Mikrokontroller laboratórium házi feladat

SÁGHY ATTILA
FZFZXF

Feladat kiírása:

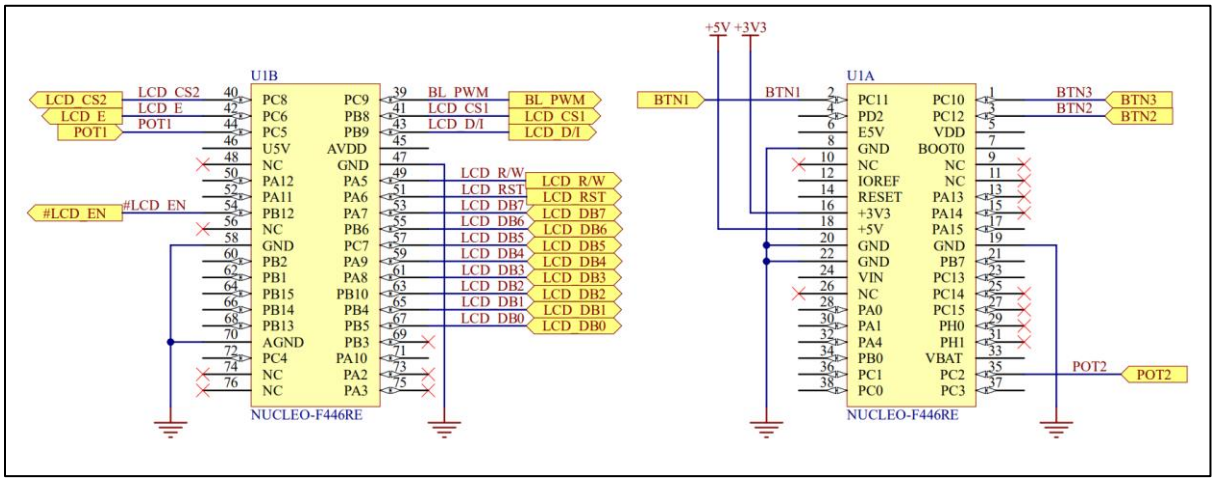
11. feladat: Elektronikus rajztábla

Illesszen 128x64 pixelből álló grafikus LCD kijelzőt az STM32 NUCLEO-F446RE kithez, valamint két potenciométert a kontroller A/D konvertereire. A két potenciométer segítségével valósítson meg a kijelzőn ún. „rázós” rajztáblát. Helyezzen el továbbá egy nyomógombot, amellyel a képernyőt törölni tudja, valamint a törlés opcionálisan egy gyorsulásérzékelő segítségével, „rázásra” is megvalósulhat. Készítsen egy PC-s kliensprogramot, amely segítségével a képet soros porton keresztül letölteni, tárolni és visszatölteni tudja. A soros kommunikációhoz virtuális soros portot használjon, melyet a kiten megtalálható USB port segítségével valósítson meg!

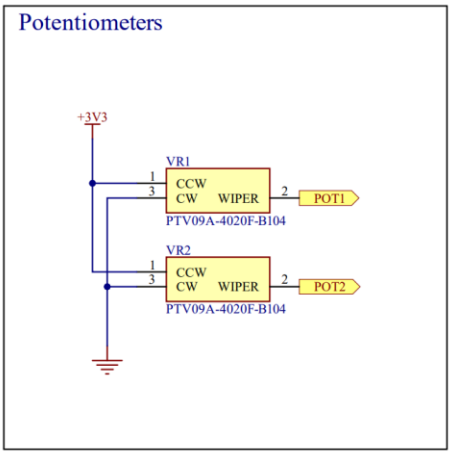
Az opcionális gyorsulásérzékelő nem került bele a megvalósításba, viszont két plusz nyomógomb igen, a menürendszer egyszerűbb kezelése miatt.

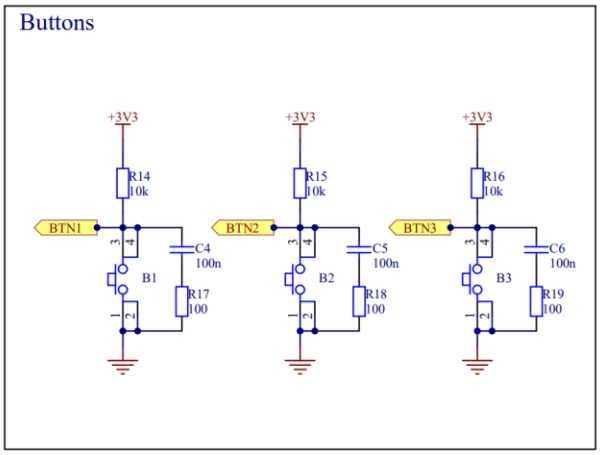
Kapcsolási rajz:

Nucleo:

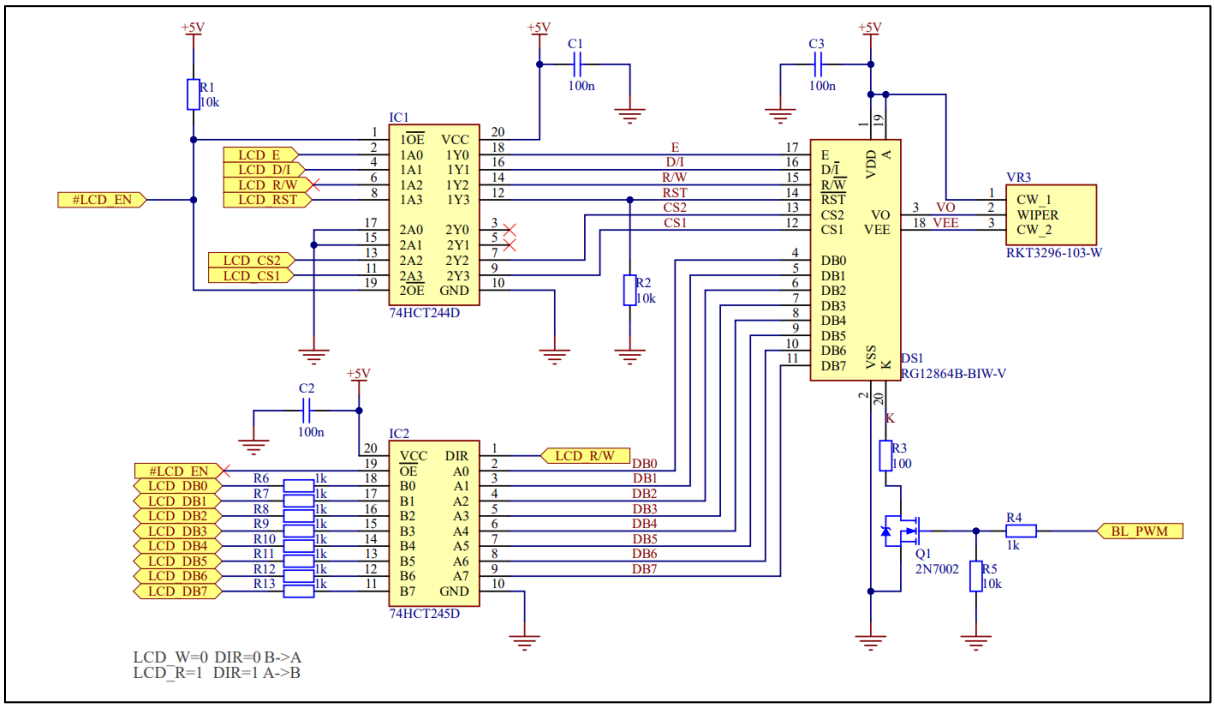


Gombok és potenciométerek:





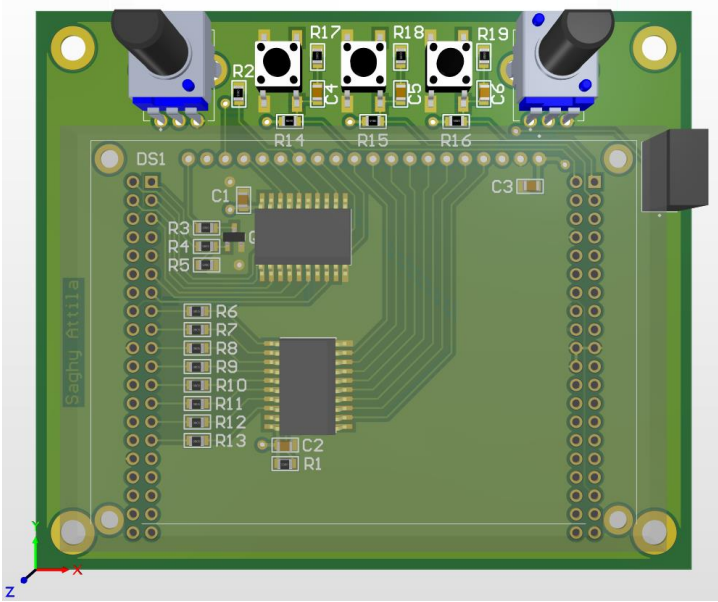
LCD:



A nucleo ki- és bemenetei 3.3V-osak, míg az LCD-é 5V, ezért szintillesztés szükséges. Ezt két HCT IC-vel oldottam meg, melyek 5V tápfeszültség mellett a bemenetükön már 2.4V-tól magas jelet érzékelnek és a kimenetükön ez esetben 5V-ot adnak ki.

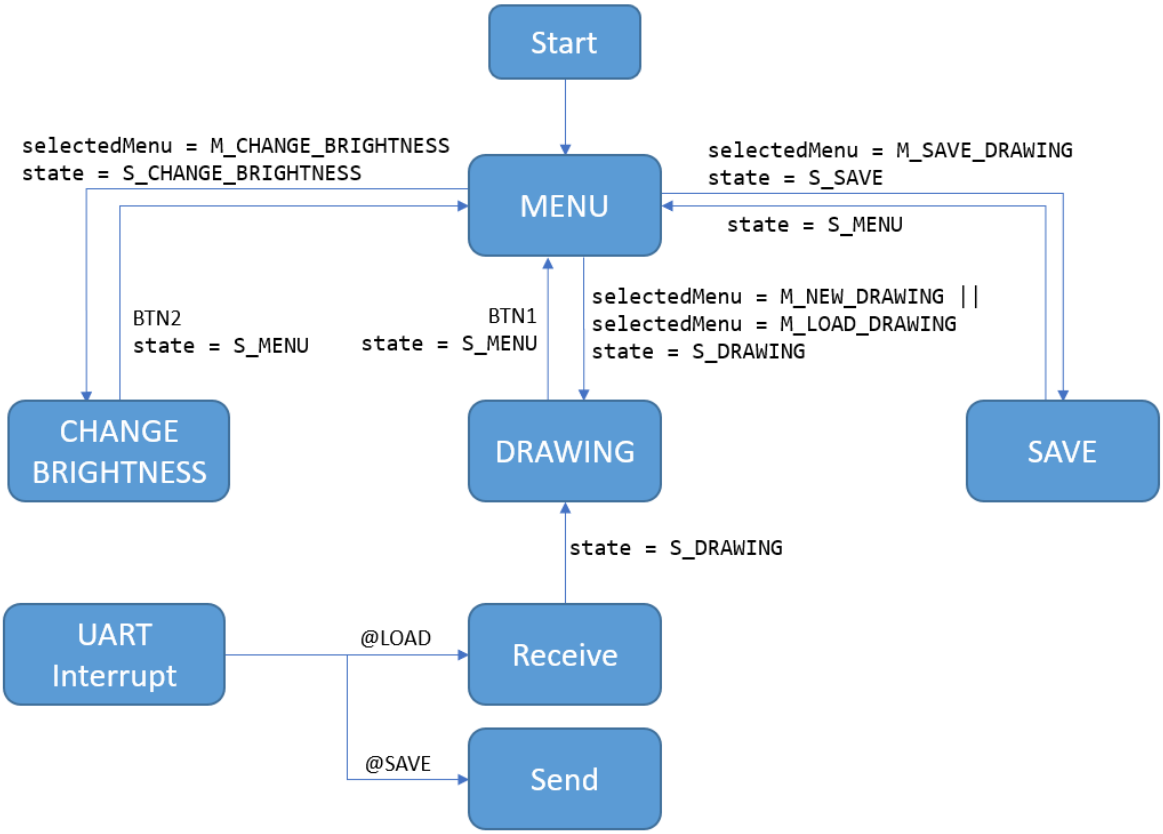
74HCT244D buffer tisztán a szintillesztés miatt kell míg, 74HCT245D ezen felül a kétirányú kommunikáció miatt lett tervezve, hogy a kijelzőből kiolvasni is lehessen. A nucleo azon pinjei melyek az LCD adatlábaival vannak összekötve 5V toleránsak, ezért visszafelé nem kell 3.3V-ra illeszteni a jelet. A sorba kötött ellenállások védelemként szolgálnak, ha véletlenül összehajtanánk a jeleket.

Végül nem használtam ki a kiolvasás lehetőségét a firmware-ben, ezért az adatbemenetekre is elég lett volna egy 74HCT244D buffer.



PCB design

Firmware:



A firmware állapot gép alapján működik. Négy állapot van, melyek között a gombokkal és UART interrupt-tal lehet váltani.

- **MENU:**
  Ez a kezdő állapot, amiben a gombokkal 4 opció közül lehet választani. A választott érték a *selectedMenu* változóba kerül mentésre, majd ez alapján változik az állapot.
- **CHANGE BRIGHTNESS:**
  Itt lehet állítani a kijelző fényerejét a két szélső gombbal. A középső gomb megnyomása során a fényerő elmentődik a flash-be, majd a program visszatér a menübe.
- **SAVE:**
  A memóriában lévő rajzot elmenti a flash-be, majd rögtön visszatér a menübe.
- **DRAWING:**
  Ebbe az állapotba 3 féleképpen is eljuthatunk. *selectedMenu* alapján lehet új rajzot kezdeni, valamint a flash-ből korábban elmentett rajzot betölteni. A harmadik módszer amikor a kliens alkalmazás UART-on keresztül rajzot küld az eszköznek.

Rajzolni a két potméter segítségével lehet. A bal oldali a toll x, míg a jobb oldali az y koordinátáját állítja. A középső gombbal lehet felemelni, illetve letenni a tollat, a bal szélsővel lehet visszatérni a menübe, a jobb szélsővel pedig törölni a kijelzőt.

A kommunikáció a kliens alkalmazással UART-on keresztül történik. Mindig a PC szólítja meg a rajztáblát egy paranccsal. Két féle parancs van melyek szintaktikája @LOAD és @SAVE.

Ha a rajztábla a @LOAD parancsot kapja akkor, az OK visszaigazolás után várja a rajznak megfelelő biteket a kliens alkalmazástól, amiket betölt a memóriába és megjeleníti azt a kijelzőn. Ezek után ezt lehet szerkeszteni is.

@SAVE esetben a rajztábla szintén küld egy OK visszaigazolást, majd a memóriájában lévő rajzot. Ilyenkor bármelyik állapotban lehetünk (akár CHANGE BRIGHTNESS), mindig a memóriában aktuális rajzot fogja küldeni a kliensnek. A PC-s alkalmazás ezek után megjeleníti a vett rajzot, ami rögtön szerkeszthető is.

Menü

Betöltött rajz a flash-ből

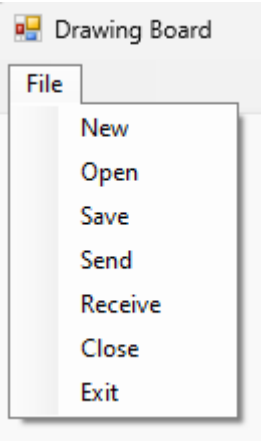
Fényerő állítása

Kliens alkalmazás:

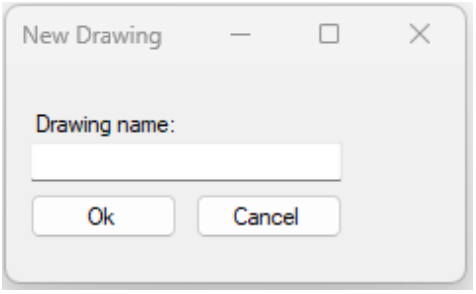
Az alkalmazás egy adott rajz pixeleit kétdimenziós bool tömbben tárolja. Ezt mentésnél, valamint az eszközre küldés előtt egy két dimenziós byte tömbre konvertáljuk a *createBitmapFromDrawing()* függvény segítségével. Így bitenként tudjuk tárolni a pixeleket, amire azért van szükség, mert a firmware memóriájában is ilyen formátumú a rajz, valamint helytakarékosabb is. Egy rajz megnyitásánál, vagy fogadásánál ugyanezt kell visszafelé megcsinálnunk a *createDrawingFromBitmap()* függvénnyel, hogy megjeleníteni, valamint szerkeszteni tudjuk azt.

Az alkalmazás Document-View architektúrát valósít meg. Egyszerre több rajzot is szerkeszthetünk különböző tab füleken. Rajzolni bal klikkel lehet, jobb klikkel radírozni, valamint a C billentyű lenyomásával az egész kijelzőt törölhetjük.

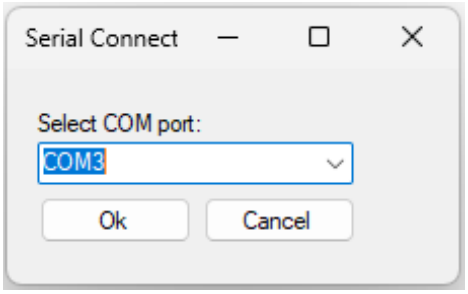
A **File** menüpont alatt 7 almenü található:



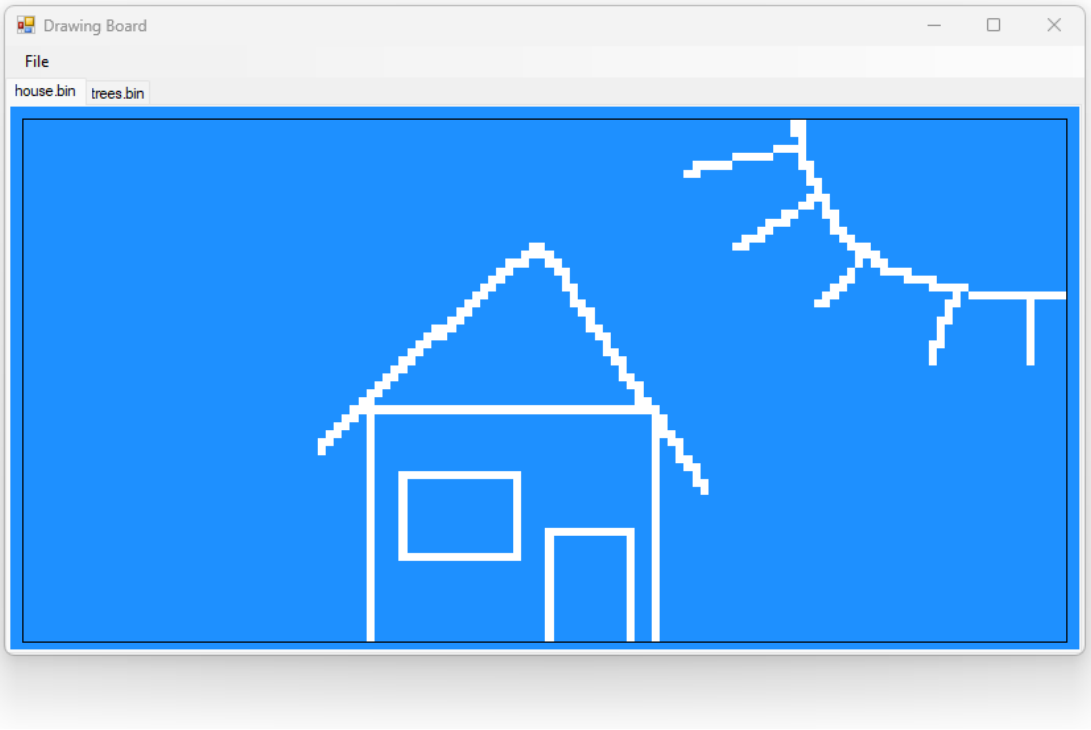
- **New:** Miután a felugró ablakban megadtuk az új rajz nevét megjelenik előttünk egy üres vászon.
- **Open:** A File Explorer segítségével megnyithatunk egy elmentett rajzot.
- **Save:** Az aktuális rajzot, amit épp szerkesztünk a File Explorer segítségével elmenthetjük.
- **Send:** Soros porton való csatlakozás után elküldi a kliens az aktuális rajzot a rajztáblának.
- **Receive:** Soros porton való csatlakozás, és a rajz nevének megadása után a rajztábla elküldi a kliens alkalmazásnak a rajzot, majd az megjelenik a vásznon.
- **Close:** Bezárja az aktuális rajzot.
- **Exit:** Bezárja a teljes programot.



Új rajz létrehozása



Csatlakozás a rajztáblához



Rajzolás különböző tab füleken