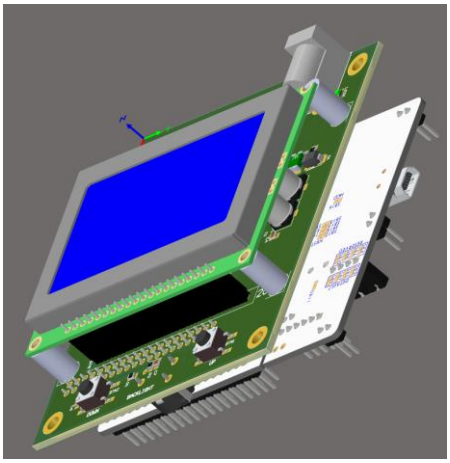
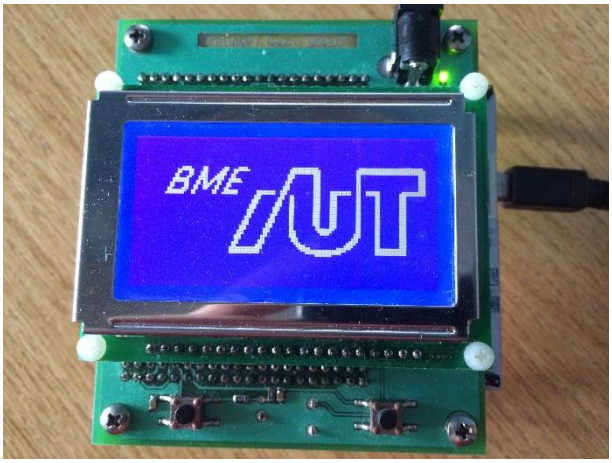
Földvári Dávid-B5PLMF





Feladat: B9, Fényújság grafikus LCD-vel (IMSc)

Készítsen kiegészítő hardver egységet az STM32 NUCLEO-F446RE kithez, amely egy grafikus LCD kijelzőt párhuzamosan illeszt a mikrokontrollerhez! A kijelző háttérvilágítása legyen állítható két nyomógomb segítségével!

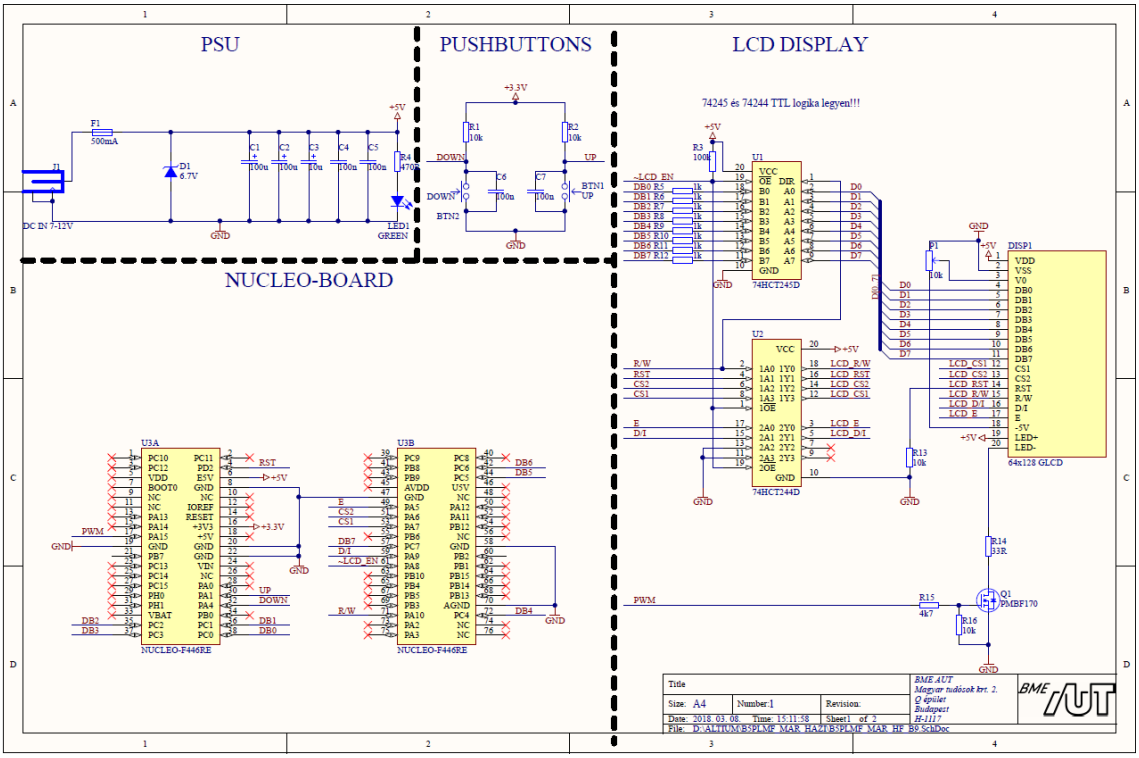
Az áramkör megtervezése, megépítése és üzembe helyezése után készítsen el egy olyan egyszerű alkalmazást, amely reklámfelületként egyetemünket, karunkat, szakirányunkat vagy ágazatunkat népszerűsíti grafikus fényújságként! Tegye lehetővé az indítás utáni alapértelmezett grafikákon felül egyéni képek letöltését PC-ről. A PC-vel történő kommunikációhoz virtuális soros portot használjon, melyet a kiten megtalálható USB port segítségével valósítson meg!

Az áramkör megtervezése, megépítése és üzembe helyezése után készítsen el egy, az eszköz bemutatására szolgáló demonstrációs célú tesztprogram rendszert, amely magában foglalja a megfelelő működést biztosító mikrokontrolleres programot, illetve a PC-s kliensprogramot.

Többletfeladat IMSc pontokért:

Hozzon létre a kliensprogramon belül egy animáció szerkesztő alkalmazást, amellyel bármely felhasználó képes lehet különböző animációk létrehozására. Elképzelhető funkciók: futó szöveg állítható iránnyal és sebességgel; animáció képkockánkénti megrajzolása; statikus képekből álló „slideshow”, különböző áttűnési lehetőségekkel.

A kiegészítő hardver



A táp bemenet 500mA-es rövidzár védelmet és egy supressor-diódás túlfeszültség védelmet kapott. az 5V-os tápfeszültségre több darab elektrolit puffer és kerámia szűrőkondenzátor került, amelyekből nem építettem be mindet. A hibakeresések megkönnyítésére egy tápfeszültség jelző LED-et is elhelyeztem az 5V-ra. A Nucleo kártya External 5V-os bemeneten keresztül kapja a tápfeszültségét.

A fényerő PWM állításához elhelyeztem 2 nyomógombot, ezek alacsony aktívak (a bemenetek így kevesebbet fogyasztanak).

Az LCD modul meghajtása 74HCT244 és 74HCT245 busz meghajtó IC-vel van megoldva. A modul 5V-os vezérlőjeleket igényel, az STM32 mikrokontrollerek pedig 3.3V-os logikai szintet tudnak kiadni. A HCT sorozatú TTL kompatibilis CMOS így a 3.3V-os meghajtó jeleket az LCD bemenetén 5V-ra illeszti.

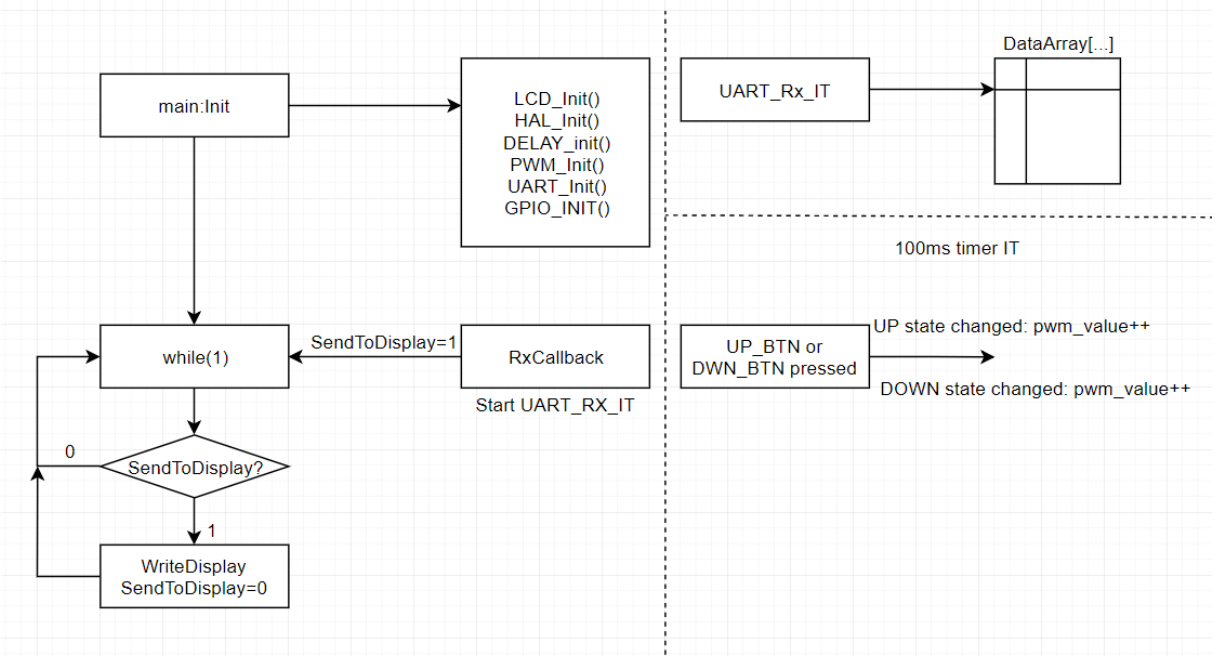
A teljes LCD-meghajtó logika kimenete egy negált jellel(~LCDEN) engedélyezhető. A BD0..7 párhuzamos adat jelek 1kOhm-os ellenálláson keresztül kapcsolódnak a két irányú busz meghajtó IC-hez a kimenetek szembekapcsolásából eredő hibák elkerülése miatt. Az R/W bemenettel írást/olvasást jelezhetünk az LCD modul felé. Az RST jellel resetethetjük az LCD-t. Alapvetően GND-re húzza az R13 ellenállás, programból kell logikai 1-est adni rá, ekkor szűnik meg a reset állapota. A modul 2db 64x64pixel meghajtására képes IC-vel valósítja meg a 128x64pixel felbontást, a CS1 és CS2 bemenetek ezeknek az IC-knek a vezérlő jelei. Az E (enable) engedélyezésre szolgál, minden LCD írási és olvasási ciklushoz szükséges. A D/I jelzi a vezérlő IC-knek hogy adat vagy parancs byte van a DB0..7 párhuzamos buszon.

A kijelző kontrasztját egy potenciométerrel lehet beállítani, amely az LCD modul -5V és GND lába közötti feszültséget osztja le a V0 bemenetre.

A kijelző háttérvilágításának fényerejének állítását egy N csatornás FET-tel oldottam meg. A FET gate-jén egy feszültségosztó van, amely a Nucleo kártya egyik PWM lábára csatlakozik.

Az R14 előtét ellenállást az $R = \frac{5V - U_{LED}}{I_{LED}}$ képlettel méreteztem.

A működtető szoftver felépítése



TIM2: CH1, PA15 GPIO kimenet 1kHz-es PWM jelét állítja elő pwm_value értéke alapján.

TIM3: A fényerő növeléséhez használt gombok pergésmentesítéséhez szükséges +/- 10 százalékonként állítható a kijelző fényereje.

TIM4: mikroszekundumos késleltető funkciót valósít meg az LCD kezelő függvények időzítéséhez.

Főprogram: Inicializálja a perifériákat, GPIO, LCD, UART, PWM, megszakításokhoz prioritást rendel, elindítja a beépített .c és .h párokban található konstans blokkokban tárolt képek vetítését a kijelzőre, a GLCD_WriteDisplay() függvény segítségével.

UART2 Soros porton fogadott frame (1024 byte) után átvált soros porton érkező kép(ek) vetítésére.

A program dupla bufferelléssel dolgozik, az aktuálisan érkezett 1024 byte és a kijelzőn megjelenített 1024 byte egy 2048 hosszú tömb egy-egy felén helyezkedik el. 2 pointer cserélgetésével valósul meg az adatok átadása a megszakítás és az LCD kezelő függvény között, illetve egy bool típusú változóval jelezzük a while ciklusban, hogy új adat érkezett. Tényleges adatmozgatás az UART-on érkező adatok mentése, illetve a kijelzőre küldése közben az adatblokkból való kiolvasása közben történik.

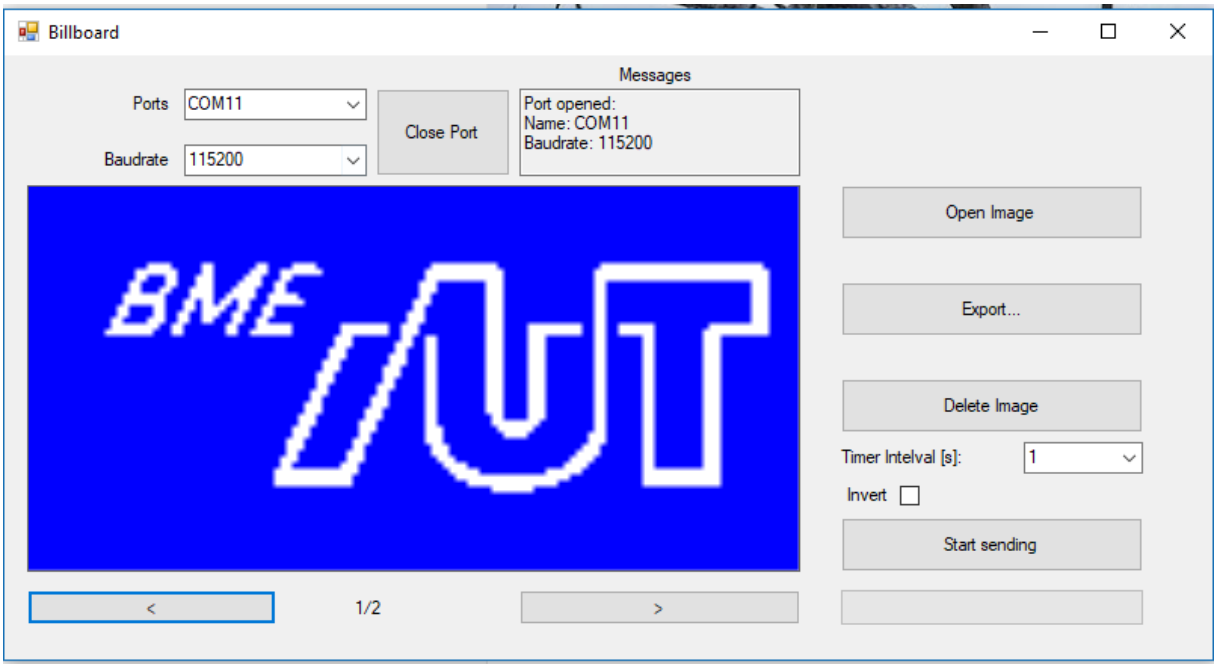
Kommunikáció: Inicializálás után a HAL library egyszerű lehetőséget ad arra, hogy meghatározott számú byteot fogadjunk. Ezt nem blokkoló módon (megszakítással) az alábbi megoldással valósítom meg:

Itt is látszik a dupla bufferelés. Minden megfelelő UART frame után pointert cserélek és jelzem a főprogramnak, hogy kirajzolhatja a legutóbb kapott képet a kijelzőre.

Működés közben bármikor előfordulhat hiba. Ha ez megtörténik, akkor a következő vételnél overrun error miatt meghívódik a hibakezelő callback függvény, ahol jelzem a kliensalkalmazásnak hogy hiba történt, majd a mikrokontrolleren újraindítom az adatok fogadását.

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    if (huart->Instance == USART2)
    {
        uint8_t *tmpPtr=SerialReceiveBufferPtr;
        SerialReceiveBufferPtr=DisplayBufferPtr;
        DisplayBufferPtr=tmpPtr;
        DisplayMode=true;
        sendtodisplay=true;
        HAL_GPIO_TogglePin(GPIOA,LD2_Pin);
        HAL_UART_Transmit_IT(&huart2, (uint8_t*)"OK", 2);
        HAL_UART_Receive_IT(&huart2,SerialReceiveBufferPtr,1024);
    }
}
void HAL_UART_ErrorCallback(UART_HandleTypeDef *huart){
    if(huart->ErrorCode == HAL_UART_ERROR_ORE){
        sendtodisplay=false;
        HAL_UART_Transmit_IT(&huart2, (uint8_t*)"FAIL", 4);
        HAL_UART_Receive_IT(&huart2,SerialReceiveBufferPtr,1024);
    }
}
```

PC-s alkalmazás

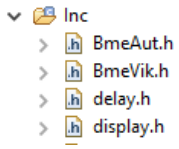


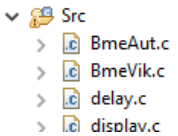
Az alkalmazásban sokféle funkciót megvalósítottam, saját megítélésem szerint többet mint az alap elvárás, de valamivel kevesebbet mint a kiírásban többletfeladatként megfogalmazott működés.

Mivel mindenképpen szükség volt virtuális soros portra ezért a felület fejlécében ezt a funkciót valósítottam meg, egy-egy legördülő menüből választható ki a port száma és a baudrate.

Az Open Image gombra kattintva megkereshetjük, előzőleg valamilyen képszerkesztő programmal létrehozott (GIMP) 64x128 1bites színmélységű képeinket. Ha a kép megfelel a ezeknek a követelményeknek, akkor a bal oldalon található panelen olyan formában jeleníti meg ahogyan a kijelzőre fog kerülni. További képeket adhatunk hozzá az Open Image gombbal.

Az Export… gombbal az aktuálisan láthatós képből generálhatunk *.c és *.h fájlokat, amelyek teljesen készek arra, hogy használjuk őket, csak hozzá kell adni a System Workbench-ben a mikrokontrolleres projektünkhöz.






```
/* USER CODE BEGIN Includes */
#include "stdbool.h"
#include "display.h"
#include "BmeAut.h"
#include "BmeVik.h"
/* USER CODE END Includes */
```

Majd ezután meghívni a GLCD_WriteDisplay(&BmeAutDataArray[0]); függvényt.

A Delete Image törli az éppen látható képet a képek listájából.

Az Invert checkbox-szal a színeket cserélhetjük fel.

A Send gomb egy kép esetén egyszer elküldi az eszköznek a kép adatait, az megjeleníti és egészen addig marad amíg nem küldünk neki mást. Egynél több kép esetén viszont megjelenik egy legördülő menü a Timer Inverval felirat mellett, ahol beállíthatunk 1…10 másodperc közötti értékeket. Ilyen időközönként küldi el a következő képet az eszköznek. Ha elértük az utolsó képet, akkor kezdi újra az elejéről. Egészen addig küldi, amíg nem állítjuk le ugyanezzel a gombbal.

A kép előnézeti ablaka alatti < és > gombokkal a megnyitott képek között váltogathatunk.

Működés közben:

Tapasztalatok a feladat megoldása során:

A hardver megtervezése nem okozott gondot, mivel kevés alkatrészt igényel és egyszerű. Ráadásul a tanszéki minta minden LCD kijelzős feladathoz erős ajánlás volt, lényegében azt másoltam le. Több időt vett igénybe a kliensalkalmazás felépítése és megvalósítása. A képek feldolgozása, manipulálása a legtömörebb információ kinyerése a képekből, stb. A mikrokontroller programjához nagy segítség volt a CubeMX környezet, a HAL library, és az hogy több gyakorlatom volt beágyazott fejlesztésben mint .NET-ben.

Amit utólag másképp csinálnék:

- A hardvert minimálisan módosítanám, terveznék a DC tápcsatlakozó mellé egy reset gombot, hogy ne kelljen mindig az eszköz alján keresgélni.
- A DC tápcsatlakozót az USB csatlakozó irányával megegyezően tervezném rá.
- A Nucleo-t egészen a NYÁK szélére raknám így teljesen szimmetrikus lenne és nem lógna ki egyik irányba sem túlságosan.
- A kliensalkalmazásban megvalósítanám a rajzolás funkciót egy egyszerűsített paint kinézettel, 1 színű tollal és szöveg elhelyezés funkcióval. Ezt valószínűleg hobbiból meg is valósítom majd.

Földvári Dávid - B5PLMF, 2018