Electric Guitar Effects Pedal

Introduction

This project is an electric guitar effects pedal built around the **ATmega328P Xplained Mini** development board.

The pedal receives the signal from an electric guitar, processes it using the microcontroller, and sends the modified signal further to an amplifier or audio system. The main effects chosen for this project are **distortion** and **tremolo**, together with a simple **clean/bypass mode**.

The project started from the idea that guitar pedals are both useful and interesting from a technical point of view. They are good examples of systems that combine analog electronics with embedded software. Instead of making a very complex multi-effect unit, the goal was to create something simpler, realistic, and possible to build by hand with common components.

This project is useful for us because it helps us apply concepts learned in the laboratories, such as ADC, PWM, timers, interrupts, GPIO, I2C, and UART, in a practical product. It can also be useful for other students or hobbyists who want to understand the basics of digital audio effects on a microcontroller.

General Description

The project can be split into a few simple functional blocks:

- **Input stage** – receives the guitar signal through a jack connector.
- **Analog conditioning stage** – prepares the signal for the microcontroller by shifting and filtering it.
- **ADC module** – samples the guitar signal.
- **Control module** – reads potentiometers, push buttons, and footswitch.
- **Effect processing module** – applies the selected effect in software.
- **PWM output module** – generates the processed output signal.
- **Output filter stage** – smooths the PWM signal and sends it to the output jack.
- **User feedback module** – can use an LCD or simple LEDs to indicate the current mode.
- **Debug module** – sends data through UART for testing and debugging.

Suggested block diagram:

Signal flow:

Guitar Input → Analog Conditioning → ADC → Effect Processing → PWM Output → Output Filter → Guitar Amplifier

Control flow:

- potentiometers adjust the effect parameters;
- buttons select the active effect;
- the footswitch enables or disables the pedal;
- an optional LCD or LEDs show the active mode.

The interaction is simple: the input signal is first prepared so it can be safely read by the microcontroller. Then the ADC samples it, the firmware applies either distortion or tremolo, and the result is sent to the PWM output. After filtering, the output becomes an analog signal again and goes to the amplifier.

Hardware Design

Main components

- 1 x **ATmega328P Xplained Mini**
- 2 x **6.35 mm mono jack connectors**
- 1 x **single-supply dual operational amplifier**
- 1 x **3PDT footswitch**
- 2 x **push buttons**
- 2 or 3 x **potentiometers**
- 1 x **status LED**
- optional: **16×2 LCD with I2C backpack**
- resistors and capacitors
- coupling capacitors
- pull-up resistors
- RC filter components
- breadboard or perfboard

Input stage

The electric guitar produces an AC signal, but the ADC of the microcontroller works with positive voltages only. Because of this, the signal has to be adapted before sampling.

The analog conditioning stage contains:

- an input capacitor for AC coupling;
- a voltage divider that creates a mid-supply bias;
- an op-amp stage for buffering or amplification;
- optional filtering to reduce noise.

This stage is important because it makes the signal compatible with the ADC and improves stability during processing.

Processing unit

The main processing unit is the **ATmega328P** from the Xplained Mini board. It handles:

- sampling of the guitar signal;
- reading control inputs;
- applying the selected effect;
- generating the output with PWM;
- communication through UART;
- optional display updates.

Output stage

The output signal is generated as PWM. Since PWM is not directly suitable as an audio signal, it is passed through an RC low-pass filter. After filtering, the signal is sent to the output jack.

An op-amp buffer can also be used here to improve output stability.

Controls

The user controls are:

- **footswitch** – turns effect on/off or bypasses it;
- **buttons** – select effect mode;
- **potentiometers** – change effect parameters;
- **LED** – shows whether the pedal is active;
- optional **LCD** – displays current mode and values.

Proposed pin usage

- **PC0 / ADC0** – guitar input
- **PC1 / ADC1** – potentiometer 1
- **PC2 / ADC2** – potentiometer 2
- **PC3 / ADC3** – potentiometer 3, if needed
- **PC4 / SDA** – optional LCD
- **PC5 / SCL** – optional LCD
- **PD5 or PD6** – PWM output
- **PD2** – footswitch / interrupt input
- **PD3** – button input / second interrupt
- **PD0 / RX** – UART
- **PD1 / TX** – UART
- **PB5** – onboard LED or status output

Schematics and diagrams

This section will include:

- input conditioning schematic;
- control wiring schematic;
- PWM output filter schematic;
- full project schematic;
- signal diagrams and, if available, simulation results.

Placeholders:

Software Design

Development environment

The firmware will be written in **C/C++** using an AVR-compatible development environment:

- **PlatformIO**
- **avr-gcc**

External libraries

The project will try to keep the code as simple as possible. Only a few external libraries may be used:

- standard AVR libraries;
- optional LCD I2C library;
- basic UART helper functions for debug.

Most of the main logic will be implemented manually.

Laboratory concepts used

This project includes functionality from multiple laboratories:

- **GPIO** – LED, buttons, footswitch
- **ADC** – input signal and potentiometers
- **Timers** – timing for tremolo and audio update
- **PWM** – audio output
- **Interrupts** – ADC complete interrupt and optional button/footswitch interrupts
- **UART** – serial debugging
- **I2C** – optional LCD communication

Software structure

The firmware can be organized into the following modules:

- `main.c`
- `adc.c / adc.h`
- `effects.c / effects.h`
- `controls.c / controls.h`
- `pwm.c / pwm.h`
- `uart.c / uart.h`
- optional `lcd.c / lcd.h`

Main logic

The general execution flow is:

1. initialize peripherals;
2. configure ADC;
3. configure timer and PWM output;
4. initialize buttons, LED, and optional LCD;
5. start sampling the input signal;
6. process each sample depending on the selected mode;
7. update output;
8. update controls and debug information in the main loop.

Effect modes

Clean / bypass

This mode sends the input further without applying a strong audio effect. In the final hardware version, the footswitch can also provide real hardware bypass.

Distortion

Distortion is implemented by increasing the amplitude of the signal and clipping it when it goes above a threshold.

Example logic:

- read sample;
- remove bias;
- multiply by gain;
- clip positive and negative peaks;
- restore bias;
- send to output.

The distortion intensity is controlled by a potentiometer.

Tremolo

Tremolo changes the signal amplitude periodically. It is implemented by multiplying the input signal with a slowly changing factor generated in software.

Parameters:

- **speed**
- **depth**

The modulation can be generated using a counter or a lookup table.

Planned functions

- `init_gpio()`
- `init_adc()`
- `init_pwm()`
- `init_uart()`
- `init_timer()`
- `read_controls()`
- `process_distortion()`
- `process_tremolo()`
- `update_led_status()`
- `send_debug_info()`
- optional `lcd_show_mode()`

Implementation notes

The project should contain more than 100 lines of original non-trivial code. The most important parts are:

- peripheral configuration;
- ADC reading;
- PWM output;
- button and footswitch logic;
- parameter control;
- audio processing functions.

Results Obtained

At this stage, the project has a complete design plan and a clear implementation direction.

The expected final result is a working prototype that:

- receives electric guitar signal;
- offers clean/bypass mode;
- offers distortion effect;
- offers tremolo effect;
- uses potentiometers for parameter control;
- uses buttons and footswitch for interaction;
- uses LED and optionally LCD for feedback;
- can send debug data over UART.

After implementation, this section will include:

- photos of the prototype;
- oscilloscope screenshots;
- screenshots or photos of the user interface, if LCD is used;
- notes about sound quality and stability;
- practical test results with guitar and amplifier.

Placeholders:

Conclusions

The project is a realistic embedded application that combines hardware and software in a clear and useful way.

Its main advantage is that it remains simple enough to be built by hand while still including important concepts from the laboratory. Instead of trying to implement many difficult effects, the project focuses on two effects that are easier to understand and demonstrate: distortion and tremolo.

The final result should be a functional guitar pedal prototype that can later be improved with more effects, better filtering, a custom PCB, or a more advanced user interface.

Download

The following files will be uploaded when the implementation is finished:

- source code
- schematics
- block diagram
- optional simulation files
- README
- ChangeLog
- build instructions
- flashing instructions

Example files:

- `guitar_pedal_sources.zip`
- `guitar_pedal_schematics.pdf`
- `README.md`
- `CHANGELOG.txt`

Example attachments:

- Source code
- Schematics
- README

Journal

tip

Bibliography/Resources

Hardware Resources

- ATmega328P datasheet
- ATmega328P Xplained Mini documentation
- operational amplifier datasheet
- LCD datasheet, if LCD is used
- basic guitar pedal electronics references

Software Resources

- AVR peripheral documentation
- ADC and PWM examples
- UART communication examples
- optional I2C examples

Course Resources

- PM laboratory materials
- PM project template

Other Resources

- similar guitar pedal projects
- audio signal processing tutorials for embedded systems

Export to PDF