

# Unidad UNO de Efectos

## Informe Proyecto Final

### Sistemas Embebidos

Manuel Garat

## Resumen

El proyecto consiste en desarrollar una unidad de efectos genérica para un instrumento musical, preferentemente una guitarra o un piano/teclado. Esta unidad consiste de un Arduino UNO que recibe el input del instrumento para luego modificarlo y enviarlo a un amplificador. Los efectos se logran en software. Una interfaz gráfica corriendo en una PC, conectada mediante el puerto serie al Arduino, indica qué efecto realizar y cómo modificarlo. La entrada y la salida de la unidad de efectos pasan por un circuito de amplificación y filtro pasa bajos, para eliminar armónicos y poder acondicionarla al ADC.

## Objetivo y motivación

El objetivo es hacer una unidad de efectos genérica (que pueda lograr más de un efecto) implementada en un dispositivo embebido o similar, controlada mediante una interfaz gráfica corriendo en una PC. Los efectos se programan en software y son:

- **Bit crusher:** para distorsionar el sonido “digitalmente”.
- **Octavador Daft Punk:** para lograr un efecto parecido al que utiliza Daft Punk en canciones como *Television Rules the Nation*.
- **Distorsión:** para distorsionar el sonido “analógicamente”.
- **Fuzz:** que es como una distorsión pero más violenta.
- **Fuzz Controlado:** una versión más controlada del fuzz.
- **Sin efecto/Clean:** para obtener la señal de entrada sin manipulación alguna (sólo su conversión de analógica a digital y viceversa, como *passthrough*).

La interfaz, además de permitir elegir qué efecto se utiliza en el momento, debe posibilitar la modificación de la magnitud del efecto, por ejemplo, el nivel de distorsión, el tiempo de delay, etc.

Las razones de por qué este proyecto y no otro son:

- Investigar cómo se comporta el sonido y cómo manipularlo en un dispositivo embebido.
- Seguir explorando la comunicación entre un programa en una PC y un dispositivo embebido.

## Hardware y software

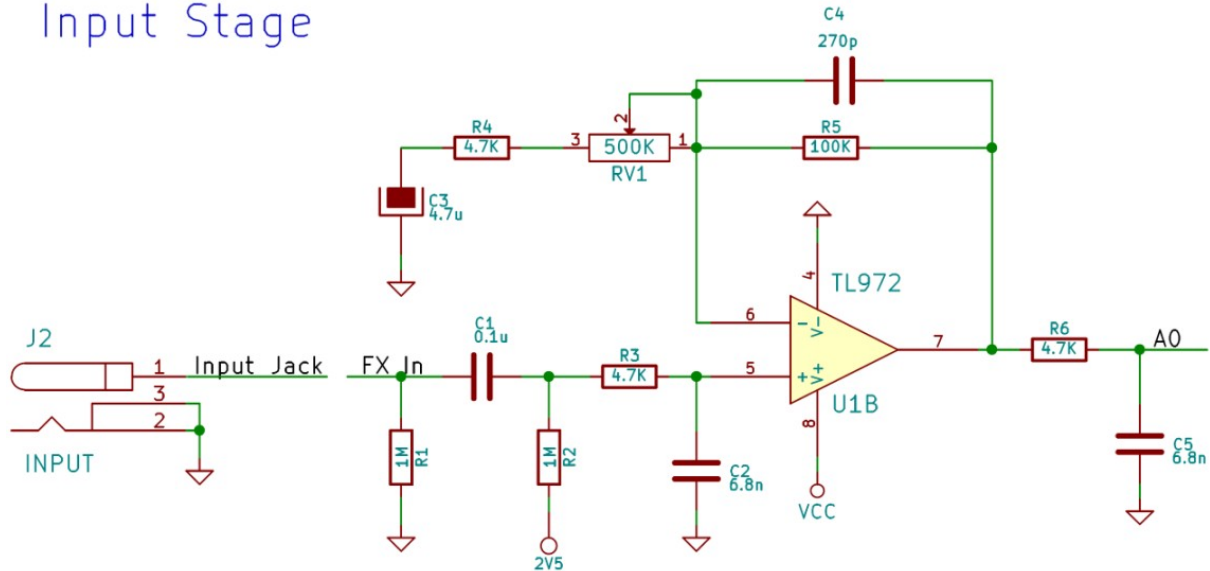
La unidad de efectos se desarrolla en un Arduino UNO porque no es una tarea muy demandante a esta escala y porque es un componente relativamente común y fácil de conseguir, si alguien quiere replicar el proyecto.

Para acondicionar la señal del instrumento al ADC del Arduino, esta debe pasar por un circuito de amplificación y filtro pasa bajos para eliminar armónicos de alta frecuencia. De manera similar, a la salida del Arduino, dos PWM en vez de uno para obtener mayor precisión (16 bits), se le debe volver a

aplicar un filtro pasa bajos antes de pasar al amplificador. Estos circuitos se desarrollan en protoboards, para facilitar modificaciones que puedan surgir.

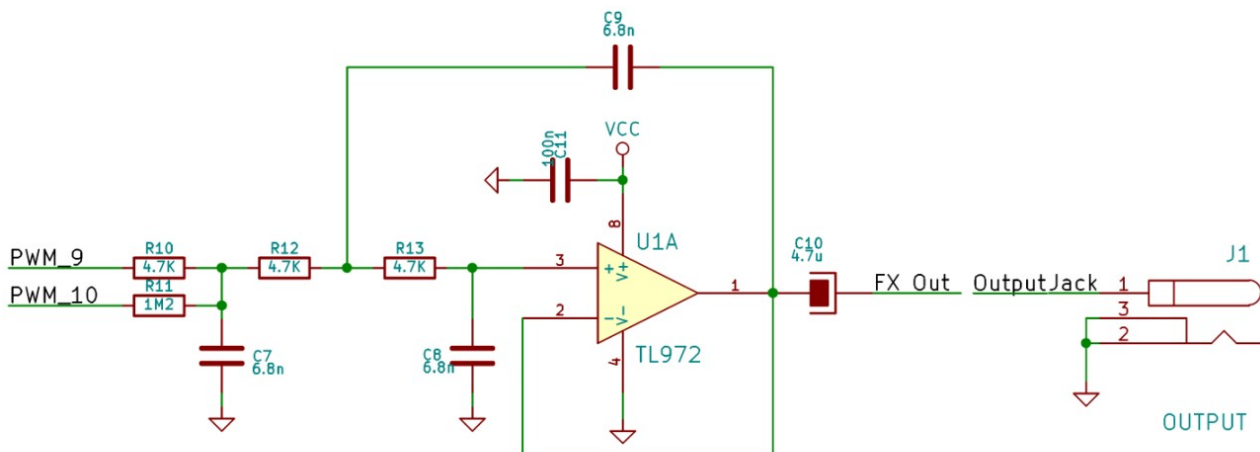
Todas las señales, del instrumento al circuito de entrada y de salida al amplificador, se proveen utilizando los clásicos cables jack, con adaptadores jack/PCB, para conectarlos al arduino.

## Input Stage



Circuito de entrada al Arduino.

## Output Stage



Circuito de salida del Arduino.

Los efectos se logran por software. Cada efecto tiene su modificador, que cambia la intensidad del efecto, en cierta medida. Cada uno consiste en:

- **Bit crusher:** hacer un *shifting* de los bits de la entrada. El modificador determina cuántos bits se *shiftean*.
- **Octavador Daft Punk:** saltar tantos ciclos como diga el modificador. En estos ciclos no se produce salida.
- **Distorsión:** llevar la salida al valor del modificador si la entrada lo supera.
- **Fuzz:** llevar la salida a la mitad del máximo si la entrada supera al modificador, o al mínimo, en caso contrario.

- **Fuzz Controlado:** si la entrada supera el valor cero, le asigna el modificador, sino le asigna el modificador negativo. Básicamente genera una onda cuadrada.
- **Sin efecto/Clean:** no realizar ninguna modificación al valor obtenido del ADC.

La interfaz gráfica corre en una PC, Linux o Windows. Esta se implementa con *python*, utilizando la librería *appjar*, para los elementos gráficos, y *pySerial* para la comunicación serial.

## Descripción de la solución

El proyecto está basado en el *pedalShield UNO* de *ElectroSmash*. Sin embargo, se debe adaptar para que funcione con una interfaz gráfica y no con botones y switches físicos. Además, el propuesto por *ElectroSmash* implementa un efecto por módulo de software, imposibilitando el cambio entre ellos sin tener que cargar de nuevo el Arduino.

### SerialEvent()

```
// si 0 -> obtener efecto La próxima
efecto = Serial.read()
// sino
modificador = Serial.read()
```

### isr\_TIMER()

```
input = ADC
switch (efect)
    case 0:
        // bit crusher
        input = input << modificador
    case 1:
        // daft punk octaver
    ...
// escribir en PWM el input modificador
PWM = input
```

Pseudocódigo de la estructura básica del módulo que se ejecuta en el Arduino.

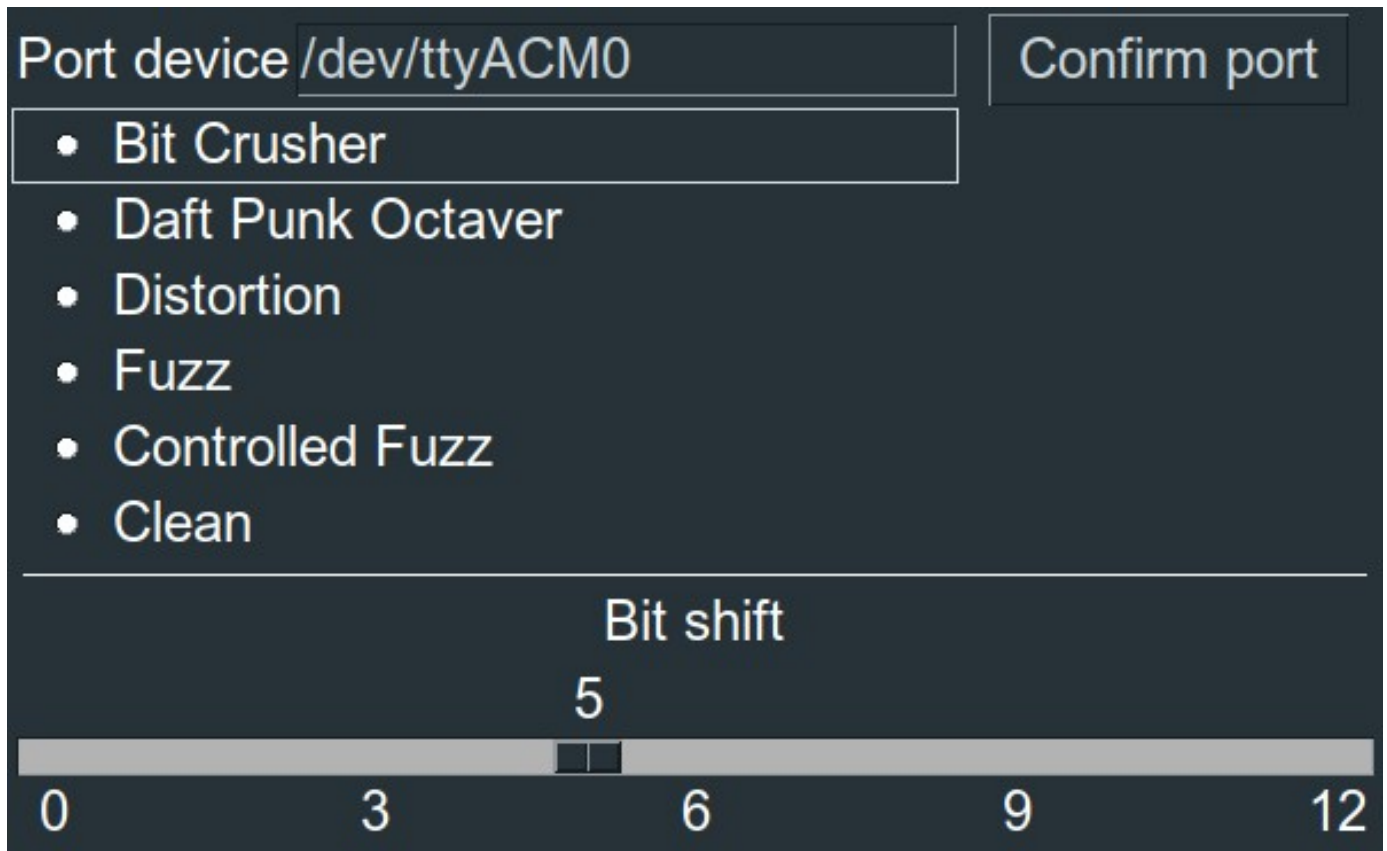
En este proyecto hay un único módulo C/C++ que hace *switching* para determinar qué efecto realizar.

Implementa una *ISR* para el timer del *PWM* que lee un valor del *ADC*, lo modifica según el efecto y lo escribe a través del *PWM*.

También define un *SerialEvent* para obtener el efecto y el modificador cada vez que lo recibe, sin necesidad de hacerlo como una tarea.

También hay otras funciones que permiten decodificar el mensaje enviado desde la interfaz. Sin embargo son funciones de manipulación de arreglos ajenas al desarrollo del proyecto.

La interfaz gráfica se programa en python. Esta provee una entrada de texto donde se escribe el puerto o dispositivo que se va a utilizar para la comunicación serie; un menú de *radio buttons* que permite elegir el efecto y un *slider* para elegir el modificador.



Port device

- Bit Crusher
- Daft Punk Octaver
- Distortion
- Fuzz
- Controlled Fuzz
- Clean

Bit shift

5

0 3 6 9 12

Interfaz gráfica que controla el efecto y el modificador del Arduino.

Cada vez que se clickea en un efecto se manda un mensaje por el puerto serie al Arduino con un "0", para indicar un cambio de efecto. Luego se manda un número del 1 al 8, que indica qué efecto se eligió, siendo 1 el *bit shifter*, 2 el *booster*, y así sucesivamente. Por último se manda un mensaje con el valor que contiene el *slider*, más uno para evitar que el cero se pueda interpretar de más de una forma. Si se mueve el slider, no se actualiza el valor del efecto: queda implícito. La longitud de los mensajes es siempre de cinco bytes y se mandan como strings de cinco caracteres, para simplificar la decodificación en el Arduino.

## Referencias

- <https://github.com/ElectroSmash/pedalshield-uno>
- <https://create.arduino.cc/projecthub/electrosmash/arduino-uno-guitar-pedal-b2ba96>
- <https://github.com/tardate/LittleArduinoProjects/tree/master/playground/pedalShieldUno> (ejemplos bajo osciloscopio de los efectos)
- <https://www.electrosmash.com/pedalshield-uno>
- <https://github.com/garatma/proyecto-embebidos> (repo de este proyecto)