# Pedală Programabilă pentru Chitară

* Autor: Jumărea Ștefan Dorin - 331CC

## Introducere

Proiectul își propune realizarea unei pedale programabile pentru chitară. Pedala va modifica sunetul produs de chitară, va fi amplasată între chitară și amplificator, și va putea fi programată pentru a simula mai multe efecte. Vor exista butoane și switch-uri prin care se vor putea modifica efectele fără a se reprograma plăcuța.

## Descriere generală

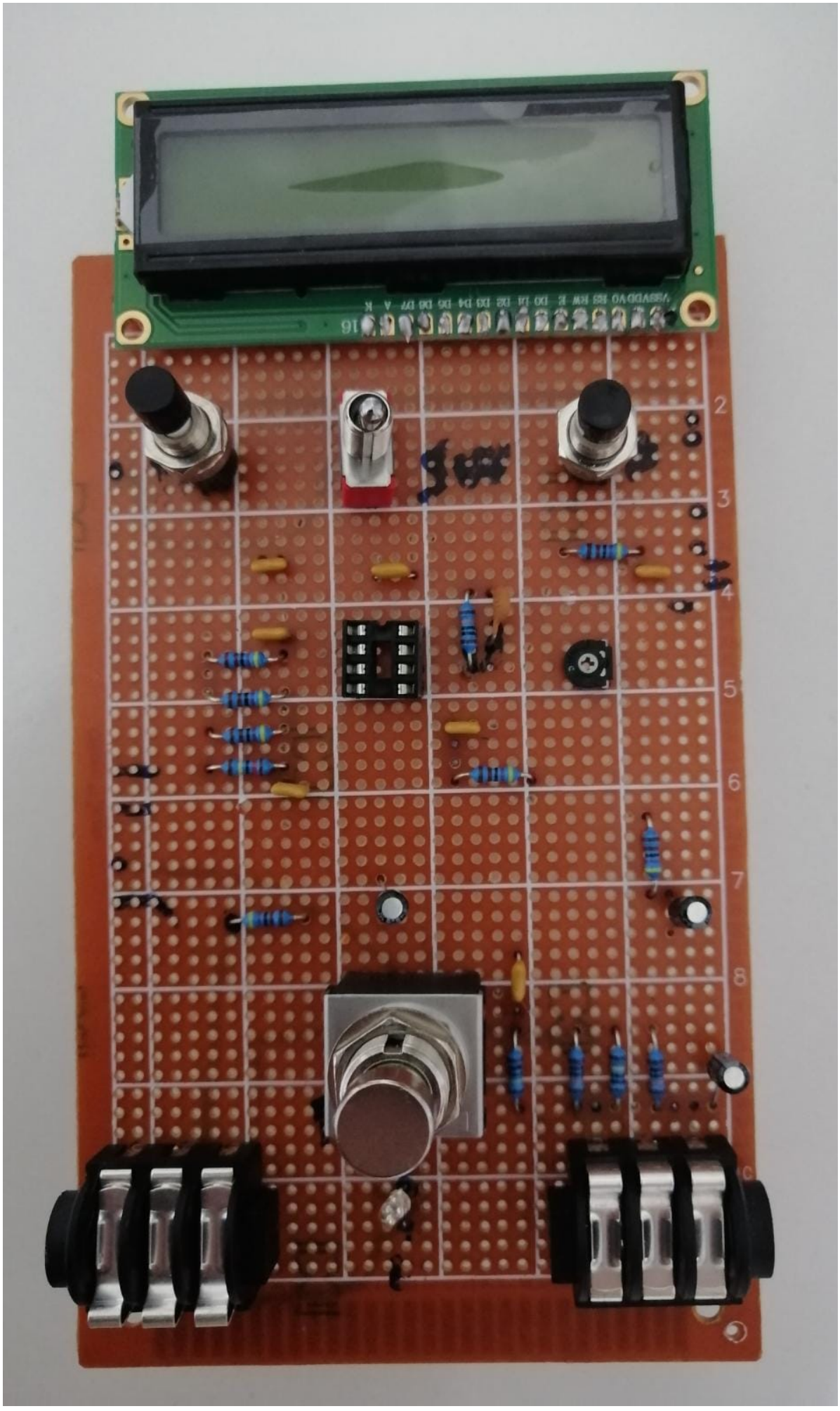
La intrare, semnalul va trece printr-un set de filtre pentru a elimina parte din zgomot. Intrarea va fi amplificată, de la 2.8V (ieșirea pe cablu Jack) la 5V. Semnalul va intra apoi într-o intrare analogică, unde va fi modificat după cerințe. Semnalul de ieșire se va genera folosind PWM. Se vor afișa pe un LCD informații despre modul curent al pedalei.

Pentru pornirea/oprirea pedalei se va folosi un footswitch. Când pedala este oprită, se va face doar bypass la semnalul de intrare spre ieșire. Vor exista butoane și switch-uri ca intrări în Arduino.

Schema electrică:

Surse de inspirație schemă electrică:

- Tone stage: <https://neatcircuits.com/doctor_pedal.htm>
- Basic stuff: <https://www.wamplerpedals.com/blog/uncategorized/2020/05/how-to-design-a-basic-overdrive-pedal-circuit/>
- More basic stuff: <https://cushychicken.github.io/posts/ltspice-tube-screamer/>



# Hardware Design

Piesele folosite vor fi:

- Footswitch pentru ON/OFF
- Conectori pentru mufe Jack
- Op Amp
- Butoane, switch-uri, etc. pentru parametrii de intrare
- Arduino UNO
- LCD pe care se va afișa efectul curent al pedalei

# Software Design

Se vor implementa mai multe efecte, printre care:

- Volume Booster
- Distortion
- Pitch Shifter
- Chorus

Totul se va implementa în întreruperea dată de ADC. În `setup()` se vor face configurările pentru IO, ADC și PWM. În funcția `loop()` doar se va aprinde ledul și se va seta efectul dorit în funcție de numărul de toogle-uri are switchului.

## Volume Booster

Pentru a realiza un efect de Volume Booster, se va prelua semnalul de la ADC și se va aduna la el o valoare pentru a modifica volumul: `input += booster_var;`, unde `booster_var` este o variablilă ce se va modifica folosind butoanele. Dacă `booster_var` este un întreg pozitiv, volumul va fi mărit, altfel va fi scăzut.

## Distortion

Pentru a se realiza un efect de Distortion, se vor selecta toate armonicele cu un volum mai mic ca o variabilă dată, și se vor ridica toate la un prag:

Mai multe detalii aici: https://en.wikipedia.org/wiki/Distortion#Harmonic_distortion

```
if (!digitalRead(PUSHBUTTON_2)) {
  if (distortion_threshold < 32768)
       distortion_threshold = distortion_threshold + 25;
```

```
}

if (!digitalRead(PUSHBUTTON_1)) {
  if (distortion_threshold > 0)
       distortion_threshold = distortion_threshold - 25;
}

if (input > distortion_threshold)
    input = distortion_threshold;
```

## Pitch Shifter

Pentru a se realiza un efect de Pitch Shifter, se va modifica frecvența cu care este citit și transmis semnalul. Astfel se poate modifica pitch-ul semnalului, prin întârzierea transmiterii semnalului.

```
if (!digitalRead(PUSHBUTTON_2)) {
  if (pitch_var < 500)
       pitch_var = pitch_var + 25;
}

if (!digitalRead(PUSHBUTTON_1)) {
  if (pitch_var > 0)
       pitch_var = pitch_var - 25;
}

counter2++;
if (counter2 >= dist_variable) {
  counter2 = 0;

  // read input
  ...

  // write output
  ...
}
```

## Chorus

Pentru a se realiza un efect de Chorus, se va combina semnalul primit cu o copie a sa, puțin întârziată. Efectul poate fi mai intens sau mai puțin intens în funcție de dimensiunea bufferului în care se reține copia semnalului.

```
if (!digitalRead(PUSHBUTTON_2)) {
  if (chorus_counter < MAX_VALUE)
       chorus_counter = chorus_counter + 1;
}
```

```
if (!digitalRead(PUSHBUTTON_1)) {
  if (chorus_counter > MIN_VALUE)
       chorus_counter = chorus_counter - 1;
}

ch_delay_cnt ++;

// Populate the buffer
if (ch_delay_cnt >= delay_depth) {
  ch_delay_cnt = 0;
  if (count_up) {
    for (p = 0; p < buff_len + 1; p++)
      chorusBuffer[delay_depth + p] = chorusBuffer[delay_depth - 1];
    delay_depth = delay_depth + buff_len;

    if (delay_depth > MAX_DELAY)
      count_up = 0
  } else {
    delay_depth = delay_depth - buff_len;
    if (delay_depth < MIN_DELAY)
      count_up = 1
  }
}

// send chorusBuffer[ch_delay_cnt] to output
```

Cod adaptat de la:
https://github.com/ElectroSmash/pedalshield/blob/master/chorus_vibrato/chorus_vibrato.ino.

# Download

jumarea_stefan_cod.zip

# Bibliografie/Resurse

Resurse pentru hardware:

- Tone Stage
- Basic Stuff, Filtre, Input/Output Stage, etc.
- More of the Same

Resurse pentru software:

- Distortion

- Pitch Shifter
- Chorus
- Workflow Cod

Export to PDF