FPGA klavir na razvojni plošči Red Pitaya

Projekt pri predmetu
Digitalna integrirana vezja in sistemi

Datum: 30. 1. 2020

Marko Remec,
64080454

Uvod

Cilj projekta je bil narediti simulacijo klavirja na razvojni plošči Red Pitaya. Ob pritisku tipke na tipkovnici dobimo na analognem izhodu ton določene frekvence in trajanja. Naloga je razdeljena na dva dela:

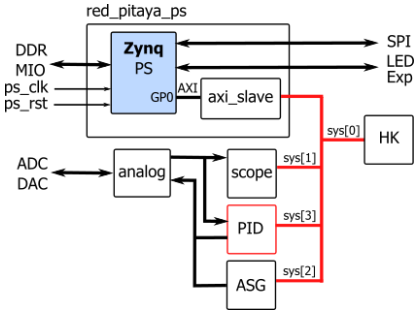
1. FPGA del, ki skrbi za generiranje tonov (verilog/VHDL) in
2. procesorski del (ARM), ki skrbi za komunikacijo z uporabnikom preko linux terminala (programski jezik C).

Končna funkcionalnost simulatorja klavirja je:

- FPGA:
  - možnost igranja dveh tonov različnih frekvenc in trajanja hkrati,
  - metronom z možnostjo nastavljanja glasnosti in hitrosti,
  - VU meter izhodnega signala na LED diodah.
- Procesor:
  - programiranje FPGA z ustrezno bitstream datoteko, ki je shranjena v programu,
  - grafični vmesnik s klavirskimi tipkami v terminalu za komunikacijo z FPGA preko AXI vodila,
  - avtomatsko igranje shranjenih pesmi v Nokia RTTTL (Ring Tone Transfer Language) formatu.

FPGA

Projekt je zasnovan na Vivaldo projektu *redpitaya-classic*, ki že vsebuje osciloskop, signalni generator in regulator PID. Glavna komponenta projekta *red_pitaya_top.sv* povezuje procesorski del s periferijo preko AXI vodila (slika 1).



Slika 1 - Blokovna shema povezave preko AXI vodila

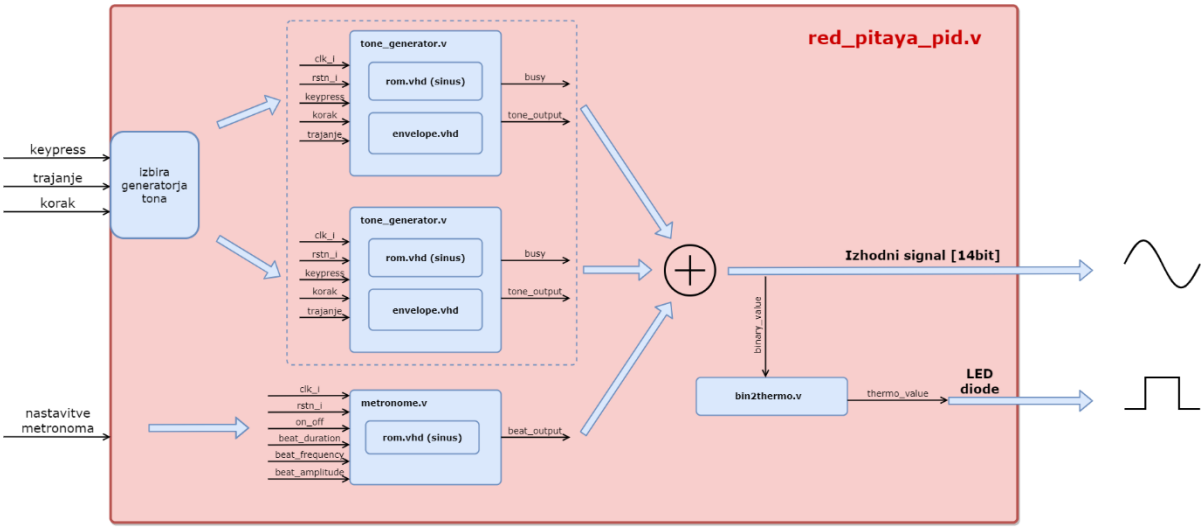
Za izvedbo projekta sem uporabil modul PID (*red_pitaya_pid.v*) v katerega sem dodal kodo za simulacijo klavirja. Uporabil sem več komponent, ki jih povezuje in nadzira modul PID:

- ROM z vrednostmi sinusa,
- ROM z vrednostmi ovojnice,
- generator tona,
- metronom,
- VU meter (pretvornik iz binarne v temperaturno kodo).

Blokovna shema povezav je na sliki 2.

Datoteke, ki sem jih glede na projekt *redpitaya-classic* spremenil/dodal so:

`red_pitaya_top.sv` `red_pitaya_pid.v` `tone_generator.v`
`envelope.vhd` `metronome.v` `bin2thermo.v` `rom.vhd`



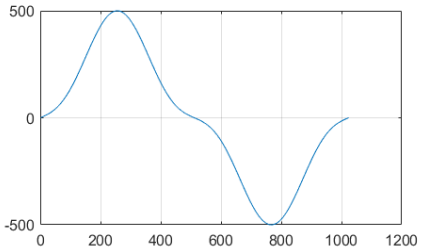
Slika 2 - Blokovna shema povezav znotraj PID modula

ROM z vrednostmi sinusa

Komponenta *rom.vhd* ima shranjenih 1k * 10bit vrednosti sinusa, oziroma eno periodo. Ker klavirski ton ni čisti sinus, so vrednosti v spominu izračunane po formuli:

$$sinval = 100 * \sin\left(2\,\pi\frac{i}{romsize}\right) + 400 * \sin\left(2\,\pi\frac{i}{romsize}\right)^3$$

S tem dobimo popačen sinus oziroma več harmonskih komponent.

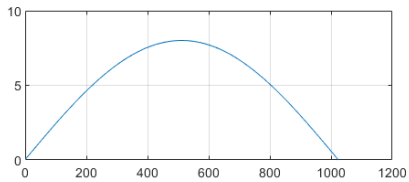


Slika 3 - Oblika popačenega sinusa shranjenega v spominu

ROM z vrednostmi ovojnice

Komponenta *envelope.vhd* ima shranjenih 1k * 10bit vrednosti ovojnice tona. Izračunane so po formuli:

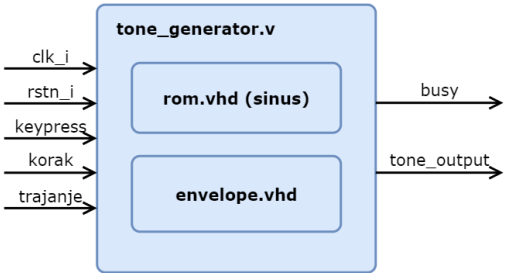
$$sinval = 8 * \sin\left(2\,\pi\frac{i}{romsize}\right)$$



Slika 4 - Oblika vrednosti ovojnice v spominu

Generator tona

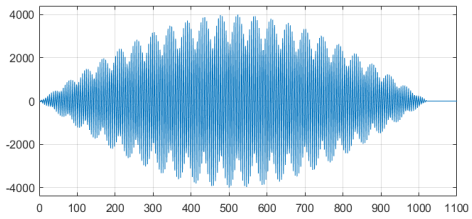
Komponenta *tone_generator.v* generira ton željene frekvence in trajanja.



Slika 5 - Komponenta generatorja tona

Sinusni signal je generiran s pomočjo števca, ki ga pripeljemo na naslovne bite spomina s sinusom. Frekvenco nastavljamo s korakom števca. Izhodni signal je generiran kot zmnožek sinusa in ovojnice – ovojnica tako predstavlja amplitudo sinusa. Ker je sinusni signal 10bitni in vrednost ovojnice največ 8, lahko uporabimo dva generatorja tona hkrati, da ne presežemo največje možne vrednosti 14 bitnega izhodnega signala.

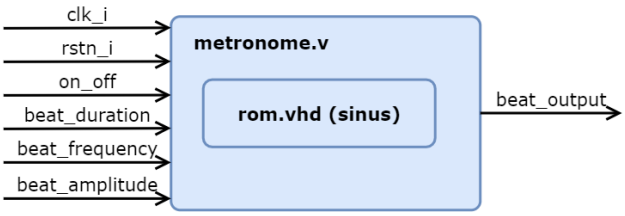
Korak števca je določen z $korak = \frac{125\,000\,000}{2^{24}} * željena\ frekvenca\ tona = 7{,}45 * f_{ton}$.



Slika 6 - Zmnožena sinusni signal in signal ovojnice

Metronom

Komponenta *metronome.v* je zgrajena podobno kot generator tona. Sinusni signal množimo s signalom amplitude. Ko je amplituda ne-ničelna dobimo signal na izhodu. Periodo in trajanje, ko je izhodni signal aktiven določimo s primerjanjem števca in registrov z nastavitvami.



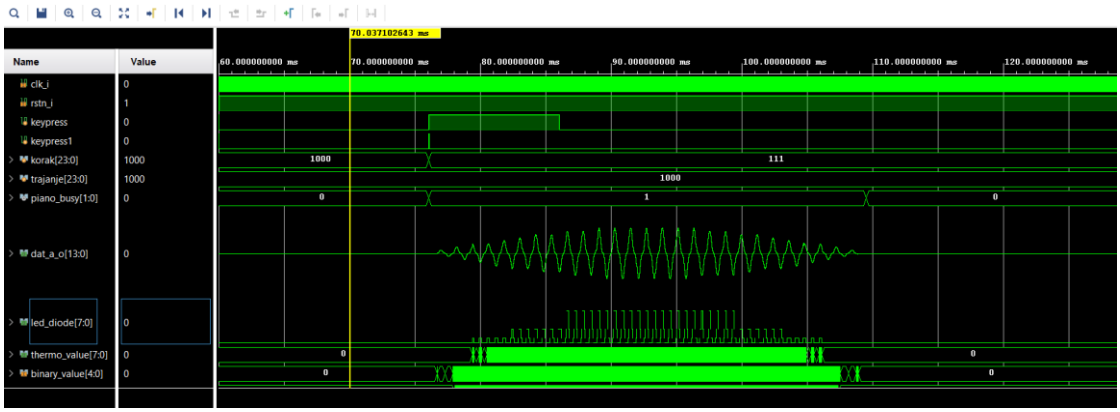
Slika 7 - Komponenta metronoma

VU meter

Komponenta *bin2thermo.v* predstavlja VU (volume unit) meter. Na platformi Red Pitaya imamo na voljo 8 oranžnih LED diod. Izhod komponente je zato 8 bitna termometerska koda. Na vhod je pripeljanih zgornjih 5 bitov izhodnega signala (predznak + 4 zgornji biti). Izračuna se absolutna vrednost, ki se ji nato priredi izhodna vrednost v termometerski kodi.

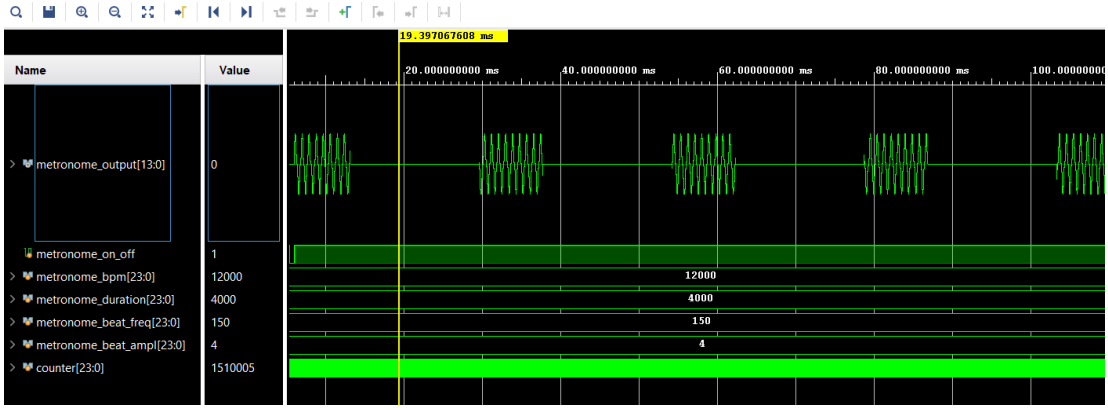
Simulacija verilog kode

Delovanje kode sem najprej simuliral v Vivaldo simulatorju. Preveril sem delovanje generatorja tona in VU metra – slika 8.



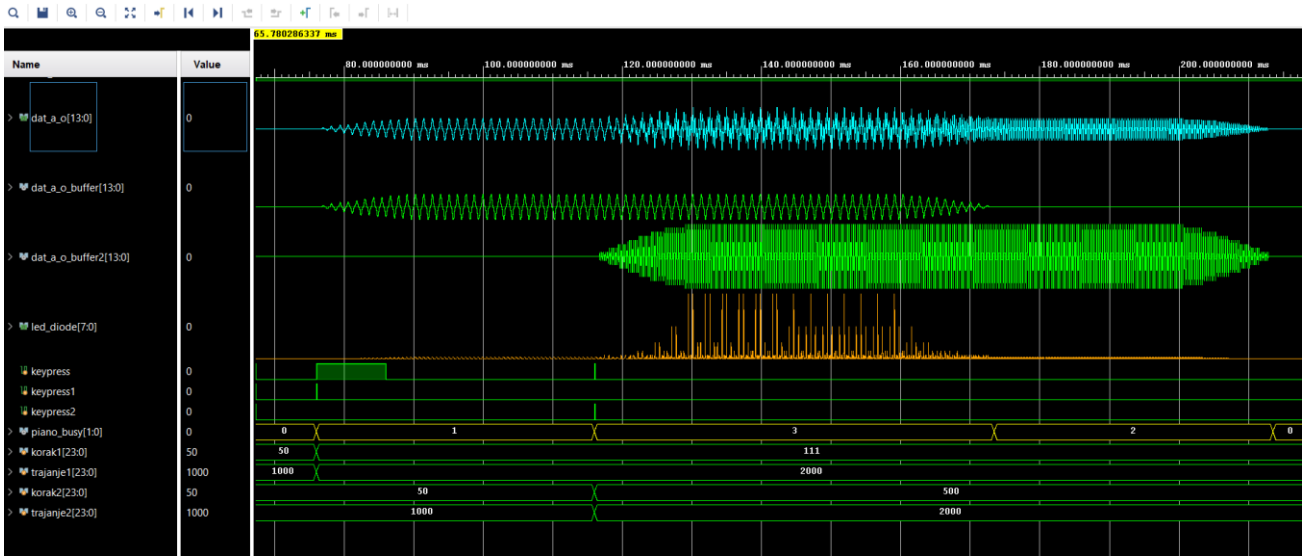
Slika 8 - Simulacija izhodnega signala in signala VU metra

Zaradi zahtevnosti simulacije sem metronom simuliral z zelo kratko periodo in trajanjem – slika 9.



Slika 9 - Simulacija izhoda metronoma

Simulacija hkratnega delovanja obeh generatorjev tona z različnima frekvencama – slika 10.



Slika 10 - Simulacija pritiska dveh tipk. Izhodni signal je modre barve, signal LED diod je oranžne barve.

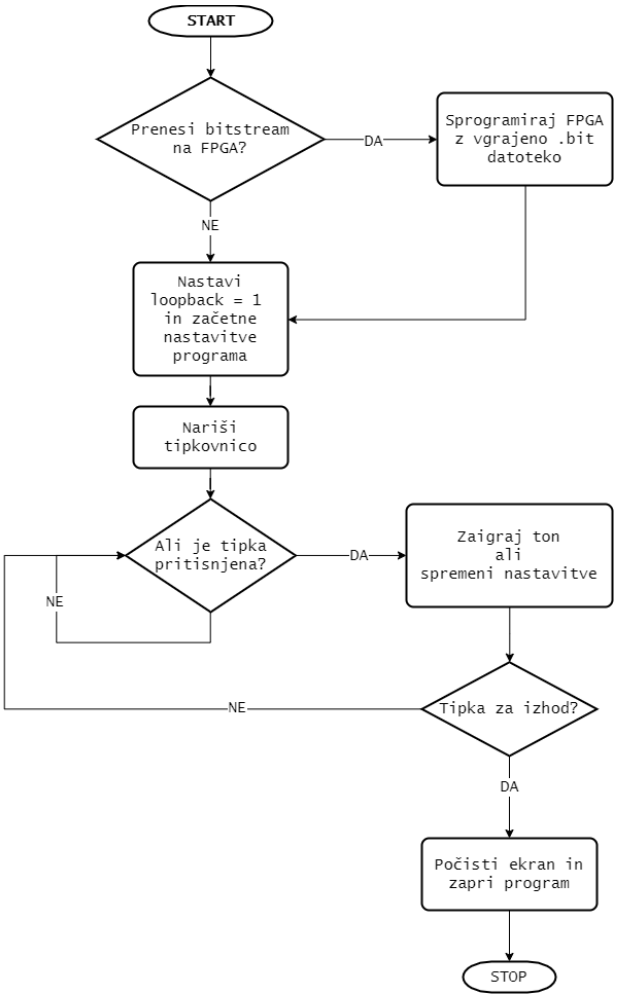
Procesorski del

Potek izvajanja programa

Osnovno delovanje C programa je predstavljeno na sliki 11. Bitstream datoteka, ki jo potrebujemo za pravilno delovanje FPGA je vključena v programu. Ob zagonu programa imamo možnost, da z njo sprogramiramo FPGA. Sledi nastavitev prevezave izhodnega signala znotraj čipa nazaj na analogni vhod, da lahko izhodni signal opazujemo na vgrajenem osciloskopu (*loopback = 1*). Sledi izris ene oktave klavirskih tipk v terminalnem oknu. Program nato vstopi v glavno zanko, ki spremlja pritiske tipk na tipkovnici. S tipkami lahko zaigramo željen ton ali spremenimo nastavitve. Klavirske tipke so povezane na:

- bele tipke: Y, X, C, V, B, N, M, vejica,
- črne tipke: S, D, G, H, J.

Mala črka pomeni kratek ton, velika črka pa dolg ton.

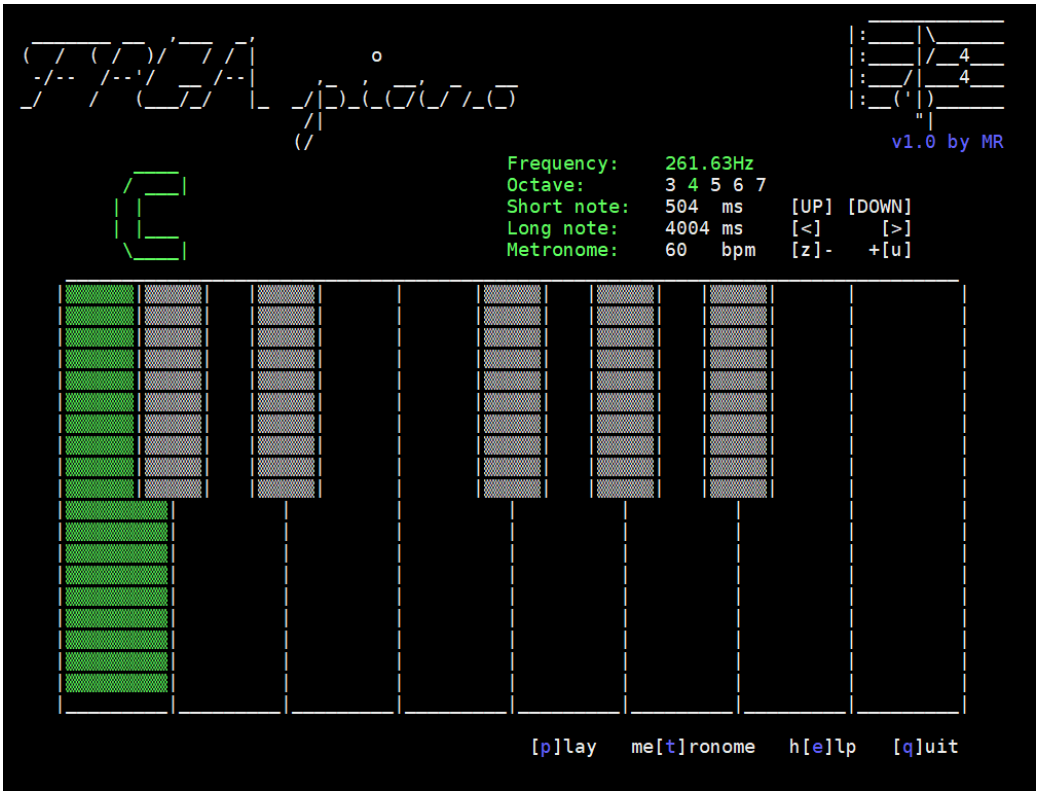


Slika 11 - Osnovno delovanje programa

Možne nastavitve so:

- izbira oktave (od C3 do C7),
- trajanje kratkega in dolgega tona,
- metronom in njegova hitrost ter glasnost,
- izbira načina delovanja: igranje, pomoč, avtomatsko igranje, igrica.

Grafični vmesnik



Slika 12 - Glavno okno programa ob pritisku tipke C

Grafični vmesnik je narejen s pomočjo knjižnice <curses.h>[1]. Knjižnica omogoča zapis znakov v buffer velikosti okna terminala in nato dejanski izris. ASCII znake je mogoče zapisovati na določene x/y koordinate v terminalu. Možna je tudi uporaba barv, ki jih terminal omogoča.

Del kode za izris tipkovnice s pomočjo vertikalnih (mvvline) in horizontalnih (mvhline) črt:

```
void draw_map(void) {
… …
    mvhline(y_origin, x_origin + 1, '_', number_of_keys * key_width - 1);
    mvhline(y_origin + key_length, x_origin + 1, '_', number_of_keys * key_width - 1);

    for (x = x_origin; x < number_of_keys * key_width + 1 + x_origin; x += key_width)
    {
        mvvline(y_origin + 1, x, '|', key_length);
… …
```

Ostale uporabljene funkcije za izris grafike so:

`clear()` – počisti okno terminala,

`refresh()` – izpiše buffer v terminal,

`mvprintw(y,x, »besedilo«)` – prirejena printf funkcija za izpis na izbrane koordinate zaslona,

`attron(atr) / attroff(atr)` – nastavitev atributov (barve).

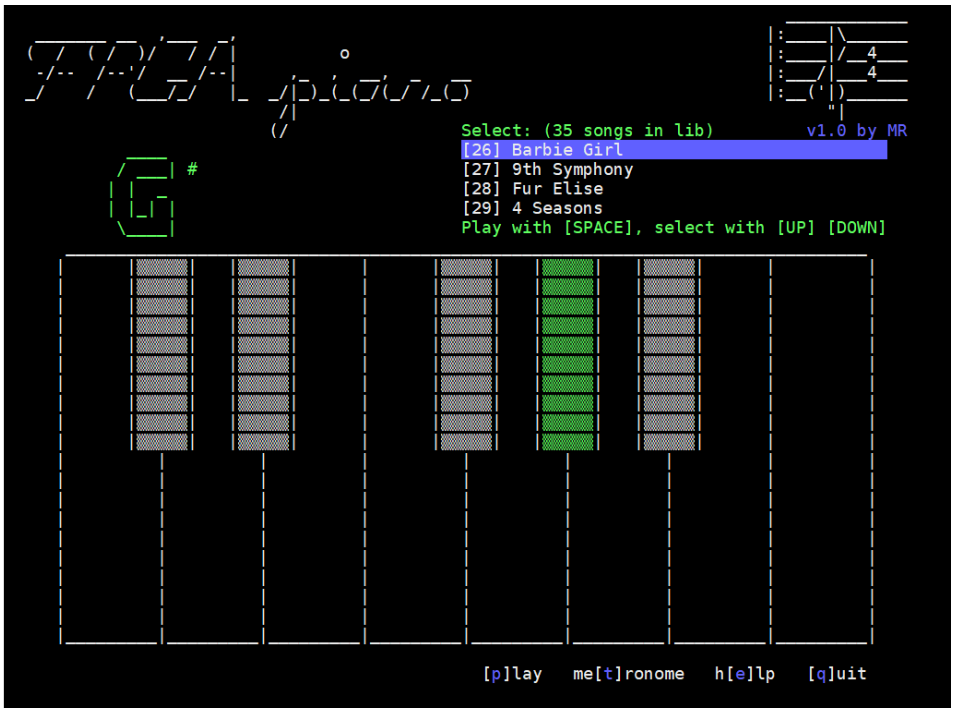
[1] https://en.wikipedia.org/wiki/Ncurses

Merjenje časa v programu je izvedeno s pomočjo funkcije *getch()*[2], ki ji lahko nastavimo zakasnitev po kateri vrne vrednost ERR. Zakasnitev nastavljamo s funkcijo *timeout(int delay)*, kjer negativna vrednost spremenljivke *delay* pomeni »blocking« zakasnitev, pozitivna pa zakasnitev v milisekundah.

Ob pritisku na tipko [e] se prikaže pomoč – slika 13.

Slika 13 - Prikaz navodil za uporabo

Zaslon za izbiro pesmi in avtomatsko igranje – slika 14.



Slika 14 - Izbira shranjenih pesmi

[2] https://linux.die.net/man/3/getch

Sporočanje pritiska tipke

Pritisk tipke sporočimo s pisanjem vrednosti 1 in nato 0 v register VAL_KEYPRESS na naslovu BASE_PID[3] + 0x20. Pred tem moramo nastaviti registra za trajanje in frekvenco tona: VAL_TRAJANJE (BASE_PID + 0x08) in VAL_KORAK_FREQ (BASE_PID + 0x00).

Funkcija pred tem preveri, ali je kateri od generatorjev tona prost, in v primeru da ni, pritisk tipke ignorira.

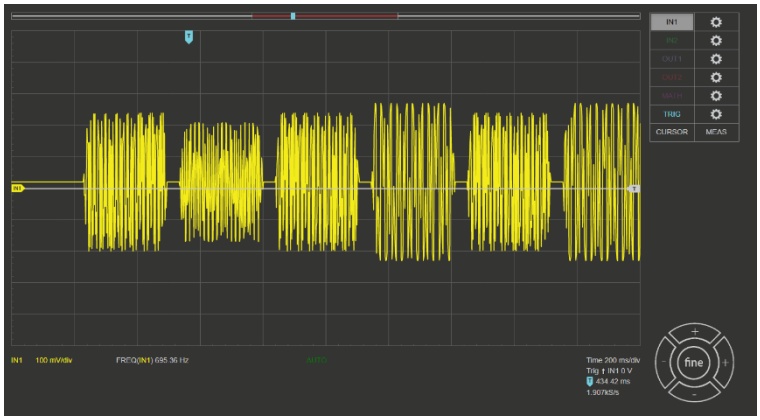
Koda funkcije *key_handler*:

```
void key_handler(int key_position, int ch) {
    if (In32(adr, VAL_BUSY) == 0) {
        set_key(key_position);
        key_name(key_position);
        stop_arr[key_position] = 0;
        counter_arr[key_position] = 0;

        Out32(adr, VAL_KORAK_FREQ, ton[key_remap[key_position] + (octave - 3) * 12]);

        if (ch > 97 || ch == ',') Out32(adr, VAL_TRAJANJE, duration_short);
        else Out32(adr, VAL_TRAJANJE, duration_long);

        Out32(adr, VAL_KEYPRESS, 1);
        Out32(adr, VAL_KEYPRESS, 0);
    }
}
```

Avtomatsko igranje pesmi

Ob pritisku tipke [p] program preide v način avtomatskega igranja shranjenih pesmi. Pesmi so shranjene v kodi, v formatu RTTTL[4]. Ta format sem izbral zato, ker je v zelo primerni obliki za tako aplikacijo in omogoča enostavno dekodiranje. Primer pesmi:

```
HauntHouse:
d=4,o=5,b=108: 2a4, 2e, 2d#, 2b4, 2a4, 2c, 2d, 2a#4, 2e., …
```



Slika 15 - Signal na osciloskopu ob igranju

Razčlemba formata:

```
Naslov pesmi: d=privzeto trajanje, o=privzeta oktava, b=privzet tempo: trajanje ton oktava, trajanje ton oktava, …
```

Za dekodiranje sem uporabil že narejeno funkcijo napisano za okolje Arduino (vir: *https://codebender.cc/sketch:109888#RTTTL%20Songs.ino*), ki sem jo priredil za potrebe aplikacije. Program prevede zaporedje v primerno obliko in nastavi potrebne registre.

[3] https://redpitaya.readthedocs.io/en/latest/developerGuide/fpga.html#register-map
[4] https://en.wikipedia.org/wiki/Ring_Tone_Transfer_Language

Kompilacija programa in zapis bitstream datoteka na FPGA

Program kompiliramo z ukazom:

```
gcc -o fpga_piano fpga_piano.c divs.o -lncurses
```

Pred tem moramo ustvariti divs.o datoteko, ki jo potrebuje linker. To storimo z ukazom[5]:

```
ld -r -b binary -o divs.o divs.bit
```

Program nato poženemo z ukazom:

```
./fpga_piano
```

Na FPGA program zapiše bitstream datoteko z ukazom:

```
f = fopen("/dev/xdevcfg", "wb");
fwrite(& _binary_divs_bit_start, _binary_divs_bit_size, 1, f);
fclose(f);
```

Registri za nastavitve

Za branje in pisanje registrov sta na voljo funkciji `In32` in `Out32`.

```
int In32(void *adr, int offset)
{
    return *((uint32_t *)(adr+offset));
}

void Out32(void *adr, int offset, int value)
{
    *((uint32_t *)(adr+offset)) = value;
}
```

Loopback nastavitve – naslovu BASE_HOUSEKEEPING prištejemo naslov registra.

```
BASE_HOUSEKEEPING               0x40000000
// Input
VAL_LOOPBACK                    0x0c
```

PID nastavitve – naslovu BASE_PID prištejemo naslov registra.

```
BASE_PID                0x40300000
// Input
VAL_KORAK_FREQ                  0x00
VAL_TRAJANJE                    0x08
VAL_KEYPRESS                    0x20
VAL_METRONOME_ON_OFF            0x40
VAL_METRONOME_BPM               0x44
VAL_METRONOME_DURATION          0x48
VAL_METRONOME_BEAT_FREQ 0x4C
VAL_METRONOME_BEAT_AMPL 0x24

// Output
VAL_BUSY                0x14
```

[5] https://gareus.org/wiki/embedding_resources_in_executables