Hangulatvilágítás

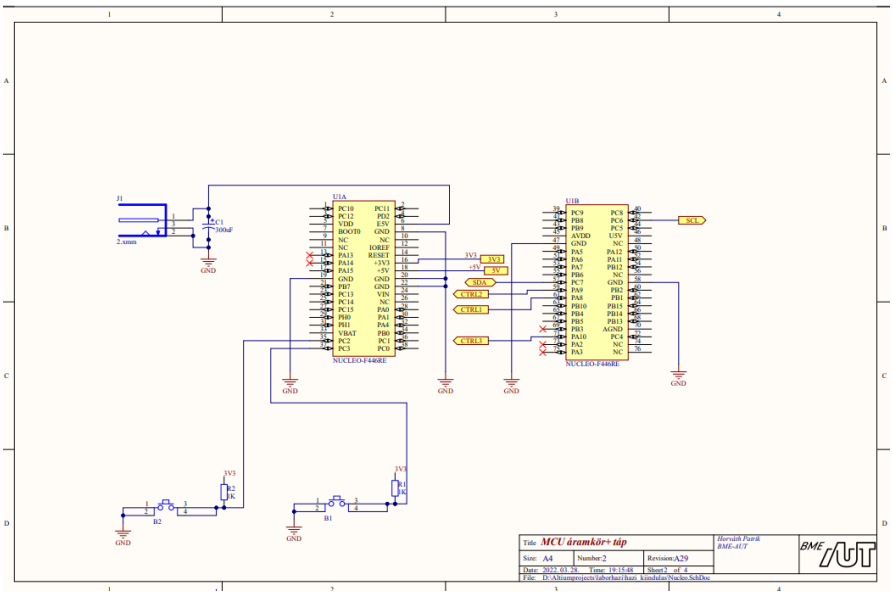
Horváth Patrik

Feladat:

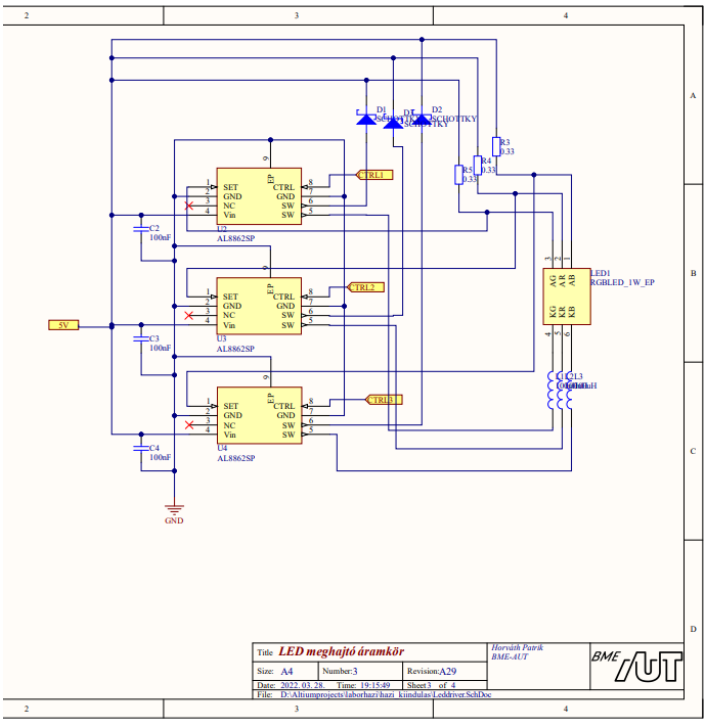
Készítsen kiegészítő hardver egységet az STM32 NUCLEO-F446RE kithez, amely programozható hangulatvilágítást valósít meg három nagy fényerejű RGB LED segítségével. A végrehajtható világítás-programok PC-n szerkeszthetőek legyenek, a hardver soros porton kommunikáljon a PC-vel. A soros kommunikációhoz virtuális soros portot használjon, melyet a kiten megtalálható USB port segítségével valósítson meg! Az így létrehozott programokat tárolja egy I2C EEPROM vagy SPI Flash memóriában. Az eszköz legyen képes programozás nélkül is hangulatfény programot generálni, amely jellege egy nyomógomb segítségével állítható. Ugyanezen gomb segítségével tudjon váltani a letöltött világítás-programok között. 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 egy PC-s kliensprogramot.

PCB

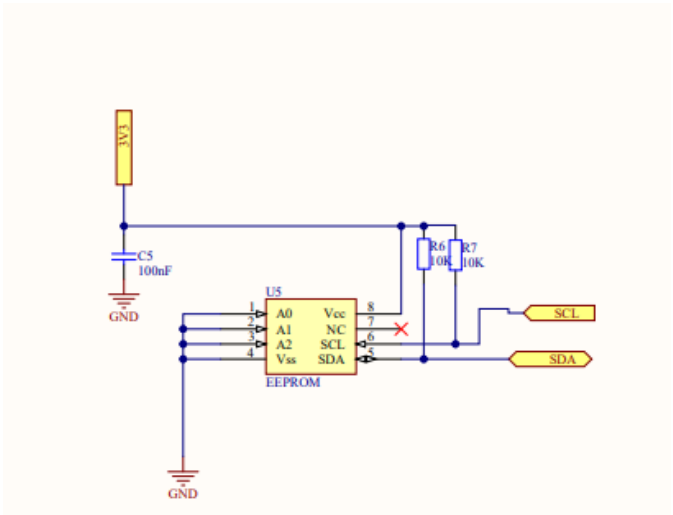
A tápot egy 300uF-os alu-elko szűrőkondenzátoron keresztül csatlakoztattam a mikrokontroller E5V lábához, a két nyomógombot egy-egy 1 kOhm-os ellenálláson keresztül felhúztam 3,3V-ra.



A ledek vezérlését Buck típusú LED Driverekkel valósítottam meg, amelyeknél a ledek áramát 100%-os kivezérlés mellett egy ellenállással lehet beállítani I=0.1/R A, ebből a képletből a közel 350mA beállításához 0,33 Ohmos ellenállásokra volt szükség. A fényerő vezérlését a CTRL lábra kötött PWM jellel lehet megvalósítani, amely nálam a TIMER1 1-es,2-es és 3-as csatornáiról érkeznek.



Az adatok tárolásához egy EEPROM-ot használtam, amely I2C-n keresztül képes adatokat küldeni illetve fogadni. Az A0-A2 lábait földre kötöttem így a 0xA0 címen lehet megcímezni. Az SDA és SCL lábakat egy-egy 10kOhmos ellenálláson keresztül fel kell húzni a tápfeszültségre az I2C működési elve miatt.



Mikrokontroller firmware

A főciklusban elindítjuk a timereket, amelyeket a Ledek PWM beállítására, valamint a gombok pergésmentesítésének megoldására használunk. Ezenkívül folyamatosan meghívjuk az uartkezelő függvényt és ha szükséges akkor lekezeljük az érkezett adatokat. Az UART-on keresztül 19 bájt érkezik, és az első alapján eldönthetjük, hogy mit szeretnénk csinálni a többivel. Az utolsó két bájt az ellenőrző összeg felső és alsó 8 bitje. Ennek segítségével könnyedén ellenőrizhetjük, hogy érvényes adatokat fogadtunk-e, és ha igen akkor végrehajtjuk az utasításokat.

```
switch(lastrecievedbyte){
case '@':                                   //új animáció hozzáadásához tartozó kód
        for(int i=1;i<17;i++){
            sum+=rxBuffer[i];
        }
        if(sum==controlamount){             //ha megegyezik az ellenőrző összeg akkor c
            animindex=rxBuffer[16];
            seteeprom(animindex);
            char letter[]={"OK"};
            HAL_UART_Transmit(&huart2, (uint8_t*)letter, 2, 100);
        }
        else{
            char letter[]={"ERROR"};            //ha nem így van akkor visszaküldjük a
            HAL_UART_Transmit(&huart2, (uint8_t*)letter, 5, 100);
        }
        break;
    case 'c':                                   //animáció törléséhez tartozó kód
        for(int i=1;i<17;i++){
            sum+=rxBuffer[i];
        }
        sum+='c';
        animindex=rxBuffer[16];
        if((animindex!=8)&&(sum==controlamount)){       //ha megegyezik az ellenőrző ös
            cleareeprom(animindex);
            char letter[]={"OK"};
            HAL_UART_Transmit(&huart2, (uint8_t*)letter, 2, 100);
        }
        else{                                           //ha nem így van akkor visszakü
            char letter[]={"ERROR\n\r"};
```

Nemutolsósorban pedig az animation függvényben megvizsgáljuk, hogy épp az eepromból vagy pedig előre megírt animációk közül választunk és ez alapján beállítjuk a ledeket. A kiválasztást a két nyomógombbal tudjuk elvégezni.

main.c

```
836 void animation(){
837     if(interruptaccepted)
838     {
839         /*char letter[]={"Gomb lenyomva\n\r"};
840         HAL_UART_Transmit(&huart2, (uint8_t*)letter, 16, 100);
841         HAL_UART_Transmit(&huart2, (uint8_t*)&lastrecievedbyte, 1, 100);*/
842         interruptaccepted=false;
843         interruptrecieved=false;
844     }
845     if(!eepromanimON){          //ha nem az eepromról akarunk játszani animációt
846     switch(animcounter){
847     //case 0:blackout(); return;
848     case 1:blue(); return;
849     case 2:green(); return;
850     case 3:moving();return;
851     }
852     }
853     else{
854         GetEEPROManim(eepromanimcounter);   //megkeressük a következő animációt
855         switch(eepromanimcounter){
856             case 0:eepromanimsetup(eepromanimcounter);return;
857             case 1:eepromanimsetup(eepromanimcounter);return;
858             case 2:eepromanimsetup(eepromanimcounter);return;
859             case 3:eepromanimsetup(eepromanimcounter);return;
860             case 4:eepromanimsetup(eepromanimcounter);return;
861             case 5:eepromanimsetup(eepromanimcounter);return;
862             case 6:eepromanimsetup(eepromanimcounter);return;
863             case 7:eepromanimsetup(eepromanimcounter);return;
864             }
```

A moving valamint az eepromanimsetup olyan függvények amelyeken belül addig vagyunk egy while ciklusban, amíg az uarton nem keresztül kapunk valamilyen üzenetet amit le kell kezelni vagy a nyomógombokkal nem választunk másik animációt.

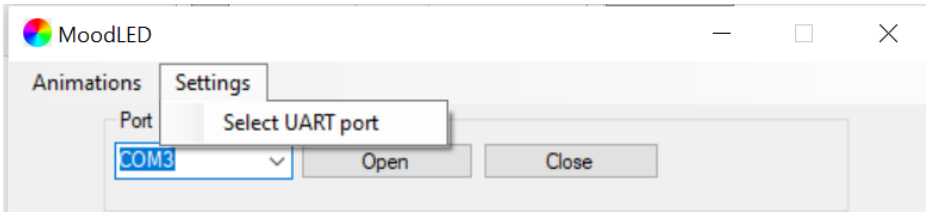
```
while(((eepromanimcounter==eanimcounter)&&eepromanimON)&&!uartrecieved){
```

A program úgy van megírva, hogy az EEPROMon lévő animációk olyan sorban kövessék egymást, mint ahogy azok a kliensalkalmazásban követik egymást, ezért ha törlünk egy animációt akkor az azt követőeket eggyel előrébb shifteljük.

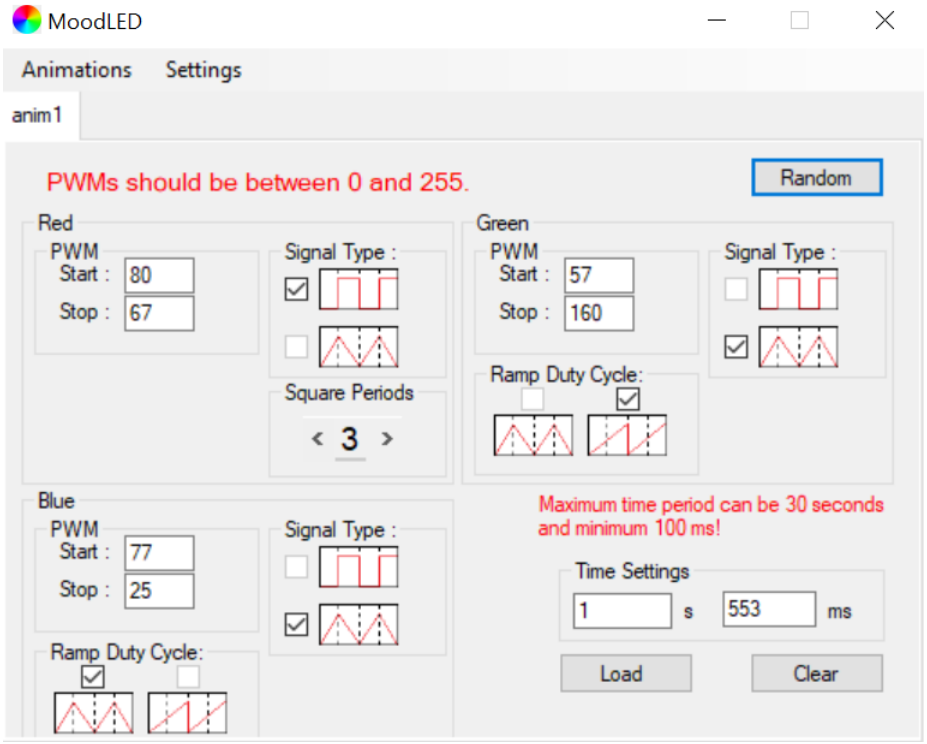
```
//kitörli a paraméterként megkapott eeprom oldalt
void cleareeprom(uint8_t index){
    uint8_t tmp[16]={0};
    if(index!=7){
        for(int i=index;i<7;i++){              //ha nem az utolsó oldalt töröljük, akkor az azt kö
            int pos=EEPROM_PAGESIZE*i;
            HAL_FMPI2C_Mem_Read(&hfmpi2c1, EEPROM_ADDRESS, pos+EEPROM_PAGESIZE, 1, tmp,EEPROM_PAGES
            HAL_FMPI2C_Mem_Write(&hfmpi2c1, EEPROM_ADDRESS, pos, 1, tmp,EEPROM_PAGESIZE, 1000);
            HAL_Delay(5);
        }
    }
    else{
        int pos=EEPROM_PAGESIZE*index;
        HAL_FMPI2C_Mem_Write(&hfmpi2c1, EEPROM_ADDRESS, pos, 1, tmp,EEPROM_PAGESIZE, 1000);
        HAL_Delay(5);
    }
    eepromanimON=false;
    eepromanimcounter=0;
    animcounter=1;
}
```

Kliensalkalmazás

Az alkalmazás megnyitása után a Settings fülön belül kiválaszthatjuk hogy melyik csatlakoztatott Portot szeretnénk használni a soros adatátvitelhez. A portokat frissítő programrészt egy timer event hív meg.

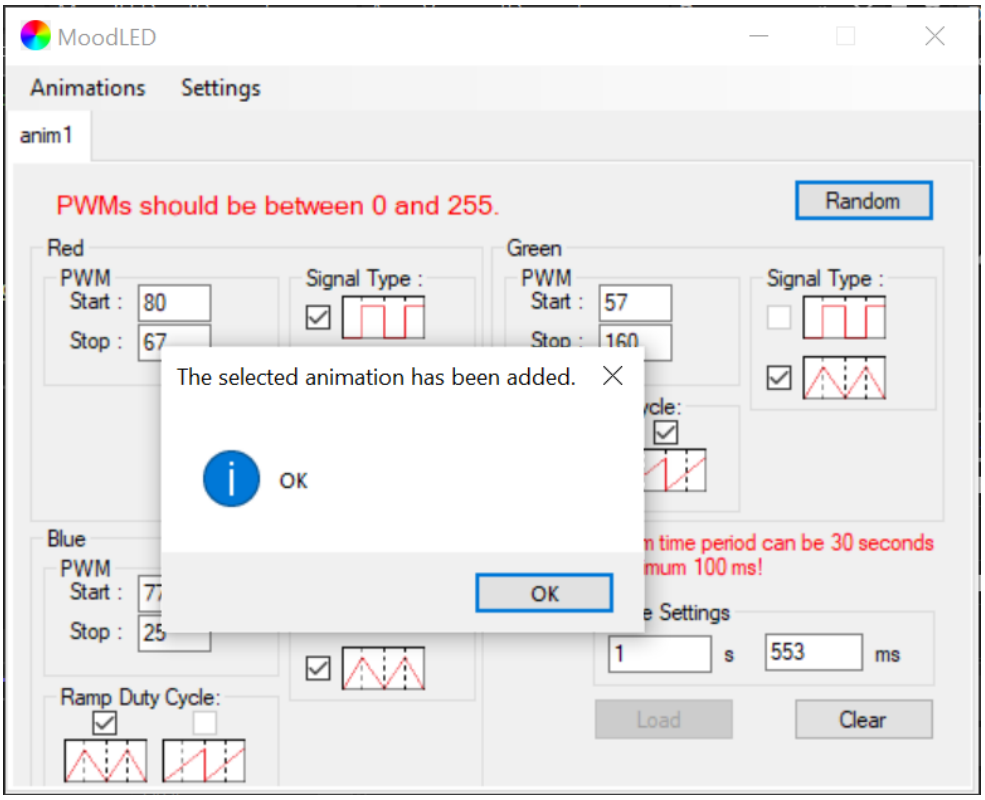


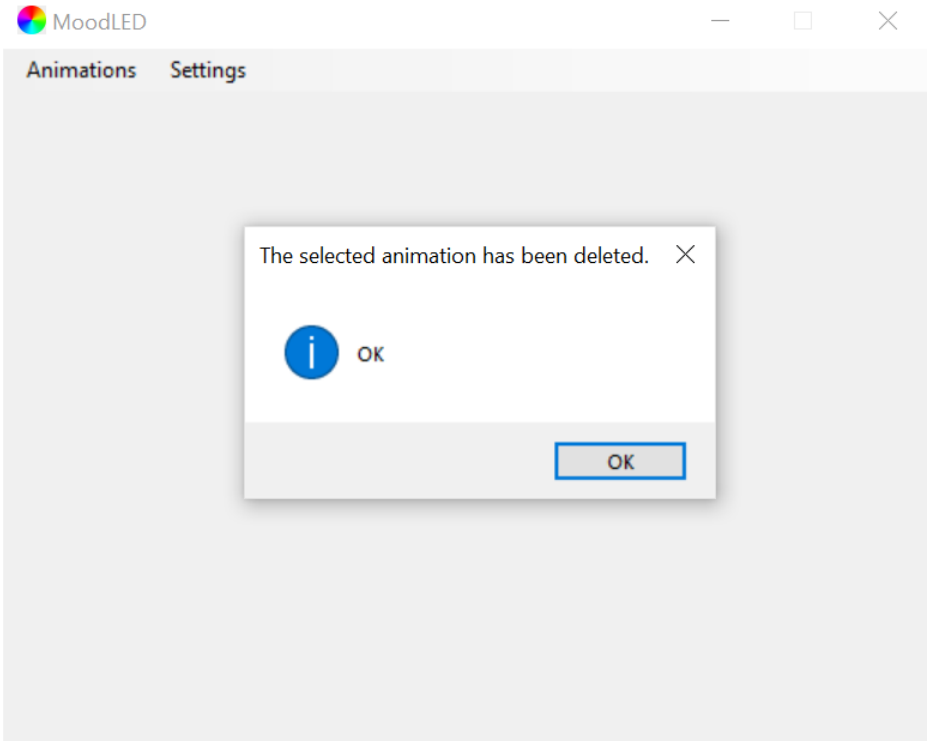
Ha a csatlakozás után készítünk egy új animációt az Animations>>New menüfül alatt akkor a Load gombunk elérhetővé válik.



Itt a Random gombot használva véletlenszerű értékeket tölthetünk be a programba vagy akár manuálisan is átírhatjuk az értékeket benne. Fontos hogy minden négyzetbe írni kell valamilyen számot/ be kell pipálni különben hibát kapunk. A Clear gombbal a beírt értékeket törölhetjük.

A Load gombot használva elküldhetjük a mikrokontroller felé a beállításokat amely-re kapunk valamilyen választ attól függően hogy a művelet hogyan sikerült.





Összegzés

A feladat megoldását hamarabb kezdtem volna, hogy több időm legyen a szoftver fejlesztésére és az apróbb hibák kiküszöbölésére. A gombokat hardveresen pergésmentesíteném, hogy azzal már ne kelljen foglalkozni a szoftverfejlesztésnél, illetve többször is átellenőrizném a layout tervet, mert kialakult 1-2 sziget amit még a forrasztás után meg kellett oldani. A minimumnál több IC-t rendelnék

és nemcsak a kisebb ellenállásokból, ha esetleg valamilyen hiba fordulna elő akkor ki lehessen cserélni.

A számítógépes alkalmazás továbbfejlesztése lehet például a több szál használata.