

Sintetizator MIDI

Introducere

Proiectul este un sintetizator cu input MIDI. Acesta va primi semnale MIDI de la un controller (in cazul meu o claviatura) si va genera unde audio pe baza notei apasate. Totodata sintetizatorul va putea reda fisiere MIDI de pe un card SD. Un ecran va afisa in timp real ce nota este redata si in ce mod de input este dispozitivul.

Descriere generală

- Clapa va trimite semnale MIDI catre USART.
- Microcontrollerul va procesa aceste semnale, activand un semnal PWM corespunzator notei primite, iar ieșirea va fi transmisa catre un difuzor sau o mufa audio.
- Un buton va fi utilizat pentru a comuta intre doua moduri de input: claviatura live sau citirea unui fisier MIDI de pe cardul SD.
- Ecranul va comunica prin protocolul I2C.
- Cititorul de carduri SD va comunica cu microcontrollerul prin SPI.
- Circuitul de iesire consta intr-un amplificator si un filtru trece-jos RC pentru a reduce zgomotul din semnalul audio si a proteja pinii de PWM.

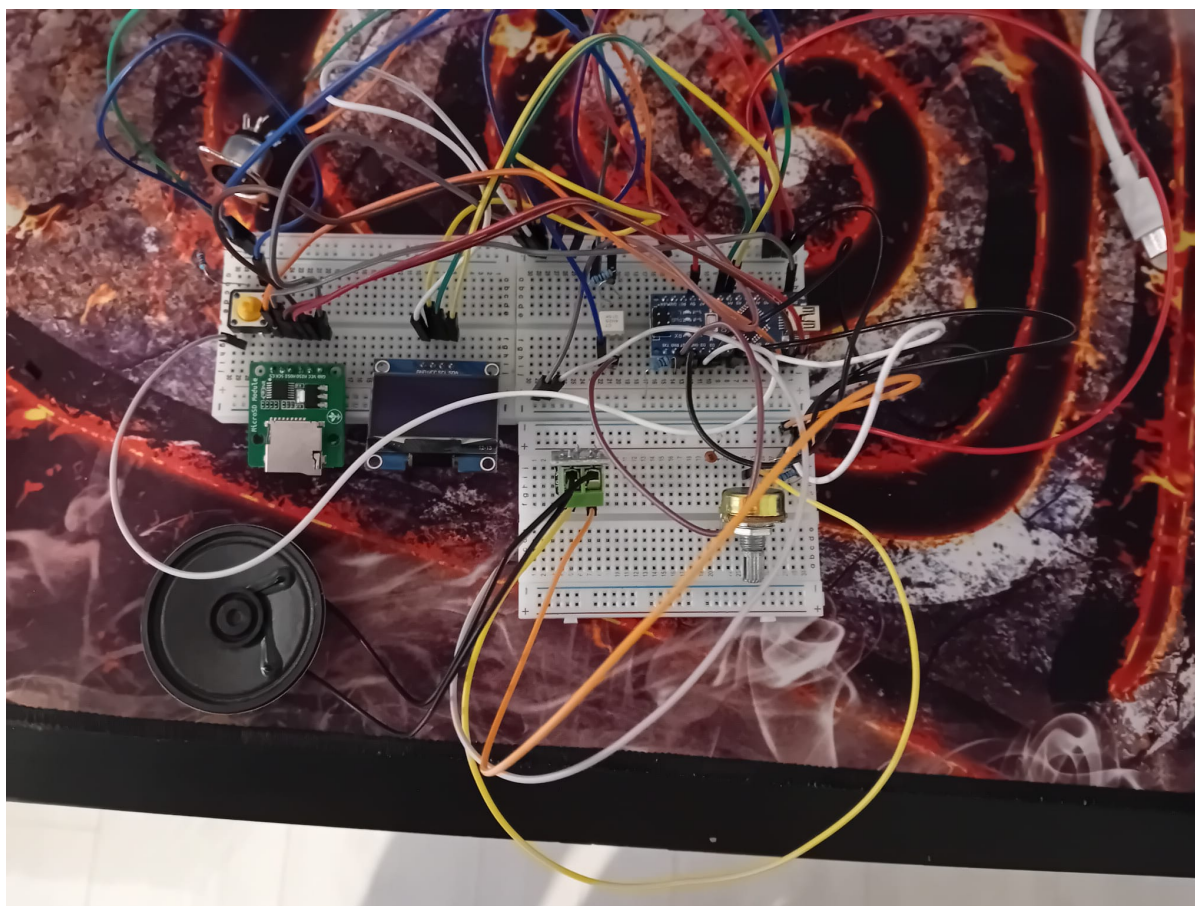


Hardware Design

Lista componente

Nr.	Componentă	Cantitate	Info
1	Arduino Nano	1	Microcontroler
2	Conector DIN 5 pini mama	1	Input pentru clapa
3	octocuplor 4n25	1	Pentru a proteja arduino-ul folosit in circuitul de input
4	dioda de protectie	1	Pentru a proteja led-ul octocuplorului
5	rezistente de pull up	2	Pentru buton si inputul midi
6	Modul micro sd	1	interfata cu sd cardul SPI
7	Ecran oled	1	I2C
8	Rezistente 10K ohmi	3	Protejeaza pinii de output PWM in caz de numai unul genereaza semnal

9	Potentiometru 100k ohmi	1	Pentru filtru trece jos
10	Condensator 10nF	1	Pentru filtru trece jos
11	Modul amplificator lm386	1	Amplifica semnalul pentru a proteja pinii de output
12	Speaker 1W	1	-



Schema Electrica



• Circuit midi input

Am folosit un octocuplor pentru a proteja pinii de intrare ai Arduino-ului. Atunci cand clapa trimite semnal LOW, pinul 5 (data line) al mufei este conectat la masa iar pinul 4 (power line) la 5V, ledul octocuplorului se aprinde. Acest lucru face ca fototransistorul din interiorul octocuplorului sa conduca, astfel pinul RX al Arduino-ului va primi un semnal de LOW. In mod similar se intampla si in cazul semnalului HIGH.

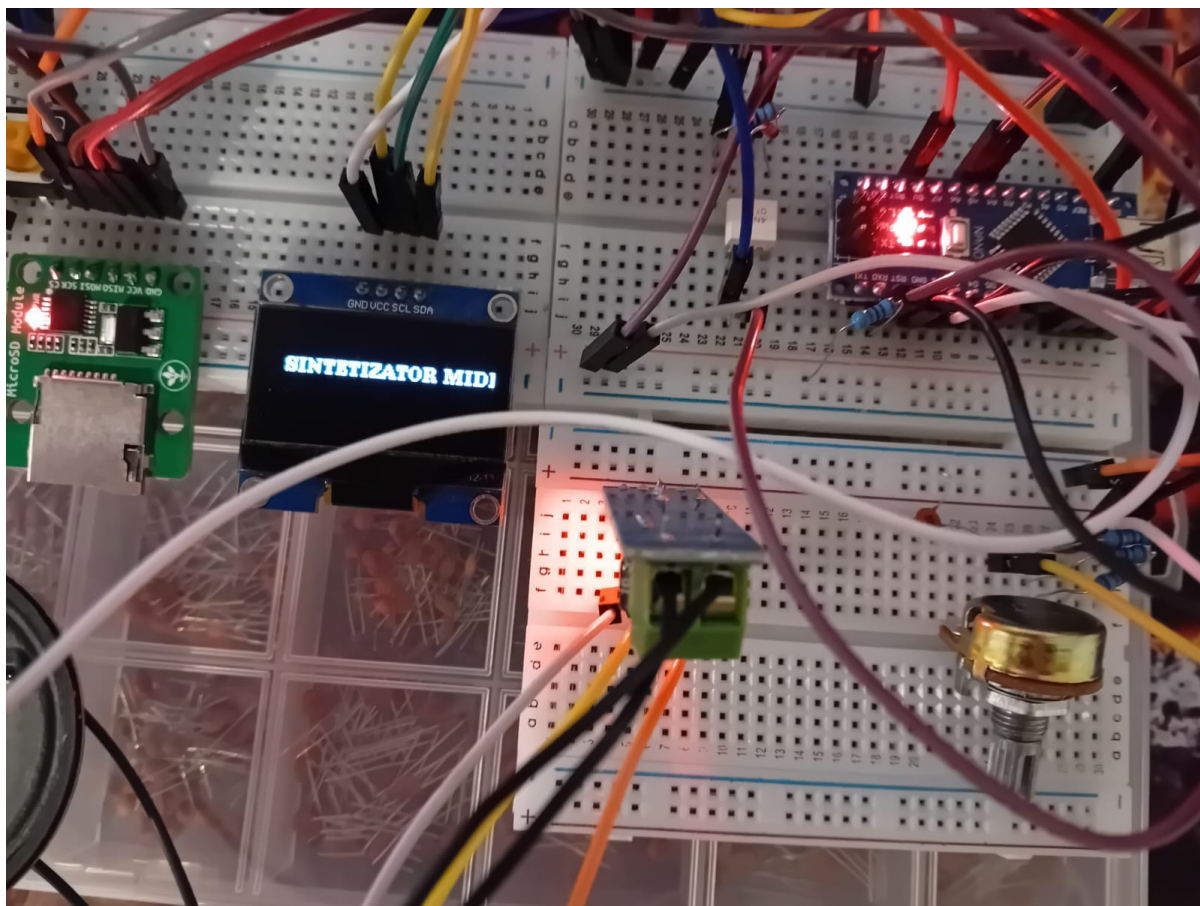
• Circuit PWM out:

Deoarece Arduino-ul are 3 timere, pot folosi 3 pini pentru a genera semnale PWM diferite. Acestia sunt conectati la un filtru trece-jos cu rezistenta variabila prin 3 rezistente de 10K ohmi pentru a proteja pinii de output. Filtrul trece-jos poate netezi o undă pătrată, transformând-o într-o undă triunghiulară și modificând astfel sunetul de la unul aspru la unul mai placut. In final, semnalul intra intr-un amplificator LM386 pentru a putea fi auzi prin difuzor si pentru a proteja pini (fiind un speaker

de 8 ohmi curentul necesar ar fi prea mare pentru pinii arduinoului fara un amplificator).

Pini Folositi

Pin Arduino	Conectat la	Descriere
5V	Vcc	Alimentare
GND	Toate GND-urile	Masă comună
D0 (RX)	Iesire Octocuplor	Input semnal midi
D1 (TX)	—	
D3	R4	Semnal PWM către difuzor
D5	R5	Semnal PWM către difuzor
D9	R6	Semnal PWM către difuzor
D2	Buton S1	Intrare digitală
D10	CS (Pin 6 MicroSD)	SPI chip select
D11	MOSI (Pin 4 MicroSD)	SPI MOSI
D12	MISO (Pin 3 MicroSD)	SPI MISO
D13	SCK (Pin 5 MicroSD)	SPI Clock
A4 (SDA)	SDA OLED	Comunicare I2C (date)
A5 (SCL)	SCL OLED	Comunicare I2C (ceas)



Dovada functionare ecran.

Aici puneți tot ce ține de hardware design:

- listă de piese
- scheme electrice (se pot lua și de pe Internet și din datasheet-uri, e.g. <http://www.captain.at/electronic-atmega16-mmc-schematic.png>)
- diagrame de semnal
- rezultatele simulării

Software Design

Am folosit Arduino IDE pentru mediul de dezvoltare.

Biblioteci folosite:

- **SdFat.h**, **MD_MIDIFile.h**:

Pentru a citi și procesa fișiere MIDI de pe cardul SD.

- **SSD1306Ascii.h**, **SSD1306AsciiWire.h**:

Pentru a afișa informații pe ecranul OLED.

Structura cod:

Sistemul are 3 stări:

1. **MANUAL** - sintetizatorul primește comenzi MIDI de la claviatura și ecranul OLED afișează informații despre notele primite.
2. **LOADING** - sintetizatorul citește fișierul MIDI de pe cardul SD și îl încarcă în memorie.
3. **PLAYING_SD** - sintetizatorul redă fișierul MIDI încărcat de pe card.

Inițializare: În Setup() se inițializează timerul pentru PWM, interruptul de pe pinul 2, cardul SD, obiectul MIDI (pentru redare din fișier MIDI) și ecranul. De asemenea, se setează starea inițială a sistemului la MANUAL.

Loop:

1. **MANUAL:** se citesc 3 bytes pe usart și funcția handleMidi() setează frecvența timerului și afișează informațiile pe ecran.
2. **LOADING:** se citește fișierul MIDI de pe cardul SD și se încarcă în obiectul MIDI, după programul trece în starea PLAYING_SD automat.
3. **PLAYING_SD:** se apelează funcția getNextEvent() pentru a reda următorul eveniment MIDI din fișierul încărcat. Astfel se simulează primirea de comenzi MIDI de la claviatura.

Trecerea între stări se face prin apăsarea butonului de pe pinul 2, care declanșează un interrupt. Aceasta trece de la starea MANUAL la LOADING sau de la starea LOADING, PLAYING_SD la MANUAL. Este implementată și debouncing pentru a evita treceri multiple între stări la o singură apăsare a butonului. Totodată la apăsarea de 2 ori a butonului sistemul trece în **teaching mode** atunci când redă fișiere, reducând viteza de redare a fișierului și afișând informații despre note pe ecran.

Funcția **showNoteName()** calculează numele și octava notei primite și le afișează pe ecran.

Notiuni din laborator aplicate:

- PWM: pentru a genera semnalul audio.
- Interrupt: pentru a schimba starea sistemului la apăsarea butonului.
- USART: pentru a primi comenzi de la clapa.

[cod sursa demo](#)

Descrierea codului aplicației (firmware):

- mediu de dezvoltare (if any) (e.g. AVR Studio, CodeVisionAVR)
- librării și surse 3rd-party (e.g. Procyon AVRlib)
- algoritmi și structuri pe care plănuieți să le implementați
- (etapa 3) surse și funcții implementate

Rezultate Obținute

Care au fost rezultatele obținute în urma realizării proiectului vostru.

Concluzii

Deși nu am putut implementa de la 0 citirea fișierelor MIDI, proiectul m-a ajutat să învăț mai mult despre acest protocol și despre cum funcționează clapa digitală. Am învățat cum sunt interpretate comenzile MIDI și cum acestea pot fi folosite pentru a genera sunete în timp real. De asemenea, am avut ocazia să experimentez cu un filtru trece-jos RC și să reușesc să trec peste bug-uri hardware.

Download

O arhivă (sau mai multe dacă este cazul) cu fișierele obținute în urma realizării proiectului: surse, scheme, etc. Un fișier README, un ChangeLog, un script de compilare și copiere automată pe uC creează întotdeauna o impresie bună 😊.

Fișierele se încarcă pe wiki folosind facilitatea **Add Images or other files**. Namespace-ul în care se încarcă fișierele este de tipul **:pm:prj20??:c?** sau **:pm:prj20??:c?:nume_student** (dacă este cazul).
Exemplu: Dumitru Alin, 331CC → **:pm:prj2009:cc:dumitru_alin**.

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** și **Resurse Hardware**.

[Export to PDF](#)

From:

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

Permanent link:

<http://ocw.cs.pub.ro/courses/pm/prj2025/avaduva/andrei.diaconu1410>



Last update: **2025/05/28 12:22**