no nos equivocamos y escogimos la mejor de las opciones!
En resumen, se podría decir que estamos orgullosos de la idea que tuvimos, pero aún más, lo estamos del resultado que hemos obtenido, y de lo que hemos conseguido a lo largo de este tiempo.

9. Glosario

Arduino:

Compañía de hardware libre y una comunidad tecnológica que diseña y manufactura placas computadora de desarrollo de hardware y software, compuesta respectivamente por circuitos impresos que integran un microcontrolador y un entorno de desarrollo (IDE), en donde se programa cada placa.

BareConductive:

Bare Conductive es una empresa de diseño y tecnología que produce una pintura eléctricamente conductora, hardware de sensores capacitivos y una gama de kits.

Raspberry

Es un computador de placa reducida de bajo coste, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.

Java:

Es un lenguaje de programación de propósito general, concurrente, orientado a objetos. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra.

JavaFX:

Es una librería gráfica para Java desarrollada por Oracle Corporation. Inicialmente fue pensada para utilizarse en la web, pero desde su versión 1.3 permite crear aplicaciones de escritorio.

Cliente/Servidor:

Es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta

TCP:

Es uno de los protocolos fundamentales en Internet. Éste garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron.

UDP:

Es uno de los protocolos fundamentales en Internet. No tiene confirmación ni control de flujo, por lo que los paquetes pueden adelantarse unos a otros; y tampoco se sabe si ha llegado correctamente, ya que no hay confirmación de entrega o recepción.

Sintetización:

Consiste en obtener sonidos a partir de medios no acústicos; variaciones de voltaje en el caso de la síntesis analógica, o por medio de programas de computadora en el caso de la síntesis digital.

Frecuencia:

Es una magnitud que mide el número de repeticiones por unidad de tiempo de cualquier fenómeno o suceso periódico.

Cuanto más frecuentes son las vibraciones (más ciclos por segundo) el oído percibe el sonido definiéndolo por tal sensación como más "agudo", y a la inversa, al ser menos frecuentes, como más "grave"

Octava:

Se denomina octava al intervalo de ocho grados entre dos notas de la escala musical.

En música, una octava es el intervalo que separa dos sonidos cuyas frecuencias fundamentales tienen una relación de dos a uno. Ejemplo de octava: el la_4 (A5 en inglés) de 880 Hz está una octava por encima respecto a la_3 (A4) de 440 Hz.

10. Bibliografía

StackOverflow - https://stackoverflow.com/
Raspberry Pi - https://www.raspberrypi.org/
BareConductive - https://www.bareconductive.com/
BitBucket - https://bitbucket.org/
Beads - http://www.beadsproject.net/
Java Api - https://docs.oracle.com/javase/8/docs/api/
JavaFX Api - https://docs.oracle.com/javafx/2/api/
RXTX - http://rxtx.qbang.org/