Pedală de efect pentru chitară electrică

Acest proiect constă în realizarea unei pedale de efect audio pentru chitară electrică, care aplică mai multe efecte sonore:

Overdrive – pentru a satura semnalul și a obține un sunet distorsionat, specific rock-ului clasic.

Tremolo – pentru a modula volumul semnalului audio într-o manieră periodică.

Reverb – simulează modul în care sunetul se reflectă și se estompează într-un spațiu închis.

Efectele sunt controlate de un microcontroler Arduino UNO R3, folosind componente precum DAC (digital to analog converter), PWM (pentru modularea semnalului audio) și GPIO (pentru controlul selecției efectelor prin butoane).

Scopul proiectului este de a crea o pedală de chitară digitală, compactă și personalizabilă, care poate fi utilizată în locul unor pedale comerciale, costisitoare sau rigide ca funcționalitate.

Ideea a pornit din pasiunea pentru muzică și electronica audio, dar și din dorința de a explora modul în care microcontrolerele pot prelucra semnal analogic și pot genera efecte sonore în timp real.

Descriere generală

Proiectul meu presupune realizarea unei pedale de efect pentru chitară electrică care include efecte audio clasice, precum overdrive și tremolo. Scopul pedalei este de a îmbunătăți experiența sonoră a chitaristului, aducând un sunet distorsionat (overdrive) și un efect de modulație al volumului (tremolo). Pedala va fi controlată printr-un microcontroler care va procesa semnalul audio. Pedala va include următoarele module:

Intrare și ieșire audio: Semnalul audio al chitarei va fi preluat și trimis prin intermediul jack-urilor audio standard.

Microcontroler: Modulul principal de procesare a semnalului, care controlează efectele, prin algoritmi software.

Selecție: Efectele pot fi selectate cu ajutorul unui buton.

Interacțiunea dintre module: Semnalul audio intră în sistem prin jack-ul de intrare, este procesat de microcontroler, iar semnalul procesat iese prin jack-ul de ieșire.

Utilizatorul poate ajusta efectele în timp real folosind butoane.

Microcontrolerul controlează aplicațiile prin algoritmi.

Hardware Design

Listă de componente hardware

Microcontroller: Arduino UNO R3

Control: Buton simplu

Intrare/ieșire audio: 2x Jack mono 6.35mm (1/4”) cu pini PCB – pentru intrare/ieșire chitară

Rezistențe si condensatoare pentru atenuare si filtrarea semnalului

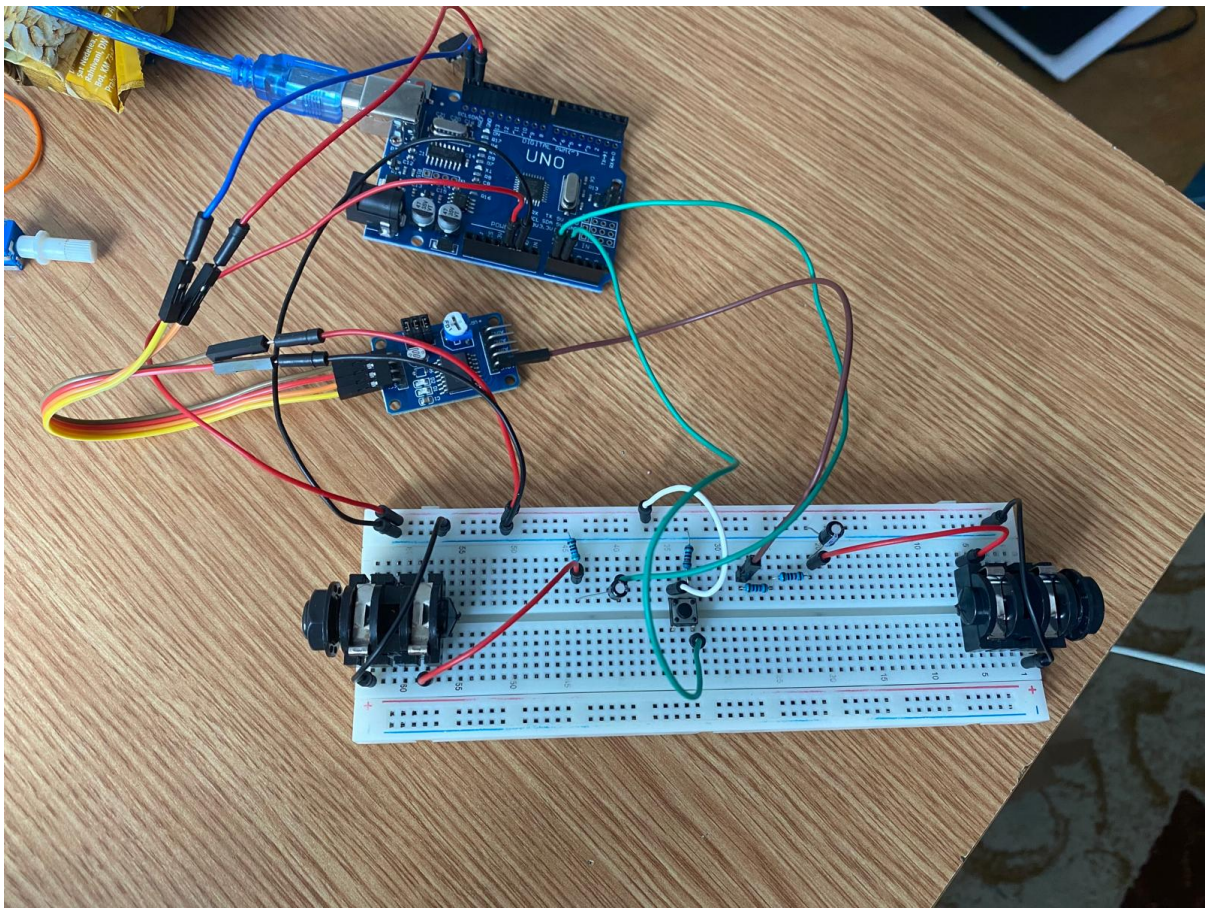
Overdrive stage: DAC, unde ma intereseaza in mod special amplificatorul operational si diodele pentru amplificarea si clipping-ul semnalului

Fire

Rezistenta de 5k legata la masa creeaza un divizor de tensiune, stabilizand semnalul ce vine de la chitara.

Condensatorul de 0.47 µF de la intrare protejeaza restul componentelor de tensiuni nedorite si pastreaza numai variatiile audio utile. (cel putin in teorie)

Filtrul trece jos de la iesire elimina zgomotele nedorite.



Software Design

Software-ul este scris în limbajul Arduino și rulează pe un microcontroller Arduino UNO R3. Acesta procesează semnalul audio de la o chitară electrică în timp real și aplică unul dintre cele cinci efecte sonore: bypass, overdrive, tremolo și reverb.

Structura principală a programului include:

Citirea semnalului audio analogic de la chitară (pin A0).

```
lastButtonState = buttonState;

  unsigned int raw = analogRead(ADC_PIN);
  raw = (raw + 1) / 4 - 1;

  unsigned int out = applyEffect(((float)raw) / SIGNAL_MAX) * SIGNAL_MAX;
```

Prelucrarea semnalului prin funcții specifice fiecărui efect audio (distorsiune, tremolo, reverb etc.).

* Tremolo

```
int index = 0;
float applyTremolo(float x) {
  // version 1
  index++;

  float Fs = TIMER / 8.0f;
  float Fx = 5.0f;
  float alpha = 0.25f;
  float trem = (1.1f + alpha * sin(2.0f * M_PI * index * (Fx / Fs)));

  return trem * x * 2.0f;
}
```

* Overdirve

```
float applyDistortion(float x) {
  x *= 1.75f;

  if (x > 1.0f) x = 1.0f;
  if (x < 0.0f) x = 0.0f;

  float y;
  if (x < 0.2f)
    y = 0.0f;
  else if (x > 0.7f)
    y = 1.0f;
  else
    y = x * x * (3 - 2 * x);  // Smooth curve in between

  return y;
}
```

* Reverb

```
uint8_t reverbBuffer[REVERB_BUFFER_SIZE];
int reverbIndex = 0;
float applyReverb(float x) {
  // Read the delayed sample from buffer
```

```
  int delayedIndex = (reverbIndex - REVERB_DELAY + REVERB_BUFFER_SIZE) %
REVERB_BUFFER_SIZE;
  float delayedSample = reverbBuffer[delayedIndex] / 255.0f;

  // Mix current input with decayed delayed sample
  float out = x + REVERB_DECAY * delayedSample;

  // Store current output in the buffer
  reverbBuffer[reverbIndex] = (uint8_t)(adjustAmplitude(out) * 255);
  reverbIndex = (reverbIndex + 1) % REVERB_BUFFER_SIZE;

  return adjustAmplitude(out);
}
```

Scrierea semnalului procesat pe un DAC extern (PCF8591) pentru ieșirea audio analogică.

Controlul efectului activ printr-un buton fizic conectat la pinul A1 – fiecare apăsare schimbă efectul curent.

```
   int buttonState = digitalRead(BUTTON);

   if (lastButtonState == HIGH && buttonState == LOW) {
     currentEffect = (currentEffect + 1) % MAX_EFFECT;  // Next effect
     Serial.println(currentEffect);
     delay(200); // Debounce simplu
   }

   lastButtonState = buttonState;
```

Algoritmi de procesare audio implementați în funcții dedicate fiecărui efect, folosind formule matematice simple și buffer circular (pentru reverb).

```
#define ADC_PIN A0        // Analog input from guitar signal
#define BUTTON A1
#define SIGNAL_MAX 255

#define TIMER 15625

#define REVERB_BUFFER_SIZE 1000  // Adjust based on available RAM
#define REVERB_DECAY 0.75f       // How much of the echo is preserved
#define REVERB_DELAY 800         // How far back in time (in samples) the
echo is

#define MAX_EFFECT 5
```

Acest software permite utilizatorului să comute dinamic între efecte și să obțină sunete expresive folosind doar hardware minim și procesare digitală în timp real.

Rezultate Obţinute

Dupa multe incercari, am reusit sa ajung la o versiune finala.

Partea de hardware ramas mai simpla decat ce mi-as fi dorit, fara valori reglate prin potentiometru.

Am experimentat cu mai multe efecte, cele mai reusite fiind cel de distors, tremolo si reverb.

Nu am reusit sa filtrez suficient de bine semnalul astfel incat semnalul ajuns la amplificator sa fie curat.

* ADC - analogRead()

* I2C - comunicarea cu DAC-ul

* GPIO - pinMode

* timere pentru tremolo

Concluzii

Nu am. }

Download

proiect_hau.zip
Jurnal

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.
Bibliografie/Resurse

Listă cu documente, datasheet-uri, resurse Internet folosite, eventual grupate pe **Resurse Software** şi **Resurse Hardware**.

Export to PDF