Sistem audio inteligent Ardu-D2 - Andrei Ionescu

Introducere

Prezentarea proiectului:

- **ce face**: In functie de directia si viteza cu care este miscat modulul scoate sunete specifice de pe un card SD
- **care este scopul lui**:Acest modul a fost gandit pentru 2 aplicatii diferite care ambele au la baza un singur scop, sa bucure copiii de toate varstele. Acest modul poate fi folosit atat pentru o tamburina inteligenta care in functie de cum este miscata scoate sunete diverse pentru a putea imbunatati creativitatea micutilor. Pentru copiii ceva mai mari, se pot schimba sunetele de pe cardul SD si se poate transforma intr-un modul pe care il pui pe o sabie obisnuita si deodata devine o sabie laser cu sunete care sa ii faca pe ceilalti copii din parc invidiosi
- **care a fost ideea de la care aţi pornit**: Voiam sa fac un instrument musical si dupa o discutie cu laborantul am ajuns la aceasta idee. Pentru design-ul hardware m-am inspirit putin din sabia de pe site-ul https://www.instructables.com/Arduino-Based-Lightsaber-With-Light-and-Sound-Effe/
- **de ce credeţi că este util pentru alţii şi pentru voi**: Acest proiect consider ca are o versatilitate destul de mare si se poate transforma dintr-o jucarie in alta relative usor. Este un device care poate lua jucarii banale si sa le transforme in ceva cu adevarat special. Pentru mine este util sa-mi aduca aminte de copilul interior pentru ca sa fim seriosi, cine nu ar vrea sa retraiasca momentele frumoase ale copilariei cand lumea era doar un joc.

Descriere generală

La Arduino se leaga accelerometrul care va comunica prin I2C datele obtinute la un anumit interval de timp
bine stabilit. Tot la Arduino avem conectat si modulul cu cardul SD care comunica prin intermediul SPI. Pe card in prealabil am descarcat niste sunete/melodii, in functie de aplicatia dorita. Atunci cand primim
informatia de la accelerometru, in functie de aceasta alegem ori un sunet specific de pe card, ori un sunet
random dintr-un pool ales de dinainte (ca jucaria sa para mai complexa si sa nu se plictiseasca copiii). Odata ce avem sunetul dorit de pe card (in format WAV), il trimitem catre speaker care ii da play.

Schema bloc pentru componente

Cum functioneaza

Jucaria are 3 moduri in care se poate afla: IDLE, TAMBOURINE si LIGHTSABER. Atunci cand deschidem pentru prima data jucaria, aceasta va prezenta singura modul de functionare. Dupa finalul mesajului vocal, se va alege o melodie random dintr-un pool de 20 de cantece. Apoi, atata timp cat se afla in modul IDLE, jucaria va da play dupa fiecare melodie la una noua imediat cum s-a terminat precedenta.

In modul TAMBOURINE poti sa ajungi din celelalte 2 moduri miscand de 3 ori consecutiv jucaria. Daca insa o misti de 3 ori atunci cand este in acest mod, atunci se trece in starea de IDLE. In modul de tamburina se extrag de la giroscop date despre cum este miscata jucaria. In functie de miscarea cea mai puternica se poate da play la unul dintre cele 6 sunete din setul curent. Sunt 6 sunete, corespunzator fiecarei directii de rotiri (pozitive sau negative) pe cele 3 axe OX, OY si OZ. Se mai poate da play si la un al 7-lea sunet in caz ca jucaria este lovita. Jucaria are 5 set-uri de cate 7 sunete din care se alege unul la intamplare atunci cand se face tranzitia in modul de tamburina.

In modul LIGHTSABER poti sa ajungi din celelalte 2 moduri lovind de 3 ori la rand jucaria. Daca insa o lovesti de 3 ori atunci cand este deja in acest mod, atunci trece in modul de IDLE. In modul de sabie laser avem mai multe sunete pentru miscari lente si miscari rapide care sunt alese la intamplare, in functie de viteza cu care este miscata sabia. Daca se detecteaza o lovitura, in functie de puterea impactului se alege un sunet random dintr-un pool prestabilit.

Jucaria are un buton de lock care o data ce este apasat blocheaza starea jucariei in starea curenta si nu se mai face schimbarea indiferent de cum este miscata jucaria. Pentru a putea schimba din nou modul de functionare, trebuie sa se mai apese o data pe buton pentru deblocare. Pe jucarie mai exista si butonul de ON/OFF, iar dupa ce inchizi jucaria, aceasta va retine modul de functionare in care era si va incepe cu acesta, dar va debloca automat jucaria, asa ca daca se doreste sa nu se mai poata schimba modul, atunci trebuie apasat din nou butonul de lock.

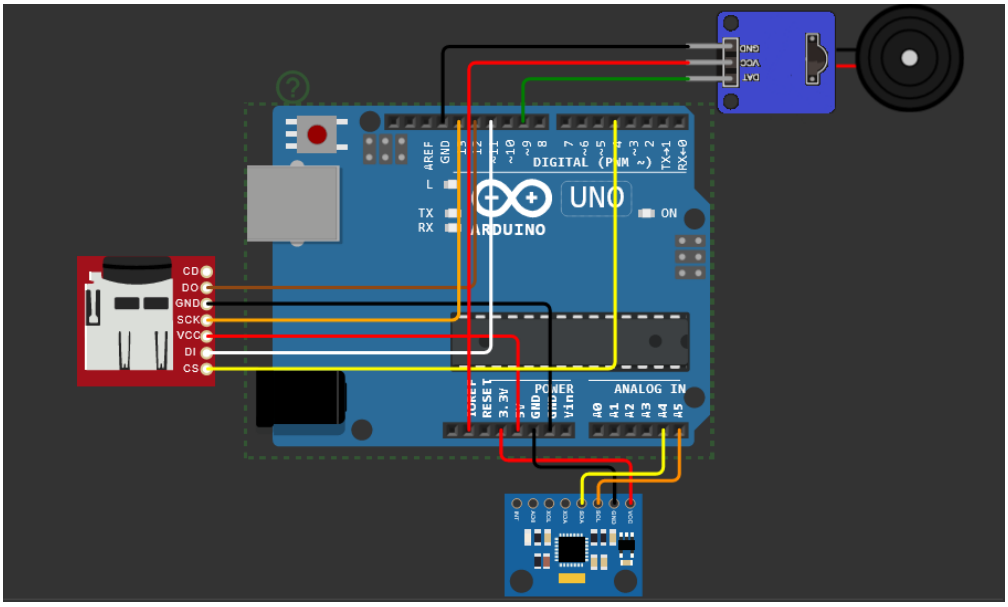
Schema bloc pentru functionalitate

Hardware Design

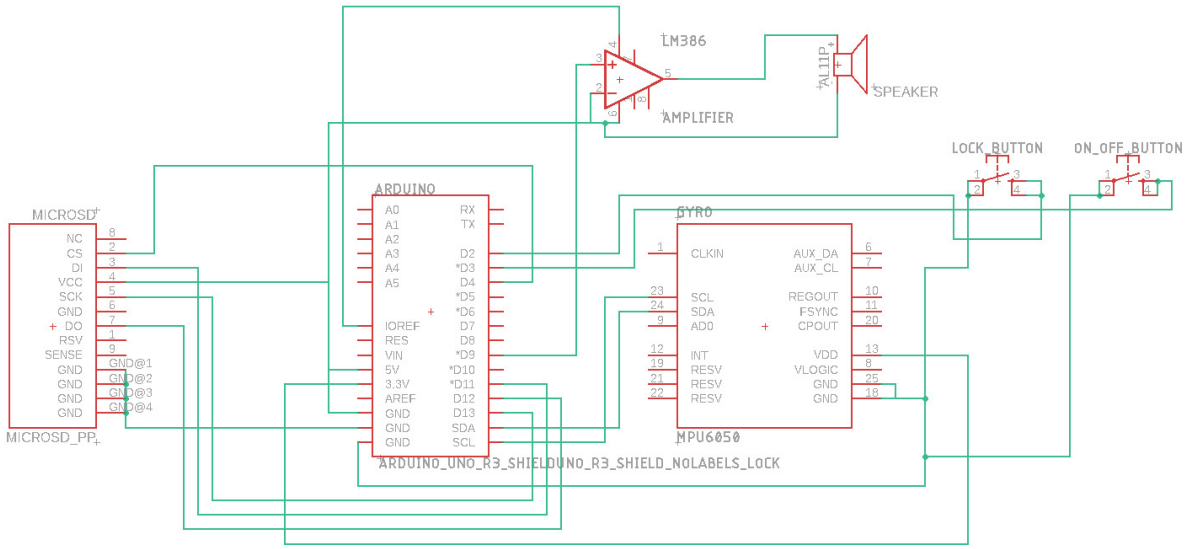
Lista piese:

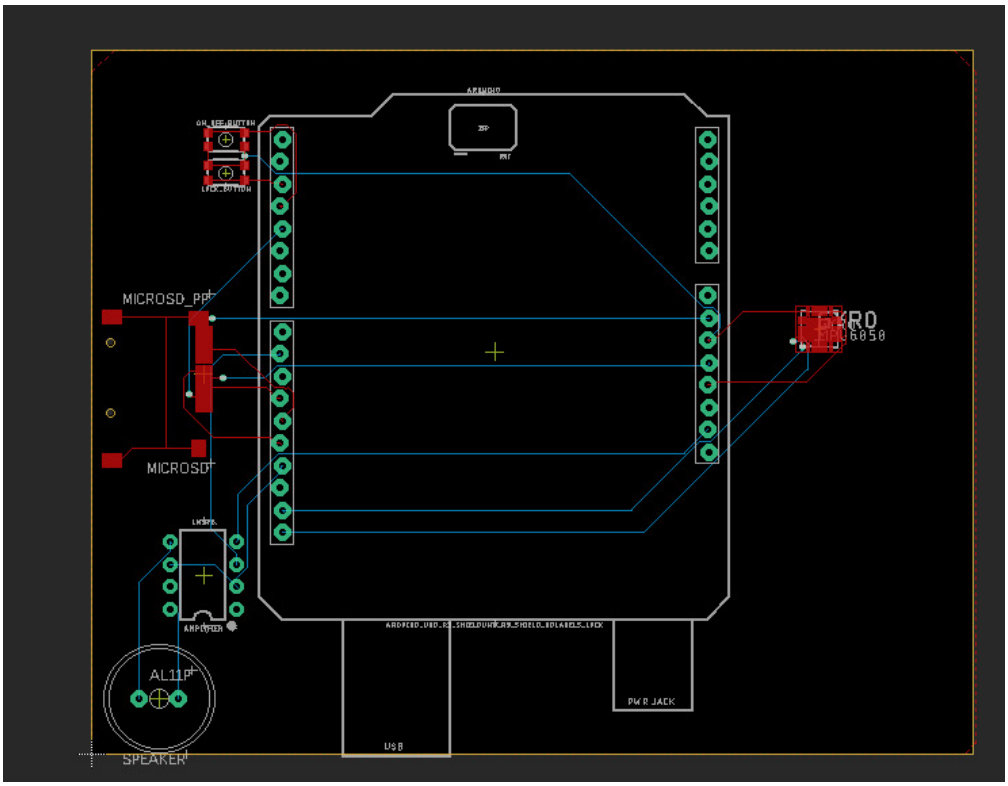
- Arduino Uno R3 ATmega328P
- Modul adaptor card micro sd
- Card micro SD
- Accelerometru si giroscop MPU6050
- Speaker 8ohmi de 1W
- Amplificator LM386
- Fire de legatura

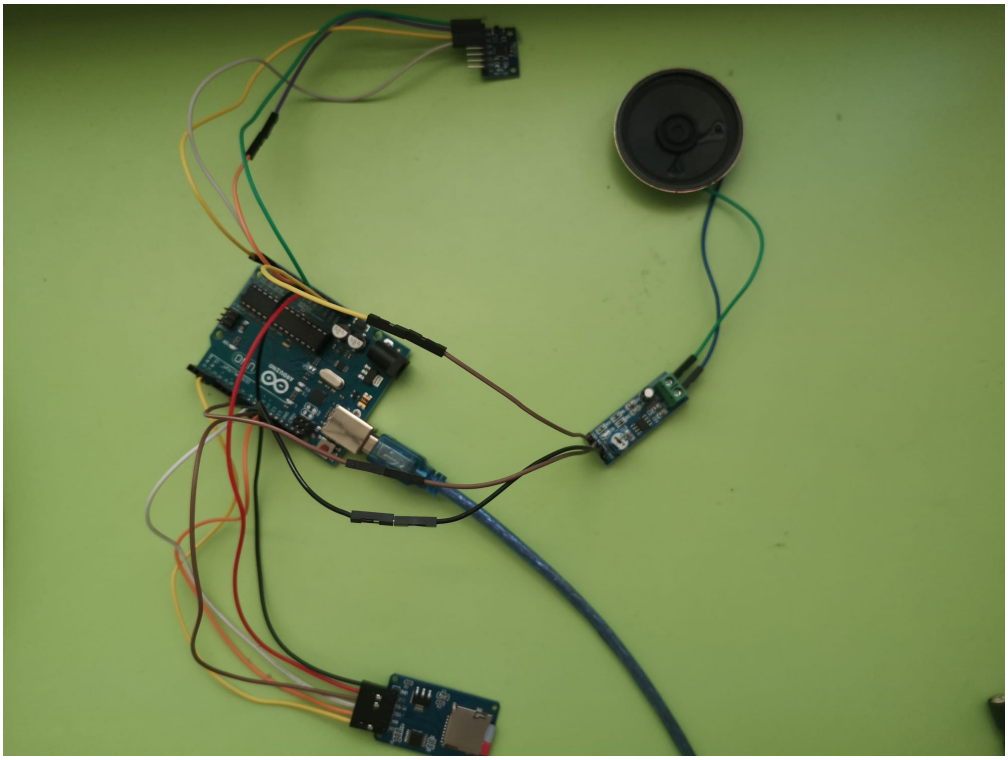
Schema circuitului:



Schema electrica:


LM386
AMPLIFIER
AL11P
SPEAKER
LOCK_BUTTON
ON_OFF_BUTTON
MICROSD
MICROSD_PP
ARDUINO
GYRO
MPU6050
ARDUINO_UNO_R3_SHIELDUNO_R3_SHIELD_NOLABELS_LOCK


MICROSD_PP
MICROSD
AL11P
SPEAKER
USB
PWR JACK



Software Design

Descrierea codului aplicaţiei (firmware):

- **Mediu de dezvoltare**: Arduino IDE
- **Biblioteci incluse**: SD.h, SPI.h, TMRpcm.h, MPU6050.h, I2Cdev.h, Wire.h
- **Descrierea modulelor software**:

ISR(TIMER0_COMPA_vect): Aici am codul de intrerupere pentru Timer-ul 0. Acesta este setat la o frecventa de 1kHz si numara 10ms pentru ca citesc datele de pe giroscop o data la 10ms. In plus, il folosesc si pentru debouncing-ul butonului de lock.

ISR(TIMER2_COMPA_vect): Aici am codul de intrerupere pentru Timer-ul 2. Acesta este setat la o frecventa de 1kHz si cronometreaza durata sunetului current pentru a stii cand sa dea play la urmatorul. In plus, functia este folosita si pentru a ajuta cu debouncing-ul butonului de on-off.

ISR(INT0_vect): Intreruperea INT0 este folosita pentru butonul de lock. Atunci cand acesta este apasat, blocam sau deblocam posibilitatea de schimbare dinamica a modului de functionare a jucariei.

ISR(INT1_vect): Intreruperea INT1 este folosita pentru butonul de on-off care inchide sau deschide jucaria.

- `void configure_mpu_timer()`: Configureaza registrele pentru Timer0

- `void configure_music_timer()`: Configureaza registrele pentru Timer1
- `void configure_buttons()`: Configureaza registrele pentru a putea folosi butoanele de lock si de on-off
- `void setup()`: In functia de setup initializam toti parametri si configuram timerele si butoanele.
- `int16_t compute_biggest_motion(int16_t gyr_x, int16_t gyr_y, int16_t gyr_z)`: Calculeaza in ce directie a fost facuta cea mai ampla miscare si intoarce un numar de la 1 la 6 care indica directia.
- `void read_from_mpu()`: Functia citeste datele oferite de giroscop
- `void tambourine_swing()`: In functie de directia si viteza miscarii se da play la sunetul de pe tamburina corespunzator.
- `void play_strike()`: MPU6050 are integrat si un accelerometru pe care il folosesc sa vad daca jucaria a fost lovita. Daca acceleratia totala depaseste un anumit threshold, inseamna ca s-a detectat o lovitura. In acest caz (daca nu este lock), incrementam contorul de lovituri si daca jucaria nu este in modul IDLE, dam play la sunetul corespunzator.
- `void play_swing()`: Daca nu s-a detectat o lovitura, dar s-a detectat o miscare, atunci incrementam contorul de swing-uri si dam play la sunetul corespunzator.
- `void idle_finished_music()`, `void lightsaber_finished_music()`, `void tambourine_finished_music()`: Cele 3 functii sunt apelate in functie de modul in care se afla jucaria atunci cand s-a terminat sunetul anterior. In caz ca suntem in modul tamburina, atunci nu se da play la nimic, altfel se pune o melodie prestabilita.
- `void set_mode_to_idle()`, `void set_mode_to_lightsaber()`, `void set_mode_to_tambourine()`: Atunci cand se atinge threshold-ul de 3 miscari/3 lovituri si jucaria nu este “locked”, atunci se trece dintr-o stare in alta. In functie de starea in care se trece, se apeleaza una dintre aceste functii care initializeaza toti parametri necesari respectivului mod si reseteaza contoarele de miscari.
- `void switch_on()`: Functia seteaza modul current sa fie ultimul mod inainte de inchidere.
- `void switch_off()`: Functia reinitializeaza toti parametri si opreste melodia curenta, daca vreuna este cantata in acel moment.
- `void loop()`: Functia loop pune totul cap la cap. Se verifica daca jucaria este deschisa, si daca da, se verifica apoi daca a fost apasat butonul de lock pentru a bloca sau debloca jucaria, pentru a face modificarile necesare. Apoi se verifica daca numarul de miscari sau de lovituri a trecut de threshold-ul de 3, caz in care in functie de starea curenta seteaza urmatoarea stare a jucariei. In continuare se verifica daca melodia care era cantata s-a terminat, si daca da, atunci in functie de mod ori se va pune o melodie dintr-un pool (IDLE), un sunet standard de “bazait” (sabie laser), sau nu se aude nimic in cazul tamburinei. Nu in ultimul rand, se verifica daca trebuie sa se citeasca datele de la giroscop si in functie de acestea se pune sau nu o melodie noua.

Rezultate Obţinute

Concluzii

Pentru acest proiect am realizat cat de importanta este legatura hardware-software si ca software-ul poate sa fie perfect scris, dar daca hardware-ul nu functioneaza, atunci e degeaba. Am invatat aceasta lectie dupa ce doar 2 componente din tot proiectul au functionat din prima, restul venind stricate de la magazin. A trebuit sa testez foarte mult hardware-ul sa ma asigur ca functioneaza corect inainte de a incepe sa scriu codul propriu zis pentru proiect.

Cea mai importanta componenta pentru proiect a fost biblioteca TMRpcm.h pentru ca era singura pe care am gasit-o sa dea play la melodii pe un speaker si nu un buzzer. Fiind insa singura, marea majoritate a problemelor au fost cauzate de ea. Am invatat ca trebuie sa fiu flexibil in dezvoltarea programelor si sa fiu deschis la schimbari rapide. Initial foloseam alta biblioteca pentru a ma ajuta sa citesc date de pe giroscop, dar a trebuit sa modific si sa caut altceva pentru ca nu functiona impreuna cu biblioteca pentru sunete. In plus, biblioteca TMRpcm.h poate sa dea play la melodii doar daca sunt de tip .wav, cu frecventa 16kHz Mono si encoding de tipul unsigned 8-bit PCM, ceea ce este o forma foarte specifica de fisier, asa ca a trebuit sa iau manual toate cele 82 de melodii si sa le convertesc folosind Audacity la tipul dorit.

Cea mai importanta lectie invatata a fost ca pentru aplicatiile embedded trebuie sa facem management de memorie, mult, foarte mult. Probabil pentru acest proiect am intalnit cel mai ciudat bug din viata mea. Biblioteca TMRpcm.h consuma multa memorie din cei 2K ai Arduino, dar mai are particularitatea ca daca folosesti mai mult de 80% din memorie, nu mai functioneaza. Am petrecut multe ore la debug aici pentru ca nu intelegeam de ce dintr-odata nu mai functioneaza speaker-ul. Dupa mult chin si jale m-am prins de problema asa ca a trebuit sa introduce toate numele de fisiere audio in memoria Flash si sa am mare grija la cum gestionez memoria.

Download

- Arhiva cu schemele electrice : pm_eagle.zip
- Arhiva cu melodiile : music.zip
- Arhiva cu codul arduino : proiect_pm.zip

Bibliografie/Resurse

Resurse Hardware

https://www.instructables.com/Arduino-Based-Lightsaber-With-Light-and-Sound-Effe/

https://www.instructables.com/Audio-Player-Using-Arduino-With-Micro-SD-Card/

https://wokwi.com/projects/305936654686749250

Resurse Software

https://github.com/AlexGyver/EnglishProjects/tree/master/GyverSaber

https://github.com/TMRh20/TMRpcm/wiki

https://github.com/ElectronicCats/mpu6050

Export to PDF