Persistence Of Vison

Nyiri Levente (ULYHQH)

Feladat pontos kiírása:

Készítsen kiegészítő hardver egységet az STM32 NUCLEO-F446RE kithez, amely segítségével ábrákat lehet megjeleníteni kihasználva a szem tehetetlenségét, feltéve, hogy megfelelő sebességgel mozgatjuk a panelt. A hardveren helyezzen el minimum 16 db LED-et egy sorban. Érzékelje gyorsulásérzékelővel a panel elmozdulását, és ennek függvényében egy szöveget vagy képet jelenítsen meg a LED-ek segítségével. A LED-ek vezérlése a mikrokontroller SPI buszán keresztül történjen (sorosan), léptető regiszter alkalmazásával. 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. A mikrokontroller és a PC közötti kommunikációt virtuális UART kommunikációval valósítsa meg USB-n keresztül.

A hardware felépítése:

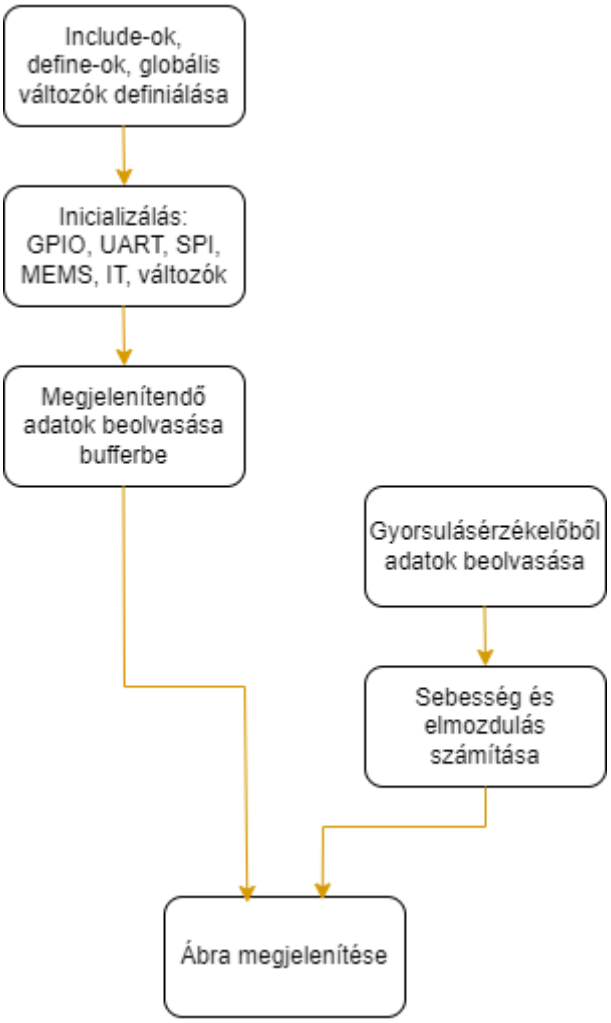
A Nucleo SPI1-évl kötöttem össze a gyorsulásérzékelővel, az SPI2-vel pedig a shift-regiszterekkel. Pontosabban az OE, LE és CLK jeleket az összes shift-regiszterre rákötöttem, az SDI-t pedig csak a legelsőre, ezután a regiszterek SDO-ját kötöttem tovább a számon következő SDI-re (daisy chain). Mivel a shift-regiszterek áram kimenetűek, ezért egyszerűen tudom biztosítani a 20mA-es forward currentet a ledek számára, ha be akarom kapcsolni őket. Ahhoz, hogy 20mA-es legyen az áram a led-vezérlők outputján, az R-EXT-re egy 1kOhm-os ellenállást kellett elhelyezni.

Symbol	Parameter	Test conditions	Min	Typ	Max	Unit
V_{IH}	Input voltage high level		0.7 V_{DD}		V_{DD}	V
V_{IL}	Input voltage low level		GND		0.3V_{DD}	V
I_{OH}	Output leakage current	V_{OH} = 20 V		0.5	10	µA
V_{OL}	Output voltage (Serial-OUT)	I_{OL} = 1 mA		0.03	0.4	V
V_{OH}	Output voltage (Serial-OUT)	I_{OH} = -1 mA	V_{OH} - V_{DD} =- 0.4 V			V
I_{OL1}	Output current	V_O = 0.3 V, R_{ext} = 3.9 kΩ	4.25	5	5.75	mA
I_{OL2}		V_O = 0.3 V, R_{ext} = 970 Ω	19.4	20	20.6	
I_{OL3}		V_O = 1.3 V, R_{ext} = 190 Ω	97	100	103	

Egy shift-regiszterrel 8db ledet lehet vezérelni, mivel nekem 48db ledem van (16rgb), ezért 6db kell belőle. Ennyi lednek a meghajtására már nem lenne elég az a táp, amit a Nucleo tud biztosítani, ezért egy 5V-os külső tápot használok. A táppal sorosan van kötve egy polyfuse, a föld és a táp közé pedig egy 6V-os zener dióda van bekötve fordított polaritással szembeni védelemre. A vezérlők és a gyosrulásérzékelő a Nucleo 3.3V-járól vannak táplálva, különben

szintillesztési problémák lépnének fel. A Nucleo is természetesen az 5V-os tápról van ellátva. Mindegyik led-vezérlő tápja hidegítő kondenzátoron keresztül földelve van.

Firmware felépítése:



Az adatok beolvasása UART-on keresztül történik interruptosan, a kliens alkalmazásban (még részletezem) amikor adatot küldök, akkor 64 word-öt küldök, byteokra osztva, tehát 128 byte-ot. Fogadásnál, egyszerre egy byte-ot fogadok, 2byteonként konkatenálom egy változóba (így fog megfelelni egy oszlopnyi megjelenítendő adatnak , mivel 16 ledem van), és ezt a változót helyezem bele egy 16x64-es tömbbe, amit végül a megjelenítő függvénynek fogok átadni (pontosabban egy rá mutató pointert, ez azért szükséges, mivel 2 bufferem van, az egyikben lévő adatot jelenítem meg, a másikba pedig fogadom az új adatot, ha megtelt, akkor megcserélem a szerepüket). Egy változót használva számlálóként figyelem, hogy történt-e 128 adat fogadás, ha igen, akkor megcserélem a buffereket.

A feldolgozást és megjelenítést egy 1ms-onként bekövetkező timer interrupttal üzemezem, ezek kevesebb mint 0.5ms alatt lefutnak. Ha több időre lenne szükségük, mint amilyen gyakran jön az interrupt, nem működne az alkalmazás.

A gyorsulás adatok begyűjtésére az LSM6DSL gyári függvényét használom, a gyorsulásérzékelőt úgy konfiguráltam, hogy 1660Hz-en működjön.

Kettő darab legfontosabb függvény az update_motion és a Display, ezekhez sok segédfüggvény is tartozik.

update_motion:

A gyorsulás adatokat centereli, majd sebességet állít elő belőle integrálással. Ezt a sebességet majd egy képaraméteres(gyorsabb alfa, lassú béta) mozgóablak átlagolással feldolgozza, hogy sima legyen a jel, és még fontosabban 0 legyen a középpontja. Ez azért nagyon fontos, mert a periódusokat a nullátmenetek alapján figyelem meg, valamint a megjelenítésnél, ha pozitív a sebesség, akkor a megjelenítendő adatokat a tömbből 0->63 kell kiolvasni, ha negatív, akkor 63->0, különben fordítva jelenítenénk meg az ábránkat a periódus második felében.
Miután a sebesség megfelelően elő lett készítve, újra integrálom, hogy elmozdulást állítsak elő belőle. Az elmozdulást minden periódus végén nullázom, mivel csak az adott periódus alatt megtett távolságra vagyok kíváncsi.
Amikor megtörtént 2 nullátmenet (1periódus) akkor azon felül, hogy nullázom a current_displacement-et elmentem az értékét a max_displacement változóba, ez majd a megjelenítésnél lesz hasznos.

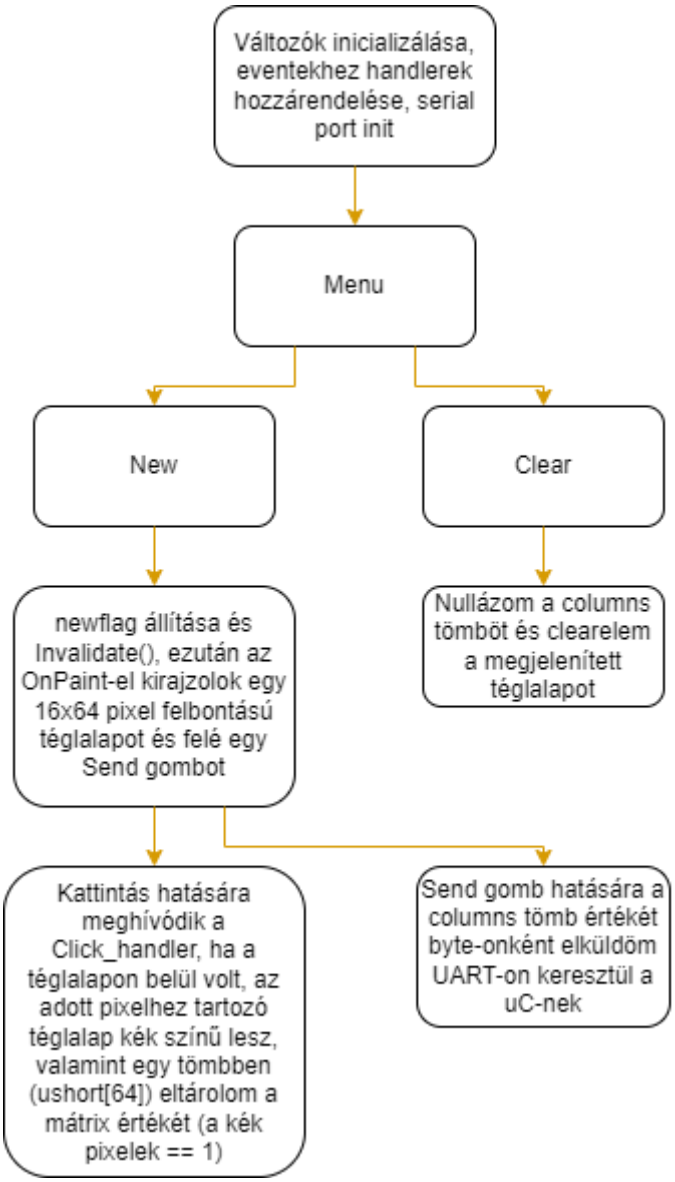
Display:

A Display függvény az update_motion-ban kiszámított adatokat, valamint a megjelenítendő tömböt kapja meg. Először is megfigyeli a sebesség előjelét, majd a current_displacement, max_displacement, start_point és last_direction(sebesség előjele) adatokat átadva a calculate index függvénynek kiszámítja, hogy a mozgás mely fázisában, melyik oszlopot kell majd kijeleznie.
calculateDisplayIndex: a megjelenítés mindig egy periódussal le kell hogy maradjon az adatgyűjtéstől. Ez azért van, mert az előző periódust mindig kalibrációra használom. Megnézem, hogy mekkora volt az előző periódusban a maximális elmozdulás, majd ezt az értéket felosztom 128 szekcióra (64 db értékem van a megjelenítendő tömbben, és oda-vissza meg kell jeleníteni). A current_displacementet ezután kvantálom 128 részre, (elosztom a max_displacementtel), és az adott sávhoz tartozó oszlopot megjelenítem.

Megjelenítés:

A megjelenítő függvényem megkap egy 16bites számot, ebből maszkolással megfigyelem, hogy melyik bitek 1-esek, majd egy 6 elemű tömbbe (6db shift-regiszter van) beletöltöm(konkatenálom), hogy mely biteket kell megjeleníteni (egy adott regiszternél az a kimenet lesz aktív, ahányadik bit 1-es). Majd a SendLEDData függvényben SPI-on keresztül elküldöm a shift-regisztereknek az adatokat, úgy, hogy hátulról olvasom ki a tömböt, mivel a daisy-chain összekötés miatt „végig fogja tolni” a regisztereken az adatokat, így ha azt akarom, hogy az utolsó regiszteren az jelenjen meg amit a tömb végére írtam, akkor azzal kell kezdenem. A HAL_SPI_Transmit előtt lehúzom a latch-et, utána engedélyezem. Az outputokat a program elején clear-elem, majd végig engedélyezem.
Mivel RGB ledeim vannak, ezért segédváltozókba definiáltam, hogy az egyes kék színű ledek aktiválásához milyen bitkombinációkat kell rátölteni a shift-regiszterekre.

PC-s alkalmazás felépítése:



A menu->new hatására kirajzolódik egy téglalap, amiben, amit lerajzolok, a send gomb megnyomása után kijelzésre kerül a ledeken.
A columns tömbben tartom számon, hogy melyik oszlopban melyik bitek aktívak, majd ha megnyomom a send gombot, akkor byte-onként kiírom a serial portra az adatokat.

A kirajzolt téglalap mérete dinamikusan változik az ablak méretétől függően, ahogyan a pixelSize is. A pixelSize és a téglalap paramétere segítségével konvertálom át a kattintás koordinátáit megjelenítendő téglalapokká, és a columns változóba elhelyezendő bitekké.

Click_handleren belül kattintásnál is ezek alapján számolom ki a megjelenítendő kis téglalap területét, majd csak azt a területet rajzolom újra, hogy ne villogjon a képernyő

Mivel a kliens programban és a uC-s megjelenítő függvényben pont fordítva kell az lsb-nek és msb-nek lennie, ezért kijelzés előtt a ReverseBits függvénnyel megfordítom a sorrendet a tömbben. Küldés előtt az adatokat byte-okká konvertálom.

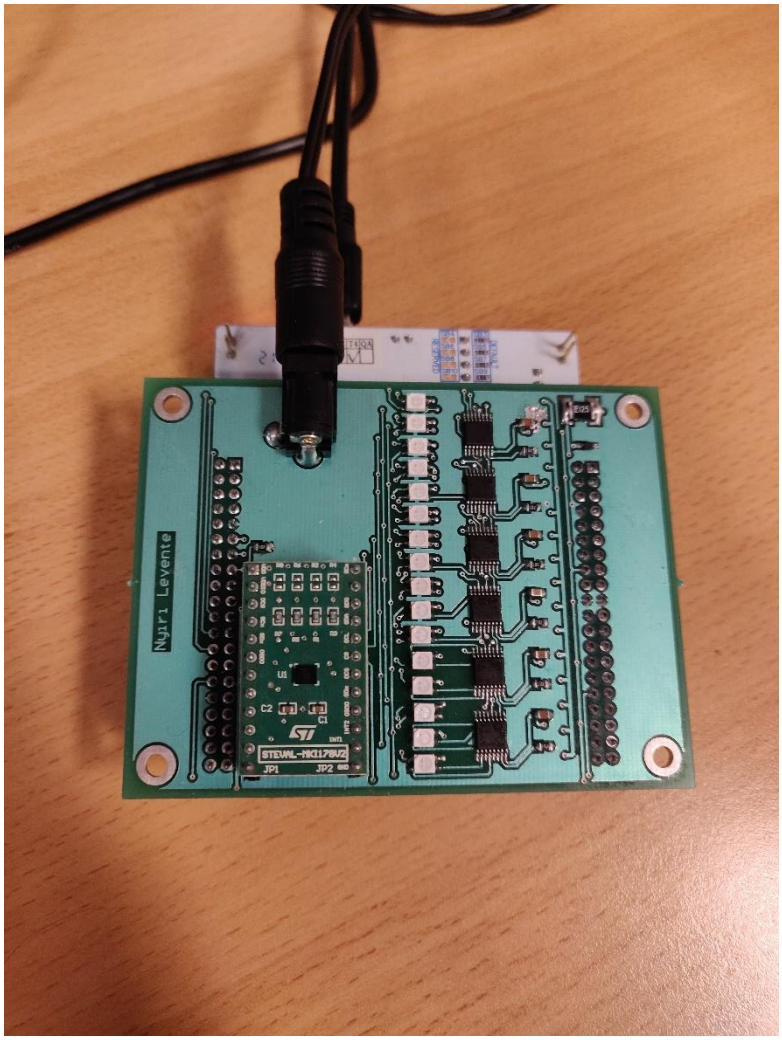
Ha bezárom a formot, lezárom a serial portot is.

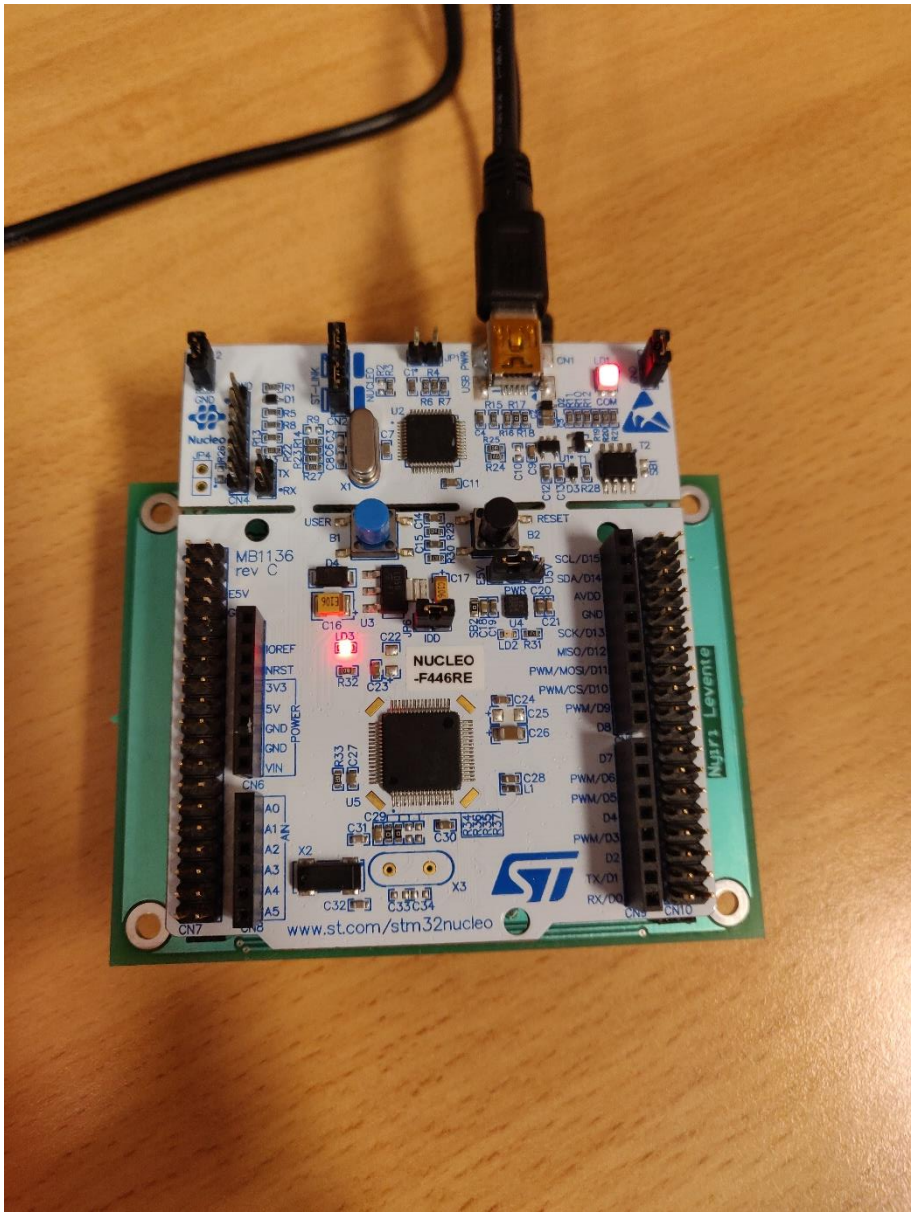
A debuggoláshoz még készítettem egy kliens alkalmazást, amivel rögzítettem egy txt file-ban a gyorsulást, sebességet, elmozdulást és az időt.

Ennek a txt filenak a tartalmának a megjelenítésére írtam egy matlab scriptet, amiben lehetőség van az idő vs gyorsulás/sebesség/elmozdulás egyenkénti, egymás melletti, vagy egymásra szuperponált megjelenítésére. Ezek az ábrák nagyon hasznosak voltak, például amíg nem mozgóablak átlagolással húztam be 0 központúra a sebességet, hanem az átlag kivonásával, akkor az rakott bele egy 90-fokos fázistolást, amit máshogy nem vettem volna észre, csak azt, hogy nem működött a program. Még sok egyéb hiba megtalálásában segített.

Értékelés:

Nagyjából ugyanígy állnék hozzá, ha most kéne elkezdenem viszont még sok funkcióval szívesen bővíteném, ha lenne rá időm. Jelenleg az RGB ledeket nem használom ki, mivel csak egy színt használok. Meg kéne fordítanom a kijelzést, hogy a kábelek alul legyenek miközben rángatom, így ergonomikusabb lenne, és a kliens alkalmazást is bővíteném (képek konvertálása megjeleníthető ábrákká, elmentett rajzok betöltése, színválasztás). Sokkal jelentősebb volt a jelfeldolgozás a feladatban, mint amire számítottam, aminek örülök, mert sokat tanultam belőle, jól jött hozzá a BAMBI-s tudás is.




MB1136
rev C
NUCLEO
-F446RE
www.st.com/stm32nucleo
Nyári Levente