Wireless Presenter

Introducere

Wireless Presenter se dorește a fi exact ceea ce sugerează și numele: o telecomandă wireless ce poate interacționa cu software de susținere a prezentărilor (Google Slides, OpenOffice Impress, Microsoft Office PowerPoint, Keynote etc.) într-o manieră simplă și cu un cost al materialelor redus. Acesta va putea să realizeze acțiunile uzuale de control ale prezentărilor (next slide, previous slide, start slideshow), cât și mișcarea cursorului sistemului de calcul ce susține prezentarea. Suplimentar, telecomanda va avea un pointer cu care prezentatorul să poată indica audienței diferite elemente de pe slide-uri.

Ideea de la care am pornit pentru acest proiect a fost nevoia de a susține prezentări. Deoarece a cumpăra o soluție comercială nu avea sens din punct de vedere economic pentru mine la acel moment, am creat o versiune proprie a unui presenter, folosind un Arduino, un senzor infraroșu și o telecomandă de la un radio vechi, ce interacționau împreună cu un software de tip daemon, ce primea date prin USB de la Arduino și le traducea în apăsări de taste pe sistemul țintă.

Utilitatea abordării pe care o folosesc acum, spre deosebire de cea anterioară, este că, prin crearea propriului hardware al telecomenzii, pot fi adăugate/modificate funcționalități după bunul plac, astfel încât telecomanda ar putea deveni folositoare în mai multe scenarii, nu doar în susținerea prezentărilor.

Descriere generală

Schema bloc

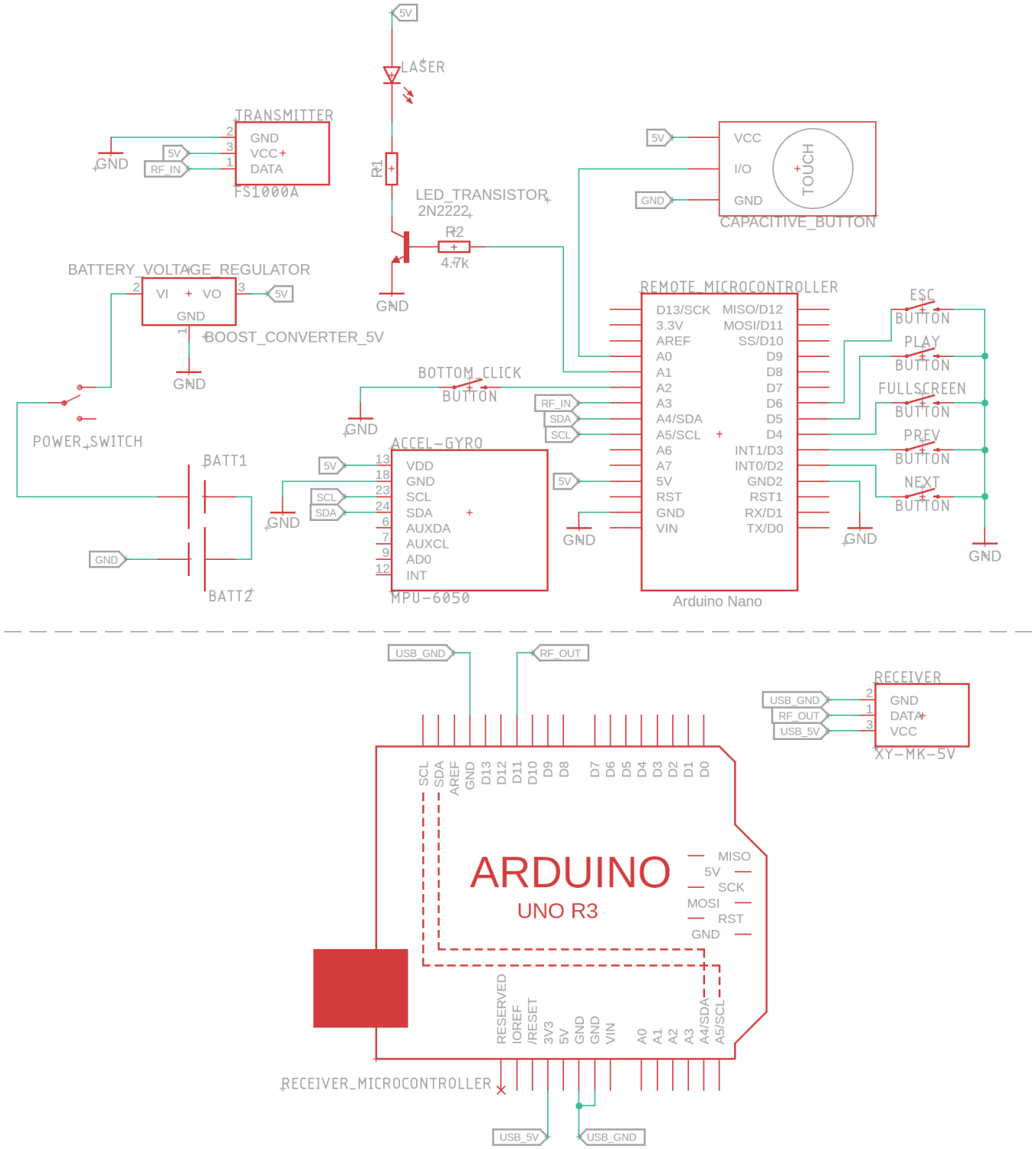
Hardware Design

Listă piese

- Arduino Nano (microcontroller telecomandă)
- Arduino Uno (microcontroller receiver)
- Modul accelerometru + giroscop MPU6050

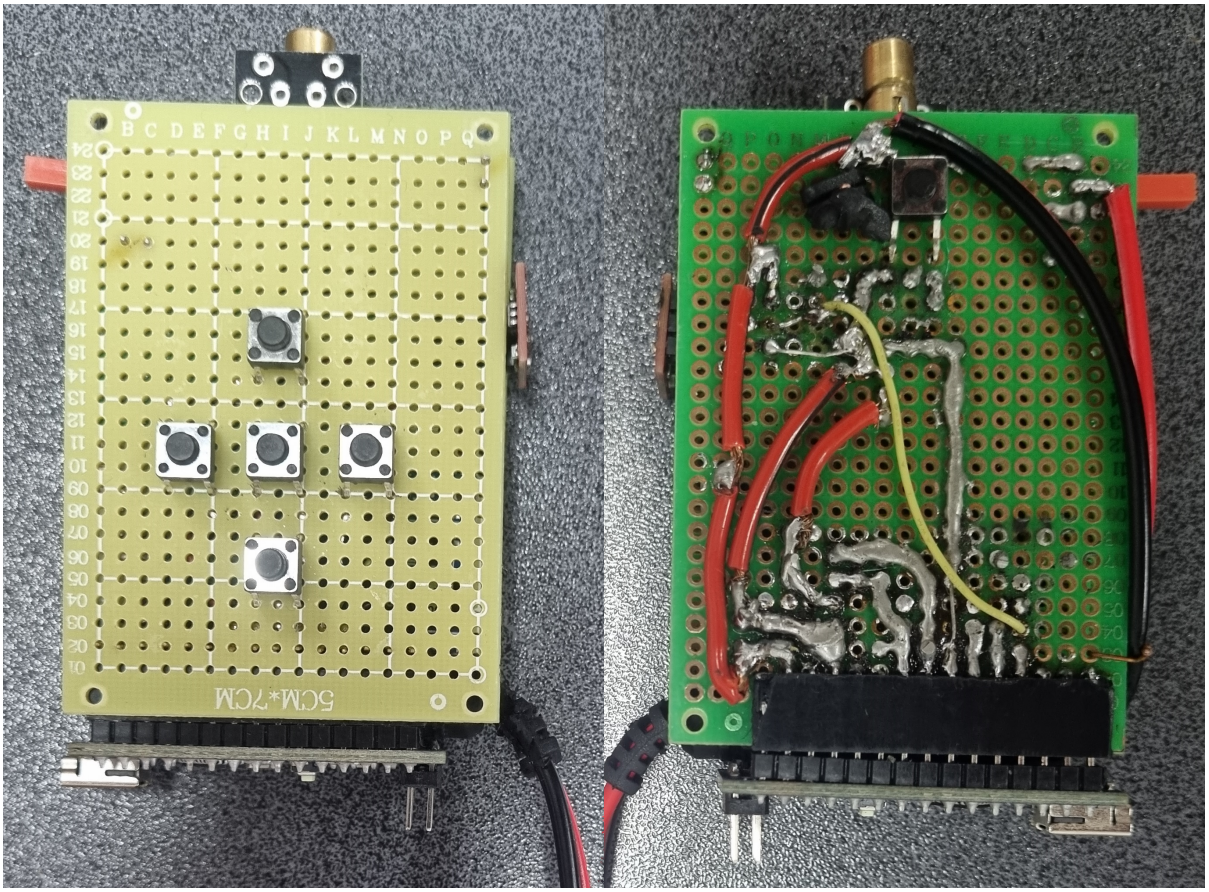
- Transmițător RF 433MHz FS1000A
- Receptor RF 433MHz XY-MK-5V
- Butoane
- Senzor capacitiv TTP223
- Laser KY-008
- Suport baterie
- Boost Converter 5V

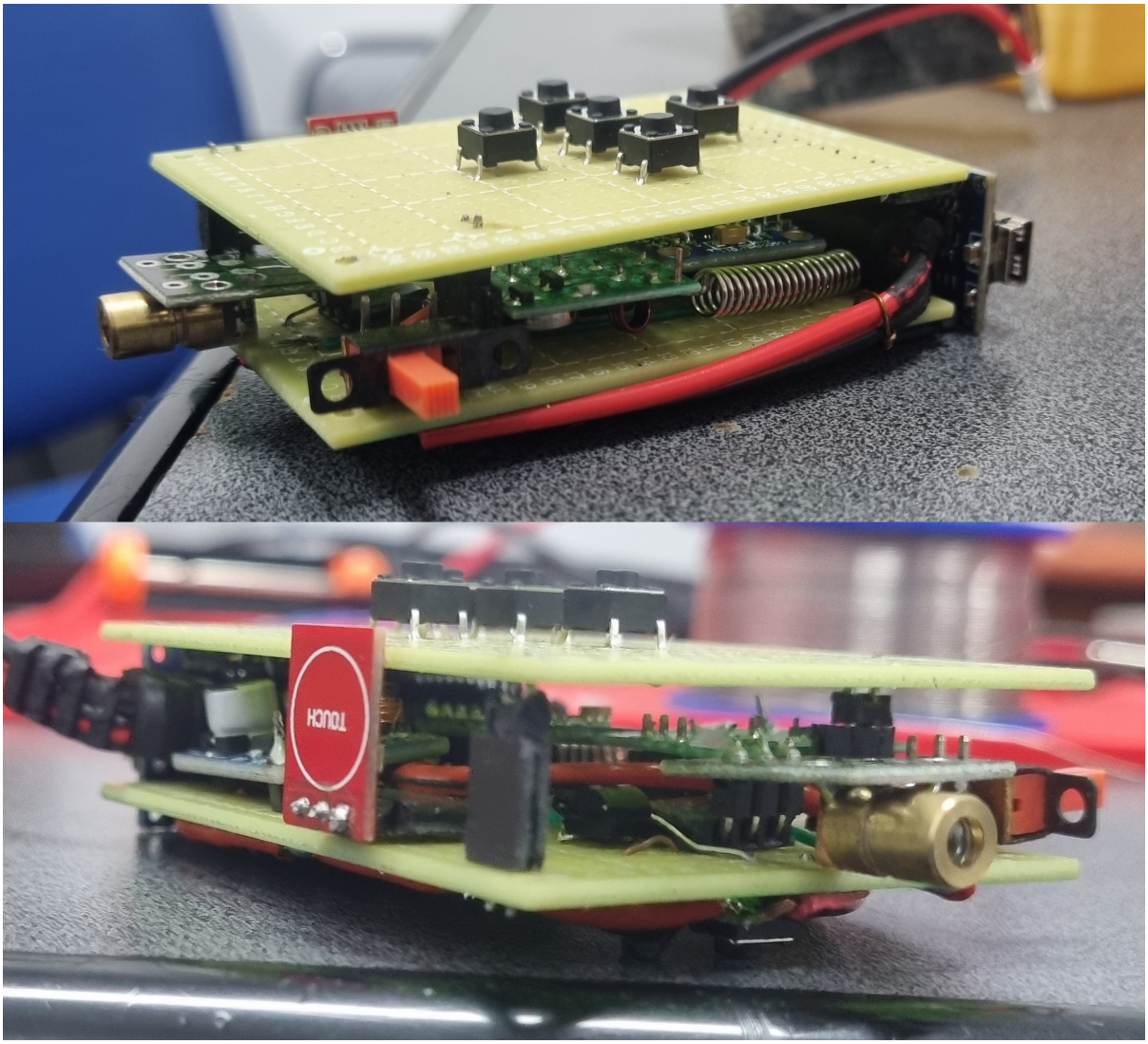
Schema electrică



Detaliere hardware

Încă de la începutul implementării hardware a telecomenzii, design-ul final pe care l-am avut în minte a fost unul bazat pe împărțirea în două categorii a componentelor proiectului: cele ce au nevoie de alimentare cu 5V și/sau să fie în partea inferioară a telecomenzii și celelalte. Astfel, pot grupa toate componentele din prima categorie în partea inferioară. În practică, pe partea superioară au rămas doar butoanele de control, care pot fi utilizate fără alimentare prin utilizarea rezistențelor de pull-up.





Componentele și utilizarea lor:

- MPU6050
  - **Scop:** detectează mișcarea mâinii utilizatorului pentru a o traduce în mișcarea cursorului pe sistemul de calcul
  - **Conectare:** fiind un modul ce folosește protocolul de comunicare I²C, am folosit pinii A4 (PC4/SDA) și A5 (PC5/SCL) specifici acestui protocol
- FS1000A
  - **Scop:** trimite comenzi către receptorul conectat la sistemul de calcul
  - **Conectare:** din motive de simplitate a organizării hardware a componentelor, am ales pinul A3 (PC3); biblioteca utilizată pentru comunicarea wireless poate fi configurată să folosească orice pin de GPIO
- XY-MK-5V
  - **Scop:** recepționează datele transmise de telecomandă
  - **Conectare:** pinul D11 (PB3) este cel default al bibliotecii pentru recepționarea de date și nu am avut niciun motiv să îl schimb
- Butoanele superioare
  - **Scop:** fiecare buton reprezintă o comandă pe care telecomanda o poate transmite sistemului de calcul
  - **Conectare:** întrucât numărul de butoane este foarte mic, am ales să folosesc pinii D2 - D6 (PD2 -

PD6), deoarece, fiind pe același port, pot activa întreruperile de schimbare ale pinilor doar pe acest port, putând astfel să detectez în mod rapid schimbarea stării butoanelor fără a recurge la polling; suplimentar, am ales ca butoanele PREV și NEXT să fie conectate la pinii D2 și D3 (PD2/INT0 și PD3/INT1), deoarece acestea vor fi folosite cel mai mult în cadrul prezentărilor, așa că vrem ca ele să aibă prioritate față de celelalte butoane din punct de vedere al întreruperilor

- Butonul inferior
  - **Scop:** în funcție de starea telecomenzii, acesta poate controla ori pointerul laser de pe telecomandă, ori transmiterea unui click al cursorului sistemului de calcul
  - **Conectare:** am ales pinul A2 (PC2) doar din considerente de rutare ale cablurilor, orice pin de pe această parte a Nano-ului ce nu are deja o utilizare consacrată poate fi utilizat la fel de bine
- TTP223
  - **Scop:** detectarea degetului utilizatorului, pentru a comuta între modul normal de funcționare al telecomenzii și cel de control al cursorului: în prezența degetului, telecomanda transmite constant mișcarea telecomenzii pentru a mișca cursorul, iar butonul inferior controlează click-ul, iar în absența degetului, butonul inferior controlează laserul
  - **Conectare:** fiind echivalentul unui buton, din motive similare cu butonul inferior, am ales pinul A0 (PC0) pentru acest senzor
- KY-008
  - **Scop:** proiectează un punct roșu laser, asemeni telecomenzilor de prezentare tradiționale, pentru a indica detalii pe slide-urile prezentării
  - **Conectare:** similar cu butonul inferior, orice pin GPIO putea fi folosit pentru laser; singurul rămas neutilizat pe partea inferioară a Nano-ului (partea stângă în schema electrică) a fost A1 (PC1), așa că pe acesta l-am utilizat
- Boost Converter 5V
  - **Scop:** ridică tensiunea bateriilor la 5V, tensiunea nominală a componentelor utilizate
  - **Conectare:** pinii de GND și 5V sunt comuni cu cei ai tuturor celorlalte componente, iar pinul de intrare a tensiunii este conectat, printr-un switch, la terminalul pozitiv al bateriilor; terminalul negativ este conectat tot la GND

Software Design

GitHub: https://github.com/Alexander1752/wireless_presenter

Am dezvoltat proiectul în Visual Studio Code cu ajutorul PlatformIO, la fel ca în cadrul laboratoarelor. Codul de pe microcontrollere este scris aproape complet folosind regiștrii și laboratoarele relevante, întrucât implementările din acestea sunt foarte optimizate, ceea ce ajută în economisirea ciclilor de ceas și a energiei consumate din baterie. Cu toate acestea, am folosit și câteva biblioteci în proiect, acestea fiind:

- VirtualWire - utilizată pentru transmisia comenzilor de la telecomandă și recepționarea lor de către receiver
- AVR Sleep - bibliotecă ce oferă apeluri simple pentru punerea în repaus a microcontrollerului, cu scopul de a economisi energie; dezactivarea perifericelor este, totuși, realizată prin interfațarea

directă cu registrul PRR (power reduction register)
- AVR WDT - folosită exclusiv pentru dezactivarea Watch Dog Timer, cu scopul de a economisi energie suplimentară

Motivul alegerii acestor biblioteci este pentru că sunt simple și eficiente: cele oferite de AVR merg până la a implementa funcțiile în inline assembly, iar VirtualWire conține minimul necesar pentru a putea realiza transmisia cu o oarecare rezistență la erori rapid, fără a consuma cicli pe microcontroller inutil (alternativa ar fi fost biblioteca RadioHead, care are funcționalități suplimentare pentru asigurarea transmiterii mesajelor, dar aduce și complexitate suplimentară, pe care am dorit să o evit cât mai mult posibil).

Citirea butoanelor se realizează prin întreruperi de schimbare a pinilor la care butoanele sunt conectate, folosind rezistențele de pull-up interne ale microcontrollerului. Pentru a realiza debouncing pentru butoane, am implementat versiunea proprie a funcției millis(), similar cu varianta din laborator, folosind timer-ul 2, singurul care se poate utiliza ca sursă de întrerupere în modurile de sleep ale procesorului. Întrucât evitarea debouncing-ului nu are nevoie de o detecție foarte precisă a timpului trecut de la ultimul eveniment, am ales să generez o întrerupere odată la fiecare 4 milisecunde, astfel că voi scoate microcontrollerul din sleep de 4 ori mai rar, fără a dăuna preciziei butoanelor, ceea ce ajută la minimizarea consumului.

Cât despre modulul accelerometru-giroscop, am creat o micro-bibliotecă proprie de comunicare cu acesta, folosind datasheet-ul componentei. Aceasta implementează doar operațiile cele mai simple de configurare și de citire a datelor, întrucât modulul are multiple funcționalități avansate de care nu am avut nevoie în cadrul acestui proiect.

În ceea ce privește structurile implementate, am creat doar structura (practic clasa) Button, care memorează ultima oară când acesta a fost apăsat (pentru debouncing), memorează dacă a avut loc un eveniment cu acel buton și memorează codul (caracterul) ce se trimite către receiver când acel buton este apăsat.

Interfațarea dintre computer și receiver este realizată prin Serial over USB; pentru aceasta am implementat un software daemon în Python care să recepționeze comenzile și să le traducă în apeluri de sistem. Astfel, am utilizat următoarele biblioteci:

- PySerial - utilizată pentru a citi comenzile de la receiver
- PySimpleGUI - oferă interfața grafică minimalistă a aplicației
- pynput - simulează evenimente de apăsări de taste și de mișcare a cursorului pentru a controla aplicația de prezentări
- threading, os, logging, queue, subprocess, math - biblioteci de bază Python, utilizate în interfațarea celorlalte și realizarea funcționalității într-un tot unitar.

Rezultate Obţinute

Telecomanda rezultată este surprinzător de ușoară și are dimensiuni rezonabile, fiind capabilă să țină o prezentare la o distanță respectabilă de receptorul conectat la calculator. Deși ceva mai greu de controlat, controlul cursorului folosind giroscopul este destul de intuitiv după o scurtă perioadă de acomodare cu această schemă de control. Butoanele sunt tactile, ceea ce ajută la a avea încredere că

acestea au fost apăsate, iar orientarea lor le face ușor de găsit și apăsat fără a te uita la ele.

Concluzii

Acest proiect a fost o oportunitate să mă apropii mai mult de hardware (este prima dată când fac un design atât de complicat și restricționat de spațiu - plus că mâinile mele tremurânde nu m-au avantajat) și să creez ceva ce voi utiliza cu siguranță la viitoarele prezentări pe care le voi ține în fața colegilor și nu numai. În plus, acum că am control total asupra software-ului ce rulează pe toate dispozitivele din acest lanț de control/comunicare, pot să particularizez oricând vreau telecomanda pentru a-mi servi și la alte scopuri decât doar susținerea de prezentări.

Download

Arhivă proiect: Alexander1752-wireless_presenter.zip

Daemon download - Windows 64-bit: daemon_v1.0.0_win64.zip

Arhivă proiect OCW: alexander1752-wireless_presenter.zip

Jurnal

- *24.04.2024* - temă finală de proiect aleasă
- *29.04.2024* - componentele principale cumpărate
- *02.05.2024* - prototipare pe breadboard a schemei electrice; testare funcționalitate componente
- *03.05.2024* - implementare incipientă a daemon-ului în Python
- *05.05.2024* - creare pagină proiect OCW, creare schemă bloc/electrică de bază
- *08.05.2024 - 13.05.2024* - implementare hardware a schemei electrice (fără baterii)
- *16.05.2024* - implementare a primei versiuni de cod pentru microcontrollere, detaliere și documentare schemă electrică
- *17.05.2024* - adăugare funcționalitate completă butoane, control cursor, economisire energie
- *18.05.2024* - proiectare carcasă 3D
- *22.05.2024 - 23.05.2024* - adăugarea bateriilor la hardware; asamblarea produsului final
- *24.05.2024* - adăugare funcționalitate laser; optimizare consum energie
- *26.05.2024* - finalizare documentație

Bibliografie/Resurse

- **Resurse Software**
  - VirtualWire Library

- Logo Python utilizat în schema bloc
- Bibliotecă EAGLE cu plăci Arduino

- **Resurse Hardware**
  - Datasheet ATmega328P
  - Datasheet MPU6050
  - Understanding the I²C Bus - Texas Instruments

Export to PDF