SimplePlot

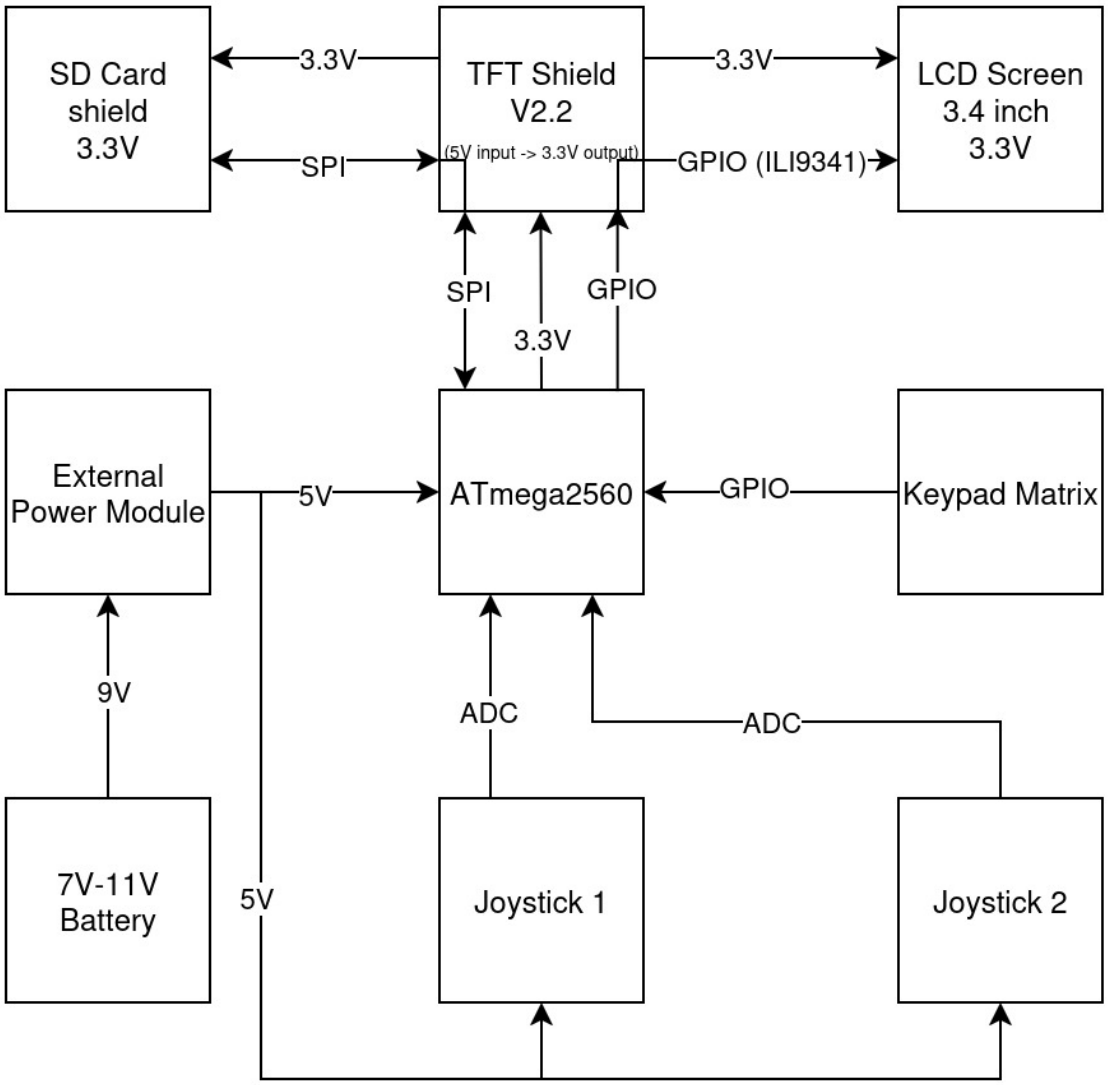
Introducere

SimplePlot este un calculator matematic care se specializează pe trasarea funcțiilor in 2D sau 3D. Utilizatorul poate introduce de la tastatură o funcție în maxim 2 variabile, în mod simbolic. Aceasta va fi parsată, eșantionată și apoi desenată pe un ecran. Utilizatorul va dispune de două joystick-uri cu care poate controla rotația în jurul originii și poziția originii relativă la punctul de origine stabilit global în 0. Userul va putea seta și alți parametri cum ar fi numărul de puncte de eșantionare, distanța intre două puncte de eșantionare (determină un fel de “zoom in”) și va putea salva pe un SD card funcțiile sale preferate (pe lângă un istoric al funcțiilor).

Descriere generală

Componentele sistemului:

- ATmega 2560: Este blocul de control unde au loc toate calculele. Comunică cu 2 joystick-uri (folosind conversii ADC), o matrice de butoane (citire directă de porturi GPIO), un ecran (protocolul folosit de ecran este ILI9341) și un SD reader (comunicare prin SPI).
- TFT Shield V2.2: Realizează conversia de la 5V la 3.3V pentru ecranul LCD și SD card shield.
- LCD Screen: Afișează pe ecran graficele dorite și setări de control.
- SD card shield: Comunică prin SPI blocul de control pentru citiri/scrieri pe SD card.
- Joystick 1/2: Folosite pentru a controla poziția originii/rotația în jurul originii.
- External Power Module: Folosit pentru a alimenta cu 5V tot circuitul.
- Battery: Baterie de 9V folosită de External Power Module pentru a obține cei 5V stabili.
- Keypad matrix: matrice de 4×4 butoane folosită pentru input.

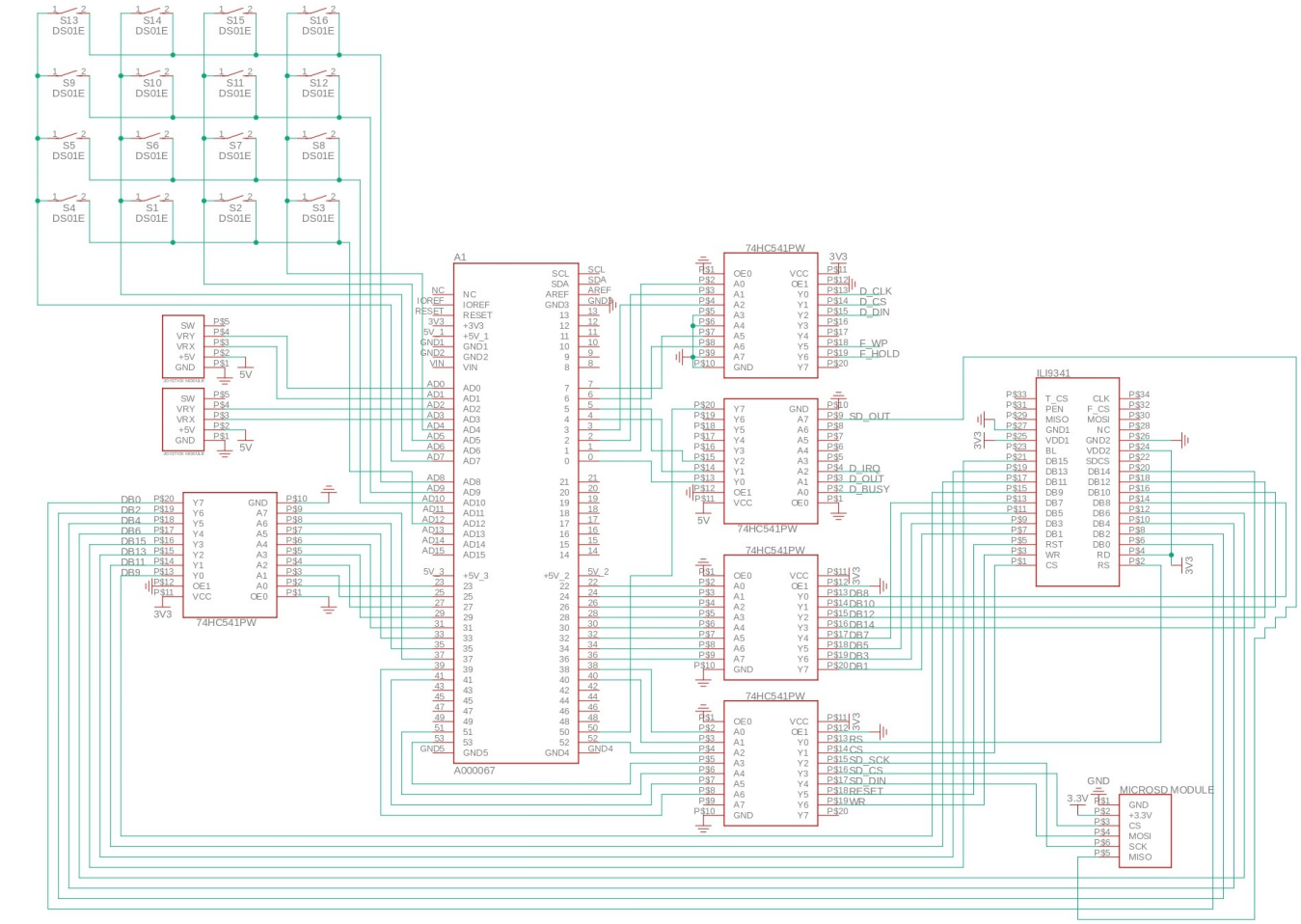


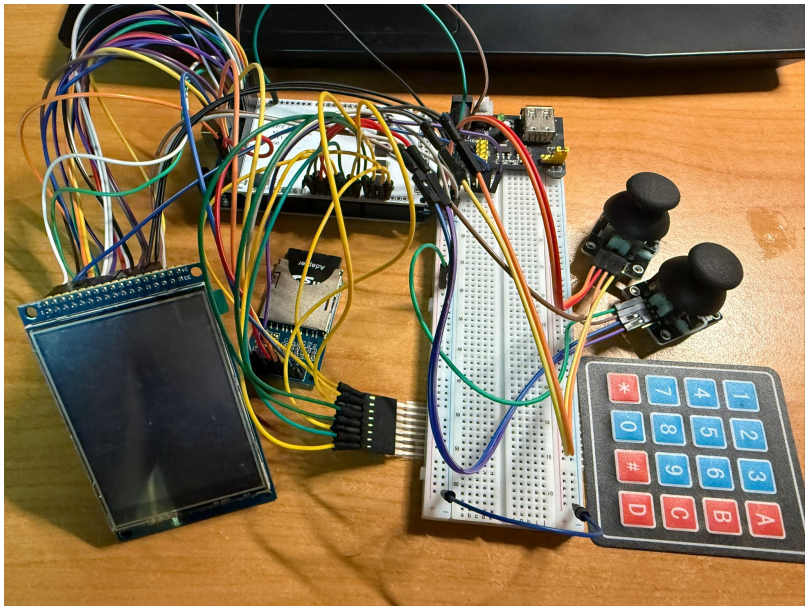
Hardware Design

Listă componente:

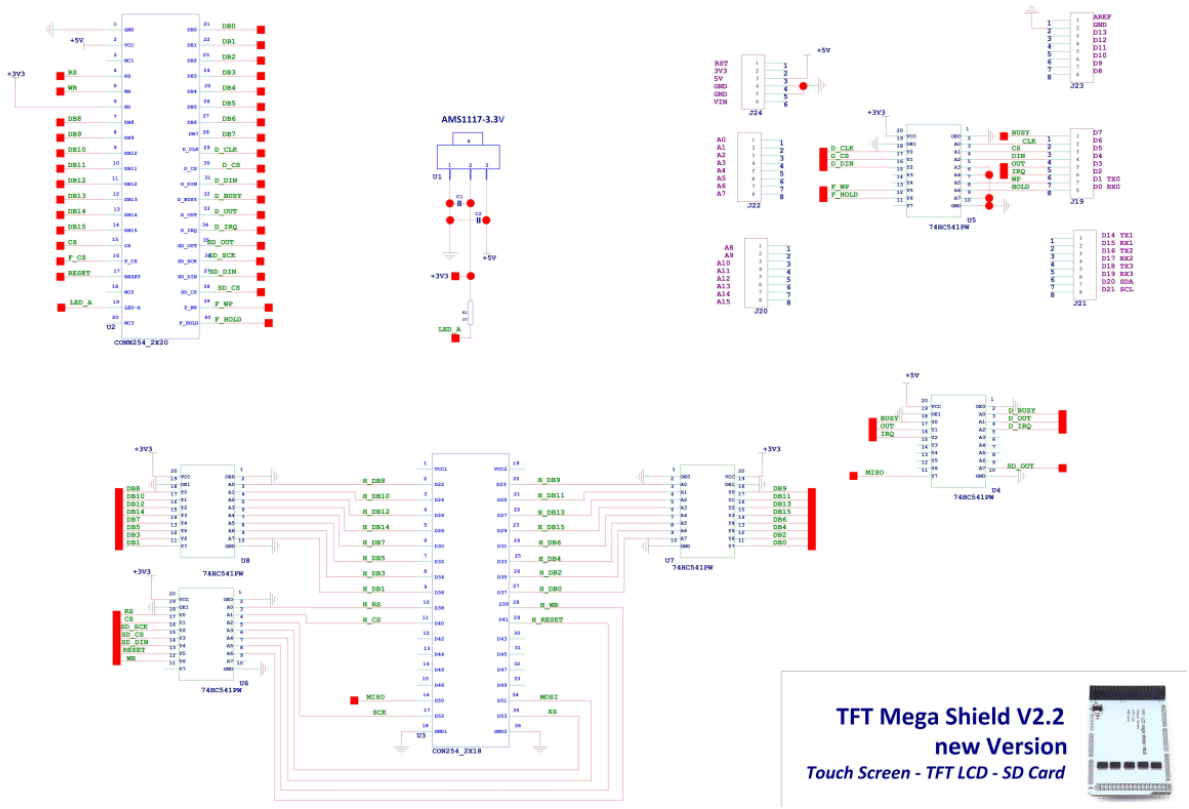
- ATmega 2560
- Joystick module KY-023 - 2 bucăți
- SD Card Module Slot Socket Reader Adapter (3.3V)
- TFT Mega Shield V2.2
- Keypad matrix 4×4
- TFT LCD 3.2 inch 320×240, ILI9341

Notă: Cele 5 module 74HC541PW fac parte din TFT Mega Shield.





Componentele principale sunt un ecran ILI9341 TFT LCD si un ATmega2560. Ecranul funcționează la 3.3V deci am avut nevoie de un shield care să efectueze conversia. Am folosit TFT Mega Shield V2.2. Acesta are următoarea schemă:



Dintre cele 5 module 74HC541PW care realizează conversia de la 5V la 3.3V, folosesc efectiv doar 4 (au fost incluși efectiv în schema proiectului). Shield-ul a fost conectat la plăcuță în felul următor:

- Ocupă toți pinii de power;
- Utilizează pinii digitali 0–7 și 22–53.

Ecranul și SD reader-ul au fost apoi conectate prin intermediul shield-ului la plăcuță (SD reader-ul funcționează și el în regim de 3.3V).

Modul de conectare al ecranului la shield:

- **CS**, **RD**, **WR** și **RS** - control ecran.
- **DB0–DB15** sunt pentru comunicare de 16 biți cu ecranul.
- **VDD** și **GND** sunt utilizați pentru alimentare.

Modul de conectare al SD reader-ului cu shieldul:

- **GND** și **3.3V** sunt conectați la liniile corespunzătoare.
- **CS** este conectat la **SD_CD** pe shield.
- **MOSI** este conectat la **SD_IN** pe shield.
- **SCK** este conectat la **SD_SCK** pe shield.
- **MISO** este conectat la **SD_OUT** pe shield.

Modul de conectare Joystick 1:

- **GND** și **5V** sunt conectate la liniile corespunzătoare.
- **VRX** este conectat la **AD0**
- **VRY** este conectat la **AD1**

Modul de conectare Joystick 2:

- **GND** și **5V** sunt conectate la liniile corespunzătoare.
- **VRX** este conectat la **AD2**
- **VRY** este conectat la **AD3**

Modul de conectare al Keypad-ului:

- **P0** - **AD4**
- **P1** - **AD5**
- **P2** - **AD6**
- **P3** - **AD7**
- **P4** - **AD8**
- **P5** - **AD9**
- **P6** - **AD10**
- **P7** - **AD11**

Deoarece SD-reader-ul nu este conectat direct la plăcuță, ci este folosit shield-ul intermediar, plăcuța va vedea următorii pini pentru comunicarea SPI:

- **SPI_MISO** - **PB3**
- **SPI_MOSI** - **PB2**
- **SPI_SCK** - **P1**
- **SPI_CS** - **SPI_CS**

Software Design

Mediu de dezvoltare:

- **Visual Studio** Code + **PlatformIO**
- **C/C++**

Librării şi surse 3rd-party:

- **UTFT**: folosit pentru comunicarea cu ecranul conform standardului ILI9341.
- **Petit FatFs** - FAT file system module: folosit pentru montarea unui sistem de fișiere FAT32, scriere și citire dintr-un fișier pre-existent (această librărie nu poate crea/modifica mărimea fișierelor).

Modul de reprezentare al numerelor reale.

Operațiile pe floats sunt lente pe ATmega2560. Am ales să folosesc operații pe virgulă fixă. Voi folosi 16 biți pentru partea întreagă și 16 biți pe partea fracțională, respectiv, o valoare reală va fi reprezentată de un `int32_t`.

Modul de randare al funcțiilor 3D

Așa cum am mai menționat, pentru a evalua o funcție, aceasta va fi evaluată pe un grid NxN. Aceste puncte, fiind în 3D trebuie proiectate în spațiul 2D al ecranului. Pentru aceasta, am definit un punct origine unde va fi centrat gridul. Acest punct va reprezenta punctul **look at** pentru proiecție. Camera va fi mereu poziționată pe o sferă de rază R (camera orbitează punctul de origine) și se va uita la punctul de look_at. Poziția camerei pe sferă va putea fi schimbată cu unul dintre joystick-uri iar pozișia originii cu celălalt. Fiecare 3D punct din grid, va fi conectat cu vecinii săi cu o linie, iar obiectul final va fi un fel de “line mesh”. Nu voi randa explicit triunghiuri. Deoarece primitivele nu vor fi triunghiuri ci linii, nu este necesar un test de adâncime, respectiv un framebuffer, care ar fi mult prea mare pentru a încăpea în cei 8KB de SRAM.

Logarea datelor

User-ul își va putea salva funcțiile preferate pe un SD card. Pentru aceasta voi folosi **SPI**. Inițierea SPI-ului este realizată în fișierul `spi.h`. Pentru montarea și lucrul cu fișiere voi folosi librăria externă Petit FatFs. Fișierul pentru funcții `data.txt` deja există pe cardul SD. Acolo voi stoca funcțiile în formă simbolică.

Algoritmi și structuri de date:

- **Geometry.h**: aici am definit diverse funcții geometrice și matematice ce operează pe tipul de date descris în secțiunea precedentă:
  - Operații primitive pe numere în virgulă fixă:
    - înmulțire
    - împărțire
    - sin
    - cos
  - Operații vectoriale:
    - Adunare/scădere
    - cross prduct
    - calcul normă
  - Funcții grafice:
    - Look at - Crează matricea de look at cu vectorii forward, right, up
    - Proiecții 3D - proiecția unui punct 3D în spațiul ecranului.
- **Parser.h** este prima structură cu care user-ul interacționează. Acesta va tasta o funcție, în mod simbolic, de la tastatură. În acest moment, este necesară parsarea acestei funcții pentru a detecta erori și a o putea evalua în diverse puncte. Această clasă va primi ca input un string care este reprezentarea simbolică a funcțieie și va returna un șir în Reverse Polish Notation sau o eroare.
- **PlotFunction.h** este un obiect ce poate fi creat pe baza unei expresii în *Reverse Polish Notation*. Șirurile în Reverse Polish Notation pot fi evaluate foarte ușor deoarece expun, în ordinea care trebuie efectuate, operațiile și operanzii corespunzători, respectiv un obiect de tip PlotFunction va

expune o singură metodă spre exterior: `int32_t get_value(int32_t x_q16, int32_t y_q16);`. Această metodă va evalua funcția într-un punct (x, y) sau (x, 0) dacă funcția are o singură variabilă.
- **Plot3D/2D.h** primește ca input o funcție învelită într-un PlotFunction, o evaluează într-un grid de NxN puncte și apoi realizează proiecția punctelor în 2. Tot această structură va ține cont de poziția camerei pe sfera imaginară și poziția originii. Datele acestei structuri se recalculează doar la schimbarea orignii (eveniment care este de așteptat să se întâmple mai rar, mai mult ne așteptăm la rotiri). Pentru plotarea funcțiilor 2D este mult mai ușor. Putem direct conecta cele N puncte 2D cu linii, nu avem rotații, doar mișcări/schimbări de origine.
- **Keyboard.h** este o clasă care ține cont de starea keyboard-ului și ce butoane sunt apăsate la moment. Acesta va funcționa pe bază de întreruperi. Există 4 pinuri pentru rânduri și 4 pentru coloane. Butonul apăsat reprezintă intersecția unei linii cu o coloane, obținută prin monitorizarea stării pinilor celor 8 pini.
- **ADC.h**: aici au loc inițializări și citiri a celor 4 pini ADC folosiți pentru cele 2 joystick-uri. Modulul ADC va funcționa în regim free running: convertorul va fi pornit automat regulat și datele scrise în ADC. Deoarece am 4 valori de citit, am decis să mențin o variabilă care indică care din cele 4 este convertită la moment. Cand o conversie este gata, valoarea se incrementează iar dacă ajunge la 4 este resetată la 0.

Elemente de UI și User Experience

UI

Pagina de pornire va permite utilizatorului să tasteze o funcție de la tastatură, încarce una deja existentă de pe micro SD sau să marcheze funcția tastată pentru salvare. Va fi un meniu simplu, în partea de sus va fi o bară unde funcția curentă va putea fi vizualizată simbolic.

User Experience

Deoarece tastatura are doar 16 butoane, voi folosi 2 “layout-uri”. Unul din cele 16 butoane va schimba layout-ul:

- Layout 0:
  - Butonul de schimbare layout (1 buton).
  - Buton de ștergere ultim caracter (1 buton).
  - Cifre 0-9 (10 butoane).
  - 2 variabile: x, y (2 butoane).
  - Operatorul '(' și ')' (2 butoane).
- Layout 1:
  - Butonul de schimbare layout (1 buton).
  - Buton de ștergere ultim caracter (1 buton).
  - Buton de finalizare, treci la plotarea functiei (1 buton).
  - Operații elementare '+', '-', '/', '*' (4 butoane).

- Operații avansate `sin`, `cos`, `exp`, `log`, `floor`. (5 butoane).
- Operatorul '(' și ')' (2 butoane).
- Toggle save for SD. (1 buton).
- Open saved functions menu (1 buton).

Rezultate Obţinute

Care au fost rezultatele obţinute în urma realizării proiectului vostru.

Concluzii

Download

O arhivă (sau mai multe dacă este cazul) cu fişierele obţinute în urma realizării proiectului: surse, scheme, etc. Un fişier README, un ChangeLog, un script de compilare şi copiere automată pe uC crează întotdeauna o impresie bună .

Fişierele se încarcă pe wiki folosind facilitatea **Add Images or other files**. Namespace-ul în care se încarcă fişierele 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** şi **Resurse Hardware**.

Export to PDF