# Mini GPU

## Introducere

**Student**: Alex-Andrei Cioc
**Grupa**: 336CC

Proiectul **Mini GPU** isi propune sa emuleze capacitatile de procesare de date RGB ale unui accelerator grafic de baza. Scopul nu este sa fie capabil sa proceseze calcule pentru grafica 3D, ci de a genera un semnal VGA, generand imagini procedurale simple sau primite prin Wi-Fi.

Am pornit de la ideea unui YouTuber cunoscut, Ben Eater, care a facut “cea mai proasta placa video” (video aici), folosind doar circuite integrate. Mi s-a parut o idee interesanta si m-am gandit sa incerc sa fac si eu o placa video putin mai buna, folosind un microcontroller.

Consider proiectul util in scop didactic, ca unealta de a invata bazele procesarii de imagini si cum functioneaza (sau *functionau* :p) monitoarele.

## Descriere generală

Pentru generarea semnalelor VGA necesare vizualizarii de imagini pe un monitor compatibil VGA (rezolutie standard 640×480), voi folosi un microcontroller ESP32, montat pe o placa de dezvoltare DevKit-C-1-N8R8. Acesta va fi capabil de a citi imaginea salvata pe un card microSD, sau de a prelua o imagine transmisa prin Wi-Fi si a o salva pe card.

Microcontroller-ul citeste imaginea de pe card si o incarca in memoria PSRAM (pentru acces rapid), transmitand datele RGB corespunzatoare catre un DAC triplu, care va genera semnalele analogice RGB necesare interfetei VGA; totodata, vor fi generate si semnale auxiliare de stingere si sincronizare pe orizontala si verticala. De asemenea, imaginea poate fi si generata procedural (**SAMPLES IN PROGRESS**).

Pentru upload de imagini este deschis un server HTTP, unde se face POST cu datele binare ale imaginii in format **TO BE DECIDED**. La primirea datelor, acestea se salveaza pe card.

De asemenea, microcontroller-ul are conectat prin SPI un ecran LCD, folosit pentru preview al imaginilor si pentru a afisa date despre monitor. Datele despre monitor si modurile de afisare suportate sunt preluate folosind protocolul DDC (Display Data Channel), care comunica cu monitorul, prin I2C, pe unul din pinii de VGA.

La baza transmiterii de semnale video RGB sta un DAC triplu de viteza mare, specializat Video, cu 3 canale pentru R, G, B. Performantele DAC-ului si ale modulului ESP32 permit generarea de semnale

pentru afisarea de imagini cu rezolutie de la 640×480 (VGA) in sus, in 16M culori (3x8bits).

## Hardware Design

Lista de piese:

- ESP32-S3-DevKitC-1-N8R8
- THS8136 Triple Video DAC, 10Bits, 180MSPS
- Amphenol D-Sub (conector VGA)
- KMR 1.8 SPI TFT 128×160 Display (cu suport microSD)
- 3x SN74LVC821ADWR 8-bit registers
- 2x TC1265-1.8VOATR voltage regulator
- SN74LVC2G17DBVT dual level shifter
- 2x BSS138 NMOS transistor
- 1.8V MMSZ4678-HF Zener diodes
- Resistors, capacitors

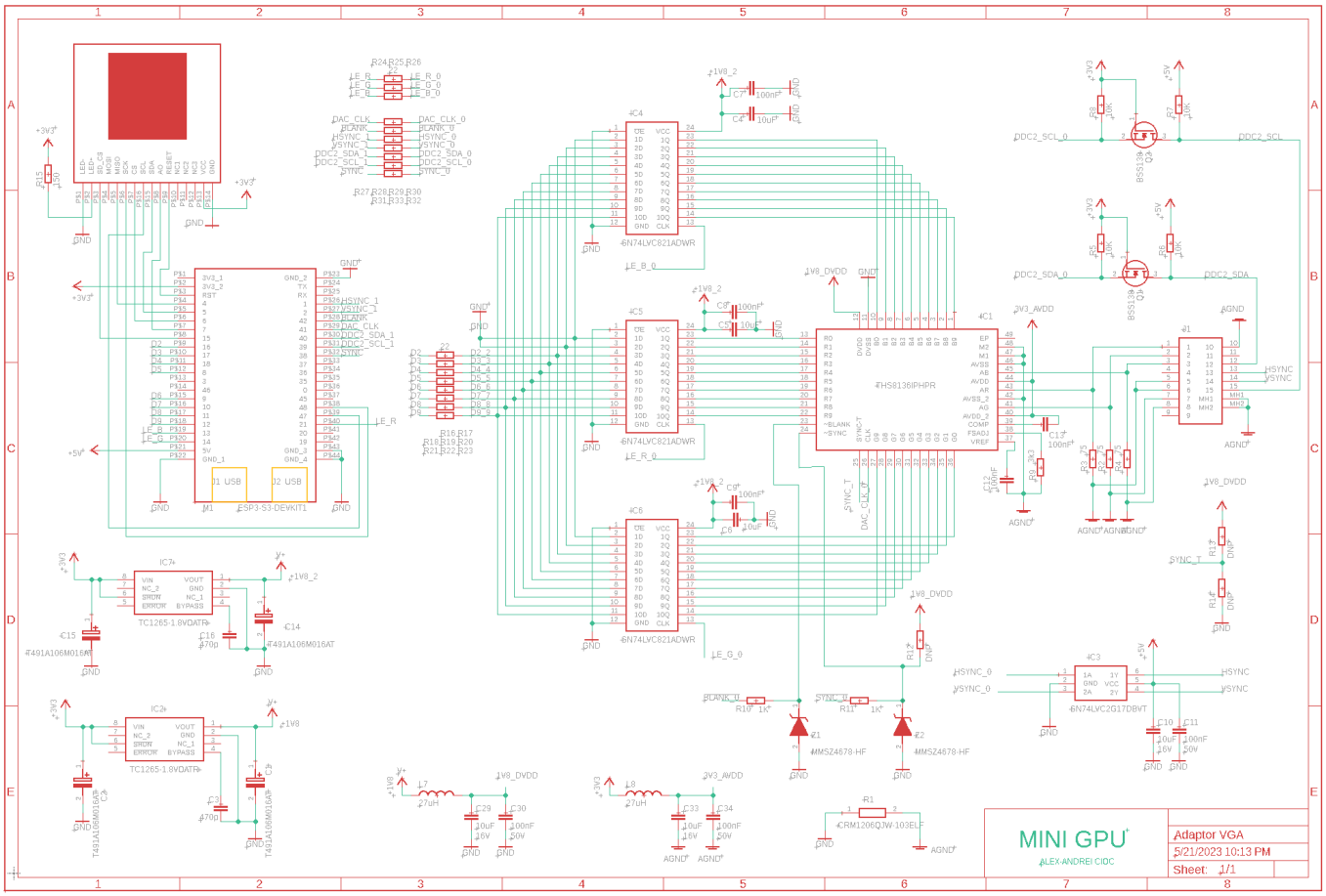
DAC-ul are 3 registri de 10 biti pentru memorarea datelor pentru RGB. Deoarece modulul de dezvoltare nu are destui pini GPIO disponibili pentru a face interfatarea direct pe 24 biti, am folosit 3 registri externi cu rol de buffer. Astfel, ESP32 are interfata de date pe doar 8 biti, la care se adauga celelalte semnale pentru controlul sincronizarilor si altele.

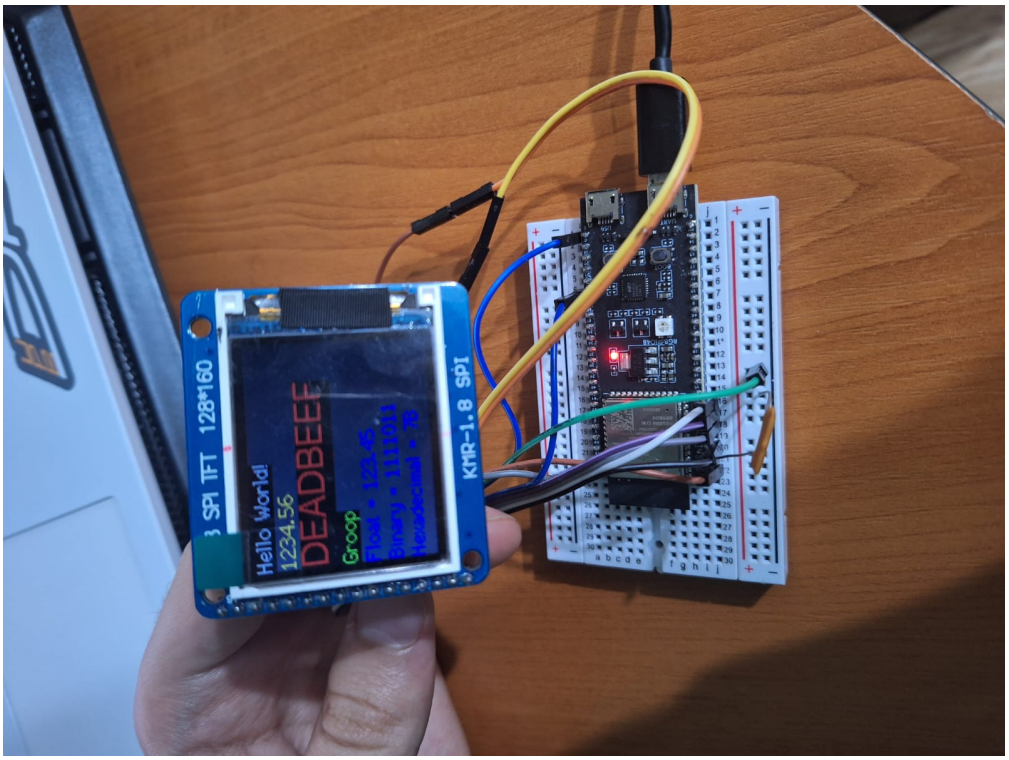
Pentru alimentare, se poate utiliza un alimentator de 5V, aplicat pe conectorul micro USB. Placa de dezvoltare are un stabilizator la 3.3V, care alimenteaza modulul de ESP32. Afisajul KMR este alimentat cu 3.3V direct din placa. DAC-ul si registrii functioneaza la 1.8V pentru partea digitala; de aceea, am avut nevoie de stabilizator de la 3.3V la 1.8V. Partea analogica se alimenteaza de la 3.3V din placa.

Level shifter-ul dual integrat este unidirectional si este folosit pentru semnalele de sincronizare HSYNC si VSYNC.

Pentru comunicatia pe protocolul DDC2 (intre ESP32 si monitor), am folosit doua level shiftere bidirectionale, pentru SDA si SCL, realizate cu tranzistoarele NMOS.

Schema electrica de mai jos am realizat-o in Autodesk Eagle. Pentru realizare fizica, mai e putin de asteptat: mai exact, pana mi se livreaza PCB-ul:))




# Software Design

Am dezvoltat proiectul in mediul PlatformIO.

Pentru acest proiect am avut nevoie de mai multe componente la nivel software. Mai exact, pe langa logica efectiva de a genera semnale VGA, am avut nevoie si de o modalitate de a incarca poze pentru a le putea citi pentru afisare pe ecran.

Astfel, am impartit munca pe cele doua core-uri ale ESP32. Un core ruleaza un server HTTP si serveste o pagina Web cu o interfata de unde se poate naviga prin filesystem-ul de pe cardul SD si face operatii precum:

- Creare de directoare
- Redenumire fisiere/directoare
- Descarcare fisiere
- Incarcare fisiere.

Intre timp, s-a defectat (sau, probabil, nu a functionat niciodata) slot-ul de card SD de pe display, asa ca a trebuit sa adaug un breaker adapter separat, si am schimbat putin conexiunile electrice.

Functionalitatea de generare de semnale VGA foloseste intreruperi precise, pentru a avea o fidelitate cat mai buna a imaginii si a respecta constrangerile de sincronizare VGA. Nu e prea mult de explicat aici, in afara de faptul ca a trebuit sa fiu foarte atent la eficienta si am folosit functiile low-level pentru lucru cu GPIO.

Repository-ul unde se afla codul se gaseste aici.

# Rezultate Obţinute

Cand ies imaginile perfect anunt aici ;)

# Concluzii

Am aflat pe viu ce inseamna sa stii ce vrei sa faci, sa stii cum sa faci, sa faci, si sa NU mearga :<

Cand e nevoie si de lucru cu hardware-ul, apar foarte des lucruri neprevazute; si e cu atat mai greu cu cat folosesti piese care nici nu au numele firmei inscriptionat pe ele. De exemplu, display-ul mi-a dat foarte multe batai de cap, caci m-am bazat pe faptul ca slotul integrat de card SD o sa imi fie util. S-a dovedit ca e destul de greu de configurat, si, probabil, trebuie sudate niste jumpere pe acolo ca sa primeasca destul curent pentru a functiona, insa faptul ca nu se gaseste nicaieri documentatie despre circuit m-a facut sa renunt la el si sa folosesc un slot extern.

Batai mari de cap mi-a dat si placuta de dezvoltare cu ESP32, mai ales alegerea pinilor. Proiectul meu necesita o latime de banda mare pentru output, si am ajuns sa folosesc toti pinii GPIO care nu sunt rezervati pentru alte lucruri importante; adica, daca mai aveam nevoie de inca un pin, probabil nu mai aveam de unde sa fac rost:)) M-am incadrat la fix numai dupa ce am facut sacrificiul de a avea un throughput de 3 ori mai mic, folosind registri pentru scrierea pe rand a bitilor de culoare RGB; astfel, in loc de 24 de biti pentru culoare, folosesc doar 8 biti, insa scriu de 3 ori. In viata nu le poti avea pe toate ;)

De asemenea, a fost o experienta (si de bonding, de altfel) sa stau cu tata sa printam un PCB de “mana”...

## Download

Repository aici.

## Jurnal

- 05/07/2023 au venit piesele principale (ESP32, DAC, LCD si diode, condensatoare, rezistoare)
- 05/18/2023 au venit registrii, stabilizatoarele de curent, level shifter-ul

## Bibliografie/Resurse

Resurse (foarte) utile:

- Documentatia oficiala pentru placuta mea: https://docs.espressif.com/projects/esp-idf/en/latest/esp32s3/hw-reference/esp32s3/user-guide-devkitc-1.html
- Laburile de PM

Biblioteci folosite:

- https://github.com/espressif/arduino-esp32/blob/master/cores/esp32/Arduino.h
- https://github.com/espressif/arduino-esp32/tree/master/libraries/WiFi
- https://github.com/espressif/arduino-esp32/tree/master/libraries/WebServer
- https://github.com/espressif/arduino-esp32/tree/master/libraries/SD
- https://github.com/Bodmer/TFT_eSPI
- https://github.com/Bodmer/TJpg_Decoder

Export to PDF