Orga electronica

Introducere

O orga electronica cu 12 butoane (clape), metronom, 2 difuzoare, efecte digitale si mai multe waveform-uri.

Am ales acest proiect ca sa ma familiarizez cu generarea si manipularea semnalelor audio, lipirea de componente (multiplexor breakout) si lucrul cu timeri si intreruperi. Nu cred ca este util pentru altii. Pentru mine este util cat timp nu e terminat.

Descriere generală

Semnalele audio produse pot fi Square (digitale pentru metronom), Sin si Sawtooth. Semnalul analog este generat folosind DAC-ul onboard al Arduino R4. Ar trebui sa mearga sa canti mai multe note in acelasi timp, dar o sa fie limitat la 2-3 note (nu include metronomul) pentru a mentine un “sampling” rate bun. Pentru Sinwave as putea sa salvez niste valorori ale functiei in memoria flash si dupa sa interpolez intre ele in functie de rezolutia de care am nevoie, sau sa calculez cu serii Taylor. Inca nu m-am decis.

Efecte digitale: tremolo, vibrato, looping (nu inregistreaza real-time, tine minte in fiecare saiprezecime/32-ime ce nota era cantata, pentru cateva masuri, asa e folosita mult mai putina memorie si BPM-ul poate fi schimbat on the fly).

Hardware Design

Piese folosite:

- Arduino Uno R4 Wifi
- Multiplexor 16-to-1 CD74HC4067
- 2 difuzoare
- ~~1 ecran LCD 20×4~~
- ~~1 Rotary Encoder~~
- ~~1 Intrerupator Basculant~~
- 1 potentiometru
- Butoane, rezistente, fire etc.

In schema scrie Arduino R3, ignorati acest lucru. O sa repar mai tarziu. Edit: schema este destul de

outdated. (eg. am schimbat pinii folositi pentru select la multiplexor astfel incat sa fie toti pe acelasi registru)

Software Design

Codul aplicatiei l-am scris in VSCode + PlatformIO. Biblioteci folosite:

- analogWave.h - pentru generarea de semnal audio pe pin-ul A0
- FspTimer.h - timer-ul pentru metronom, looper si arpeggiator (m-am organizat foarte prost si nu prea am avut timp sa ma uit pe datasheet-ul microprocesorului RA4M1)
- R7FA4M1AB.h - pentru a avea access direct la registri, folosit in interactiunea cu multiplexorul

Nu am folosit nicio structura de date speciala in program. Arpeggiator-ul ar fi putut fi un linked list, dar avand in vedere ca are dimensiunea de maximum 8 note nu cred ca ar fi avut o performanta mai buna la eliminarea elementelor fata de array + memmove. (TODO: sa testez?)

Partea de cod cu looper-ul e cam messy, eram grabit si foarte nedormit. Metronomul, arpeggiator-ul si looper-ul functioneaza pe baza unui timer de frecventa (BPM/60) * RESOLUTION. eg. Indexul arpeggiator-ului este incrementat odata la 8 callback-uri.

Metronomul si looperul sunt redate pe un difuzor conectat la un pin cu iesire digitala. Arpeggiator-ul este conectat la DAC (A0) (waveform sawtooth). Din pacate m-am organizat prost si n-am mai apucat sa generez eu semnalele, asa ca ma folosesc de analogWave :(

Surse, functii:

notes.h: un fisier header care contine macro-uri cu frecventa notelor (nu are sens sa-l pun aici)

utils.h:

```
#define IS_SET(n,x)   (((n & (1 << x)) != 0) ? 1 : 0)
#define MULTIBIT_SET(addr, mask, value)
(((addr)&(~(mask)))|((value)&(mask)))
#define SET_BIT(n, x) n |= (1 << x)
#define RESET_BIT(n, x) n &= ~(1 << x)

//output register
#define PODR(X) X + 16
//direction register
#define PDR(X) X
//input register
#define PIDR(X) X

//Mux select&signal pins
#define MP_S0 4    //P104
#define MP_S1 5    //P105
#define MP_S2 6    //P106
#define MP_S3 7    //P107
#define MP_SIG 12  //P112
```

```
const uint32_t MP_CH_MASK = (1 << PODR(MP_S0)) | (1 << PODR(MP_S1) ) |
                            (1 << PODR(MP_S2)) | (1 << PODR(MP_S3));
const uint32_t MP_CHANNELS[MP_NUM_CH] = {
    0,
    (1 << PODR(MP_S0)),
    (1 << PODR(MP_S1)),
    (1 << PODR(MP_S1)) | (1 << PODR(MP_S0)),
    (1 << PODR(MP_S2)),
    ...
    }

    void addArpNote(int noteIdx) {
    if (arpPos[noteIdx] != NO_ARP_POS || arpLen >= ARP_SIZE) return;
    arpPos[noteIdx] = arpLen;
    arp[arpLen++] = noteIdx;
}

void delArpNote(int noteIdx) {
    if (arpPos[noteIdx] == NO_ARP_POS) return;
    if (arpLen <= 0) {
        Serial.println("ERROR");
    }
    int idx = arpPos[noteIdx];
    for (int i = 0; i < NUM_NOTES; i++) {
        if (arpPos[i] > arpPos[noteIdx] && arpPos[i] != NO_ARP_POS){
            arpPos[i]--;
        }
    }
    arpPos[noteIdx] = NO_ARP_POS;
    memmove(arp + idx, arp + idx + 1, arpLen - (idx + 1));
    arpLen--;
}
```

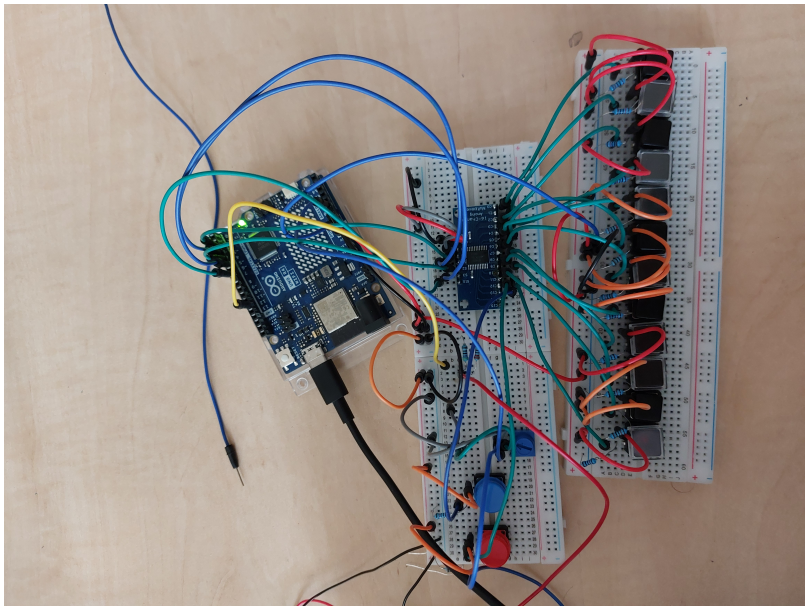
main.cpp:

Citirea de pe multiplexor:

```
for (int i = 0; i < 14; i++) {
    R_PORT1->PCNTR1 = MULTIBIT_SET(R_PORT1->PCNTR1, MP_CH_MASK,
MP_CHANNELS[i]);
    delayMicroseconds(1); //multiplexor takes about 500ns to switch channels
    int sig = IS_SET(R_PORT1->PCNTR2, PIDR(MP_SIG));
    if (sig) ...
```

Rezultate Obţinute

https://youtube.com/shorts/Bl8J0orEg_M



Concluzii

Download

Restul codului pe github: https://github.com/TheodorL/Proiect-PM

Nu e foarte readable

Jurnal

Am incercat sa lipesc un multiplexor fara sa aplic flux. Nu recomand. Luni cumpar flux si inca cateva multiplexoare.

Bibliografie/Resurse

Datasheet CD74HC4067: https://www.ti.com/lit/ds/symlink/cd74hc4067.pdf

Datasheet RA4M1: https://cdn.sparkfun.com/assets/b/1/d/3/6/RA4M1_Datasheet.pdf

Export to PDF