Budapesti Műszaki és Gazdaságtudományi Egyetem

ARM Cortex magú mikrovezérlők

Földvári Dávid

MÁTRIX KIJELZŐS ÓRA NULCEO KÁRTYÁVAL

BUDAPEST, 2018

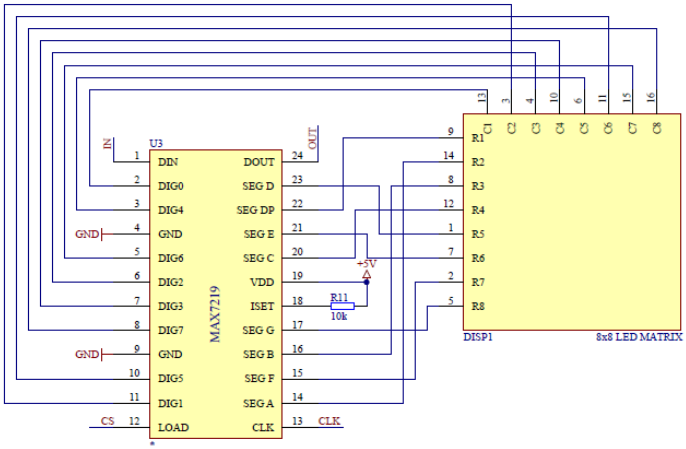
1. A motiváció.

Szerettem volna egy órát, amely nem hagyományos hétszegmenses kijelzőn jeleníti meg az időt, hanem valami olyanon, amelyre mást is ki lehet jelezni. LCD kijelzőt nem szerettem volna alkalmazni, de a buszokról jól ismert mátrixos kijelzők mindig is tetszettek. Szerencsére léteznek erre a feladatra alkalmas modulok és meghajtó IC-k. A Maxim MAX7219 típusú SPI buszos meghajtójára esett a választás. A gombos időbeállítás helyett kényelmesebb megoldás kellett, ezért UART-on küldöm el a beállítani kívánt időt és dátumot.

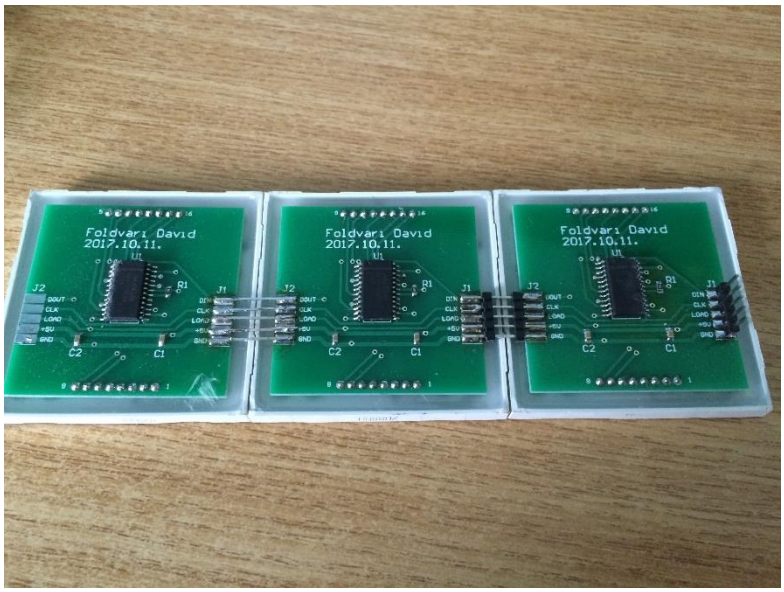
2. A kapcsolási rajz.

A felhasznált modulokat szabadidőmben hobbiból terveztem az őszi félévben. Most a tárgynak köszönhetően kipróbálhattam azt is, hogy az egyes modulokat kaszkádosítva is ugyanúgy működik mint egyesével. Szerencsére igen.

A kijelző meghajtása:



A meghajtó IC adatlapi ajánlása szerint kötöttem be a LED mátrix modult. Fontos erre figyelni, ha nem akarjuk, hogy később a programban a kijelezni kívánt adatokat bitenként kelljen összepakolni. Az R11 ellenállással állítható be a maximális áram, amit a kimenet a LED-ekre kapcsolhat. Egy ilyen LED modul 15-20mA-es áram mellett már maximum fényerővel világít. A kapcsolás kaszkásodítható: DIN_2 ->$DOUT_1$. A CS (Chip Select) jelet minden meghajtó közösen kapja meg, ez a MAX7219-nél LOAD-dal van jelölve. A CLK (SPI clock) jelet szintén minden modul megkapja közösen. 3 kijelzőt és 3 meghajtót használtam fel. Az utolsó meghajtó kimenetét szabadon kell hagyni.



3. A működtető szoftver.

STM32L476RG típusú Nucleo kártyán valósítottam meg a feladatot. Ebben van beépített RTC periféria, amit CubeMX segítségével konfiguráltam fel. Ezen kívül használok még SPI perifériát, amelyen a MAX7219-es IC-kkel kommunikálok. Az UART-kommunikációhoz felkonfiguráltam a Nucleo kártya USB-illesztőjéhez kapcsolódó UART perifériát megszakításos adatfogadáshoz.

Az RTC másodpercenként kivált egy megszakítást.

```
void HAL_RTC_AlarmAEventCallback(RTC_HandleTypeDef *hrtc){
    Point=!Point;
    if(Mode==Time){
        CreateFrameFromTime();
        seconds++;
    }
    if(seconds==10){
        CreateDateData();
        CreateDisplayDataArray(TextArray);
        Mode=Date;
        seconds=0;
    }
}
```

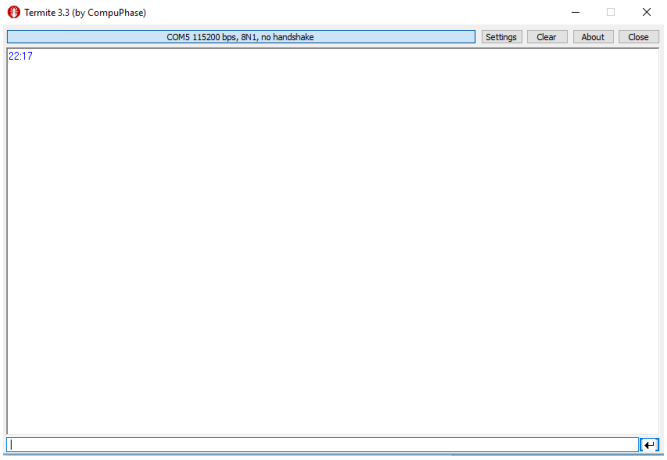
A kijelzés két módban lehet. Date vagy Time. Amikor Time módban van, akkor az aktuális időt jeleníti meg a kijelzőn másodpercenként villogó kettősponttal. Minden másodpercben növelünk egy second változót és ha elérünk egy adott értéket (itt 10), akkor átváltunk Date módba. Ilyenkor a CreateDateData() függvény a TextArray-ben létrehozza azt a szöveget amely magában hordozza a dátumot. Én ide a „A mai dátum: YYYY.MM.DD.,Weekday” formátumot találtam ki. Ezt a szöveget fogja léptetni a kijelzőn. A CreateDisplayDataArray() függvény a neki megadott szövegből (TextArray benne az előtte kiolvasott dátummal) létrehozza az oszlopokat amelyek a kijelzőre kerülnek. Ez a tömb egy nagy méretű tömb mivel minden betű 6 bájtos (egy ASCII karakter 6 oszlopból áll) ezért 6-szor akkora, mint a TextArray, amely 256 karakteres szövegeket tartalmazhat maximum. Az adatok előállítása után engedélyezem a scrollozást, ezzel jelezve a TIM3 megszakításnak, hogy léptetheti az adatot a kijelzőn.

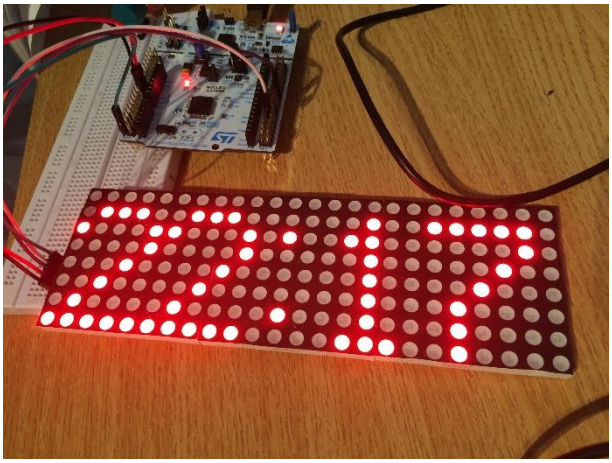
```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
    UNUSED(htim);
    if(htim->Instance==TIM3){
        if (ScrollText) {
            if (StartFrom == ((TextLength * 6) - 24)) {
                ScrollText = false;
                Mode=Time;
                CreateFrameFromTime();
            } else {
                SendToDisplay(StartFrom);
                StartFrom++;
            }
        }
    }
    if(htim->Instance==TIM4){
    }
```

Amikor a szöveg végére érkezünk, akkor vissza kell váltani Time módra. Ameddig ez nem történik meg addig, minden megszakításban egy 24 oszlopos ablakot mozgatunk a kijelzőre kikerülő adatokon. Így mindig egyet lép előre a szöveg a kijelzőn az utolsó oszlopban belép az adat, az első oszlopból pedig eltűnik. Régebben próbálkoztam AVR-rel megvalósítani egy hasonló feladatot, de ott a kijelző multiplexelését is a mikrokontroller végezte (itt ugye nem, hiszen erre van a MAX7219).

Továbbá ott nem tudtam ilyen hatékonyan megoldani a léptetést. Itt nagyon egyszerű a viszonylag nagy méretű belső memória miatt és nem kell folyamatosan előállítani a kijelzőre kiírni kívánt adatot, hanem csak egy mozgó ablakot végig léptetni.

Az idő beállítását UART-on végzem egy fix formátum szerint.





```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart){
    if (huart->Instance == USART2)  {
        if('0'<=UartBuff[0]&&UartBuff[0]<='9' &&'0'<=UartBuff[1]&&UartBuff[1]<='9'&&UartBuff[2]==':'&&
                '0'<=UartBuff[3]&&UartBuff[3]<='9'&&'0'<=UartBuff[4]&&UartBuff[4]<='9')
        {
            RTC_TimeTypeDef Time;
            RTC_DateTypeDef Date;
            Time.Hours=(UartBuff[0]-'0')*10+(UartBuff[1]-'0');
            Time.Minutes=(UartBuff[3]-'0')*10+(UartBuff[4]-'0');
            Time.DayLightSaving=RTC_DAYLIGHTSAVING_NONE;
            HAL_RTC_SetTime(&hrtc,&Time,RTC_FORMAT_BIN);
            HAL_RTC_GetDate(&hrtc,&Date,RTC_FORMAT_BIN);
            HAL_RTC_GetTime(&hrtc,&Time,RTC_FORMAT_BIN);
        }
        HAL_UART_Receive_IT(&huart2,UartBuff,5);
    }
}
void HAL_UART_ErrorCallback(UART_HandleTypeDef *huart){
    if(huart->ErrorCode == HAL_UART_ERROR_ORE){
        HAL_UART_Receive_IT(&huart2,UartBuff,5);
    }
}
```

4. Tapasztalatok, konklúzió, további ötletek.

- Ha több időm lett volna beüzemeltem volna egy fényerő szenzort, amellyel beállítható a fényerő.
- Tovább szeretném fejleszteni az őszi eszközépítő versenyre, nem csak 3 modulból, hanem felhasználni mind a 12 ilyen modulomat, hogy egy 80 centis kijelzőt kapjak, amire időjárás adatokat vagy éppen hallgatott zene címét kiírhatom.

Földvári Dávid

2018