Dokumentace
&
Zpráva

Obsah dokumentace & zprávy

Dokumentace

Popsání projektu v úvodním zadání:

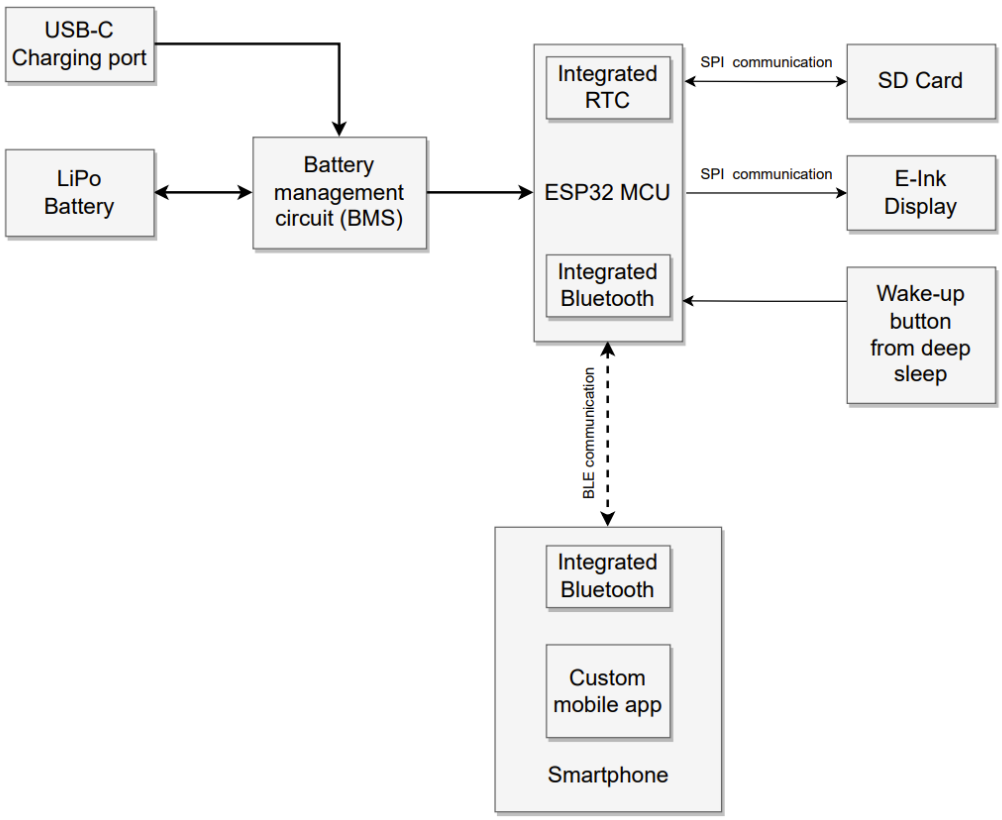
Projekt stojí na E-Ink displeji, tento display jsem zvolil z důvodu malé spotřeby při změně obsahu a nulové spotřebě při zobrazování statického obsahu. Díky tomu může být foto rámeček napájen z akumulátoru po dobu dlouhých měsíců (Zatím jsem dělal jen hrubý odhad, stále nevím jak velkou kapacitu akumulátoru zvolím a jaká bude spotřeba hotového projektu) Cíl je aby zařízení stačilo nabíjet maximálně jednou za půl roku.

Foto rámeček se bude moct dát nabít pomocí USB-C jako všechna moderní zařízení. Pro zajištění bezpečnosti bude mít akumulátor BMS, které bude hlídat zda nedošlo ke zkratu, přebití nebo podbití.

Všechny obrázky budou uloženy na SD kartě. Obrázky se budou na SD kartu nahrávat pomocí mobilní aplikace. Aplikace umožní uživateli načíst zvolený obrázek a postará se o převedení obrázku do správného barevného prostoru(E-Ink display umí pouze 7 barev) a následně obrázek odešle pomocí Bluetooth, který jej zapíše na sd kartu. Aplikace také bude uživatele informovat o počtu fotek nahraných na foto rámečku a umožní uživateli nastavit čas jak dlouho se má každý obrázek zobrazovat a umožní také uživateli vybrat pořadí fotek, ve kterém se budou zobrazovat.

Wake-up tlačítko plánuji použít, pro probuzení MCU, protože stále mám problém s integrací low energy Bluetooth, aby bylo zařízení pořád dostupné k připojení a spotřeba zařízení přitom splňovala limit. Proto nejspíše zvolím cestu, že pokud uživatel bude chtít změnit obrázek přijde k foto rámečku zmáčkne tlačítko a tím probudí MCU a obnoví se Bluetooth komunikace. Při dlouhém zmáčknutí tlačítka bude fungovat jako reset, pokud by se cokoliv pokazilo, tak aby šlo zařízení nastavit do továrního režimu.

Blokové schéma projektu



Softwarové blokové schéma

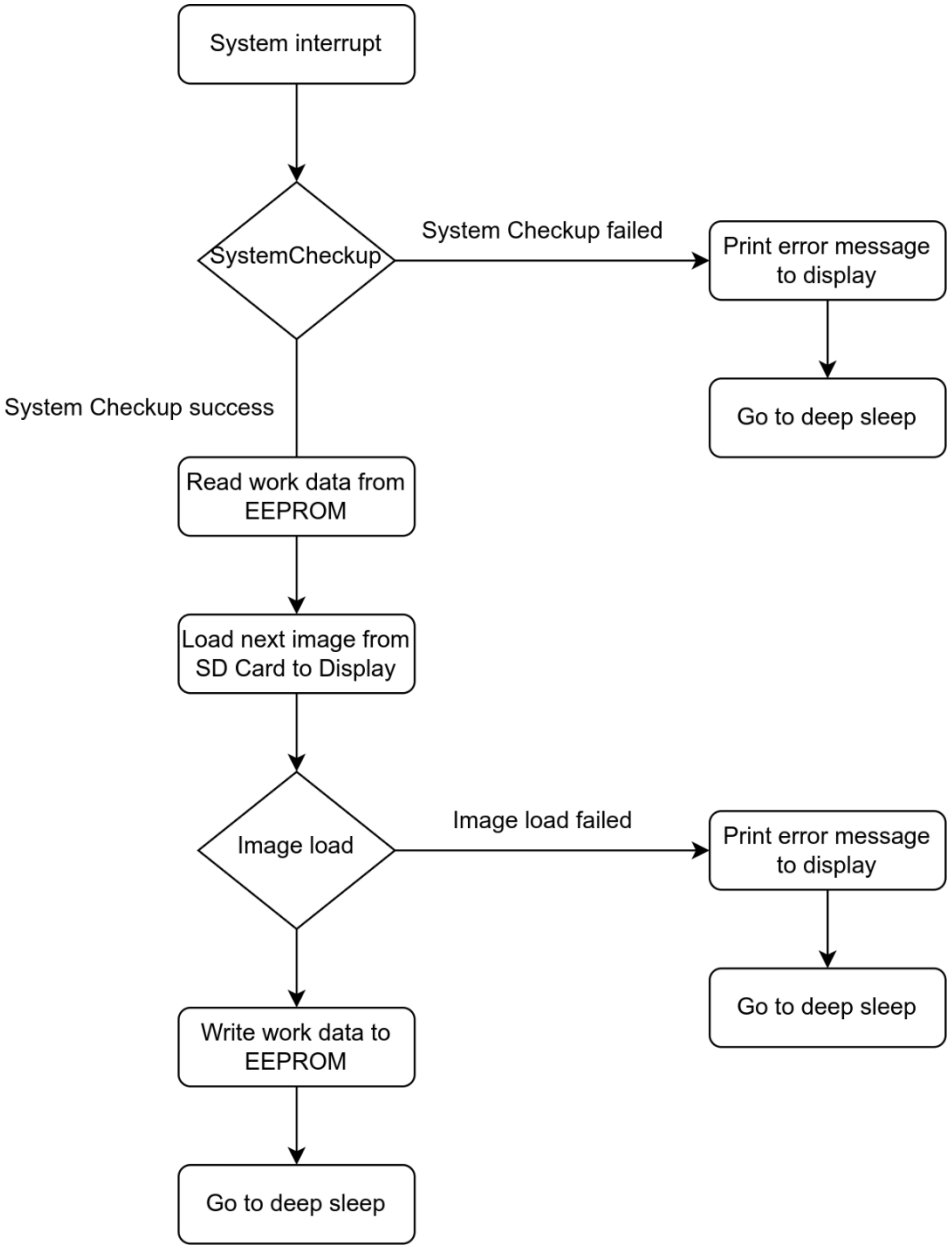
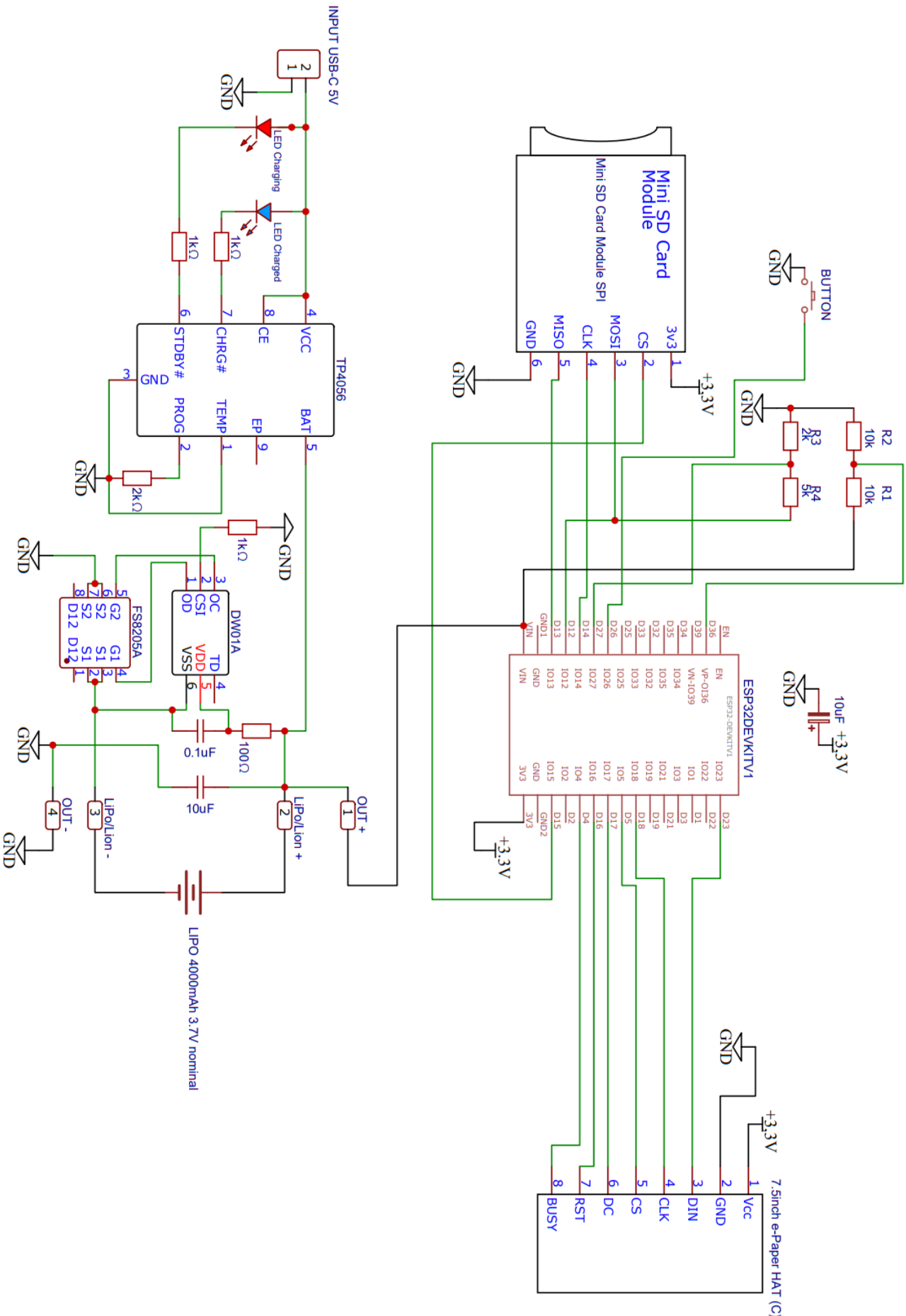


Schéma zapojení hardwarové části



Potřebná teorie

- Nutnost je znalost základů programování v jazyce C a práce s běžnými knihovnami.
- Deklarace a naplnění proměnných, následná manipulace s nimi (např: převádění datových typů).
  Tvorba a práce s metodami (funkcemi) jak pracovat s návratovými typy, zacyklení, větvení (while, for, if, switch…) Jak funguje přetečení proměnné a jak se dá této funkce využít v náš prospěch.
- Znalost knihoven jako: <Arduino.h>, <SD.h>, <SPI.h>…
- Také jak probíhá komunikace pomocí BLE (Bluetooth low energy protocol), navazování bezpečného spojení a zápis přijatých dat na SD kartu v čitelném formátu.
- Práce se systémovým přesušením
- Znalost RTC a jak jej správně použít
- Znalost knihoven pro práci e E-INK displeji
- Znalost vývoje mobilních aplikací.
- Znalost barevných palet, jak funguje Floyd-Steinberg dithering algoritmus (Algoritmus pro distribuci chyb). Co je to převádění barevných palet, jak funguje formát obrázku typu BMP, jak jej číst a převést čistě na pole bajtů.
- Pro větší uživatelskou přívětivost je vhodné mí v mobilní aplikaci možnost lehké editace obrázků. Jako: oříznutí, otočení, škálování na 800x480 rozlišení, saturace a jas.

Seznam použitého hardwaru pro tento projekt

- MCU - ESP32 - https://randomnerdtutorials.com/getting-started-with-esp32/
- 7 barevný E-Ink displej - https://www.waveshare.com/product/displays/e-paper/7.3inch-e-paper-hat-f.htm
- microSD Modul - https://www.laskakit.cz/microsd-card-modul-spi-3-3v/
- TP4056 s USB-C verzí - https://www.laskakit.cz/nabijecka-li-ion-clanku-tp4056-s-ochranou-microusb/
- 16GB micro SDHC V10
- 2x 10kΩ 0,25W THT rezistor
- 1x 2kΩ 0,25W THT rezistor
- 1x 5kΩ 0,25W THT rezistor
- 1x tlačítko
- 1x LI-PO 4000mAh akumulátor

Seznam použitého softwaru/knihoven pro vývoj

- Arduino.h
- SPI.h
- SD.h
- FS.h
- Hlavní knihovna pro práci s E-INK displejem https://github.com/ZinggJM/GxEPD2
- VS Code s Platform IO IDE
- Visual Studio 2022 WASM web app
- Aplikace pro převod obrázků https://github.com/mcraiha/Dithery

Popis řešení

Při spuštění:

Mikrokontroler spustí vlastní diagnostiku, zkontroluje, zda je vložena SD karta je možnost komunikace s SD kartou, následně ošetří zda funkční komunikace s displejem a udělá kompletní obnovu panelu (terminologie z technologie e-ink tzv. Fullscreen-refresh) a zobrazí zprávu, že je připraven. Následně dojde k načtení prvního obrázku z SD karty, pokud se zde obrázek nachází.

Následně dojde k přečtení vnitřního stavu z konfiguračního souboru na SD kartě. Načte se poslední uložený čas také časy během dne, kdy má dojí k obměně obrázku a také o stavu kolik obrázků se nachází na sd kartě.

Při běhu:

Po inicializaci, přejde mikrokontroler do režimu spánku a jednou za 500ms pošle BLE notifikaci, že poslouchá a je připraven se napárovat a spojit s aplikací. A zároveň čeká na systémové přerušení z RTC, které probudí mikrokontroler a ten následně vykreslí následující obrázek. Po vykreslení nového obrázku, přejde mikrokontroler opět do režimu spánku a cyklus se opakuje.

Při navázání spojení s mobilním telefonem a aplikací na ovládání čeká mikrokontroler na data z mobilního telefonu, může přijít aktualizace RTC času v ESP32 včetně seznamu časů, kdy má dojít k obměně obrázků a v jakém pořadí se mají cyklit. Následně také může přijít příkaz ke smazání obrázku anebo také může přijít nový obrázek, který se ihned začne ukládat na SD kartu.

Před každým vykreslením se zkontroluje stav baterie, pokud je napětí baterie po určitou úrovní na display v rohu se připíše zpráva ohledně nízkého stavu baterie. Tento stav pokračuje, dokud BMS neodpojí mikrokontroler od napájení.

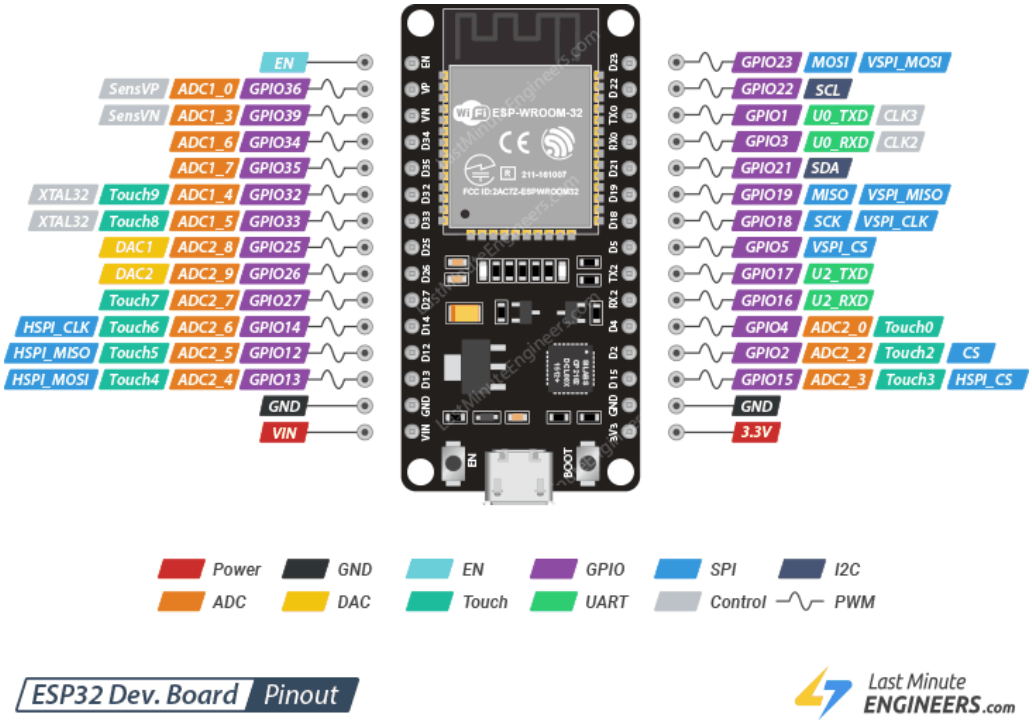
Zpráva

Problémy s realizací.

Během realizace jsem narazil na více problémů, než jsem, jakkoliv v průběhu očekával, což bohužel vedlo k tomu, že projekt není zdaleka hotoven a není ve stavu, ve kterém bych si ho představoval.

- Výrobce neposkytuje řádné knihovny na všechny platformy a pokud ano vzorové ukázky jsou velice osekané bez jakékoliv dokumentace a vysvětlení.
  Výrobce poskytuje manuál, jak podrobně komunikovat s E-Ink displejem, jak po SPI sběrnici posílat data a instrukce, časování, které přesně definují, jak postupně ovládat pixel po pixelu, zápis hodnot do interní bufferu displeje, ale abych si dokázal postavit vlastní knihovnu, která by byla bez chyb by zabralo hodně času.
  Na platformu ESP32 ani vzorová knihovna není výrobcem poskytována na tento 7 barvený displej.
- Komunitní knihovny to naštěstí zachraňují, bohužel ale převážně podporují pouze černobílé displeje, pro barevné nic moc, povedlo se mi najít knihovnu s mnoho funkcemi, bohužel ale kompletně bez dokumentace, jediné co dobré, že obsahuje mnoho vzorových příkladů, ze kterých se dá něco pochytit, problém je že obsahuje často podobné názvy metod a bez zjevného popisu rozdílů, takže je opět potřeba zajít do zdrojového kódu a hledat, nebo zají na fórum, které funguje od roku 2017 a procházet doslova tisíce dotazů a odpovědí, které se nevětví do stromu, ale všechny na sebe navazují.

- Vhodná knihovna disponuje mnoho funkcemi na vykreslování textu a 2D vektorové grafiky, jako jsou úsečky, obdélníky, kruhy, elipsy a jednobarevné bitmapy. To se ale nedá říct o vykreslování 7 barevných bitmap, na ty neexistuje funkce. Jedna možnost byla „drawRaw", která doslova vzala bajty v poli, kde každý bajt reprezentuje 2 pixely. A jeden za druhým pošle na displej, bez jakékoliv logiky. Při rozlišení 800x480 je to to ale nepoužitelné, protože se to nevejde do operační paměti. Knihovna zmateně, nebo aspoň dle mého názoru vykresluje obrázky v každém vzorovém demu, protože postup je pouze uveden v examplech, viz1 a viz2.
- Mikrokontroler ESP32 disponuje 2x sběrnicí SPI, jedna slouží ke komunikaci s displejem a druhá slouží ke komunikaci s SD kartou, ale na druhé sběrnici(HSPI) je pin GPIO12, který slouží jako MISO, ale z principu funkčnosti protokolu je výchozí hodnota v klidovém stavu HIGH, to ale během přepisování FLASH paměti a bootu nesmí být hodnota LOW, protože pin slouží k nastavení napětí pro zápis do flash paměti a také je využíván při přepisování interního firmwaru.



Tak první krok vedl k tomu, že zkusím použít tedy jen jednu sběrnici(VSPI) a pomocí CS(Chip Select) budu přepínat mezi SD kartou a displejem, bohužel to se nelíbilo knihovně, zkousil jsem několik úprav, ale po pár hodinách jsem to vzdal, tak jsem přemýšlel, jak docílit LOW hodnoty, napřed mě napadlo použít běžný tranzistor, to ale nepůjde protože budu mít úbytek napětí 0,7V a povede to ke špatné reprezentaci datového signálu, další možnost byla použít MOSFET tranzistor, abych eliminoval úbytek, protože ale zapojení bude na HIGH úrovní, potřeboval bych P-typ, ale doma všechny MOSFETY co jsem našel jsou typu N. Nakonec jsem to vyřešil pomocí rezistorů **R3, R4**, které jsou vidět na schématu a slouží jako dělič napětí. Při bootu, je těch 7kΩ dostačující na to udržet napětí <1V, ale při komunikaci by to mohlo dělat problémy, proto po inicializaci je dělič napojen na pin 26, který je následně během komunikace nastaven na hodnotu HIGH.

- Problém s převodem obrázků, aby uživatelé nemuseli obrázky složitě převádět sami, tak jsem součástí projektu byla také mobilní aplikace, která se postará o převod obrázků, protože jsem byl už ale v tomto momentě hodně ve skluzu, rozhodl jsem se pro vytvoření desktopové aplikace, která načte BMP obrázek a následně aplikuje Floydova–Steinbergův algoritmus a převede obrázek do 7 barevného prostoru, ale přitom uloží obrázek ve 24bitové barevné hloubce, protože to je co na algoritmu z knihovny fungovalo. Očekával jsem že jednoduše najdu knihovnu a aplikuji algoritmus společně s barevným profilem, ale bohužel se to ukázalo jako nefunkční řešení povedlo se mi udělat jednu verzi, ale obrázky se nechovali stejně jako ty co jsem vyexportoval z GIMP2 programu s vlastní paletou, kterou jsem vytvořil. Použil jsem knihovnu LibDithering a také ImageProcessor, nic z toho, ale nefungovalo, tak jak jsem chtěl.
  Proto jsem upravil již dostupný projekt využívající knihovnu LibDithering se jménem Dithery
  Jedná se WASM Blazor aplikaci, upravil jsem ji, aby podporovala mojí dostupnou barevnou paletu a exportovala BMP ve 24bitovém formátu, místo PNG. I přesto se vyexportovaný obrázek nechová ta jak má. Měl jsem se předem podívat, jak vlastně BMP formát funguje a na co si dát pořád a opět jsem ztratil desítky hodiny práce.
  Protože jsem na WASM naposled narazil tak před rokem a od té doby jsem s tím nepracoval, tak se mi nepodařilo webovou aplikaci spustit mimo lokální server, takže stránka bohužel není dostupná, pokus je stále online a dostupný prozatím zde: https://colorcovnersion-bi-ard.netlify.app/

  GitHub repositář s ESP32 kódem a dokumentací: https://github.com/Vassterak/ESP32-SmartPhotoFrame-EInk-V1

  GitHub repositář s kódem pro webovou aplikaci sloužící pro převod obrázků, jedná se fork Dithery, moje upravená verze: https://github.com/Vassterak/WASM-Fork-ColorConversion