

Rareș-Mihail VISALOM - Sintetizator Minimal de Sunet

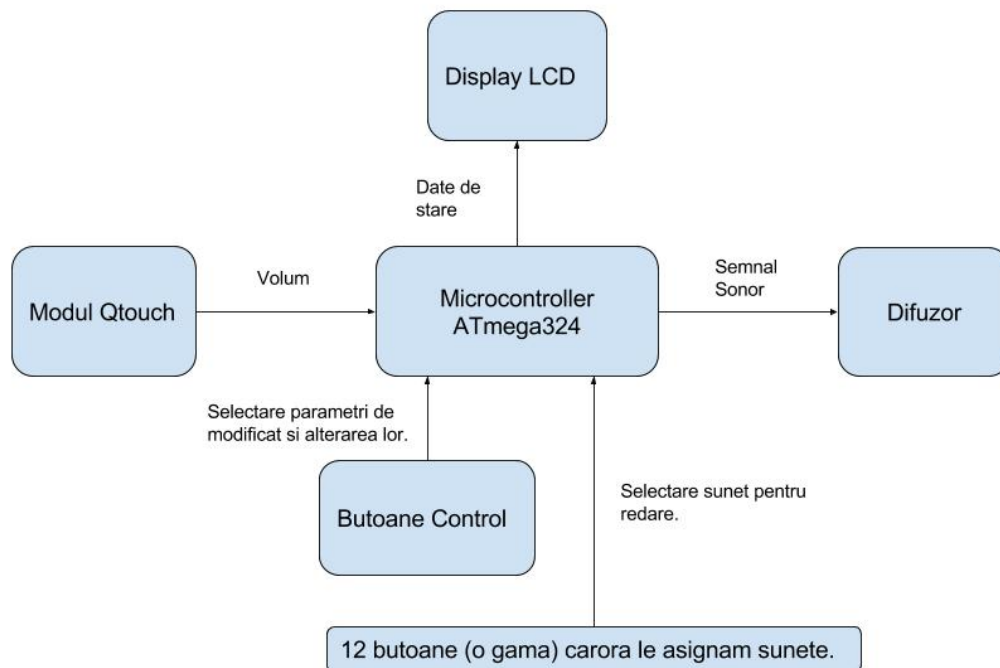
Autorul poate fi contactat la adresa: **Login pentru adresa**

Introducere

Un sintetizator de sunet este un dispozitiv electronic care produce semnale electrice pe care un difuzor le convertește în sunet. Scopul unui sintetizator este simplu: redarea sunetului. Partea complexă este generarea [proprietăților unui sunet\[4\]](#), precum frecvența (lungimea de undă), durata, volumul (amplitudinea) sau timbrul (forma undei). Calitatea sunetului este dată de granularitatea cu care un semnal analog este convertit în unul digital. Această caracteristică se numește **frecvența de esantionare** (sau [sampling rate\[5\]](#) în engleză). Sintetizatorul va fi capabil să redea sunete de o anumită frecvență, pentru o anumită durată de timp, la un anumit volum și cu un anumit timbru. Pe lângă partea de sintetizare de sunet, dispozitivul va dispune și de 12 butoane cărora le putem asigura diferite sunete. Ca funcționalitate, aceste 12 butoane fac proiectul să se asemene cu o mini-orgă electronică. Toți parametrii sunetului vor putea fi vizualizați pe LCD-ul atașat și vor putea fi modificați prin alte câteva butoane. Volumul sunetului va putea fi modificat prin intermediul modulului de Qtouch, dar și prin butoanele de control.

Descriere generală

Diagrama Bloc



Descriere Blocuri

- **Butoane Control** Configurarea sunetelor de pe cele 12 clape se realizeaza prin butoanele de control (in numar de 4 sau 6). Odata realizata maparea notelor pe fiecare dintre cele 12 clape, apasarea uneia dintre ele va duce la redarea sunetului respectiv.

Un obiectiv aditional ar fi combinarea diferitelor sunete la apasarea simultana a doua sau mai multe clape.

- **12 butoane** Reprezinta clapele ce redau sunetele mapate lor.
- **Modul QTouch** Folosit pentru reglarea volumului sunetelor redade de difuzor (lucru realizabil si prin butoanele de control).
- **Difuzor** Un simplu difuzor.
- **Display LCD** Display LCD text.

Hardware Design

[Notepad](#) (cu suport pentru AVR).

Biblioteci folosite

Pentru a folosi display-ul grafic am folosit biblioteca [UTFT\[3\]](#) facuta de cei de la Rinky-Dink Electronics. Biblioteca este una generica ce suporta o multitudine de controllere. Deoarece eu aveam nevoie doar de controller-ul ST7735 [2] (deoarece acesta este controllerul folosit de ecranul ITBD02-1.8SP[1]), am sters din biblioteca tot codul generic, astfel pastrand doar codul particular lui ST7735. Urmatorul pas a fost sa portez functiile/macrourele specifice Arduino (biblioteca UTFT este scrisa pentru Arduino, deci trebuie translatata acea logica pentru microprocesorul ATmega) si anume: [portOutputRegister\(\)](#), [digitalPinToPort\(\)](#), [digitalPinToBitMask\(\)](#) si [pinMode\(\)](#) (functie definita in fisierul [wiring_digital.c](#) din core-ul Arduino, si importata in [Arduino.h](#)). Aceste functii/macroure sunt trivial de tradus in cod AVR. Odata portate toate functiile initiale din biblioteca, am eliminat functiile ce nu erau utile proiectului, astfel pastrand doar functionalitatea necesara a bibliotecii, deci obtinand un fisier .hex mai mic in dimensiune.

O alta varianta pe care o puteam folosi se afla [aici](#).

Structuri de Date si Algoritmi

Proiectul nu implica algoritmi complicati, si nici structuri de date sofisticate, intrucat acestea ar aduce un overhead considerabil pentru un microprocesor precum ATmega324.

Structura de date de baza a acestui proiect o reprezinta automatul de stare ce retine ce frecvente are fiecare buton. Acesta este locul in care datele sunt stocate si de unde sunt luate, la nevoie, pentru a fi folosite. Modulul de redare sunet de aici isi ia datele si tot aici si cele 4 butoane de navigare/control scriu informatia.

Surse

Structura de baza ce incapsuleaza ecranul LCD:

```
struct utft {
    byte fcolorr,fcolorg,fcolorb; // foreground color
    byte bcolorr,bcolorg,bcolorb; // background color
    byte orient; // orientation: either LANDSCAPE or PORTRAIT
    long disp_x_size, disp_y_size; // display size
    regtype *P_RS, *P_WR, *P_CS, *P_RST, *P_SDA, *P_SCL; // pointers to
the used registers
    regsize B_RS, B_WR, B_CS, B_RST, B_SDA, B_SCL; // masks of the used
bits inside the used registers
    _current_font cfont; // fond used to print characters
};
```

Cu structura de mai sus se interactioneaza prin urmatorul API (au fost pastrate numele originale din

biblioteca UTFT[3]):

```
/* High Level Functions */
void UTFT_init(struct utft *display);
void InitLCD(struct utft *display, byte orientation);
void drawPixel(struct utft *display, int x, int y);
void drawLine(struct utft *display, int x1, int y1, int x2, int y2);
void fillScr(struct utft *display, byte r, byte g, byte b);
void clrScr(struct utft *display);
void drawRect(struct utft *display, int x1, int y1, int x2, int y2);
void fillRect(struct utft *display, int x1, int y1, int x2, int y2);
void setColor(struct utft *display, byte r, byte g, byte b);
void setBackColor(struct utft *display, byte r, byte g, byte b);
void print(struct utft *display, char *st, int x, int y);
void setFont(struct utft *display, const uint8_t* font);

/* Low Level Functions */
void LCD_Writ_Bus(struct utft *display, char VH,char VL);
void LCD_Write_COM(struct utft *display, char VL);
void LCD_Write_DATA_HL(struct utft *display, char VH,char VL);
void LCD_Write_DATA_L(struct utft *display, char VL);
void setPixel(struct utft *display, byte r,byte g,byte b);
void drawHLine(struct utft *display, int x, int y, int l);
void drawVLine(struct utft *display, int x, int y, int l);
void printChar(struct utft *display, byte c, int x, int y);
void setXY(struct utft *display, word x1, word y1, word x2, word y2);
void clrXY(struct utft *display);
```

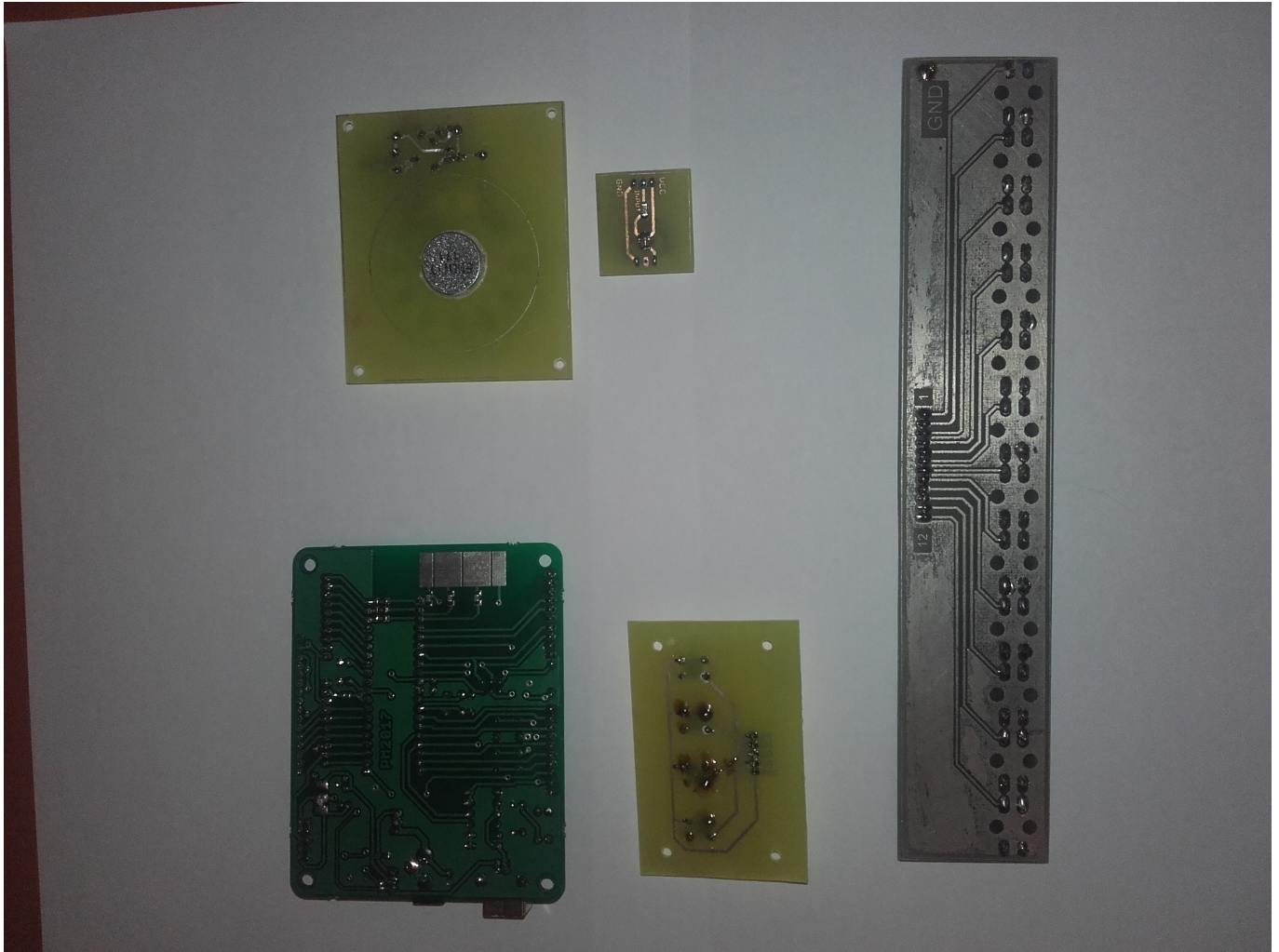
Rezultate Obținute

<imgcaption image0|Montajele sunt reprezentate de: placuta de dezvoltare formata din pachetul de baza si cele aditionale, doua difuzoare si doua tastaturi (una de control si una ce contine 12 butoane de dimensiune 12×12).>



</imgcaption>

<imgcaption image1|Pentru realiza cablajele pentru difuzoare si tastaturi am folosit placi de sticlătextolit cu cupru pe o singura parte, peste care am imprimat desenul cablajului (in prealabil printat la o imprimanta cu toner) folosind un fier de calcat. Apoi placutele au fost puse in clorura ferica pentru a coroda cuprul aflat in exces (cel neacoperit de tonerul imprimat cu fierul de calcat). Ultima etapa a fost punerea placutelor intr-o baie de staniu, etapa optionala, cu rol estetic (adauga o pelicula de staniu peste cupru, pentru a proteja cuprul de oxidare).>



</imgcaption>

<imgcaption image2|Aceasta este o poza cu montajul proiectului, aflat intr-un stadiu intermediar.>



</imgcaption>

Concluzii

În cadrul proiectului am învățat modul în care se lucrează cu deviceurile periferice (exemplu: LCD) și modul în care se structurează datele într-un program ce se vrea a fi folosit pe microcontroller. Apar probleme de dimensiune a codului și eficiența care în mod normal nu sunt întâlnite atunci când se programează pe PC.

De asemenea, am învățat mai bine utilitatea și funcțiile unor pini ai plăcuței (precum SDA, SCL și SS) și modul în care ne alegem pinii care să interacționeze cu un device.

Pe partea de software am învățat cum se ia o bibliotecă deja existentă, cum se rezolvă erorile de compilare și cum se adaptează cerințelor particulare ale unui proiect.

Download

Jurnal

23.04.2017 - Actualizare wiki: tema, descriere, schema bloc, lista de piese si resurse.
07.05.2017 - Adaugare schema electrica.
24.05.2017 - Actualizare pagina wiki (hardware design, software design, bibliografie/resurse).
25.05.2017 - Actualizare pagina wiki (schema electrica, descriere API, poze rezultate obtinute).

Bibliografie/Resurse

- Documentația în format [PDF](#)
- [0] Datasheet ATmega324A: http://cs.curs.pub.ro/wiki/pm/_media/doc8272.pdf
- [1] Datasheet Display LCD TFT ITBD02-1.8SP:
ftp://imall.iteadstudio.com/IM120419001_ITDB02_1.8SP/DS_IM120419001_ITDB02_1.8SP.pdf
- [2] Datasheet Controller Display ST7735:
ftp://imall.iteadstudio.com/IM120419001_ITDB02_1.8SP/DS_ST7735.pdf
- [3] Biblioteca UTFT: <http://www.rinkydinkelectronics.com/library.php?id=51>
- [4] https://en.wikipedia.org/wiki/Sound#Elements_of_sound_perception
- [5] [https://en.wikipedia.org/wiki/Sampling_\(signal_processing\)#Sampling_rate](https://en.wikipedia.org/wiki/Sampling_(signal_processing)#Sampling_rate)

From:

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

Permanent link:

<http://ocw.cs.pub.ro/courses/pm/prj2017/ddragomir/sintetizatorsunetminimal>



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