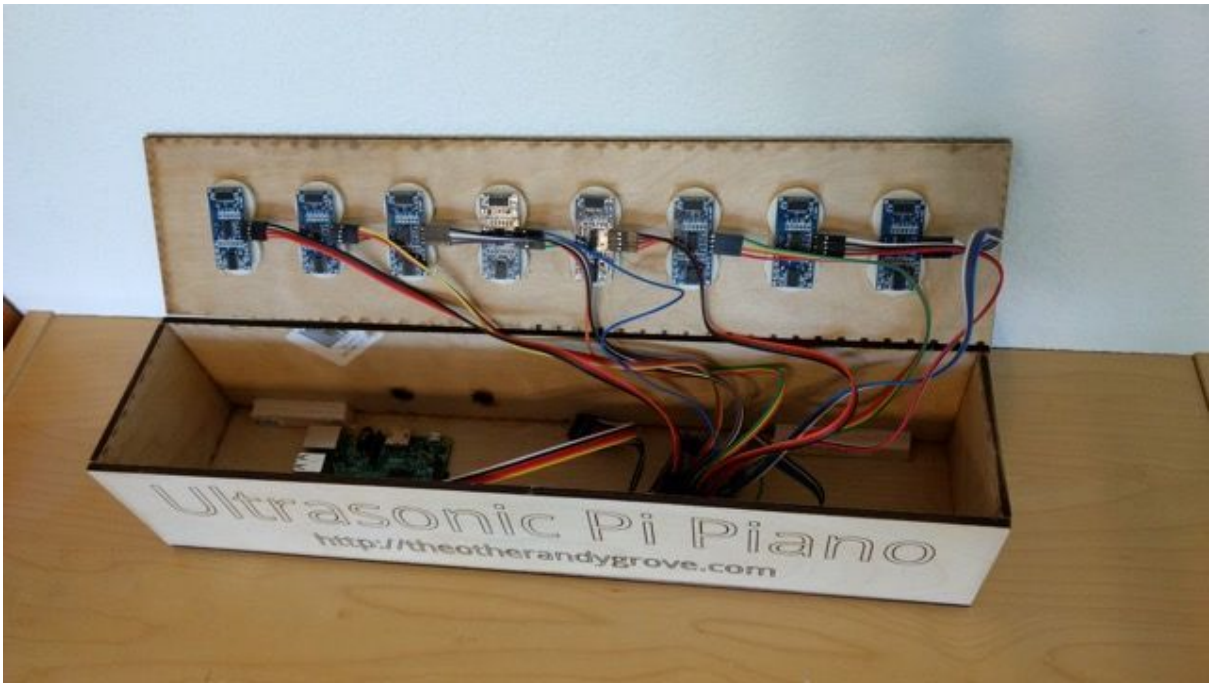
Raspberry Ultrasonic Piano

CFGS Administració de Sistemes Informàtics i Xarxes



Nom estudiants participants

Nicolás Fernández Aguilar

Curs acadèmic

2n ASIX

Data Lliurament

31 de març de 2019

Se construirá un instrumento musical que funcione mediante gestos captados por sensores de ultrasonido, que serán reproducidos por altavoces. Será necesario soldar los componentes de la placa que captará las señales de los sensores, y fabricar una carcasa que mantenga la integridad de todas las piezas.

A musical instrument will be built that works by means of gestures captured by ultrasound sensors, which will be reproduced by loudspeakers. It will be necessary to weld the components of the board that will pick up the signals from the sensors, and manufacture a case that maintains the integrity of all the parts.

Palabras clave: Raspberry Pi, Placa, Sensores, Ultrasonido, SPI, Soldadura, Electrónica.

1 - Introducción

En este proyecto se va a construir un instrumento musical basado en una Raspberry pi 3 como sistema central, conectado a ocho sensores de ultrasonido que registran las señales de entrada a partir de la proximidad (Ya sea de la mano u objetos). Estos sensores transmitirán la señal a una placa que se encargará de comunicarse con la Raspberry mediante SPI (Serial Peripheral Interface). Estas señales se interpretarán para reproducir sonidos MIDI a través de los altavoces.

Además, el control gestual permitirá cambiar de instrumentos y detener la raspberry dependiendo de la posición de las manos.

1.1 Objetivos a cumplir

Construir una caja de música que funciona por control gestual, debe permitir cambiar de instrumentos y detenerse la reproducción de música a partir de gestos. Aprender sobre la comunicación SPI y el estándar General Midi. Además de crear y mantener todo el proyecto bajo una licencia de software libre.

1.2 Análisis de los requisitos

Hardware:

- Raspberry Pi 3 y tarjeta SD.
- 8 HC-SR04 - Sensores de ultrasonido.
- Octasonic Breakout Board.
- Bi-Directional Logic Level Converter.
- 32 x 12" Female-Female Jumper Wires para conectar los sensores.
- 13 x 6" Female-Female Jumper Wires para conectar la Raspberry, la placa Octasonic y el convertidor de nivel lógico bi-direccional.
- Una fuente de alimentación para la Raspberry.
- Altavoces.
- Carcasa.

Software:

- FluidSynth: Software de sintetización de audio basado en SoundFont.
- Rust: lenguaje de programación desarrollado por Mozilla. De ser posible se utilizará Python en su lugar.

Sistema Operativo:

- Raspbian.

1.3 Materiales

Componentes	Cantidad
Sensores	8
Breakout Board	1
Raspberry	1
Fuente de Alimentación	1
Convertidor de nivel Lógico	1
Jumpers FxF	2
Carcasa	1

*Se venden en paquetes de 40.

1.4 Planificación

La planificación inicial va adjunta en un fichero HTML llamado **PlanificacioInicial.html** para una visualización más cómoda.

Esta planificación está dividida en tareas, asignando un recurso específico a cada una, dependiendo de la naturaleza de la misma. Existen tres recursos: *Builders* que son los que se encargan de soldar el Hardware y realizar las conexiones e implementación de los diferentes componentes, además de ajustes que fueran necesarios. *Administrators*, están a cargo del sistema operativo y la configuración del software utilizado en el proyecto, como la activación del protocolo SPI en la Raspberry, al igual que de las tareas de testeo e implementación del código fuente.
Los *Managers* se ocupan de redactar la memoria y decidir cambios en la planificación.

Las tareas están distribuídas en semanas o días, dependiendo de su duración, y especificadas en horas de clase.

2 - Diseño Hardware

2.1 Raspberry Pi o Arduino

La primera elección de diseño a la que ha habido que enfrentarse fue posiblemente también la más fácil de tomar. A la hora de plantear el proyecto surgían dos posibilidades, realizarlo sobre una Raspberry Pi, o con Arduino. Ambos tienen ventajas y desventajas, mientras que la Raspberry opera a 3.3V, la placa que recogerá las señales de los sensores y las transmitirá lo hace a 5V, esto significaba tener que utilizar un componente intermedio que transformara la diferencia de potencial, mientras que Arduino ya trabaja a 5V, por lo cual esto sería innecesario.
Sin embargo este no es un componente caro o difícil de conseguir. La Raspberry, en su modelo 3 b o b+ ofrece más potencia y una calidad de sonido más nítida. Además, el factor decisivo ha sido contar ya con una Raspberry 3 b+ antes de realizar el proyecto, lo que abarata el coste del mismo.

2.2 Sensor HC-SR04

A la hora de diseñar el proyecto se barajó la idea de utilizar sensores lumínicos en lugar de sensores de ultrasonido, debido a su precio más reducido. La idea se descartó debido a diversos factores, siendo el principal que Raspberry Pi tiene algunos bugs documentados en cuanto a la compatibilidad con estos sensores. Por ejemplo se puede observar en el proyecto de Linda Zhang, que será mencionado más adelante, que para resolver el problema de captar la señal de un sensor lumínico tuvo que utilizar una placa arduino. También era un factor a tener en cuenta que la placa Octasonic estuviera pensada para dar soporte a estos sensores de ultrasonido.

El sensor HC-SR04 consta de un emisor y un receptor de ultrasonidos que trabajan a una frecuencia de 40KHz (una frecuencia inaudible para las personas).

Fotografía tomada en clase de un sensor HC-SR04

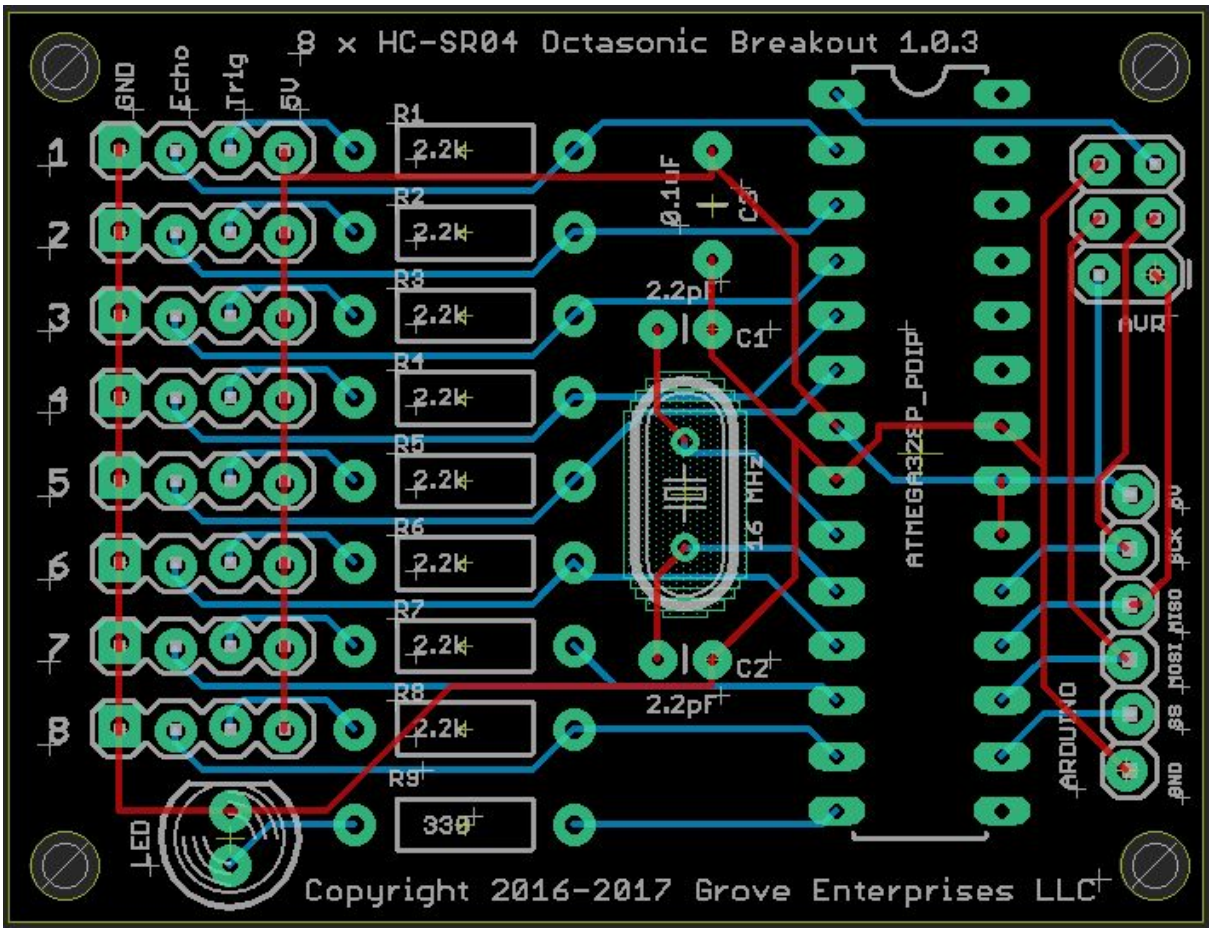
El principio en el que se basa su funcionamiento es muy sencillo, tan solo hay que generar una onda sónica en el emisor mediante un pulso en la patilla que pone "trig" (trigger o disparador), esta onda al encontrarse con algún obstáculo rebotará, volviendo al sensor y siendo registrada por el receptor, traduciéndose esta en un pulso en la patilla "Echo". Con esto se puede hacer dos cosas, detectar un obstáculo esperando simplemente que el microcontrolador reciba un "Echo" o contar el tiempo que transcurre desde que se manda el pulso por el trigger hasta que se recibe uno en Echo, de esta forma, y conociendo cual es la velocidad del sonido, se puede determinar de forma muy sencilla la distancia exacta a la que se encuentra el objeto en el que está rebotando la señal.

2.3 Placa “Octasonic Breakout Board”

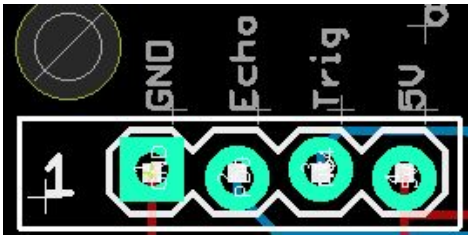
La placa Ocatonic Breakout Board está diseñada por Andy Grove específicamente para conectar los ocho sensores de ultrasonido HC-SR04 a una Raspberry o Arduino. Estos sensores que serán los encargados de capturar la señal de proximidad (input) que la raspberry transformará en sonido (output), utilizan cuatro conexiones: 5V, GND, Trigger, y Echo. Normalmente las conecciones Trigger y Echo se conectarían por separado a un microcontrolador o directamente a la raspberry, pero no es algo práctico al utilizar ocho de estos sensores (harían falta dieciséis pines).

Para eso entra en juego esta placa, que tiene conexiones dedicadas para soportar hasta ocho sensores, y su propio microcontrolador con un Firmware que monitoriza las señales de los sensores y los envía a la Raspberry Pi mediante SPI.

A continuación podemos ver un esquema de las conexiones de la placa y los diferentes componentes de la misma:

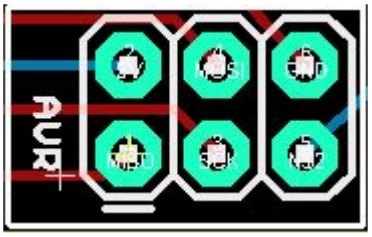


Por desgracia, al momento de realizar este proyecto no quedaba Stock de esta placa en tindie, web en la que su creador la vende. Por suerte el diseño de la misma es libre y está disponible en el github de Andy Grove. Gracias a este diseño en un archivo .brd se ha podido deducir los componentes de la placa. Desde las ocho resistencias de 2.2 kohms para los sensores y la resistencia de 330 ohms para el LED. Los capacitores de 2.2pF y 0.1µF y el oscilador cristal de 16mHz y todos los cabezales que servirán para realizar las conexiones.



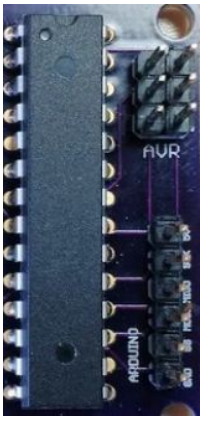
Cabezales (pines) de tipo macho para las conexiones SPI de los sensores de ultrasonido HC-SR04.

A pesar de estar señalado como **Arduino** estas conexiones serán las que sirvan para conectar la placa al convertidor de nivel lógico bidireccional, que hará de puente de la conexión con la Raspberry por motivos que se explicarán más adelante.



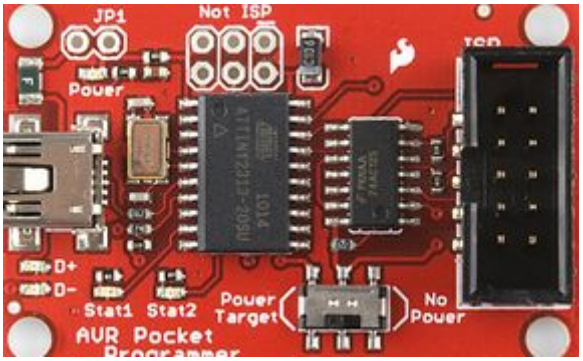
Las conexiones AVR servirán para quemar el Firmware en el microcontrolador Atmega238. Pero no únicamente, dado que el pin inferior izquierdo es también el cabezal de alimentación, que se conectará directamente al pin de 5V de la Raspberry Pi.

El oscilador de cristal es el componente que proporciona la frecuencia del reloj.



De estos componentes el más importante es el microcontrolador ATMEGA238. Dicho microcontrolador tiene seis conexiones en la placa que permiten quemar el Firmware, que habrá sido compilado anteriormente con AVR GCC Toolchain.

AVR GCC Toolchain es un compilador que servirá para compilar el código que más tarde se quemará en el microcontrolador. Sin embargo este programa no hace nada por realizar este último paso. Para eso es necesario AVR Downloader UploaDEr (avrdude). Con AVR GCC se creará un archivo hex.
Será necesario un programador AVR para poder quemar el código del Firmware dentro del microcontrolador, este hardware, que se puede ver en la siguiente captura, sirve para conectar el microcontrolador con el ordenador desde el que se quemará el Firmware.



AVR Programmer adquirido en Sparkfun, conecta el ordenador con el microcontrolador para quemar el firmware.

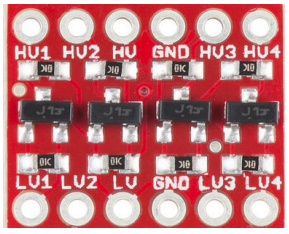
2.4 Conexiones con el Convertidor de nivel Lógico.

Los sensores HC-SR04 y la placa Octasonic requieren 5V, por otro lado, la Raspberry Pi funciona solo con 3.3V, debido a este problema es que se utiliza un convertidor de nivel lógico. Este convertidor tiene dos lados, uno de alto voltaje y otro de bajo voltaje. La Raspberry se conectará al lado de bajo voltaje (LV) y la placa al de alto voltaje (HV).
Pero estas conexiones requieren de un orden adecuado.

Estas conexiones se realizan en el panel lateral derecho de la placa donde se pueden encontrar seis cabezales de conexiones:

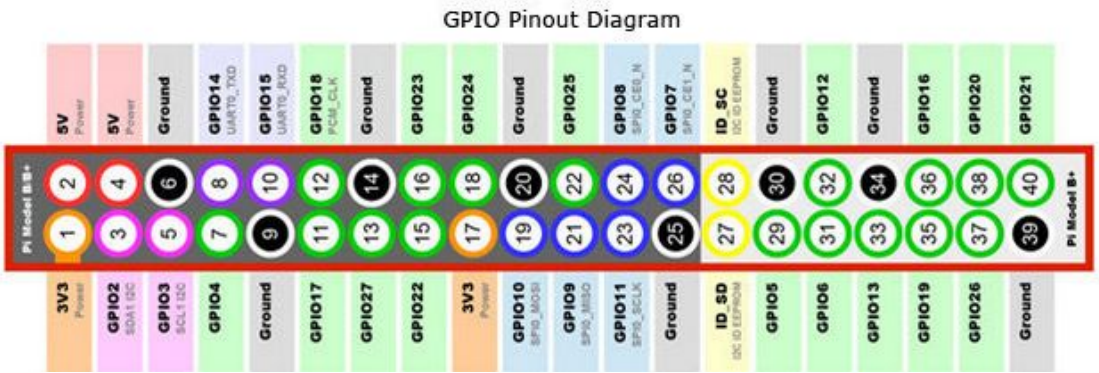
- 5V
- SCK Serial Clock
- MISO Master In, Slave Out
- MOSI Master Out, Slave In
- SS Slave Select
- GND (Toma a Tierra)

Estas conexiones se han de establecer con el convertidor lógico en el siguiente orden:

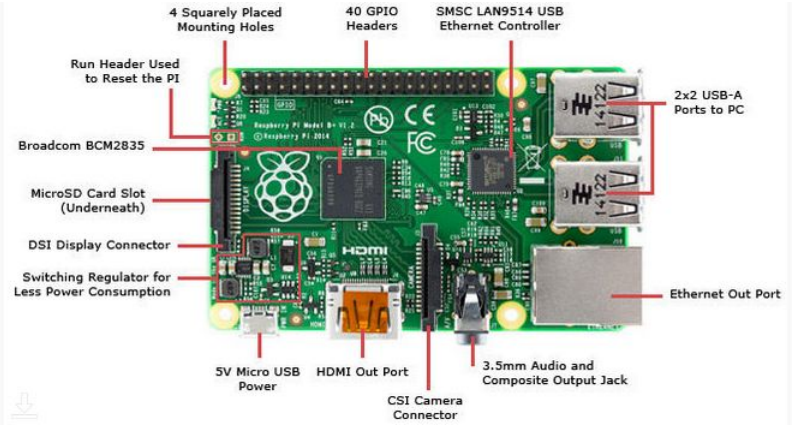


5V a HV
SCK a HV4
MISO a HV3
MOSI a HV2
SS a HV1
GND a GND

A la hora de conectar la Raspberry Pi también hay que seguir un orden determinado. En este caso es más complicado al no estar los cabezales indicados de forma tan clara como en la placa. Para descubrir cuales son las conexiones adecuadas se ha de mirar el siguiente esquema:



Estos cuarenta cabezales están ubicados en la Raspberry de la siguiente manera:



Gracias a este esquema se puede saber cual es cada cabezal de los que se quieren conectar al convertidor de nivel lógico. Siguiendo el orden mencionado a continuación:

3.3V a LV
GPIO11 (SPI_SCLK) a LV4
GPIO09 (SPI_MISO) a LV3
GPIO10 (SPI_MOSI) a LV2
GPIO08 (SPI_CE0_N) SS a LV1
GND a GND

Por último, la placa necesita 5V para su alimentación, para eso hay que conectar el cabezal de 5V de la Raspberry con el cabezal de alimentación de la placa, este es el que está en el extremo inferior izquierdo del panel AVR de la placa.

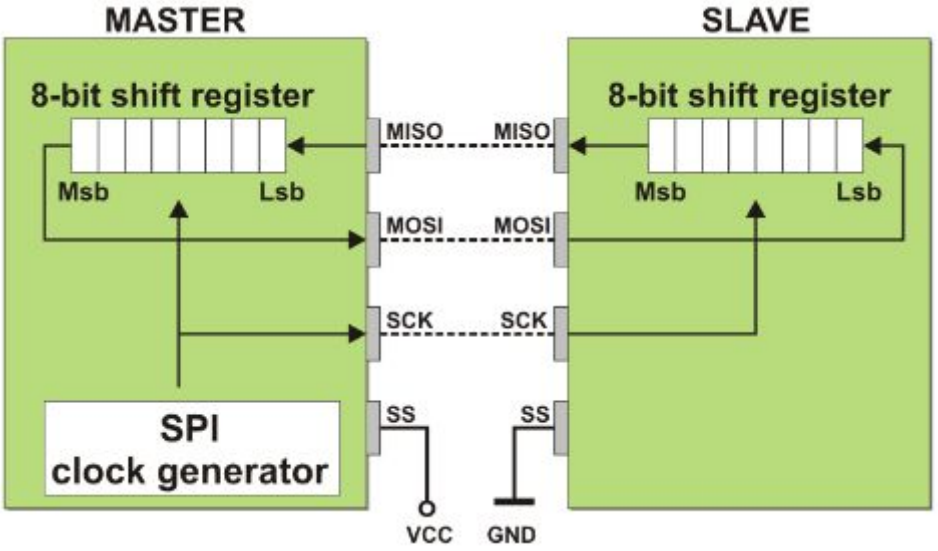
2.5 Serial Peripheral Interface

El bus de interfaz de periféricos serie o bus SPI es un estándar para controlar casi cualquier dispositivo electrónico digital que acepte un flujo de bits serie regulado por un reloj (comunicación sincrónica).
Incluye una línea de reloj, dato entrante, dato saliente y un pin de chip select, que conecta o desconecta la operación del dispositivo con el que uno desea comunicarse. De esta forma, este estándar permite multiplexar las líneas de reloj.

La ventajas de un bus serie es que minimiza el número de conductores, pines y el tamaño del circuito integrado. Esto reduce el coste de fabricar, montar y probar la electrónica.Y

tanto los sensores como la placa están diseñados para funcionar con este estándar, de hecho, tienen los cuatro cabezales (pines) que se utilizan.
El hardware consiste en señales de reloj, data in, data out y chip select para cada circuito integrado que tiene que ser controlado. La sincronización y la transmisión de datos se realiza por medio de 4 señales:

SCLK *(Clock)*: Es el pulso que marca la sincronización. Con cada pulso de este reloj, se lee o se envía un bit.
MOSI *(Master Output Slave Input)*: Salida de datos del Master y entrada de datos al Esclavo.
MISO *(Master Input Slave Output)*: Salida de datos del Esclavo y entrada al Master.
SS/Slave Select: Para seleccionar un Esclavo, o para que el Master le diga al Esclavo que se active.



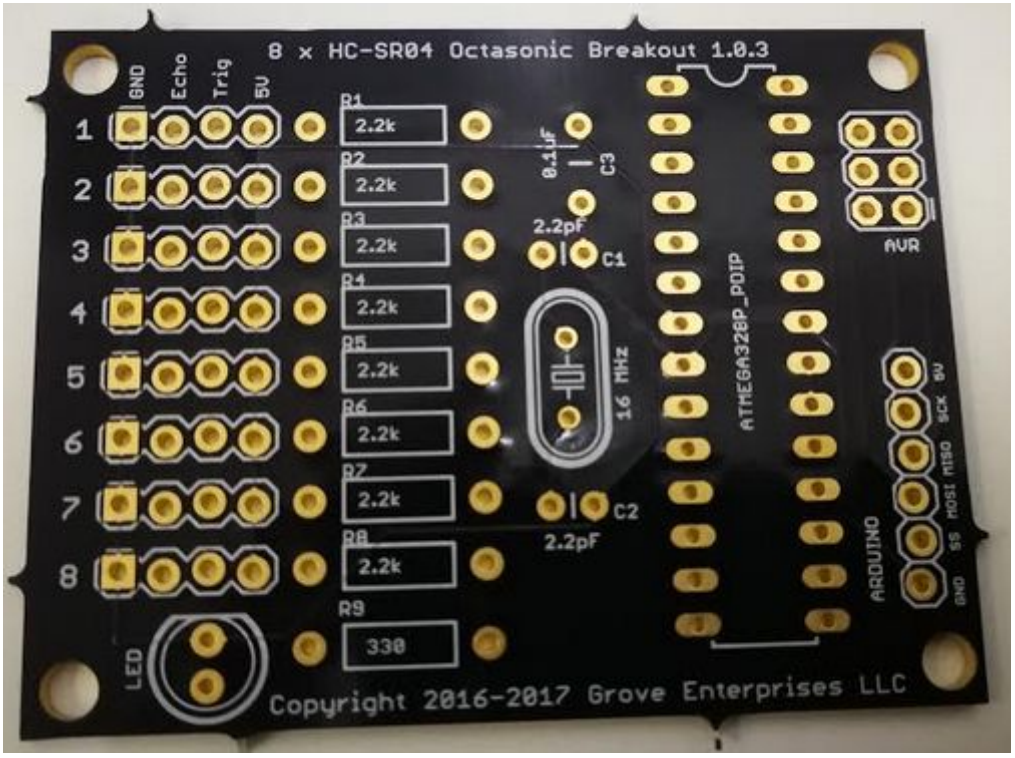
La Cadena de bits es enviada de manera síncrona con los pulsos del reloj, es decir con cada pulso, el Master envía un bit. Para que empiece la transmisión el Master baja la señal SS a cero, con esto el Esclavo se activa y empieza la transmisión, con un pulso de reloj al mismo tiempo que el primer bit es leído.

Las desventajas más importantes de este estándar son su falta de señal de aceptación, lo que podría significar que un servidor estuviera enviando datos a pesar de no haber ningún cliente conectado, sin saberlo. El hecho de que solo funciona en distancias cortas, por lo que el largo del cable es importante. Ninguna de estas desventajas (además de otras que tiene) son realmente importantes para este proyecto, por lo que este estándar es no solo el más adecuado sino el que está pensado para utilizarse por estos componentes.

3 - Montaje

3.1 Placa Octasonic

La placa Octasonic Breakout Board requiere que los componentes sean soldados. Para esto se encargaron las láminas a OHS Park, una empresa dedicada a la impresión de circuitos electrónicos a medida. En la web de dicha empresa se puede encontrar ya esta placa para ser fabricada y enviada, y también permite cargar esquemas en formato .brd para ser fabricados.
De este esquema se hizo un pedido de tres placas PBC, mientras que el resto de componentes de la placa se compraron por separado.



Placa Octasonic Breakout fotografiada en clase, sin los componentes soldados.

A la hora de soldar los componentes electrónicos a la placa se siguió el orden indicado por Andy Grove en la documentación encontrada en su página web. Todos los enlaces se pueden encontrar en la sección de bibliografía.
Dicho orden se establece de la siguiente manera:

ATMEGA
Oscilador de cristal
Capacitores
Resistencias (8 x 2.2k 1 x 330 ohm)
Pines
LED

A la hora de soldar, hay dos componentes que deben estar en una orientación concreta, estos son el microcontrolador atmega (el notch debería estar en la parte superior) y el LED, cuyo lado plano debería estar orientado hacia arriba. En la siguiente imagen se puede apreciar una placa completamente soldada:

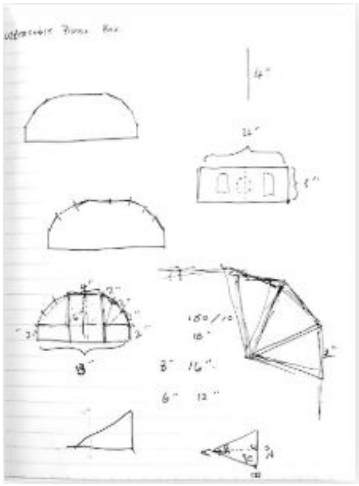
Soldar los componentes a la placa fue un proceso delicado y complicado. El principal contratiempo encontrado durante el mismo fue contar con un estaño de pésima calidad (Tan sólo un 25% de estaño), lo cual convirtió la soldadura en algo engorroso. En total se tardaron unas cinco horas en soldar todos los componentes. Durante los cuales se sufrió alguna quemadura menor, y se perdió tiempo cuando una gota de estaño se quedó insertada en uno de los espacios para los pines, al final hubo que sacarla con la punta de una broca tras probar diferentes métodos.

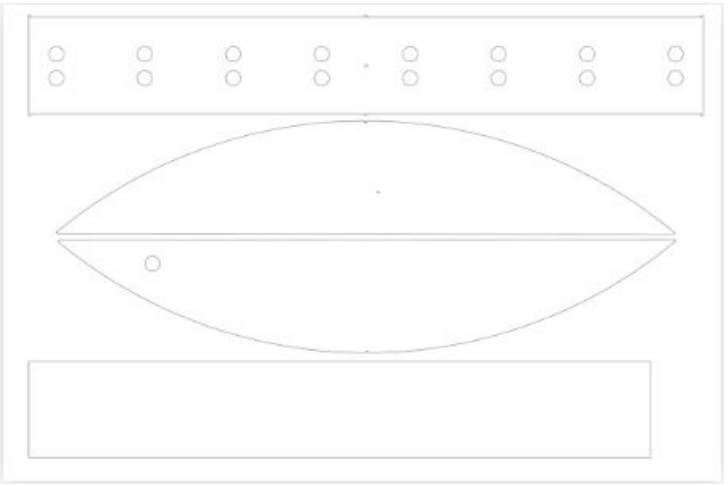
Había que tener un especial cuidado a la hora de soldar el microcontrolador con no calentar el mismo demasiado para no estropearlo, por lo que era necesario parar de vez en cuando.

3.2 Carcasa

El montaje de la carcasa fue el último paso en la realización del proyecto, una vez se habían realizado todos los tests y comprobado que la captación de señales y reproducción de sonido funcionaban. Sin embargo no fue ni el menos importante, ni el más fácil. Como se apreció en un proyecto de Georgia Tech, en el que una estudiante hizo su propia versión del del Ultrasonic Pi Piano, la disposición de los sensores en la carcasa puede afectar a la hora de causar interferencias en el control de la posición, debido a que los sensores pudieran registrar el movimiento de una mano por encima de ellos sin que este fuera deseado.

Para resolver la situación se diseñó una carcasa curva, que dispusiera los sensores en un ángulo de treinta grados. Se incluirá un enlace a su documentación en la sección de Bibliografía. A continuación se puede ver un boceto del diseño de **Linda Zhang,** del Instituto de Tecnología de Georgia:





Implementar dicho diseño habría sido lo ideal, pero no contar con una cortadora láser, además de materiales más flexibles que la madera, hizo que no fuera posible de implementar. Se decidió volveral estilo clásico de caja de música propuesto por Andy Grove al idear el piano ultrasónico.

Sin embargo para evitar las interferencias se decidió crear una carcasa de ochenta centímetros, pudiendo así dejar un espacio entre cada sensor lo suficientemente amplio para que colocar una mano encima no interfiriera con los que tenga a los lados.

Para crear la carcasa se cortó un tablón de ochenta por cuarenta centímetros en dos, dando lo que posteriormente serían la tapa superior e inferior de ochenta por veinte. Los laterales largos se pegaron a la base inferior utilizando silicona y un nivelador para que estuvieran tan rectos como fuera posible, y los laterales cortos dejaron un espacio vacío para pasar cables y para que el aire pudiera fluir por el interior de la carcasa y evitar que la placa se sobrecaliente.

El paso más complicado fue realizar los agujeros para los sensores. Estos tienen un diámetro de dieciséis milímetros, por lo que se consiguió una broca para madera del mismo tamaño. Lo que representaba un reto era posicionar el punto central de cada agujero para poder cortarlos. Lo primero que se hizo fue, con una hoja de papel y un lápiz, marcar ambos círculos de un sensor, colocando la hoja sobre el mismo y rayando con cuidado su superficie con el lápiz.

Una vez con un esquema en papel, se marcó el centro de ambos círculos a través de este con una broca sobre una lámina de madera y se agujereó para utilizar de molde.

Sobre la tapa superior de la carcasa se tomaron medidas para situar cada controlador a una distancia equidistante y simétrica en el centro de la tapa, sin embargo el esquema de madera era poco práctico y se dibujó uno sobre papel adhesivo. Este adhesivo se transparentaba bastante bien por lo que era fácil de colocar sobre las marcas hechas en la caja, de forma que se pudo marcar el centro de cada perforación clavando la punta de una broca en él.

Finalmente se incluyeron unas bisagras para que pudiera abrirse y cerrarse a voluntad por motivos de comodidad, y para facilitar enchufar los cables.

4 - Diseño Software

4.1 Sistema operativo

Raspberry puede funcionar bajo una gran cantidad de sistemas operativos hechos dedicadamente para la misma. Los oficiales y más utilizados son los siguientes:

GNU/ Linux – Uso PC o servidor

- **RASPBIAN** Debian Jessie
- **RASPBIAN Lite** Debian Jessie

GNU/Linux – Uso Media Center Kodi XBMC

- **OSMC**
- **LibreELEC**

Solo Raspberry Pi 2 y 3

- **Ubuntu Snappy Core** solo servidor sin escritorio gráfico.
- **Ubuntu MATE**

Además de un sinfin de sistemas no oficiales entre los que se incluyen CentOS, ARCH Linux, Kali, Devuan, openSUSE… Muchos de estos sistemas se descartan debido a que están hechos para funcionar con un objetivo diferente al que busca este proyecto. En esencia, lo único que necesita el sistema operativo es ser capaz de recibir señales mediante SPI.
De todos los sistemas se eligió Raspbian por la familiaridad adquirida a lo largo de los años en este instituto con los sistemas Debian, además del hecho de que varios tutorial a seguir en la realización del proyecto utilizan este sistema o sistemas basados en Debian.

Elegir Raspbian Lite también era una opción, sin embargo al contar con una tarjeta SD de 16GB no era necesario decantarse por una versión reducida del sistema operativo.

4.2 Lenguaje de Programación

A la hora de elegir un lenguaje de programación no habían muchas opciones. El Firmware para el microcontrolador de la placa está diseñado por Andy Grove y no hay ninguna necesidad de cambiarlo en este proyecto (se podrían tocar cosas en el código como la cantidad de sensores que se conectan a la placa, pero dado que el soporte máximo de la misma es para ocho, esto solo sería necesario en caso de ampliar el diseño de la misma).

Donde se ofrece una elección es a la hora de programar el código que interpretará las señales recogidas en la Raspberry y las transformará en sonido. En este apartado, se

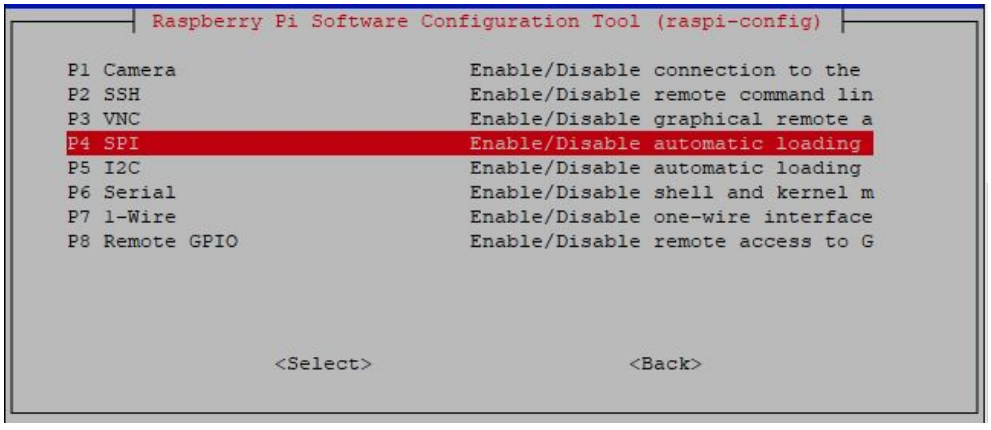
puede elegir la versión del código programada en Rust y una versión para Python. Dado que Python es un lenguaje con mucho más reconocimiento y recorrido que Rust, es el lenguaje que se usará en este proyecto a priori. Sin embargo, dada que esta versión es posterior y no ha sido probada tanto como la versión del código en Rust, de encontrarse problemas no habría ninguna complicación en cambiar de lenguaje.

Finalmente, tras probar con ambos lenguajes, ha sido elegido el código programado en Rust, debido a que el código python sólo dispone de una versión de prueba. Ambos son muy similares y no supuso ningún reto utilizar el código en Rust. Únicamente fue necesario instalar un compilador que interpretara dicho lenguaje.

4.3 Activar SPI

La Raspberry Pi tiene tres tipos de interfaz de serie incluidos en la cabecera GPIO. El puerto de serie UART, que permite abrir un inicio de sesión en una aplicación de terminal de serie, como por ejemplo PuTTY que es un cliente de telnet y SSH. Las otras dos son SPI (de la cual se ha hablado anteriormente) e I2C. Mientras que en raspberry se permite conectar hasta dos dispositivos mediante SPI, I2C permite conectar un número mayor, siempre que sus direcciones no entren en conflicto.

Activar SPI en la Raspberry no requiere de una gran complicación. Este estándar está desactivado por defecto, y para habilitarlo es necesario utilizar raspi-config. Este comando nos ofrecerá una interfaz en modo texto desde la cual cambiar la configuración de nuestra raspberry, y entre sus opciones se puede activar o desactivar la carga automática de SPI.



Esta configuración no tendrá efecto hasta una vez reiniciada la raspberry.

4.4 FluidSynth

FluidSynth es un software de sintetización de audio en tiempo real. Está basado en las especificaciones de SoundFont2. En sí, este software no cuenta con interfaz gráfica,

aunque varias aplicaciones y sistemas integrados móviles hacen uso de esta. Tiene una potente API y será el software a partir del cual se sintetice el sonido que reproducirá este proyecto.

En otras palabras, el código fuente se encargará de procesar las señales recibidas desde los sensores de ultrasonido para reproducir el sonido que sintetiza FluidSynth. Este sonido tendrá un formato MIDI y se reproducirá a partir de cualquier tipo de altavoz que sea compatible con una Raspberry (prácticamente cualquier altavoz para ordenador).

¿Por qué utilizar FluidSynth? Este software es de código abierto, y cuenta con muchas ventajas, como el ser multiplataforma (funciona al menos en Linux, MacOS y Windows), el hecho de recibir actualizaciones y ser un proyecto activo a día de hoy (la última actualización fue publicada el 19 de abril de este mismo año). El soporte para SoundFont 2 y SoundFont 3, y el playback de archivos MIDI.

Como alternativa al mismo está Swami Project, que es una colección de software libre que utiliza MIDI para editar o reproducir instrumentos musicales. Dicho proyecto es mucho más amplio, y hace uso de SoundFont al igual que FluidSynth. De hecho, entre muchas otras cosas, incluye el mismo FluidSynth como sintetizador. Dado que esta es una aplicación mucho más compleja que lo necesario para este proyecto, se descartó su uso en favor de FluidSynth.

Otros sintetizadores que podrían cumplir la función de FluidSynth son ZynAddSubFX, un sintetizador que funciona en múltiples plataformas, o Absynth 5.

4.5 General MIDI

Se trata de un estándar para instrumentos musicales electrónicos que responden a mensajes MIDI. Fue desarrollado por MIDI Manufacturers Association (MMA) y la Japan MIDI Standards Committee (JMSC) y publicado por primera vez en 1991. Este estándar especifica que todos los instrumentos que lo sigan deben al menos ser capaces de interpretar veinticuatro notas a la vez (Polifonía) y une diversas interpretaciones a parámetros y mensajes de control que anteriormente se habían dejado sin especificar. Como por ejemplo, definir el sonido de los instrumentos para los 128 posibles números programables.

En concreto, los instrumentos que siguen este estándar deben de ser capaces de:

Permitir veinticuatro voces activas a la vez (Incluyendo al menos dieciséis melódicas y ocho percusivas).
Responder a la velocidad de las notas.
Soportar los dieciséis canales simultáneamente (Con el canal diez reservado para la percusión).
Soportar la polifonía (Múltiples notas sonando simultáneamente) en cada canal.

En MIDI el sonido del instrumento (o “programa”) para cada uno de los dieciséis canales MIDI es seleccionado mediante el mensaje de Cambio de Programa, que cuenta con un parámetro llamado Número de Programa. Estos Números de Programa son 128 en el estándar General MIDI, con el canal 10 reservado únicamente para instrumentos de percusión. Estos números suelen ir del 0 al 127 en los sintetizadores, y ser mostrados desde el 1 al 128 en la mayoría de instrumentos, tal como se pueden apreciar en la siguiente tabla:

Piano [edit]
- 1 Acoustic Grand Piano
- 2 Bright Acoustic Piano
- 3 Electric Grand Piano
- 4 Honky-tonk Piano
- 5 Electric Piano 1
- 6 Electric Piano 2
- 7 Harpsichord
- 8 Clavinet

Chromatic Percussion [edit]
- 9 Celesta
- 10 Glockenspiel
- 11 Music Box
- 12 Vibraphone
- 13 Marimba
- 14 Xylophone
- 15 Tubular Bells
- 16 Dulcimer

Organ [edit]
- 17 Drawbar Organ
- 18 Percussive Organ
- 19 Rock Organ
- 20 Church Organ
- 21 Reed Organ
- 22 Accordion
- 23 Harmonica
- 24 Tango Accordion

Guitar [edit]
- 25 Acoustic Guitar (nylon)
- 26 Acoustic Guitar (steel)
- 27 Electric Guitar (jazz)
- 28 Electric Guitar (clean)
- 29 Electric Guitar (muted)
- 30 Overdriven Guitar
- 31 Distortion Guitar
- 32 Guitar Harmonics

Bass [edit]
- 33 Acoustic Bass
- 34 Electric Bass (finger)
- 35 Electric Bass (pick)
- 36 Fretless Bass
- 37 Slap Bass 1
- 38 Slap Bass 2
- 39 Synth Bass 1
- 40 Synth Bass 2

Strings [edit]
- 41 Violin
- 42 Viola
- 43 Cello
- 44 Contrabass
- 45 Tremolo Strings
- 46 Pizzicato Strings
- 47 Orchestral Harp
- 48 Timpani

Ensemble [edit]
- 49 String Ensemble 1
- 50 String Ensemble 2
- 51 Synth Strings 1
- 52 Synth Strings 2
- 53 Choir Aahs
- 54 Voice Oohs
- 55 Synth Choir
- 56 Orchestra Hit

Brass [edit]
- 57 Trumpet
- 58 Trombone
- 59 Tuba
- 60 Muted Trumpet
- 61 French Horn
- 62 Brass Section
- 63 Synth Brass 1
- 64 Synth Brass 2

Reed [edit]
- 65 Soprano Sax
- 66 Alto Sax
- 67 Tenor Sax
- 68 Baritone Sax
- 69 Oboe
- 70 English Horn
- 71 Bassoon
- 72 Clarinet

Pipe [edit]
- 73 Piccolo
- 74 Flute
- 75 Recorder
- 76 Pan Flute
- 77 Blown bottle
- 78 Shakuhachi
- 79 Whistle
- 80 Ocarina

Synth Lead [edit]
- 81 Lead 1 (square)
- 82 Lead 2 (sawtooth)
- 83 Lead 3 (calliope)
- 84 Lead 4 (chiff)
- 85 Lead 5 (charang)
- 86 Lead 6 (voice)
- 87 Lead 7 (fifths)
- 88 Lead 8 (bass + lead)

Synth Pad [edit]
- 89 Pad 1 (new age)
- 90 Pad 2 (warm)
- 91 Pad 3 (polysynth)
- 92 Pad 4 (choir)
- 93 Pad 5 (bowed)
- 94 Pad 6 (metallic)
- 95 Pad 7 (halo)
- 96 Pad 8 (sweep)

Synth Effects [edit]
- 97 FX 1 (rain)
- 98 FX 2 (soundtrack)
- 99 FX 3 (crystal)
- 100 FX 4 (atmosphere)
- 101 FX 5 (brightness)
- 102 FX 6 (goblins)
- 103 FX 7 (echoes)
- 104 FX 8 (sci-fi)

Ethnic [edit]
- 105 Sitar
- 106 Banjo
- 107 Shamisen
- 108 Koto
- 109 Kalimba
- 110 Bagpipe
- 111 Fiddle
- 112 Shanai

Percussive [edit]
- 113 Tinkle Bell
- 114 Agogo
- 115 Steel Drums
- 116 Woodblock
- 117 Taiko Drum
- 118 Melodic Tom
- 119 Synth Drum
- 120 Reverse Cymbal

Sound effects [edit]
- 121 Guitar Fret Noise
- 122 Breath Noise
- 123 Seashore
- 124 Bird Tweet
- 125 Telephone Ring
- 126 Helicopter
- 127 Applause
- 128 Gunshot

Este estándar es el seguido por FuidSynth, lo que permite en el código asignar un instrumento no solo a la interpretación de las señales captadas por los sensores, sino el cambio de instrumentos a gestos definidos en el mismo código, como por ejemplo, cambiar de instrumento al mantener las manos sobre dos sensores determinados, una determinada cantidad de tiempo.

4.6 Código Fuente

El siguiente paso es compilar el código fuente que interpretará las señales recibidas por los sensores. Dicho código está disponible en una versión para Rust y otra para Python. La primera será la que se utilice como se ha podido apreciar en las decisiones de diseño.

Lo que hace el código es asociar un estado a cada “tecla”, es decir, a cada uno de los sensores. Estos estados representan la distancia a la que se captura una señal en dicho sensor, siendo 0 el estado cuando no se captura ninguna señal dentro de una distancia predeterminada, y cambiando este estado. El cómo cambia se puede calcular de dos formas, la lineal que consiste en dividir la distancia a la que se recibe la señal, y la modular, que utiliza el módulo de esta división, y que parece dar un mejor resultado.

En la siguiente captura se puede ver la distancia de captura de señal de los sensores para cada nota, al igual que el código que captura los gestos para cambiar de instrumento y para apagar la Raspberry Pi:

```
// determine the max distance to measure
let mut cm_per_note = 5;
let mut mode_string = "linear".to_string();

let mut gesture_change_instrument = 129_u8; // two outermost sensors
let mut gesture_shutdown = 24_u8; // middle two sensors
```

En la primera línea se puede ver que cada cinco centímetros de distancia con el sensor suponen una nota diferente, aumentando la escala de la misma. El método utilizado para medirlo es el lineal, y en las líneas inferiores, se ve como se captura el gesto para cambiar de instrumentos y apagar la Raspberry (Con los sensores de los laterales, o con los dos sensores centrales).

Para reproducir el sonido se recurre a un bucle que está esperando que se reciba una señal de los sensores y comprueba que esta esté dentro de la distancia máxima, lo que entiende como que ha sido cubierta. En ese momento calcula que nota debería reproducir para ese sensor y a esa distancia. Si había otra nota reproduciéndose en ese momento, la detiene y reproduce la nueva nota.

```
// get sensor reading
distance[i] = octasonic.get_sensor_reading(i as u8);

// is the key covered?
if distance[i] < max_distance {

  // the key is covered, so figure out which note to play
  let scale_start = start_note + octave_offset * i as u8;
```

Todas las funciones octasonic vienen incluídas en la librería del mismo nombre creada por Andy Grove. Creador también de este código.

5 - Conclusiones

El diseño general del proyecto no es muy costoso, a pesar de lo cual los gastos de envío acrecientan en gran medida el precio del mismo, llegando a suponer aproximadamente un cincuenta por ciento del mismo. Esto se debe a que algunos componentes, como la placa y el convertidor de nivel lógico se compraron en tiendas americanas. Un examen más exhaustivo de proveedores podría ayudar a abaratar mucho el coste del proyecto, si en un futuro se quisiera repetir.

El principal reto del proyecto está en su montaje, las fases de soldadura y la construcción de la carcasa.
El código fuente del mismo deja mucho espacio a la experimentación y ampliación del proyecto, del mismo modo que se puede ampliar desde el punto de vista electrónico. Una idea interesante, que es la que lleva a cabo Linda Zhang en su proyecto, es la de combinar el piano con sensores lumínicos para que, dependiendo de la iluminación del ambiente, se reproduzcan sonidos diferentes.

Otra idea para la ampliación del mismo que surge de mi, y en la que trabajaré más adelante, es la de grabar todas las notas que se reproducen mientras se está utilizando el piano en un fichero, de forma que puedan ser reproducidas desde este directamente, lo que permitiría hacer grabaciones y composiciones con el instrumento.

Ha sido un proyecto emocionante e instructivo especialmente en el apartado tecnológico relacionado con el Hardware del mismo.

6 - Glosario

Todos los términos del glosario incluyen un link hacia la fuente o su página web principal.

Arduino: Arduino es una plataforma de creación de electrónica de código abierto, la cual está basada en hardware y software libre, flexible y fácil de utilizar para los creadores y desarrolladores. Esta plataforma permite crear diferentes tipos de microordenadores de una sola placa a los que la comunidad de creadores puede darles diferentes tipos de uso.

AVR: Es una CPU de arquitectura Harvard. Tiene 32 registros de 8 bits. Algunas instrucciones sólo operan en un subconjunto de estos registros. La concatenación de los 32 registros, los registros de entrada/salida y la memoria de datos conforman un espacio de direcciones unificado, al cual se accede a través de operaciones de carga/almacenamiento. A diferencia de los microcontroladores PIC, el stack se ubica en este espacio de memoria unificado, y no está limitado a un tamaño fijo.

Fue diseñado desde un comienzo para la ejecución eficiente de código C compilado. Como este lenguaje utiliza profusamente punteros para el manejo de variables en memoria, los tres últimos pares de registros internos del procesador son usados como punteros de 16 bits al espacio de memoria externa, bajo los nombres X, Y y Z.

Capacitor: Un condensador eléctrico (también conocido frecuentemente con el anglicismo capacitor, proveniente del nombre equivalente en inglés) es un dispositivo pasivo, utilizado en electricidad y electrónica, capaz de almacenar energía sustentando un campo eléctrico. Está formado por un par de superficies conductoras, generalmente en forma de láminas o placas, en situación de influencia total (esto es, que todas las líneas de campo eléctrico que parten de una van a parar a la otra) separadas por un material dieléctrico o por la permitividad eléctrica del vacío. Las placas, sometidas a una diferencia de potencial, adquieren una determinada carga eléctrica, positiva en una de ellas y negativa en la otra, siendo nula la variación de carga total.

Aunque desde el punto de vista físico un condensador no almacena carga ni corriente eléctrica, sino simplemente energía mecánica latente, al ser introducido en un circuito, se comporta en la práctica como un elemento "capaz" de almacenar la energía eléctrica que recibe durante el periodo de carga, la misma energía que cede después durante el periodo de descarga.

I2C: Circuito inter-integrado (I²C, del inglés Inter-Integrated Circuit) es un bus serie de datos desarrollado en 1982 por Philips Semiconductors (hoy NXP Semiconductors, parte de Qualcomm1). Se utiliza principalmente internamente para la comunicación entre diferentes partes de un circuito, por ejemplo, entre un controlador y circuitos periféricos integrados.

Jumper: En electrónica y en particular en informática, un jumper o saltador es un elemento que permite cerrar el circuito eléctrico del que forma parte dos conexiones. Esto puede hacerse mediante soldadura (se derrite suficiente estaño para cerrar el circuito), soldando un cable o alambre entre ambos puntos o, lo más frecuente, conectado dos pines en hilera

o paralelo mediante una pieza de plástico que protege el material conductor que cierra el circuito. Los más habituales tienen tamaños de 2,54 mm, 2 mm y 1,27 mm.

LED: Un diodo emisor de luz o led (también conocido por la sigla LED, del inglés light-emitting diode) es una fuente de luz constituida por un material semiconductor dotado de dos terminales. Se trata de un diodo de unión p-n, que emite luz cuando está activado. Si se aplica una tensión adecuada a los terminales, los electrones se recombinan con los huecos en la región de la unión p-n del dispositivo, liberando energía en forma de fotones. Este efecto se denomina electroluminiscencia, y el color de la luz generada (que depende de la energía de los fotones emitidos) viene determinado por la anchura de la banda prohibida del semiconductor.

Microcontrolador: Un microcontrolador (abreviado µC, UC o MCU) es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica. Un microcontrolador incluye en su interior las tres principales unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada/salida.

MIDI: (abreviatura de Musical Instrument Digital Interface) es un estándar tecnológico que describe un protocolo, una interfaz digital y conectores que permiten que varios instrumentos musicales electrónicos, ordenadores y otros dispositivos relacionados se conecten y comuniquen entre sí. Una simple conexión MIDI puede transmitir hasta dieciséis canales de información que pueden ser conectados a diferentes dispositivos cada uno.

Oscilador de Cristal: Es un oscilador electrónico que utiliza la resonancia mecánica de un cristal vibratorio de material piezoeléctrico para crear una señal eléctrica con una frecuencia precisa. Esta frecuencia se utiliza comúnmente para controlar el tiempo, como en los relojes de cuarzo, para proporcionar una señal de reloj estable para circuitos integrados digitales y para estabilizar las frecuencias de los transmisores y receptores de radio.

Python: Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Raspberry Pi: Raspberry Pi es un ordenador de placa reducida, ordenador de placa única u ordenador de placa simple (SBC) de bajo coste desarrollado en el Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de informática en las escuelas.

Raspbian: Raspbian es una distribución del sistema operativo GNU/Linux y por lo tanto libre basado en Debian para la placa computadora (SBC) Raspberry Pi. El lanzamiento inicial fue en junio de 2012.

Técnicamente el sistema operativo es un port no oficial de Debian armhf para el procesador (CPU) de Raspberry Pi, con soporte optimizado para cálculos en coma flotante por

hardware, lo que permite dar más rendimiento en según qué casos. El port fue necesario al no haber versión Debian armhf para la CPU ARMv6 que contiene el Raspberry PI.

Resistencia: Se le denomina resistencia eléctrica a la oposición al flujo de corriente eléctrica a través de un conductor. La unidad de resistencia en el Sistema Internacional es el ohmio, que se representa con la letra griega omega (Ω), en honor al físico alemán Georg Simon Ohm, quien descubrió el principio que ahora lleva su nombre.

Rust: Rust es un lenguaje de programación compilado, de propósito general y multiparadigma que está siendo desarrollado por Mozilla. Es un lenguaje de programación multiparadigma, soporta programación funcional pura, por procedimientos, imperativa y orientada a objetos.

SoundFont: SoundFont es el nombre de una marca que conjuntamente se refiere a un formato de archivo y una tecnología asociada para cerrar la brecha entre grabación digital y síntesis de audio, especialmente para los propósitos de composición musical por computadora.

7 - Bibliografía

Firmware: https://github.com/andygrove/octasonic-firmware

Sparkfun: https://www.sparkfun.com

OSH Park: https://oshpark.com

Octasonic pthon library: https://github.com/andygrove/octasonic-python

Octasonic rust library: https://github.com/andygrove/octasonic-rs

AVR GCC Toolchain:
http://maxembedded.com/2015/06/setting-up-avr-gcc-toolchain-on-linux-and-mac-os-x/#step2

Activación de SPI en raspberry:
https://learn.sparkfun.com/tutorials/raspberry-pi-spi-and-i2c-tutorial

Proyecto Ultrasonic Light Piano de Linda Zhang:
https://www.ziyinzhang.com/project/ultrasonic-piano.html

Documentación sobre Octasonic: https://theotherandygrove.com/octasonic/

Código fuente: https://github.com/andygrove/UltrasonicPiPiano

8 - Anexos

Flashear el microcontrolador AVR

Lo primero que hay que hacer es instalar el compilador AVR GCC Toolchain:

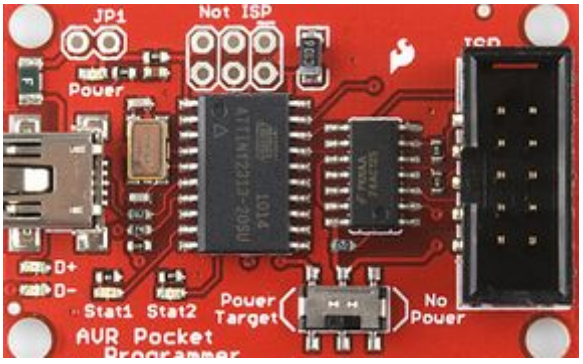
sudo apt-get install gcc-avr binutils-avr avr-libc
sudo apt-get install gdb-avr

A continuación hay que instalar AVR dude, el programa que servirá para quemar el firmware en el microcontrolador:

sudo apt-get install avrdude

Una vez hecho esto, el código del firmware se puede encontrar en el github de Andy Grove.
https://github.com/andygrove/octasonic-firmware

Una vez clonado, y antes de continuar, hay que conectar el microcontrolador a la máquina mediante la placa AVR Programmer:



Dicha placa tiene un cable con seis conexiones que van enchufadas a los seis pines de la sección AVR de la placa octasonic. Una vez conectado sólo queda por situarse dentro del directorio firmware y ejecutar los siguientes comandos:

make clean
make flash
make fuses

Habilitar el código Rust

Lo único necesario para poder trabajar con código Rust es ejecutar el siguiente comando:

curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh

Activación Automática

Para que nada más iniciarse el sistema operativo de la Raspberry se ejecute el programa para utilizar el piano es necesario añadir las siguientes líneas al fichero **/etc/rc.local**:

```
. /home/pi/.cargo/env
cd /home/pi/UltrasonicPiPiano
./run.sh > /var/log/ultrasonic-pi.log 2>&1
```