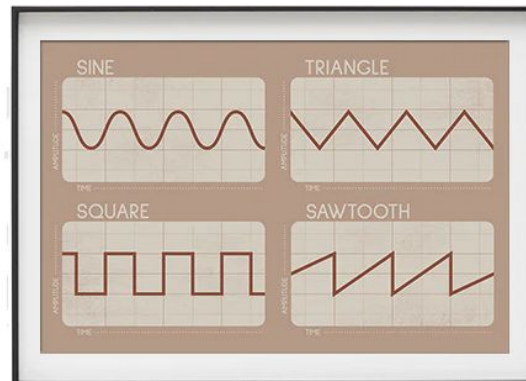


# Creación de sintetizadores utilizando Arduino y la biblioteca Mozzi.

Esp. Ing. Ernesto Gigliotti



# Ernesto Gigliotti



Ingeniería electrónica. UTN FRA



Especialización en sistemas embebidos.  
FIUBA



<http://www.tortoiseinstruments.com.ar>



<https://www.facebook.com/tortoiseinstruments>



[@tortoiseinstruments](https://www.instagram.com/tortoiseinstruments)

## Introducción:

- Arduino
- Biblioteca Mozzi
- DAC
- PWM
- Agregar biblioteca al IDE
- Circuito para los ejemplos

## Ejemplos:

- Ejemplo 1: Osciladores
- Ejemplo 2: Arpegiador
- Ejemplo 3: ADSR
- Ejemplo 4: LPF
- Ejemplo 5: Map y Clip
- Ejemplo 6: WaveFolder
- Ejemplo 7: Reverb y Mix

# Arduino

- Microcontrolador
  - Memoria de programa
  - Memoria de datos
  - GPIOs
  - Módulos digitales integrados: PWM, UART
  - 8bit
- PCB
  - Conectores a GPIOs
  - Fuente de alimentación
  - Conversor USB-serial
  - Cristal para clock del microcontrolador



# Biblioteca Mozzi



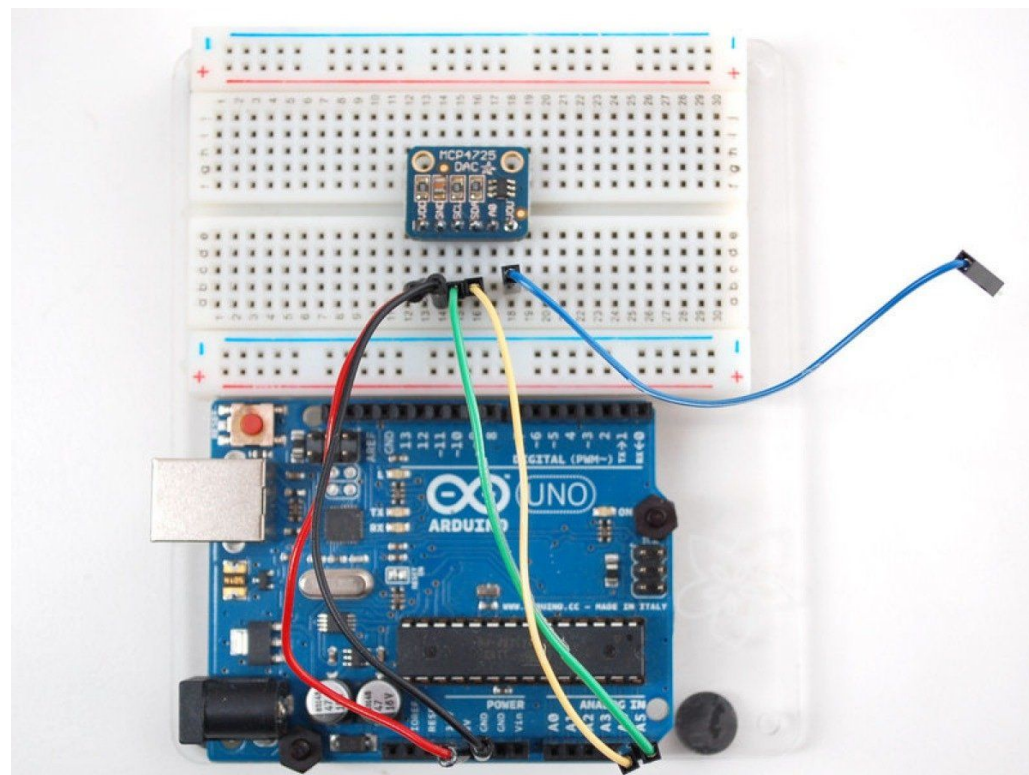
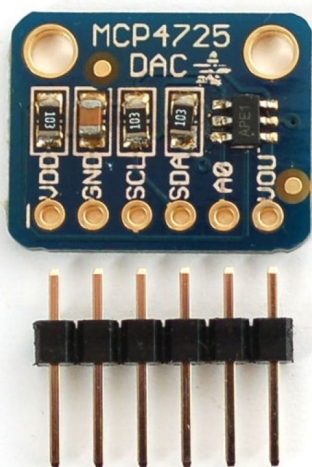
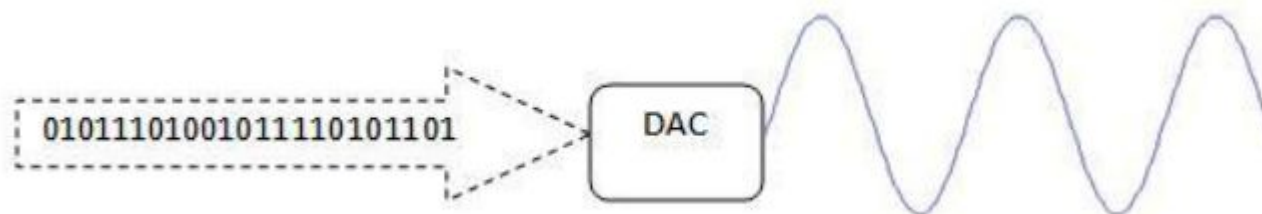
- Biblioteca de terceros
- C++
- Múltiples plataformas
- Representa unidades conocidas de síntesis
  - OSC
  - ADSR
  - Filtros
- DAC con PWM

<https://sensorium.github.io/Mozzi>

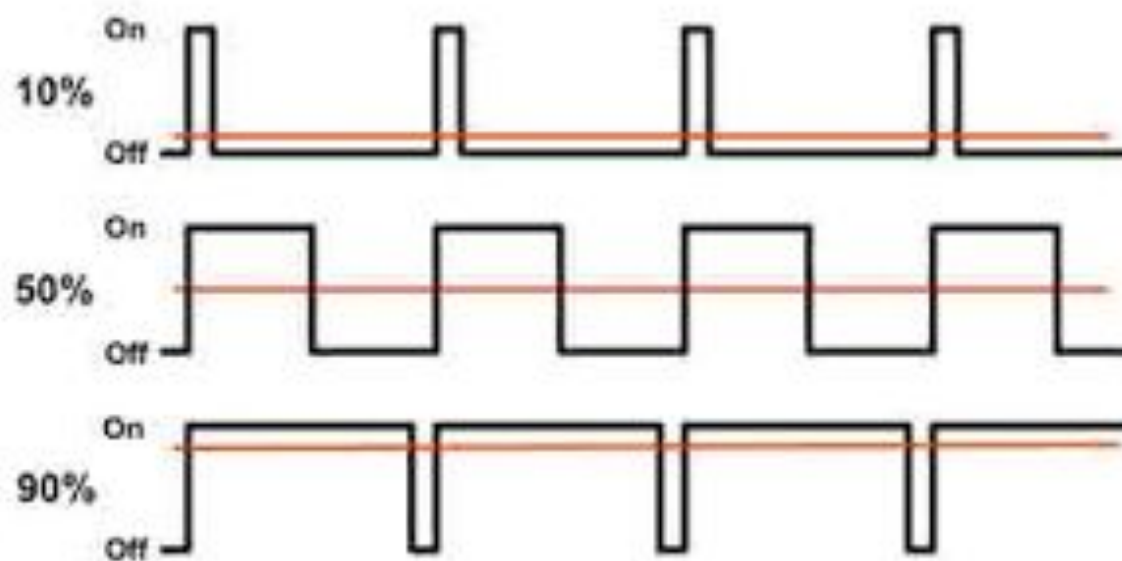
<https://sensorium.github.io/Mozzi/doc/html/index.html>

<https://github.com/sensorium/Mozzi>

# DAC

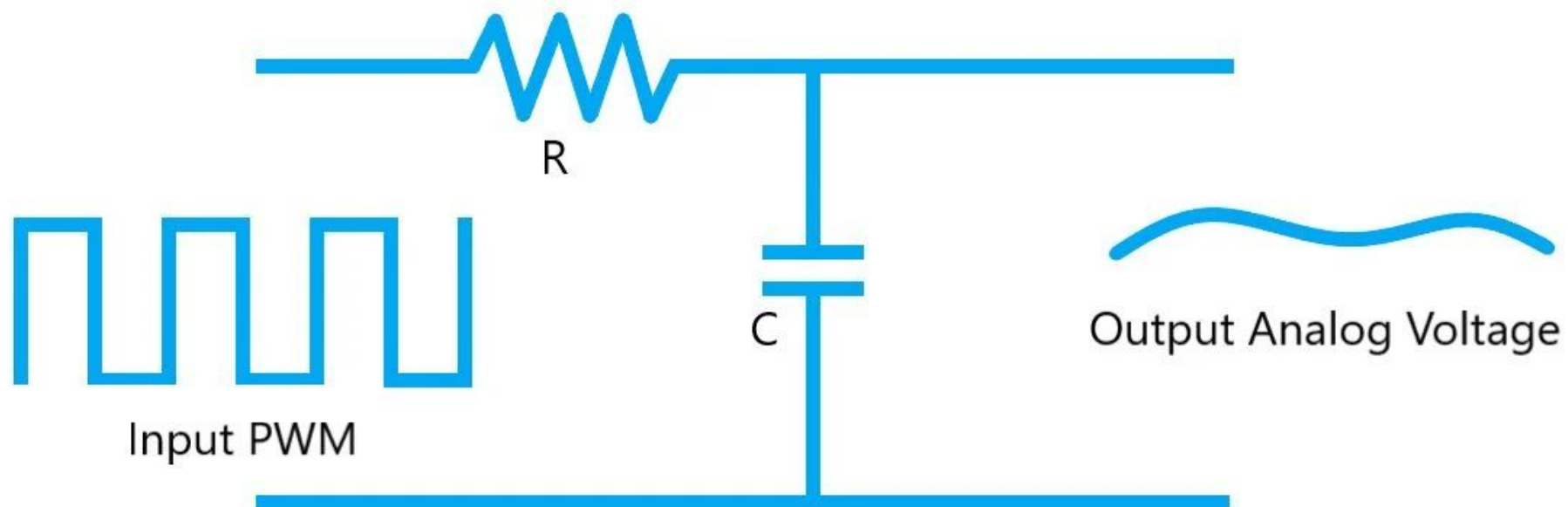


# PWM

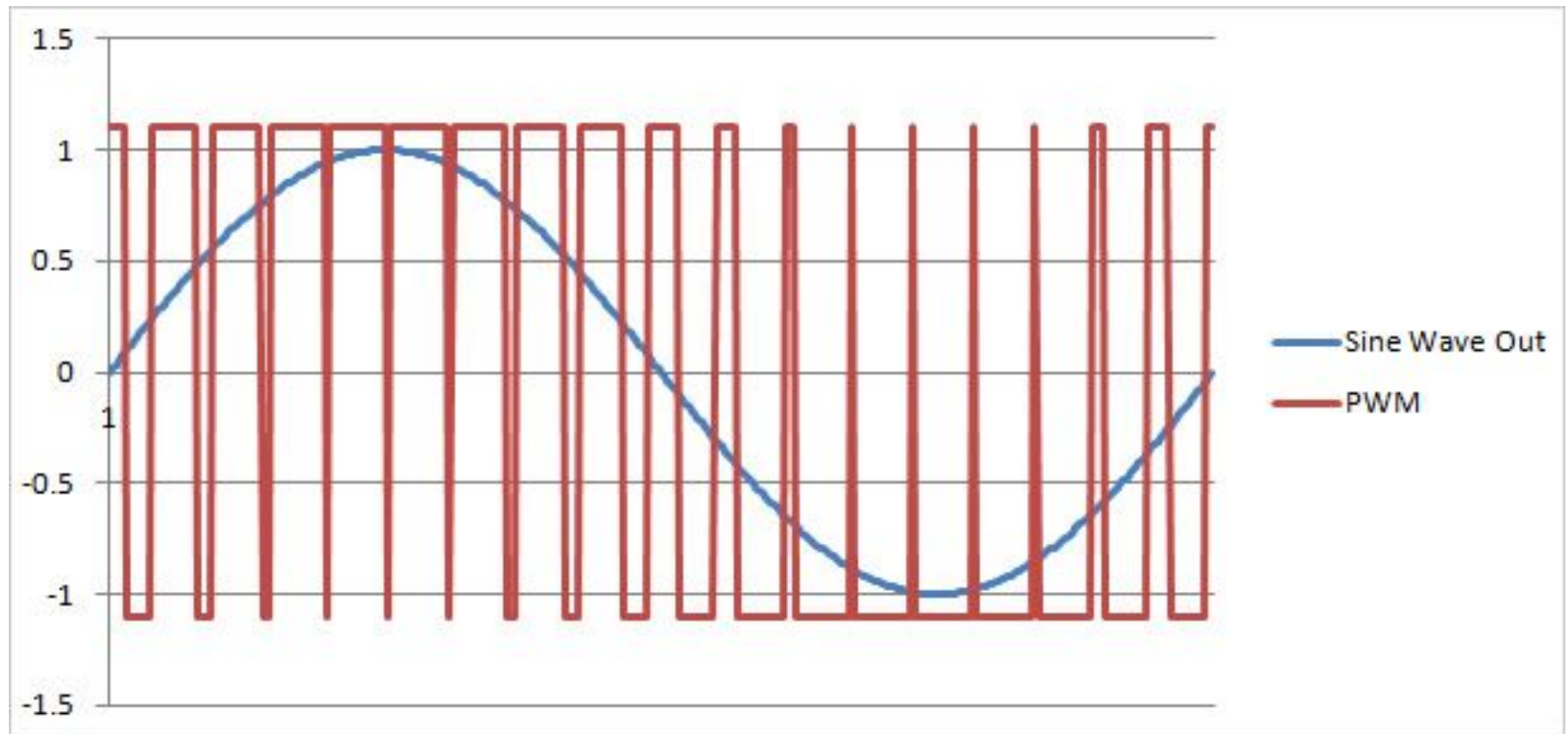




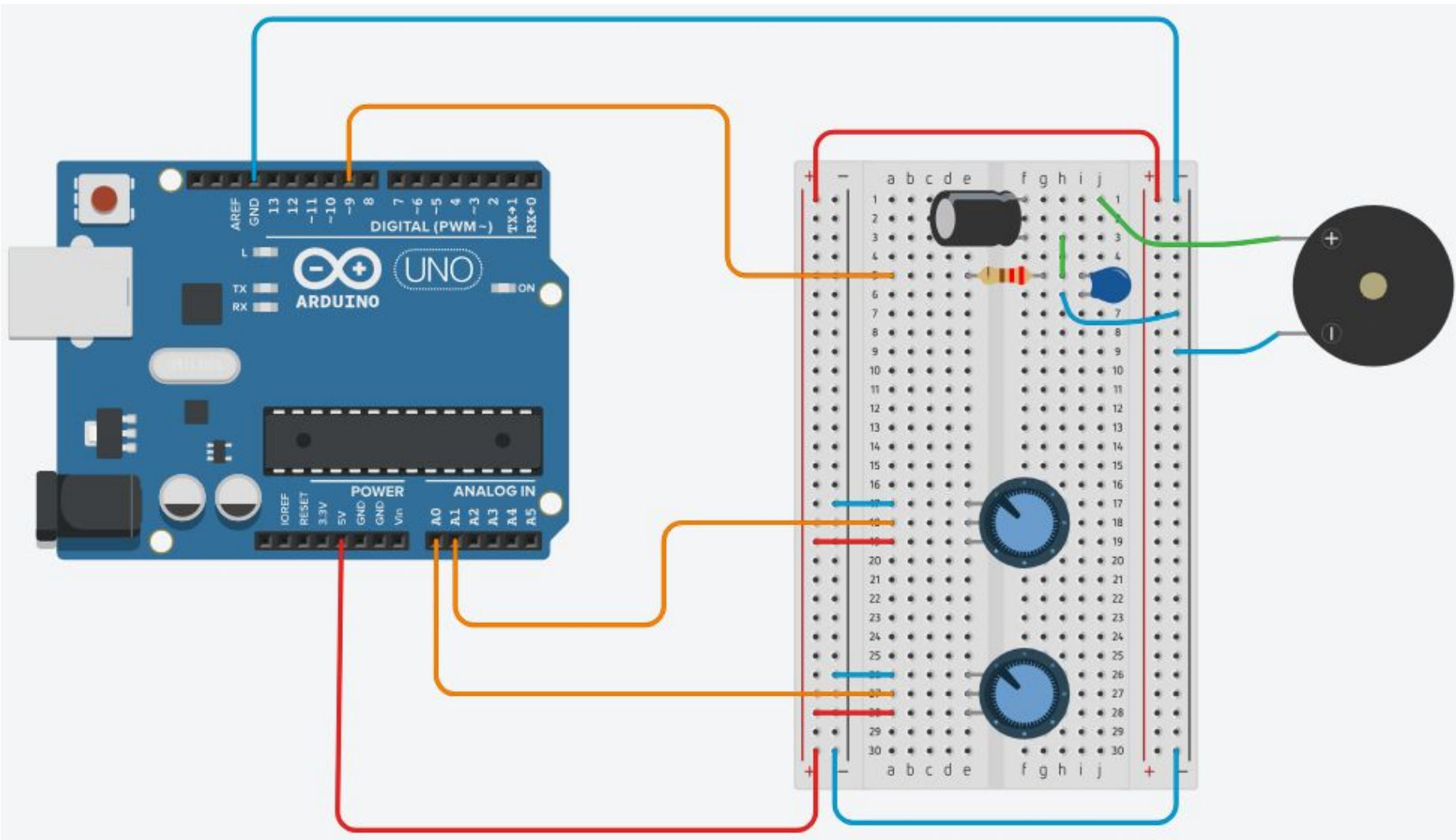
# PWM



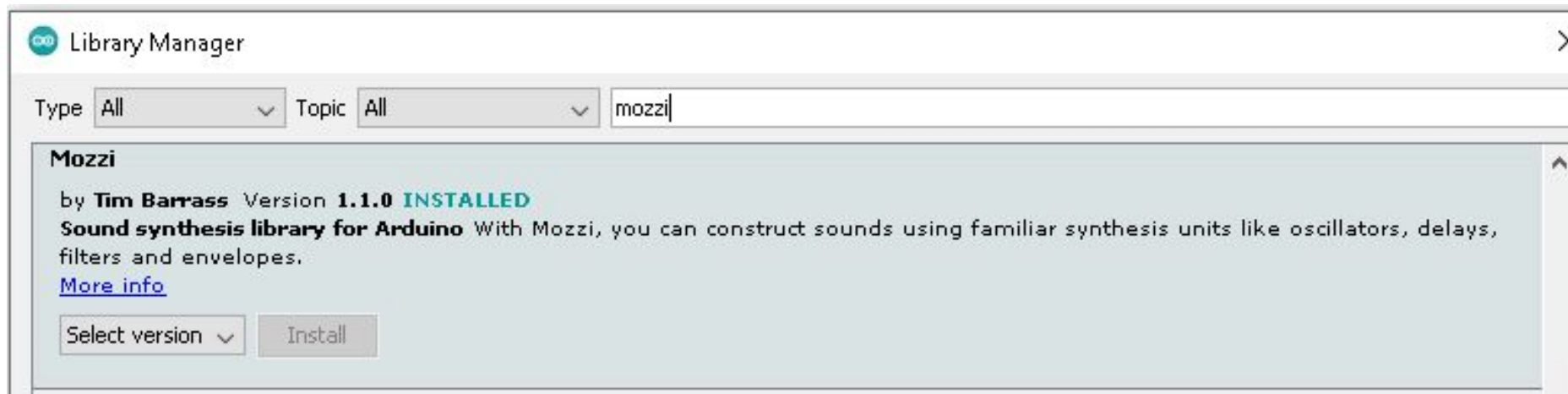
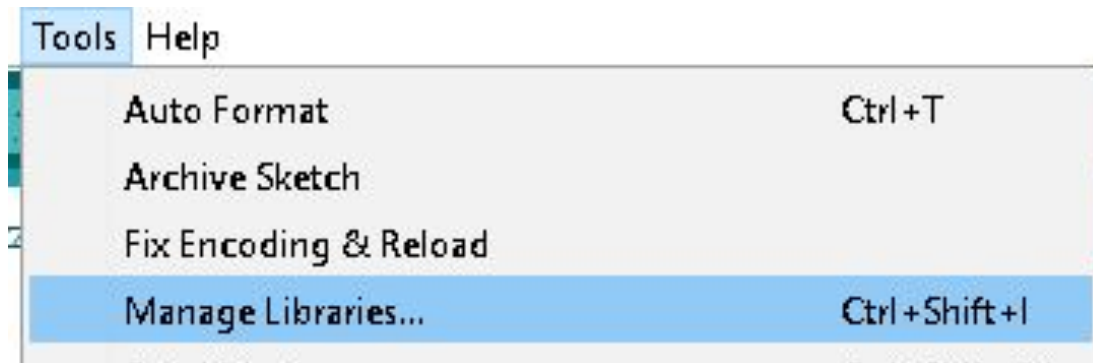
# PWM



# Circuito Arduino + filtro



# Instalación Mozzi en IDE



# Arquitectura del código

sketch\_sep24a | Arduino 1.8.15

File Edit Sketch Tools Help



```
void setup() {  
    // put your setup code here, to run once:  
  
}  
  
void loop() {  
    // put your main code here, to run repeatedly:  
  
}
```

# Arquitectura del código con Mozzi



```
sketch_sep24a$  
#include <MozziGuts.h>  
  
void setup() {  
  
}  
  
void updateControl() {  
  
}  
  
int updateAudio() {  
  
}  
  
void loop() {  
    audioHook(); // required here  
}
```

# Arquitectura del código con Mozzi

- **updateAudio():**
  - Se llama de forma periódica.
  - Alta velocidad: 16384Hz (61uS)
  - Devolvemos la “muestra” a representar
  - No debemos hacer muchos cálculos
- **updateControl():**
  - Se llama de forma periódica.
  - Baja velocidad: 64Hz (15.6mS)
  - Hacemos cálculos
  - Lectura de parámetros (entradas analógicas, MIDI, botones, etc.)



# Osciladores (Ejemplo 1)

Ejemplo\_1

```
#include <MozziGuts.h>
#include <Oscil.h>
#include <tables/sin2048_int8.h>

Oscil <SIN2048_NUM_CELLS, AUDIO_RATE> aSin(SIN2048_DATA);

void setup() {
  startMozzi();
  aSin.setFreq(440); // 440Hz
}

void updateControl() {
}

int updateAudio() {
  return aSin.next();
}

void loop() {
  audioHook(); // required here
}
```



# Formas de onda

<https://github.com/sensorium/Mozzi/tree/master/tables>

- `cos2048_int8.h`
- `saw2048_int8.h`
- `sin2048_int8.h`
- `pinknoise8192_int8.h`
- `square_no_alias_2048_int8.h`
- `triangle2048_int8.h`



Mozzi / tables /	
noise_static_1_16384_int8.h	Move pgmspace macros / functions to a separate header file.
phasor256_int8.h	Move pgmspace macros / functions to a separate header file.
pinknoise8192_int8.h	Move pgmspace macros / functions to a separate header file.
saw1024_int8.h	Move pgmspace macros / functions to a separate header file.
saw2048_int8.h	Move pgmspace macros / functions to a separate header file.
saw256_int8.h	Move pgmspace macros / functions to a separate header file.
saw4096_int8.h	Move pgmspace macros / functions to a separate header file.
saw512_int8.h	Move pgmspace macros / functions to a separate header file.

## Modificando notas (Ejemplo 2)

```
Oscil <SAW2048_NUM_CELLS, AUDIO_RATE> oscSaw(SAW2048_DATA);
```

```
// C4 E4 G4 E4
```

```
float ARP[4]={261.6256,329.6276,391.9954,329.6276};
```

```
int arpIndex=0;
```

```
float centerFreq;
```

```
int tick=0;
```

[https://en.wikipedia.org/wiki/Piano\\_key\\_frequencies](https://en.wikipedia.org/wiki/Piano_key_frequencies)

# Modificando notas

```
void updateControl()
{
    tick++; // 1 tick:15,625ms
    if(tick==15) // 256bpm=234ms
    {
        centerFreq = ARP[arpIndex];

        oscSaw.setFreq(centerFreq);

        arpIndex++;
        if(arpIndex>=4)
            arpIndex=0;

        tick=0;
    }
}
```

# Arquitectura del procesador

- **Evitar:**

- Cálculos float.
- Divisiones.

- **Usar:**

- Números enteros.
- Multiplicaciones.
- Divisiones por  $2^n$

`out = out*0.33;` ❌

`out = out/3;` ❌

`out = (out*85) /256;` ✅

`out = (out*85)>>8;` ✅

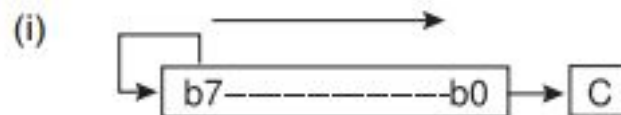
# Arquitectura del procesador

## 10. ASR – Arithmetic Shift Right

### 10.1. Description

Shifts all bits in Rd one place to the right. Bit 7 is held constant. Bit 0 is loaded into the C Flag of the SREG. This operation effectively divides a signed value by two without changing its sign. The Carry Flag can be used to round the result.

Operation:



Syntax:

(i) ASR Rd

Operands:

$0 \leq d \leq 31$

Program Counter:

$PC \leftarrow PC + 1$

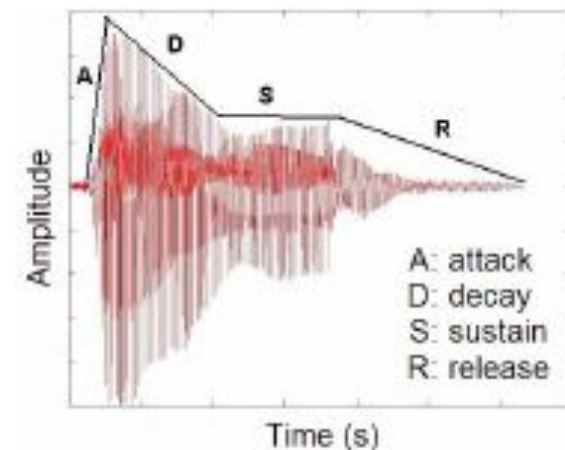
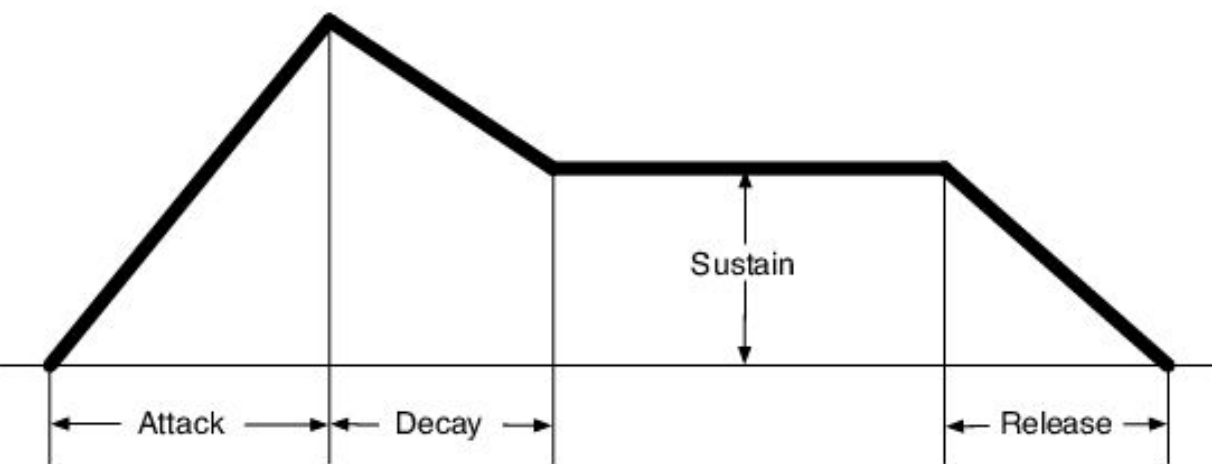
```
out = out>>1;
```

# Agregando Envelope (Ejemplo 3)

```
#include <ADSR.h>
```

```
ADSR <CONTROL_RATE, AUDIO_RATE> envelope;
```

```
void setup() {  
  //...  
  envelope.setADLevels(255, 64);  
  envelope.setTimes(50, 100, 100, 250); // en ms  
}
```



## Agregando Envelope (Ejemplo 3)

```
void updateControl() {  
    //...  
    envelope.update();  
}  
  
int updateAudio() {  
    return (envelope.next() * oscSaw.next()) >> 8;  
}
```

## Agregando Envelope (Ejemplo 3)

`out = (E * 0) / 256`

`(envelope.next() *`

`E=0`

`oscSaw.next()) >> 8;`

`out = (0 * -128) / 256 = 0`

`out = (0 * 0) / 256 = 0`

`out = (0 * 127) / 256 = 0`

`E=64`

`out = (64 * -128) / 256 = -32`

`out = (64 * 0) / 256 = 0`

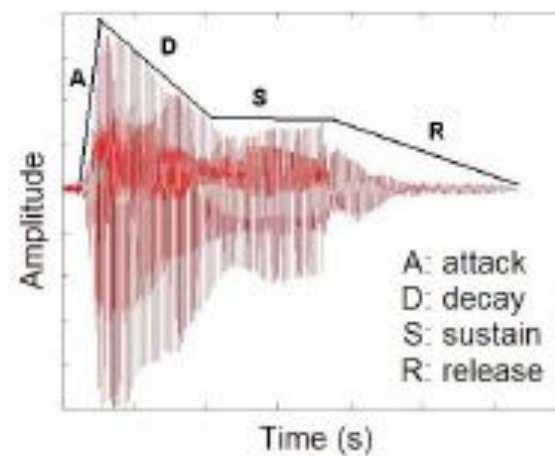
`out = (64 * 127) / 256 = 31`

`E=255`

`out = (255 * -128) / 256 = -127`

`out = (255 * 0) / 256 = 0`

`out = (255 * 127) / 256 = 126`





## Agregando Envelope (Ejemplo 3)

- **envelope.noteOn():**
  - Llamar cuando se ejecuta la nota.
  - Inicializa el envelope.
- **envelope.noteOff():**
  - Llamar cuando deja de sonar la nota.
  - Finaliza el envelope

Debemos detectar el evento de “nueva nota” para llamar a `noteOn()`.

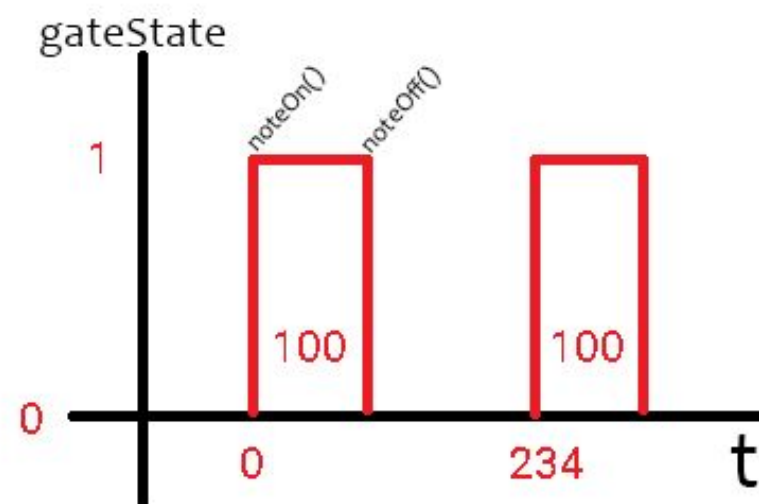
Luego de un tiempo debemos llamar a `noteOff()`.

## Agregando Envelope (Ejemplo 3)

```
int timeoutGate=0;
int gateState=0;

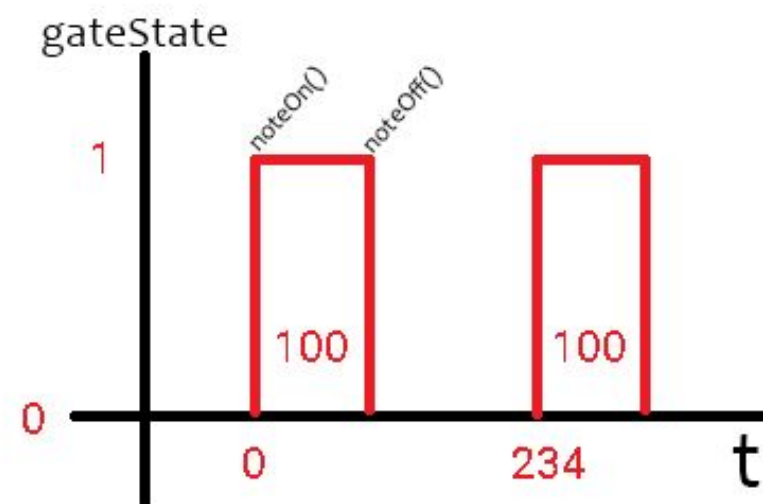
void updateControl() {
  tick++; // 1 tick:15,625ms
  if(tick==15) // 256bpm=234ms
  {
    centerFreq = ARP[arpIndex];
    oscSaw.setFreq(centerFreq);
    envelope.noteOn();
    timeoutGate=0;
    gateState=1;
    arpIndex++;
    if(arpIndex>=4)
      arpIndex=0;

    tick=0;
  }
}
```



## Agregando Envelope (Ejemplo 3)

```
void updateControl() {  
  // ...  
  
  if (gateState==1)  
  {  
    timeoutGate++;  
    if (timeoutGate==7) //100ms  
    {  
      gateState=0;  
      envelope.noteOff();  
    }  
  }  
}
```



# Agregando Filtro LPF (Ejemplo 4)

```
#include <StateVariable.h>
```

```
StateVariable <LOWPASS> svf;
```

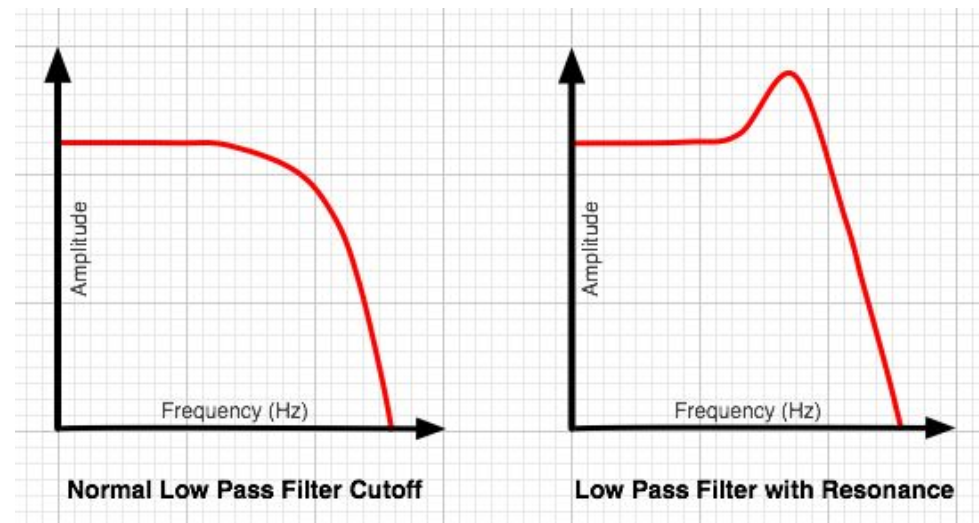
```
void updateControl() {  
  // ...
```

```
  // 20Hz a 2.066Hz
```

```
  svf.setCentreFreq(mozziAnalogRead(A1)*2 + 20);
```

```
  svf.setResonance(mozziAnalogRead(A0)/4); // 0 a 255
```

```
}
```



## Agregando Filtro LPF (Ejemplo 4)

```
int updateAudio() {  
  
    int out = (int) (envelope.next() * oscSaw.next()) >> 8;  
  
    return svf.next(out) >> 2;  
  
}
```

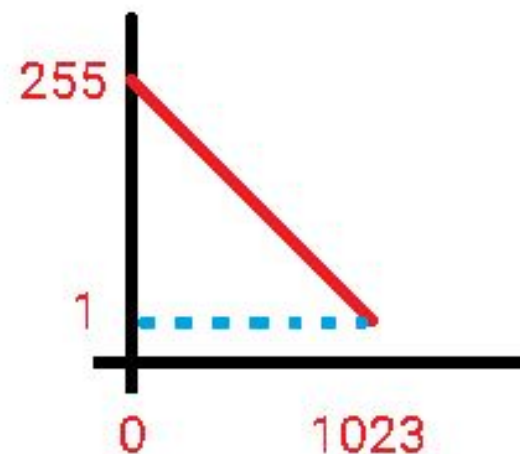
## Ajustando rangos (Ejemplo 5)

```
#include <IntMap.h>
```

```
// 0-> 255 (res min) ; 1023 -> 1 (res max)
```

```
IntMap mapResonance(0,1023,255,1);
```

```
svf.setResonance(mapResonance(mozziAnalogRead(A0)) );
```



El rango de la resonancia es de:  
1 (máximo) a 255 (mínimo)

Usamos el objeto IntMap para escalar e invertir el rango de la entrada analógica.

## Ajustando rangos (Ejemplo 5)

```
int updateAudio() { // -244 a 243
    int out = (int) (envelope.next() * oscSaw.next()) >> 8;
    out = svf.next(out);
    //clip
    if(out > 243)
        out = 243;
    else if(out < -244)
        out = -244;
    //_____
    return out;
}
```

Hacemos un clip para que el audio no se pase de los límites que soporta el DAC.

## Ajustando rangos (Ejemplo 5)

```
#include <AudioOutput.h>
```

```
int updateAudio() { // -244 a 243
    int out = (int) (envelope.next() * oscSaw.next()) >> 8;
    out = svf.next(out);

    out = CLIP_AUDIO(out);

    return out;
}
```

Hacemos un clip para que el audio no se pase de los límites que soporta el DAC.



## Agregamos Wavefolder (Ejemplo 6)

```
#include <WaveFolder.h>
```

```
WaveFolder<> wf;
```

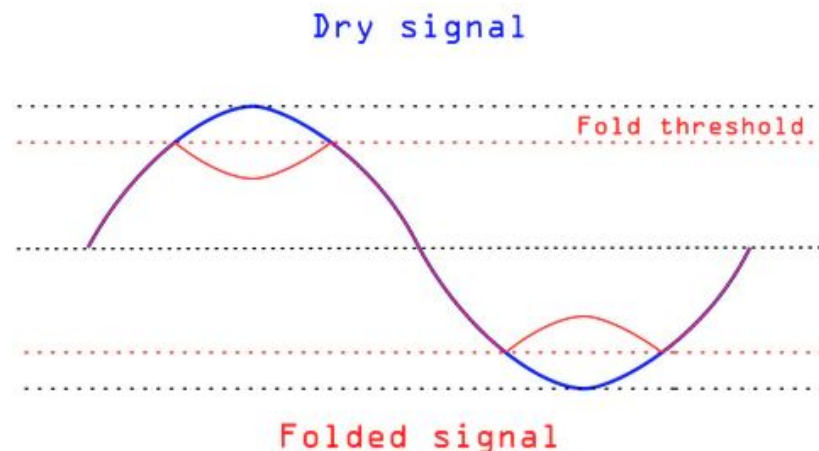
```
void updateControl() {
```

```
    //...
```

```
    int lim = mozzianalogRead(A0)/4 + 2;
```

```
    wf.setLimits(-1*lim, lim);
```

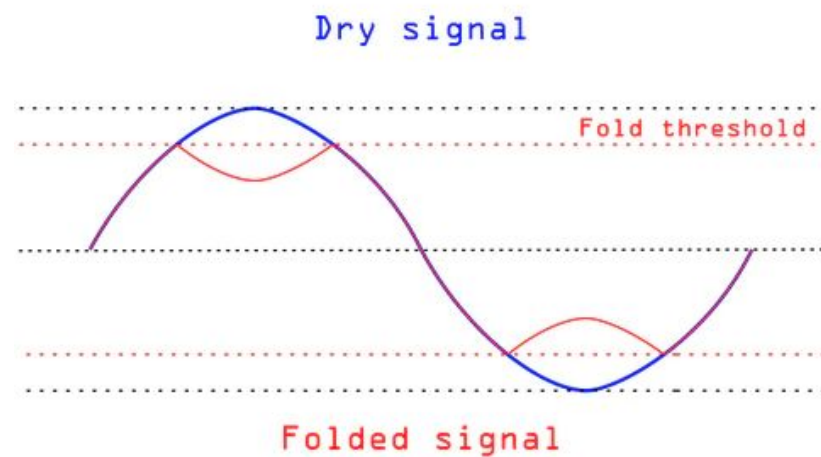
```
}
```



El objeto WaveFolder tiene el método setLimits() donde se setean los niveles.

## Agregamos Wavefolder (Ejemplo 6)

```
int updateAudio() {  
    int out = (int) (envelope.next() * oscTri.next()) >> 8;  
    out = wf.next(out);  
    //clip  
    if(out > 243)  
        out = 243;  
    else if(out < -244)  
        out = -244;  
    //_____  
    return out;  
}
```



# Reverb Tank (Ejemplo 7)

```
#include <ReverbTank.h>
```

```
ReverbTank reverb;
```

```
int dryWet;
```

```
void updateControl() {
```

```
    //...
```

```
    dryWet = mozziaAnalogRead(A0)/4; // 0 a 255
```

```
}
```

```
int updateAudio() {
```

```
    int out = (int) (envelope.next() * oscTri.next()) >> 8;
```

```
    int rev = reverb.next(out);
```

```
    out = ( (out*dryWet) + (rev*(255-dryWet))) >> 8; // MIX
```

## Mix de dos señales (Ejemplo 7)

```
out = ( (out*dryWet) + (rev*(255-dryWet)))>>8; // MIX
```

**Si dryWet=0**

$\text{out} = 0\% \text{out} + 100\% \text{rev}$

**Si dryWet=255**

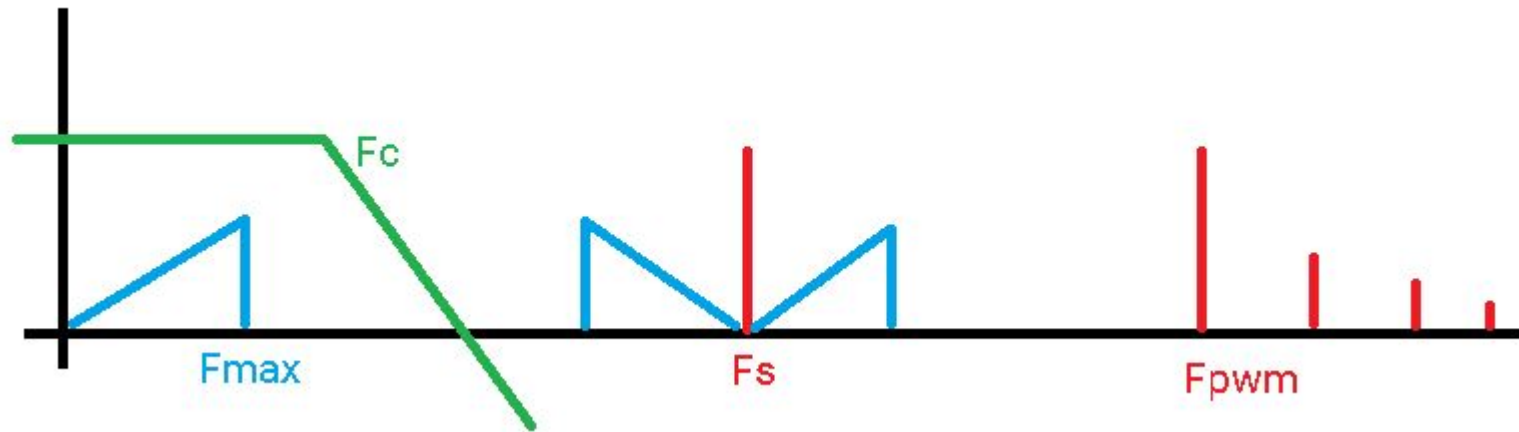
$\text{out} = 100\% \text{out} + 0\% \text{rev}$

**Si dryWet=128**

$\text{out} = 50\% \text{out} + 50\% \text{rev}$

# Ventajas y desventajas

- Velocidad sampleo
- Bits de salida
- Tiempo de procesamiento
- Filtro



# Preguntas

# Gracias!



# Bibliografía

<https://sensorium.github.io/Mozzi/learn/>

[https://github.com/sensorium/Mozzi/blob/master/mozzi\\_fixmath.h](https://github.com/sensorium/Mozzi/blob/master/mozzi_fixmath.h)

<https://www.culturasonora.es/auriculares/que-es-un-dac/>

<https://blog.adafruit.com/2012/09/06/mcp4725-12-bit-dac-tutorial-add-an-analog-output-to-your-microcontroller/>

<https://ww1.microchip.com/downloads/en/devicedoc/atmel-0856-avr-instruction-set-manual.pdf>