Daniel FERA (78412) - Procesor de efecte pentru chitară

Autorul poate fi contactat la adresa: **Login pentru adresa**

Introducere

Dispozitivul pe care îl voi realiza in cadrul proiectului este un procesor de efecte pentru chitară. El alterează semnalul generat de o chitară electrică astfel încât sunetul redat să capete efecte care pot schimba mult modul în care se aude o melodie cântată la chitară.

Efectele pe care le va crea acest dispozitiv sunt:

- Distortion
- Tremolo
- Flanger
- Echo
- Compression

Ideea de la care am pornit este reprezentată de un proiect din anii trecuți, văzând astfel că un astfel de dispozitiv este realizabil cu procesorul ATMEGA 324A-PU. Eu cred că acest proiect este util atât pentru mine, cât și pentru oricine cântă la o chitară electrică, pentru că acest dispozitiv poate schimba foarte mult experiența pe care o are atât cel care cântă, cât și un alt ascultător.

Descriere generală

Diagrama bloc:

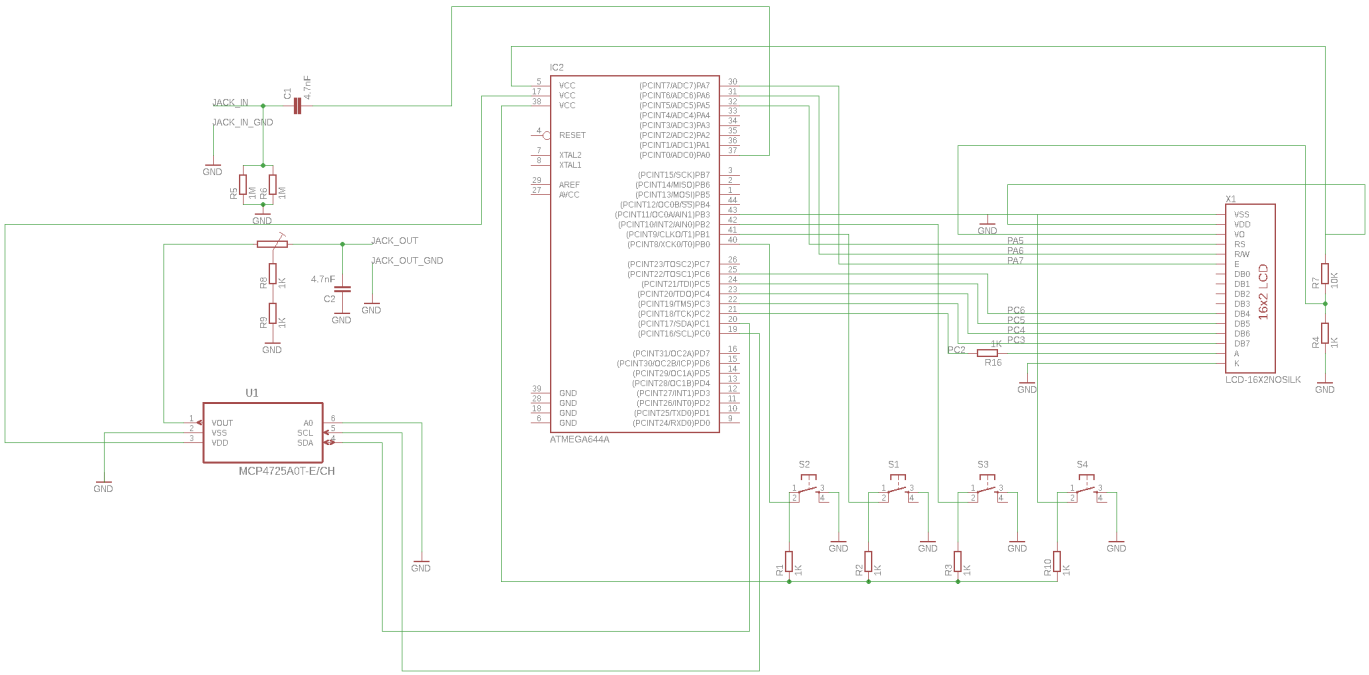
Hardware Design

Lista de piese:

Număr piese	Nume piesă	Observații
1	Placa de bază ATMega 324A-PU	

1	LCD Text 2×16	http://www.adelaida.ro/display-lcd-16x2-80x36x13-2mm-blue-rc1602bb.html
2	Mufă Jack 6.3mm Mamă	3614-PRIZA-AUDIO-6-35-MM-MONO-DE-CABLAJ.html
2	Rezistențe de 1kΩ	http://www.adelaida.ro/r-1-4w-1k.html
2	Rezistențe de 1MΩ	http://www.adelaida.ro/r-1-4w-1m.html
2	Condensatoare 4.7nF	http://www.adelaida.ro/100pf-3kv-condensator-ceramic-cc1k-4n7.html
4	Butoane pentru control și setări	http://www.adelaida.ro/microtach-tact-65k-spst-no.html
1	Potențiometru 10k	http://www.adelaida.ro/potentiometru-mono-10k-lin-r16110n-b10k.html
1	DAC MCP4725 cu interfata I2C (TWI)	https://www.optimusdigital.ro/ro/altele/1327-modul-dac-mcp4725-cu-interfaa-i2c.html

Schema electrică:



Software Design

- Mediu de dezvoltare: AVR-GNU-toolchain, Sublime Text
- Biblioteci folosite: lcd.c & lcd.h (din laboratorul 1), twi.c & twi.h (comunicarea prin I2C cu modulul DAC)
- Implementare:

1. Distortion: return (1-es_distortion_depth) * x + es_distortion_depth * tanh(es_distortion_gain * x);
2. Tremolo: float modulation = sin((2 * M_PI / OCR1A) * TCNT1);
   modulation = (1 - es_tremolo_depth) + es_tremolo_depth * modulation * modulation;
   return x * modulation;
3. Flanger: float max_sample_delay = es_flanger_delay * MAX_OUTPUT_HISTORY;

```
float current_sin = fabs(sin((2 * M_PI / timer0_top) * TCNT0E));
int16_t current_delay = (int16_t)ceil(current_sin * max_sample_delay);
float out_delayed = (float)output_history[(output_history_last + (MAX_OUTPUT_HISTORY - 1) -
current_delay) % MAX_OUTPUT_HISTORY];
out_delayed /= SGN_MAX;
return (1 - es_flanger_depth) * x + es_flanger_depth * out_delayed;
```

4. Echo:
```
float out_delayed = (float)output_history[(output_history_last + (MAX_OUTPUT_HISTORY - 1)
- es_echo_delay) % MAX_OUTPUT_HISTORY];
out_delayed /= SGN_MAX;
return (1 - es_echo_depth) * x + es_echo_depth * out_delayed;
```
5. Compression:
```
const float slope = 1.0f / 5.0f;
static float compressor_gain = 1.0f;
static float state = 0.0f;
const float max_abs = fmax(x, es_compression_kmin);
const float max_abs_dB = log(max_abs);
const float overshoot = max_abs_dB - es_compression_knee_treshold;
const float rect = fmax(overshoot, 0.0f);
const float cv = rect * slope;
const float prev_state = state;
if (cv <= state)
    state = es_compression_alpha_attack * state + (1 - es_compression_alpha_attack) * cv;
else
    state = es_compression_alpha_release * state + (1 - es_compression_alpha_release) * cv;
compressor_gain *= exp(state - prev_state);
return x * compressor_gain;
```

Rezultate Obţinute

În acest videoclip voi prezenta cum funcționează procesorul de efecte:

<WRAP center round todo 60%> Pozele de la PM Fair... </WRAP>

Concluzii

În concluzie, pot spune că proiectul a fost unul foarte interesant, mai ales că înaintea acestui semestru nu aveam nici o idee despre cum se realizează un montaj, iar acum pot spune că mă descurc destul de bine să înțeleg o schemă electrică și să o transpun în mediu fizic. Consider că

~~puteam așeza chiar mai bine legăturile cu PCB-ul, astfel încât să pot încapsula proiectul, dar fiind~~ începător, consider că se acceptă și așa cum l-am realizat.

Download

fera_daniel.zip

Jurnal

Bibliografie/Resurse

- pedalShield pedalshield
- Arduino Guitar Pedal Arduino-Guitar-Pedal
- DSP Audio effects distortion
- Documentația în format PDF

From:
http://ocw.cs.pub.ro/courses/ - **CS Open CourseWare**

Permanent link:
http://ocw.cs.pub.ro/courses/pm/prj2018/astratulat/feradanielguitardsp

Last update: **2021/04/14 15:07**