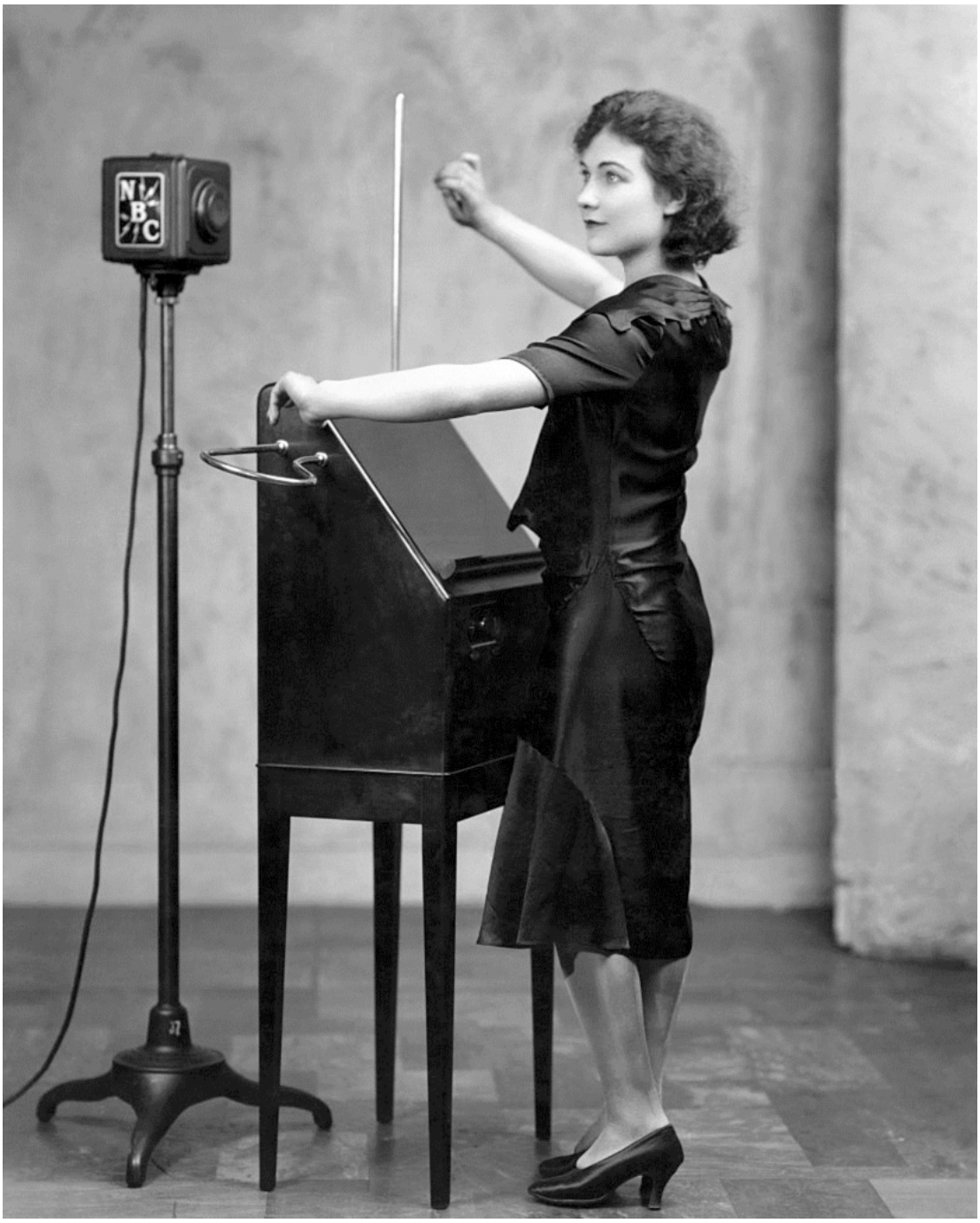
# Trabajo de la asignatura Sistemas Empotrados 2

**Pedro Ramoneda**




**GITHUB. https://github.com/Petruselgrande/ultrasonic-theremin-rpi-zero**

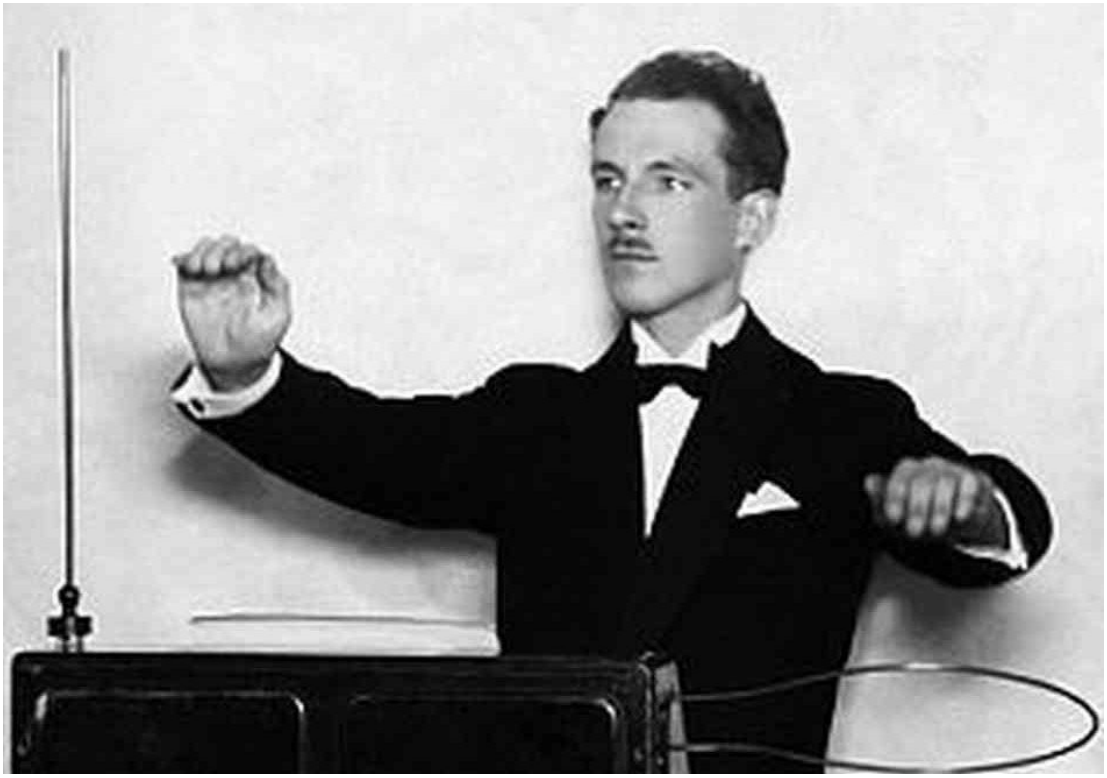
# Audio, Linux y tiempo real.

La idea del trabajo era hacer una primera aproximación a un **sistema de audio en tiempo real** y con sistema operativo. Por ello, lo mas sencillo para hacer un sistema completo ha sido hacer un controlador MIDI. Pero lo importante no era el controlador en si mismo, sino el aprendizaje sobre este campo que se ha hecho con miras a futuros proyectos.

En concreto, se ha hecho un **theremin MIDI**. Lo bueno de usar este protocolo es que desacopla la activación del sonido a la sintetizacion del sonido. Dicho de otra manera, que por una parte se pueden hacer una serie de controladores desde los cuales calcular las variables del sonido, mandar el sonido a otro dispositivo por medio de MIDI y este último dispositivo que sintetice el sonido como sea debido.

Desde el punto de vista del **intérprete** se han de conseguir buenos **tiempos de reacción** de los instrumentos electrónicos, para tener un buen manejo y percepción del sonido. En concreto los tiempos de latencia han de estar entre 15ms y 30 ms para que se pueda considerar "tiempo real". Por eso, este era el principal requisito. Todo esto se ha extraido de varias directrices que da en sus articulos Ge Wang sobre la creación de nuevos instrumentos electronicos musicales (https://www.gewang.com/publish/).

## Theremin



El theremin, llamado en un principio eterófono, thereminófono, termenvox o thereminvox,es considerado uno de los primeros instrumentos musicales electrónicos. Se controla sin que haya contacto físico entre el intérprete y el instrumento.

El instrumento está formado por dos antenas metálicas que detectan la posición relativa de las manos del thereminista y los osciladores para controlar la frecuencia (altura) con una mano y la amplitud (volumen o intensidad) con la otra. Las señales eléctricas se amplifican por medio de un amplificador y son emitidas por altavoces.

Se ha elegido este instrumento por su relativa sensillez. Así como la librería *portMidi y wiringPi . Aunque a posteriori, habría sido interesante utilizar el framework JUCE https://www.youtube.com/watch?v=jq7zUKEcyzI .

## MIDI

> 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. (Wikipedia)

La ventaja fundamental que nos ha ofrecido este prtocolo es su eficiencia debida a su pequeño tamaño de mensaje. Además de la fácil manipulación, modificación y selección de los instrumentos y las notas que permite. Todo esto así comola librería libre y multiplataforma utilizada (PortMidi de PortMedia (http://portmedia.sourceforge.net)) se ha aprendido del libro *The audio programming book* y sobre todo del cd con capitulos adicionales sobre MIDI que lleva asociado. Se adjuntan en la carpeta tutoriales los capítulos del cd.

## Sobre la sincronización dispositivo ordenador

Uno de los objetivos del proyecto era hacer un instrumento que fuese independiente de un ordenador "host" concreto. Dicho de otra manera, que pudiera ejecutarse conectado a cualquier ordenador sin tener que instalar ningún tipo de software concreto. Por ello, usando el protocolo MIDI, lo mas sencillo para conectarlo a un ordenador era usar la especificación **MIDI-USB** implementada por defecto en cualquier ordenador que tenga usb (https://www.midi.org/articles-old/basic-of-usb). La principal ventaja de usar esto es que no hay que implementar un driver específico para cada ordenador al que se vaya a conectar el theremin. La desventaja es que el theremin ha de ser visto "desde fuera" como un dispositivo.

Para sincronizar las notas en el dispositivo MIDI y en el ordenador se ha utilizado un bit del protocolo MIDI llamado change program. Esto ha sido posble gracias a que el theremin es monódico, es decir que como mucho suena una sola nota a la vez. Sin embargo, en el caso de un instrumento polifónico habría que haber utilizado para sincronizar otro protocolo superpuesto, por ejemplo MIDI clock, que es un protocolo bastante antiguo, dificil de usar, poco intuitivo y mal documentado. Por ello, se está utilizando mucho en los últimos años la alternativa también libre de ableton **link** (https://ableton.github.io/link/). Esta librería requiere de un sistema operativo completo para ser ejecutada, esta es una de las razones por la que los controladores MIDI mas actuales están hechos con empotrados con sistema operativo, en concreto, el privativo elk OS (https://www.mindmusiclabs.com/).

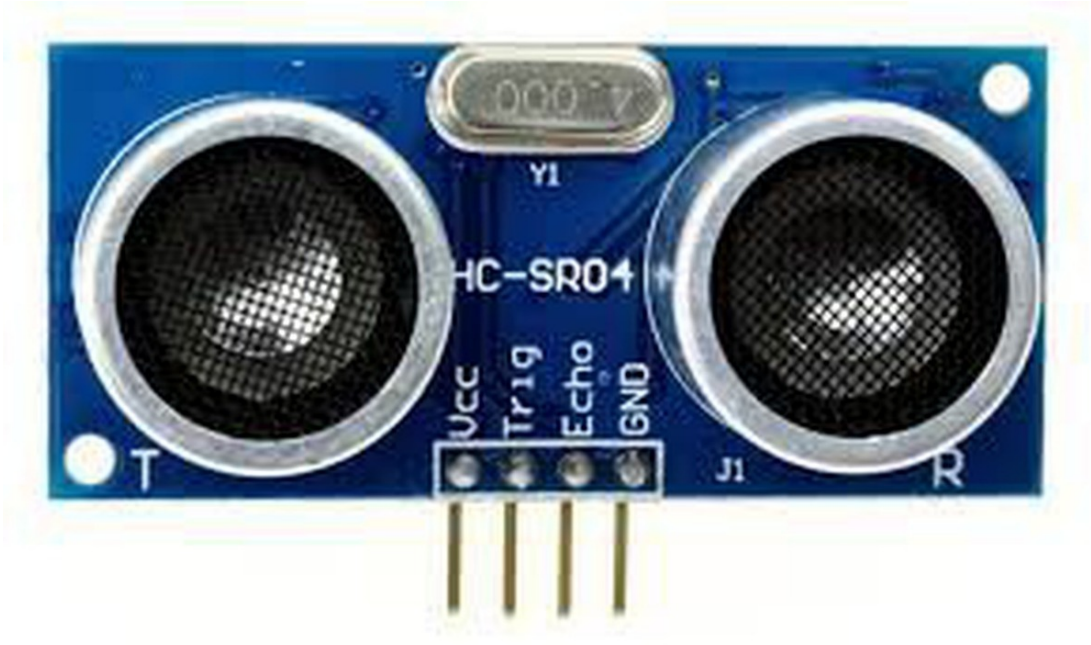
## Plataforma de desarrollo elegida

Como se ha dicho anteriormente, el theremin ha de ser visto "desde fuera" como un dispositivo. Por ello, se han planteado tres alternativas.

- Shield externo con USB tipo B (el cuadrado) y un controlador que lo maneje. Aunque la mayoría de controladores MIDI que hay en el mercado llevan uno de este tipo, la realidad es que apenas hay tiendas que lo distribuyan como "shield de desarrollo".
- Utilizar una rpi zero, ya que su usb se puede configurar como OTG y por ello se puede configurar como dispositivo (https://en.wikipedia.org/wiki/USB_On-The-Go).
- Comunicar cualquier plataformo de desarrollo con un arduino UNO que tiene un USB tipo B.

Debido a que había bastante información sobre como convertir el USB de una rpi zero a un gadget (sobre todo del tipo serial aunque para g_midi se ha de la misma forma) se ha optado por esta plataforma. Además, en un principio se encontró interesante utilizar el mínimo hardware posible, aunque visto con perspectiva habría dado mas juego la tercera opción. (https://www.raspberrypi.org/forums/viewtopic.php?t=67727) (https://www.raspberrypi.org/forums/viewtopic.php?t=204782) (https://raspberrypi.stackexch the-pi-zero-act-as-an-usb-peripheral-device/40626#40626)

Para medir la distancia de los sensores a las manos se han utilizado dos **sensores de ultrasonidos hc-sr04** . Sin embargo, habría sido interesante invertir en unos sensores mejores o incluso de infrarojos para tener una mejor precisión. Simplemente, en un pin con una señal se activa el sensor mandando un ultrasonido y se espera hasta que otro pin de una señal de que ha sido recibido el rebote del ultrasonido, se mide el tiempo que ha pasado, y como tarda 29,4 us en recorrer un cm, por cada 58,8 us (ida y vuelta) hay una distancia de un cm.




## Discreto vs Continuo

En un origen, el theremin era un instrumento *con sonidos continuos*, ya que como ya se ha dicho se basaba en amplificar la señal de una una corriente eléctrica que aumentaba o disminuía según la distancia de las manos a las antenas. Sin embargo, el sistema predominante en la cultura occidental es **el temperado** el cual es discreto. Por ello, se ha evaluado para tener un buen tiempo de respuesta si se podía hacer con sistemas alternativos **al temperamento igual** o si en caso contrario, es decir, discretizando la distancia de las manos al sensor de ultrasonidos permitía un funcionamiento adecuado.

## Ints vs Floats

Para poder utilizar otro sistema de temperamento, continuo (lo que permite los floats y los sensores) o con menor intervalo de las notas del temperamento igual se puede realizar con MIDI con su séptimo byte o PITCH relativo a la nota que se está tocando.

El problema es que las operaciones con float en la raspberry son muy lentas. En el repositorio la mayoría de los ficheros salvo `pruebaSensor3.c` y `main.c` se han realizado con floats. Medir las dos distancias, flitrarlas para asociarlas a la nota que qeueremos y tocarla tarda aproximadamente 25ms. Además, los sensores utilizados no tienen muy buena precición 30, 40 cm arriba o abajo y no son continuos no toman todos los valores, según varios foros como mucho 3 medidas dentro de cada cm pero esto no es del todo cierto.

Cada muestra se tomaba cada medio segundo subiendo una carpeta poco a poco y dspués bajándola y la realidad es que los sensores eran muy poco precisos. Además, se generaban valores anómalos que se filtraban ya en la propia función getCM().

(la tabla está al final)

## El espacio de la nota "dinámico"

Un problema bastante importante de usabilidad es que cuando estas justo en la distancia entre varias notas a veces suena una y a veces otra (sobre todo debido a la **mala precisión** del sensor). Por lo cual, con la frecuencia final de muestreo, que tiene un periodo de 20 ms, se hace muy incomódo. Al solamente haber la posibilidad de cambiar de semitono en semitono (medio tono) se produce una especie de trémolo sin quererlo en algunas ocasiones. Al final, simplemente se ha ignorado ya que bajaba mucho el rendimiento y si se retiraban las manos la referencia se iba al techo (esto se podría haber solucionado) o se desplazaban los registros de las notas mucho (esto no tenía solución).

La idea es que la distancia de la nota a la siguiente nota fuera **dinámica** , no estática. Por ejemplo, se toma una distancia que equivale a un DO si se quiere producir un sonido RE# hay que subir exactamnte entre un cm y dos cm y con ese primer valor se toma otra medida respecto a la cual se sube o se baja la nota.

Está implementado en el fichero `mainFloatsYDinamicos.c`.

## Patch real time y overclocking

Para mejorar el rendimiento del procesador se ha querido utilizar el kernel RT como en las prácticas de la asignatura. Sin embargo, el USB OTG no se puede utilzar ya que deja de funcionar todo y hay aue configurar la rpi otra vez. Además, en la rpi3 B y en la zero sin modificar el usb el rendimiento al usar USBs decae un 15%, como indica este artículo https://lemariva.com/blog/2018/03/raspberry-pi-zero-w-preempt-rt-kernel-performance.

Con el overclocking tampoco se ha experimentado mucho, siguiendo este tutorial (https://github.com/RetroPie/RetroPie-Setup/wiki/Overclocking) dejó de funcionar y debido a todos los problemas se decidió que era mejor centrarse en la aritmética con enteros. Pero se explorará en la siguiente asignatura.

## Arquitectura Software de la aplicación.

Es una aplicación muy sencilla: se leen los datos de las distancias, se flitran para que correspondan a una nota determinada y mandan por el USB de nota en nota.

### Arquitectura Multicore (Alternativas a los dispositivos utilizados).

Sería interesante en el caso de que se hubiese utilizado por ejemplo una rpi 3 B conectada a un arduino. Esto permitiría separar lo que ahora es secuencial, por un lado se leería un sensor, por otro lado otro, por otro se realizarían los cálculos y por otro se gestioría la entrada salida con el USB. Además, se podría hacer overclocking y poner el Kernel preemt-RT sin roblema. Esto reduciría bastante la frecuencia para poder tener unos tiempos de respuesta mejores.

## Conclusiones

Se ha hecho un prototipo mejorable pero funcional que era el objetivo que me había planteado en un principio. Pero sobre todo, he aprendido una gran cantidad de cosas, y aunque solo he aplicado una mínima parte en este proyecto, tengo una primera aproximación para el desarrollo de aplicaciones de audio en tiempo real. Además, he conectado un poco mas mis estudios artísticos y técnicos, junto con proyecto software esta asignatura ha sido la que mas me lo ha permitido. Para la siguiente asignatura me gustaría seguir por esta linea aunque con protocolos de Audio mas complejos.

## Implementación de la aplicación

### Aligerando Linux

En mi caso solamente he desactivado los servicios que tenía instalados. Ya que utilizaba Raspbian Lite y muchas cosas no las tenía.

Fuente: https://wiki.linuxaudio.org/wiki/raspberrypi

```
## Stop the ntp service
sudo service ntp stop

## Stop the triggerhappy service
sudo service triggerhappy stop

## Stop the dbus service. Warning: this can cause unpredictable behaviour wher
sudo service dbus stop

## Stop the console-kit-daemon service. Warning: this can cause unpredictable
sudo killall console-kit-daemon

## Stop the polkitd service. Warning: this can cause unpredictable behaviour w
```

```
sudo killall polkitd

## Only needed when Jack2 is compiled with D-Bus support (Jack2 in the AutoSta
#export DBUS_SESSION_BUS_ADDRESS=unix:path=/run/dbus/system_bus_socket

## Remount /dev/shm to prevent memory allocation errors
sudo mount -o remount,size=128M /dev/shm

## Kill the usespace gnome virtual filesystem daemon. Warning: this can cause
killall gvfsd

## Kill the userspace D-Bus daemon. Warning: this can cause unpredictable beha
killall dbus-daemon

## Kill the userspace dbus-launch daemon. Warning: this can cause unpredictabl
killall dbus-launch

## Uncomment if you'd like to disable the network adapter completely
#echo -n "1-1.1:1.0" | sudo tee /sys/bus/usb/drivers/smsc95xx/unbind
## In case the above line doesn't work try the following
#echo -n "1-1.1" | sudo tee /sys/bus/usb/drivers/usb/unbind

## Set the CPU scaling governor to performance
echo -n performance | sudo tee /sys/devices/system/cpu/cpu0/cpufreq/scaling_g
```

## Raspbian setup

Para raspberry normal

https://randomnerdtutorials.com/installing-raspbian-lite-enabling-and-connecting-with-ssh/

Sin teclado ni pantalla. Setup del wifi.

https://howchoo.com/g/ndy1zte2yjn/how-to-set-up-wifi-on-your-raspberry-pi-without-ethernet

Para conectarse por ssh:

Escaneo mi red local con Angry Ip Scanner para ver en que dirección está la raspberry.

`contraseña: raspberry`

Se configura como un dispositivo midi el USB OTG.

https://raspberrypi.stackexchange.com/questions/38576/can-the-pi-zero-act-as-an-usb-peripheral-device/40626#40626

Para comprobar que se detecta como midi desde fuera.

```
ioreg -p IOUSB
```

## Jugando con midi

### portMidi en Mac

Instalación:

```
brew install portMidi
```

Tras instalar portMidi en Mac....

```
gcc -o prg pmidiout.c -I/usr/local/include -L/usr/local/lib -lportmidi
```

### portMidi en Linux

Instalación:

```
sudo apt install libportmidi-dev
```

Ejecución:

Los fuentes están en el repositorio de github.

```
gcc -o prg prueba1midi.c -I/usr/local/include -L/usr/local/lib -lportmidi
```

```
0: Midi Through Port-0
2: f_midi
choose device: 2
```

## Para probar sensor ultrasonico

Instalar Bcm2385.h

En un principio se exploró la idea de utilizar esta librería, pero se desechó debido
al gran numero de problemas que causaba y su alto tiempo de respuesta.

```
sudo apt-get install html-xml-utils
mkdir -p bcm2835 && (wget -qO - `curl -sL http://www.airspayce.com/mikem/bcm28
cd bcm2835
./configure
make
sudo make install
```

```
gcc -o prg main.c HC_SR04.c -l bcm2835
```

**La libreria de arriba va como el culo.... PlanB**

### WiringPi

```
gpio -v
sudo apt-get install git-core
sudo apt-get update
sudo apt-get upgrade
cd
git clone git://git.drogon.net/wiringPi
cd ~/wiringPi
git pull origin
cd ~/wiringPi
./build
gpio -v
gpio readall
```

NOTE: To compile programs with wiringPi, you need to add: -lwiringPi

```
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>

#define TRUE 1

#define TRIG 11
#define ECHO 15

void setup() {
        wiringPiSetup();
        pinMode(TRIG, OUTPUT);
        pinMode(ECHO, INPUT);

        //TRIG pin must start LOW
        digitalWrite(TRIG, LOW);
        delay(30);
}

int getCM() {
        //Send trig pulse
        digitalWrite(TRIG, HIGH);
        delayMicroseconds(20);
        digitalWrite(TRIG, LOW);

        //Wait for echo start
        while(digitalRead(ECHO) == LOW);

        //Wait for echo end
```

```
        long startTime = micros();
        while(digitalRead(ECHO) == HIGH);
        long travelTime = micros() - startTime;

        //Get distance in cm
        int distance = travelTime / 58;

        return distance;
}

int main(void) {
        setup();

        printf("Distance: %dcm\n", getCM());

        return 0;
}
```

## Con dos sensores de ultrasonido HC-SR04. Mapa de conexiones.

### HC SR04 - 1

Pin 11 - (GPIO 17) - Wiring PI 0 TRIG

Pin 15 - (GPIO 22) - Wiring PI 3 ECHO

ground -> pin 6 vcc -> pin 2

### HC SR04 - 1

Pin 32 - (GPIO 12) - Wiring PI 26 TRIG

Pin 36 - (GPIO 16) - Wiring PI 27 ECHO

ground -> pin 39 vcc -> pin 4

### HC SR04 - 1

## Posibles mejoras

- Meterle vibratto ya sea por un controlador midi o por el paper http://www-classes.usc.edu/engr/ise/599muscog/2004/projects/yang/
- Que vaya por un glisando hasta la siguiente nota (hecha desde ableton)

- Hacer un theremin con CV
- Hacer un theremin con un iPhone y un iWatch

### Compilación del programa

```
gcc -o prg main.c -I/usr/local/include -L/usr/local/lib -lportmidi -lwiringPi
```

### Ejecución del programa

```
./main
```

### Utilidades

```
scp ./main.c pi@10.231.204.177:main.c
```

Como hacer un sintetizador que parezca un theremin para Ableton Live.

https://music.tutsplus.com/tutorials/quick-tip-how-to-emulate-a-theremin-sound-in-ableton-live–audio-6198

Como maximo se va a medir 64 cm que son 64 * 58.4 = 3738us.

| tiempo | distancia |
|---|---|
| 1151 us | 11.655172cm |
| 735 us | 4.482759cm |
| 728 us | 4.344828cm |
| 702 us | 3.982759cm |
| 698 us | 3.862069cm |
| 673 us | 3.413793cm |
| 671 us | 3.396552cm |
| 679 us | 3.500000cm |
| 646 us | 2.982759cm |
| 677 us | 3.517241cm |
| 687 us | 3.672414cm |
| 673 us | 3.413793cm |
| 823 us | 5.913793cm |
| 764 us | 5.000000cm |
| 839 us | 6.310345cm |
| 885 us | 7.120690cm |
| 928 us | 7.810345cm |

| tiempo | distancia |
|---|---|
| 961 us | 8.396552cm |
| 1049 us | 9.913794cm |
| 1104 us | 10.844828cm |
| 1127 us | 11.258620cm |
| 1167 us | 11.982759cm |
| 1187 us | 12.327586cm |
| 1203 us | 12.551724cm |
| 1256 us | 13.465517cm |
| 1315 us | 14.500000cm |
| 1404 us | 16.051723cm |
| 1537 us | 18.327587cm |
| 1654 us | 20.293104cm |
| 1724 us | 21.534483cm |
| 1829 us | 23.327587cm |
| 1946 us | 25.362068cm |
| 2002 us | 26.379311cm |
| 2074 us | 27.568966cm |
| 2109 us | 28.172413cm |
| 2182 us | 29.465517cm |
| 2236 us | 30.396551cm |
| 2319 us | 31.534483cm |
| 2386 us | 32.965519cm |
| 2491 us | 34.793102cm |
| 2589 us | 36.500000cm |
| 2618 us | 36.931034cm |
| 2664 us | 37.741379cm |
| 2732 us | 38.896553cm |
| 2873 us | 41.344826cm |
| 2885 us | 41.551723cm |
| 2942 us | 42.534481cm |
| 3081 us | 44.948277cm |
| 3105 us | 45.362068cm |
| 3184 us | 46.724136cm |
| 3327 us | 49.189655cm |
| 3286 us | 48.500000cm |
| 3456 us | 51.396553cm |
| 3352 us | 49.637932cm |
| 3344 us | 49.413792cm |
| 3379 us | 50.086208cm |
| 3594 us | 53.775864cm |
| 3599 us | 53.913792cm |
| 3703 us | 55.637932cm |
| 3594 us | 53.775864cm |
| 3901 us | 59.068966cm |

| tiempo | distancia |
|---|---|
| 3992 us | 60.620689cm |
| 3134 us | 45.862068cm |
| 2581 us | 36.327587cm |
| 1999 us | 26.275862cm |
| 1445 us | 16.706896cm |
| 1205 us | 12.603448cm |