



E-SCHAAKBORD

Professionele Bachelor in de Elektronica-ICT

Verslag in het kader van Practice Enterprise 2

Door: Matthias Hendrickx

Academiejaar 2020-2021
Campus De Nayer, Jan De Nayerlaan 5, BE-2860 Sint-Katelijne-Waver

INHOUDSTAFEL

INHOUDSTAFEL	1
VOORWOORD	2
INLEIDING	3
1 HARDWARE	4
1.1 Blokschema.....	4
1.2 Schema's.....	5
1.3 PCB's.....	7
1.4 Beschrijving	9
1.4.1 Schakelende voeding	9
1.4.2 LDO Voltage Regulator.....	11
1.4.3 µC.....	11
1.4.4 RGB LED's.....	12
1.4.5 Voltage Level Translator	13
1.4.6 Rotary Encoder	14
1.4.7 EEPROM.....	15
1.4.8 Timer Display	15
1.4.9 7-segment Driver.....	15
1.4.10 Magneet Sensor.....	16
1.4.11 Shift Register PISO.....	18
2 TESTEN	19
2.1 Voltage Translator	19
2.2 RGB LED's	19
2.3 Klok	22
2.4 Positie uitlezing	25
3 BESLUIT	28

VOORWOORD

Mijn project voor Practice Enterprise 2 is tot stand gekomen met de hulp van een aantal mensen tot wie ik persoonlijk een dankwoord wil richten.

Eerst en vooral wil ik mijn ouders en zus bedanken om mij op zo veel mogelijk vlakken te helpen. Ze hebben mij het volledige proces gesteund en gemotiveerd.

Verder wil ik enkele leerkrachten bedanken namens Mr.Dams, Mr.Steen. Zij hebben mij altijd bijgestaan met raad en daad en hebben zich mede ingezet om ideeën omtrent het project te kunnen realiseren. Mr.Dams wist ook altijd in moeilijkere tijden mij te motiveren en terug op het juiste spoor te leiden.

Ik wil ook Mr.De Weerdt bedanken voor het helpen met het solderen van enkele moeilijkere SMD-componenten.

Daarnaast dank ik ook mijn klasgenoten die mij goede raad konden geven en helpen bij de kleinste zaken en meer.

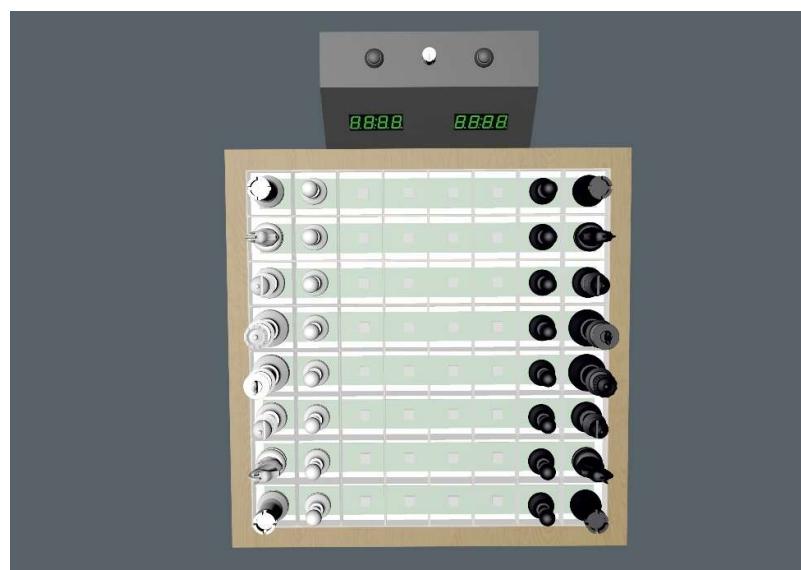
En uiteraard kon ik ook altijd op de steun van mijn vrienden rekenen, waar ik altijd aan iets kon vragen of om mijn ideeën te kunnen realiseren.

INLEIDING

Deze bundel gaat over het project dat ik gemaakt heb voor het vak Practice Enterprise 2, hierbij waren de minimale vereisten dat ik een PCB ontwerp, Software schrijf (liefst in C op een µController) en ten slotte het project een nuttige toepassing heeft. Het is ook belangrijk dat er een veilig project gekozen werd en de kostprijs niet te hoog lag.

Na enkele projectjes dat ik heb voorgesteld koos ik om een elektronisch educatief schaakbord te maken. Hiermee zouw u dan de mogelijkheid krijgen om te leren hoe men iedere pion zet en de werking van een schaakklok. Daarnaast bevat het schaakbord over een puntensysteem dat bijgehouden wordt.

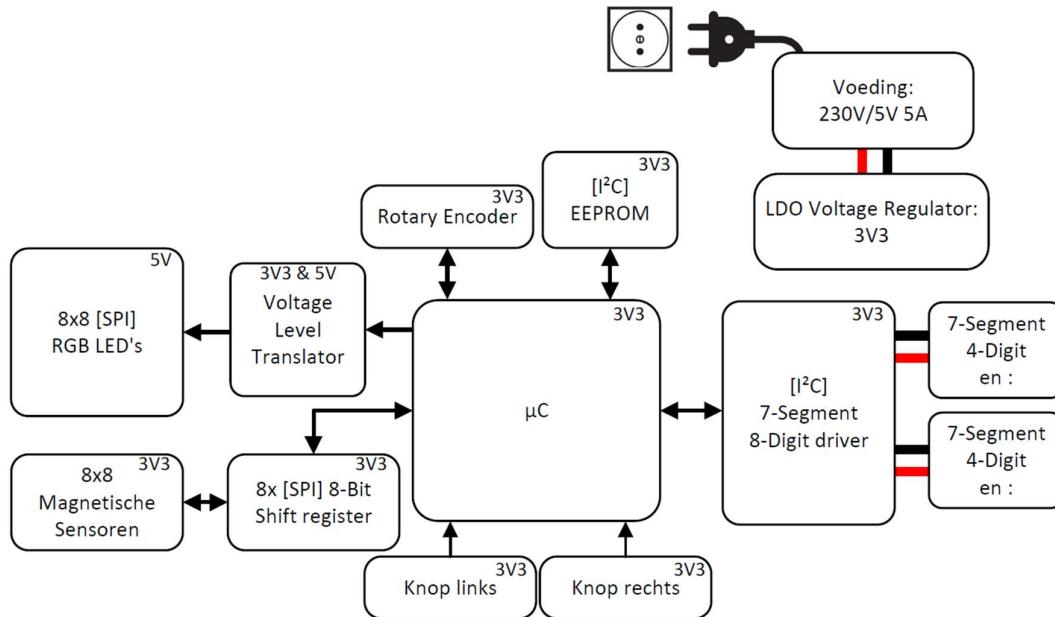
Er is ook een behuizing die zeer nauwkeurig moet zijn wegens de standaard van het schaken, hier onder is een 3D-schets van hoe het ganse project er zal uit zien:



1 HARDWARE

1.1 Blokschema

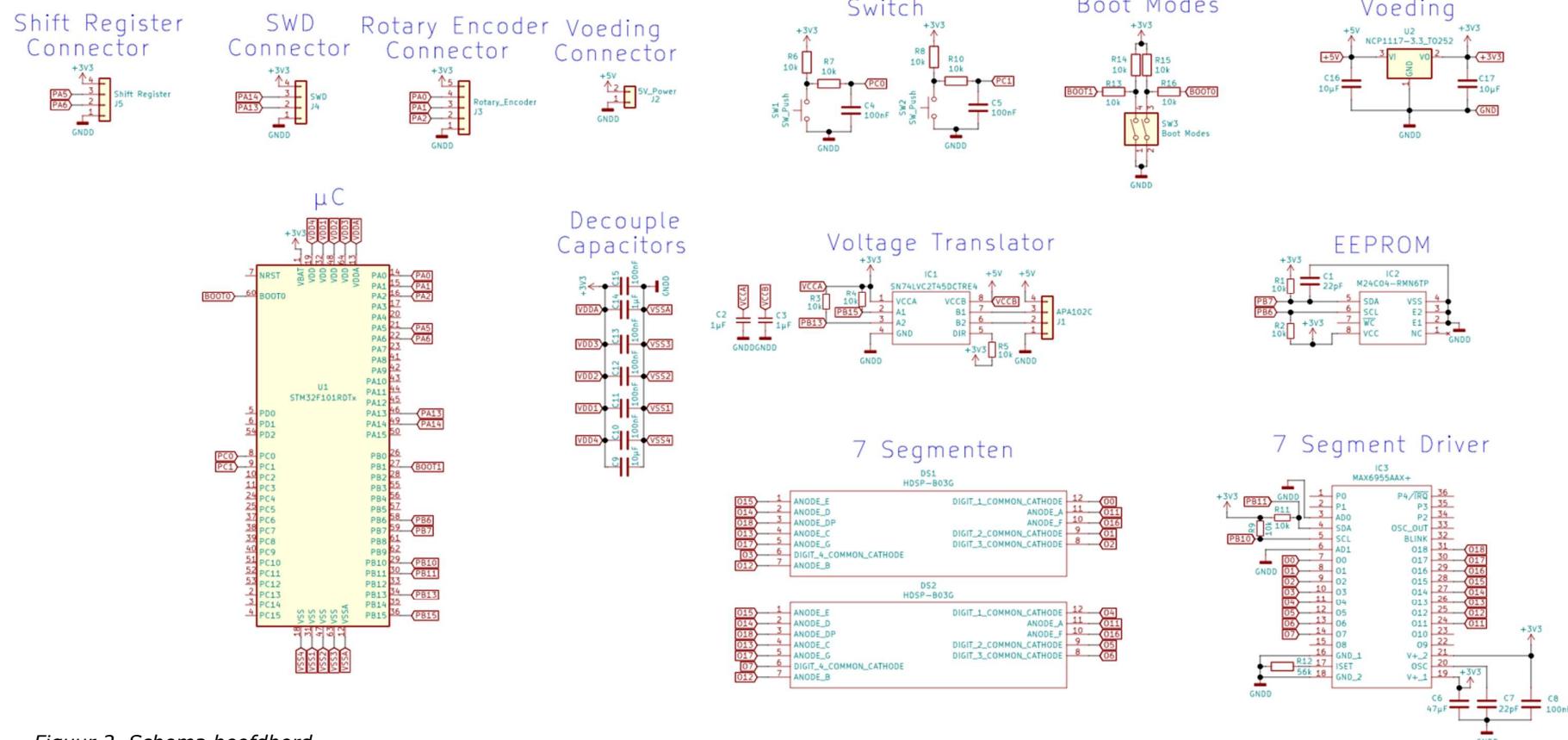
In het blokschema zien we hoe de verschillende blokken verbonden zijn met elkaar maar dit in een vereenvoudigde weergave. Hierbij kan je zien hoe alles logisch met elkaar verbonden is d.m.v. voedingsaansluitingen en communicatielijnen.



Figuur 1, Blokschema

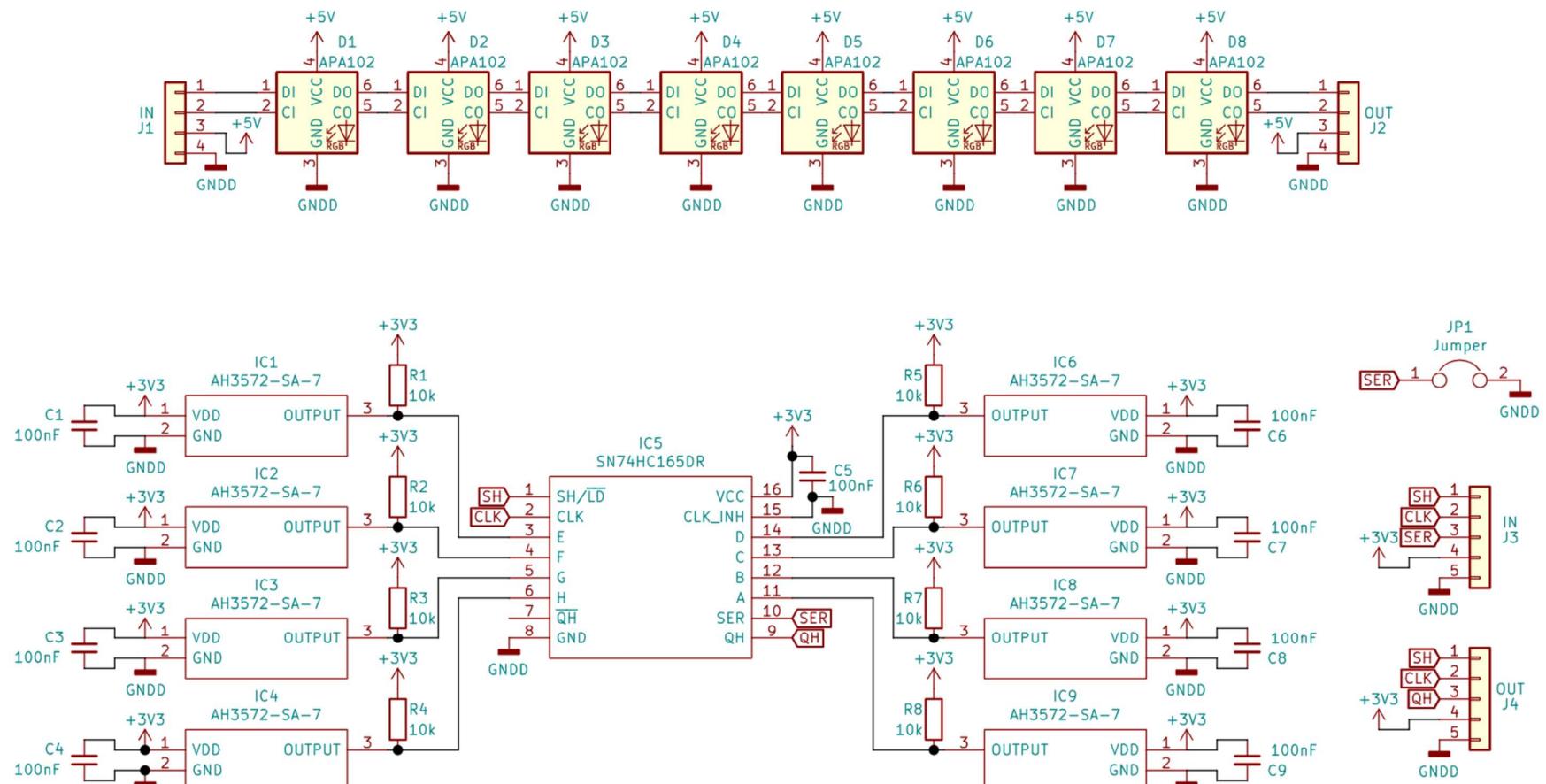
1.2 Schema's

Dit is het aansluiting schema van het hoofdbord, hierin zit: de µC, 7 Segmenten, Rotary Encoder ... Later wordt ieder deel nog besproken.



Figuur 2, Schema hoofdbord

Dit is het aansluiting schema van de borden die zich onderaan de pionnen liggen, Later wordt ieder deel besproken.

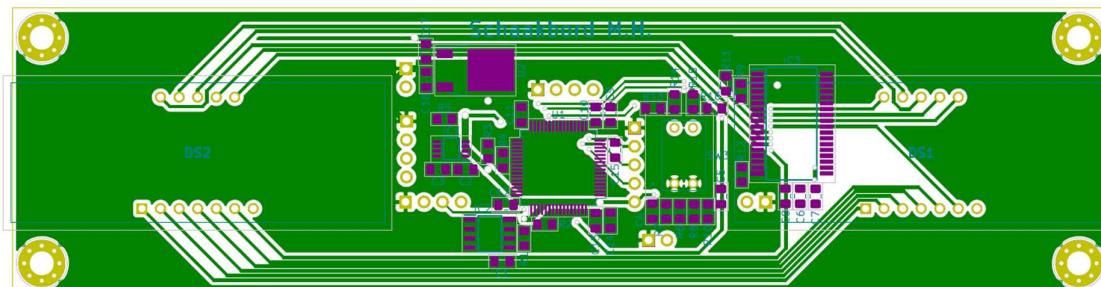


Figuur 3, Schema pionnen

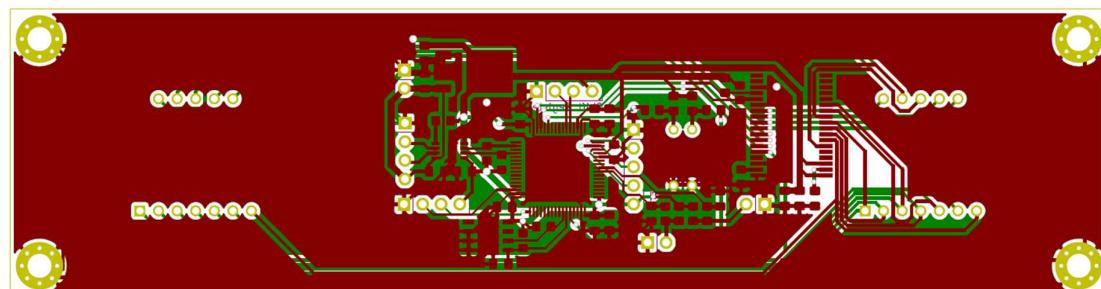
1.3 PCB's

De PCB's zijn helemaal zelf ontworpen en de footprint die ik gebruik stammen af van de website Mouser waar de componenten besteld zijn en de paden zijn dubbel checkt.

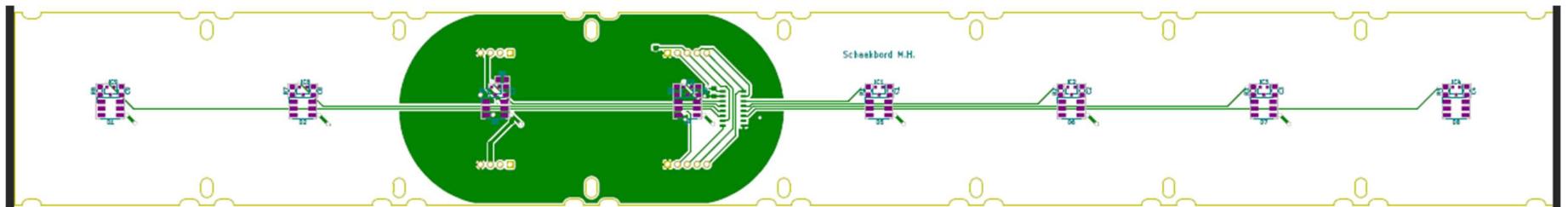
Bovenkant Hoofdbord:



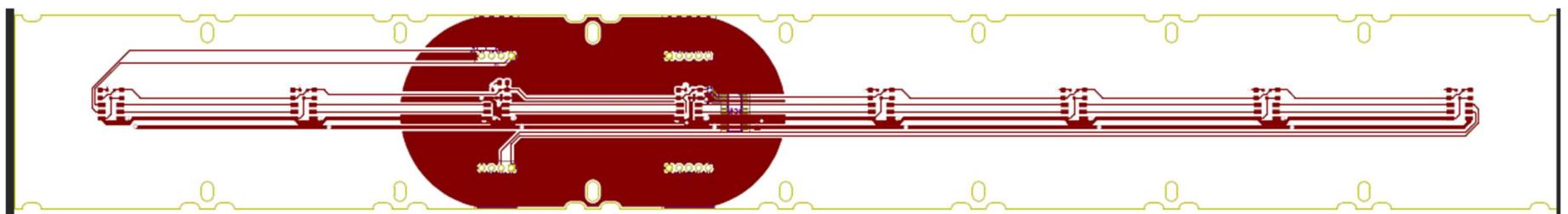
Onderkant Hoofdbord:



Bovenkant Hoofdbord:



Onderkant Hoofdbord:



1.4 Beschrijving

Hierin ga ik ieder deel bespreken en uitleggen hoe ik eventueel iets berekend heb.

1.4.1 Schakelende voeding

Het E-Schaakbord wordt gevoed door de 230V AC netspanning die wordt omgezet naar 5V DC d.m.v. een schakelende voeding. De voeding die ik gekozen/berekend heb kan maximaal 5A leveren of in andere woorden $5V * 5A = 25W$.

Dit heb ik berekend a.d.h.v. het maximaal vermogen dat gebruikt kan worden op te tellen van ieder component.
Daarmee heb ik gebruik gemaakt van de maximale spanning en stromen die uit de datasheet komen van ieder component.



Figuur 4, Voorbeeld voeding

Vermogen berekening*						
MAX6955 (7 Segment Driver)						
Uin(V)	Uled(V)	Duty Cycle	Iled(mA)	Ndigit	Itot(mA)	Ptot(mW)
3,3	2	0,9375	25	8	73,899	243,866
APA102C (RGB Addressable LED's)						
Uin(V)	Iin(mA)	Aantal	Itot(A)	Ptot(W)		
5	73,5	64	4,704	23,52		
STM32F101RDTx (μC)						
Uin(V)	Iin(mA)		Itot(mA)	Ptot(mW)		
3,3	20		20	66		
AH3572 (Magnetic Sensor)						
Uin(V)	Iin(mA)	Aantal	Itot(mA)	Ptot(mW)		
3,3	3	64	192	633,6		
NCP1117 (LDO)						
Uin(V)	Uuit(V)	Iq(mA)	Itot(mA)	Ptot(mW)		
5,2	3,2	10	20	40		
(*) De totale stromen onder 1mA wordt niet meegerekend. (EEPROM, Shift Register en Voltage Translator)						
5V voor Schakelende voeding						
I _{max} (mA)		P _{max} (W)				
5,010		24,503				
3V3 voor LDO						
I _{max} (mA)		P _{max} (mW)				
305,899		983,466				

Tabel 1, Vermogen berekening

Zoals u merkt heb is mijn voeding's keuze redelijk krap genomen, dit is doordat ik het maximale berekend heb en daarmee juist gekozen heb voor 5A. Het verbruik wordt beperkt d.m.v. de software die ik er insteek, omdat ik nooit al mijn RGB LED's tegelijk de kleur wit ga maken zal de stroom dus beperkt blijven.

APA102C (RGB Addressable LED's): kleur Wit

$$Itot = Uin * Iin * Aantal = 5V * 24,5mA * 3 * 64 = 4,704A$$

$$Ptot = Uin * Itot = 5V * 4,704A = 23,52W$$

APA102C (RGB Addressable LED's): kleur Blauw

$$Itot = Uin * Iin * Aantal = 5V * 24,5mA * 64 = 1,568A$$

$$Ptot = Uin * Itot = 5V * 1,568A = 7,84W$$

MAX6955 (7 Segment Driver):

$$P_{tot} = (U_{in} * 35mA) + (U_{in} - U_{led})(DUTY * I_{seg} * N_{digit})$$

$$P_{tot} = (3,3V * 35mA) + (3,3V - 2V)(15/16 * 25mA * 8) = 243,866mW$$

$$I_{tot} = \frac{P_{tot}}{U_{in}} = \frac{243,866mW}{3,3V} = 73,899mA$$

STM32F101RDTx (μ C):

The screenshot shows the CubeMX PCC Tool interface for power consumption simulation. It includes the following sections:

- Power/Memory:** Power Mode (RUN), Power Scale (No Scale), Memory Fetch Type (FLASH), V_{DD} (3.3), Voltage Source (Vbus).
- Clocks:** CPU Frequency (36 MHz), Interpolation Ranges, User Choice (Hz), Clock Configuration (HSI PLL), Clock Source Frequency (8 MHz).
- Optional Settings:** Step Duration (1 s), Additional Consumption (0 mA).
- Results:** Step Consumption (18.83 mA), Without Peripherals (15.6 mA), Peripherals Part (3.23 mA (A: 0 nA - D: 3.23 mA)), Ta Max (°C) (102.2).

I_{tot} is gesimuleerd a.d.h.v. CubeMX zijn PCC Tool.

Hierbij heb ik alles ingesteld op het maximum dat ik kan gebruiken i.v.m. het schaakbord: HSI PLL, de CPU Frequency, I/O en meer.

$$I_{tot} \approx 20mA$$

$$P_{tot} = U_{in} * I_{tot} = 3,3V * 20mA = 66mW$$

Figuur 5, Simulatie stroomverbruik

AH3572 (Magnetic Sensor):

$$I_{tot} = U_{in} * I_{in} * Aantal = 3,3V * 3mA * 64 = 192mA$$

$$P_{tot} = U_{in} * I_{tot} = 3,3V * 192mA = 633,6mW$$

NCP1117 (LDO Voltage Regulator):

$$I_{tot} = (U_{in} - U_{uit}) * I_q = (5,2V - 3,2V) * 10mA = 20mA$$

$$P_{tot} = (U_{in} - U_{uit}) * I_{tot} = (5,2V - 3,2V) * 20mA = 40mW$$

1.4.2 LDO Voltage Regulator

Nadat de 230V AC omgezet is naar 5V DC zal ik deze 5V omzetten naar 3V3, hiermee zal ik de andere componenten voeden buiten de RGB LED's.

Ik heb gekozen voor een LDO omdat:

- Het maximaal vermogen onder de 1W was en dus dit vermogen nog gedissipeerd kan worden d.m.v. een LDO.
- Het voordeel is ook dat LDO's qua structuur simplistisch zijn en dus onafhankelijk zijn van een externe spoel i.t.t. een Buck-Converter.

A.d.h.v. de vorige berekeningen (Tabel 1, Vermogen berekening) kwam ik uit dat de LDO een minimaal vermogen van $\pm 983\text{mW}$ gedissipeerd wordt.

Hiermee ben ik dan aan de slag gegaan en opzoek gegaan naar een LDO en kwam ik uit op de NCP1117.

Deze heb ik gekozen door de volgende berekeningen te maken:

$$P_{DMAX} = \frac{T_J - T_A}{\theta_{JA}} = \frac{150^\circ C - 40^\circ C}{67^\circ C/W} = 1642\text{mW}$$

$$P_{MAX} < P_{DMAX} \rightarrow 983\text{mW} < 1642\text{mW}$$

Deze berekeningen heb ik geraadpleegd via een handig document van Microchip:

<http://ww1.microchip.com/downloads/en/appnotes/00761b.pdf>

1.4.3 μC

Dit is een zeer belangrijk keuze die gemaakt moet worden, daarmee heb ik genoeg research gedaan op het internet d.m.v. online guides op te zoeken en meer.

De eerste vraag die ik mezelf stelde is of ik een 8-bit of 32-bit microcontroller kies. Ik koos voor een 32-bit microcontroller wegens volgende redenen:

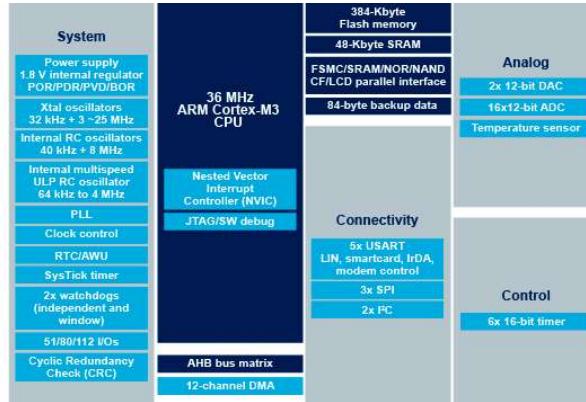
- Communicatieprotocollen zoals SPI & I²C zijn eenvoudiger te implementeren.
- Om lengte/grootte van software te beperken.
- Eenvoudiger te debuggen.

Nu was de keuze voor welke fabrikant ik zal gaan, hier waren vooral 2 uitblinkers bij de STM32 of PIC32. Ik koos voor de STM32 wegens volgende redenen:

- Meer bekend in hoe deze ge programmeert wordt en debugt.
- Zeer stabiel en handige Toolchain/IDE ondersteuning.

Ten slotte ben ik opzoek gegaan naar een STM32 die een genoeg aantal I²C en SPI-protocollen had en die genoeg geheugen heeft.

Toen kwam ik uit op de STM32F101RDT6, deze bevindt zich in een LQFP-64 package dat nog handsoldeerbaar is en ook de mogelijkheid heeft in dezelfde package een groter geheugen te kiezen of een andere reeks.



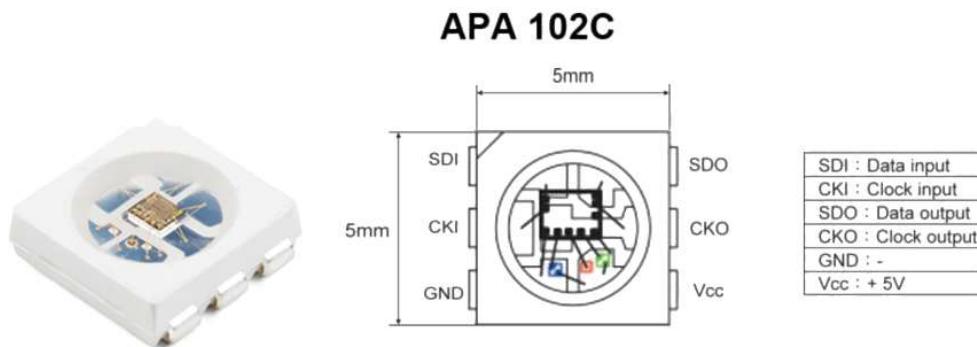
Figuur 6, Circuit Diagram

1.4.4 RGB LED's

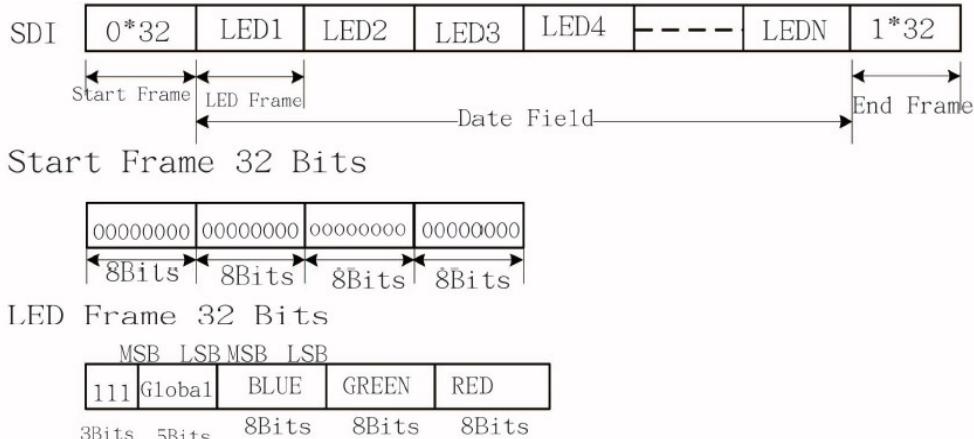
Om de schaakvakken een kleur te geven maak ik gebruik van adresseerbare RGB LED's, hierdoor kan ik elk vak individueel een andere kleur geven wat dat belangrijk is voor aan te duiden wat de pion hun mogelijke zetten zijn en meer.

Ik heb gekozen om gebruik te maken van de APA102C LED's omdat:

- Worden aangestuurd via het standaard SPI-Protocol.
- Heeft een snellere CLK-frequentie maximaal 1,2MHz.
- Beschikt over een aparte PWM-instelling om de lichtsterkte te regelen.



Figuur 7, Illustratie APA102C



Figuur 8, APA102C SPI-Protocol

Zoals u hierboven ziet wordt de APA102C's aangestuurd:

Eerst wordt er een Start Frame gestuurd met 4 bytes die '0' zijn.

Daarna wordt er de data van de eerste LED gestuurd met als inhoud:

1. "111" om te beginnen, 3 bits.
2. Daarna de data wat de licht intensiteit is van al de LED's, 5bits.
3. De lichtintensiteit van de kleur blauw, 1 byte.
4. De lichtintensiteit van de kleur groen, 1 byte.
5. De lichtintensiteit van de kleur rood, 1 byte.

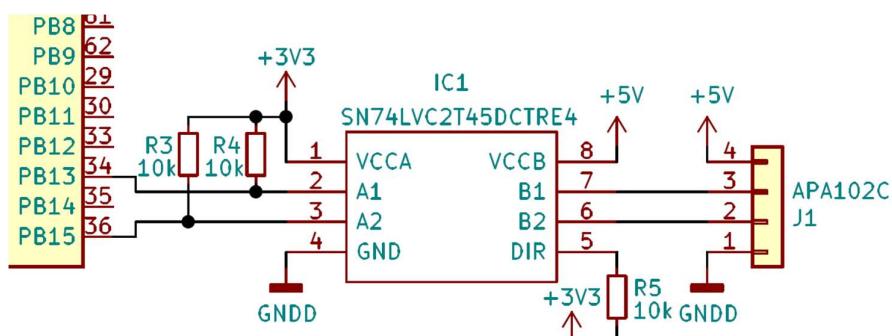
Vervolgens stuurt men de data door van de tweede LED.

Ten slotte kan men een eind frame sturen die bestaat uit allemaal enen, de grote van deze frame is gelijk aan het totaal aantal LED's delen door 2.

Dus het schaakbord bestaat uit 64 RGB LED's, dat wil zeggen dat er 32 keer 4 bytes die '1' zijn (eind frame) verstuurt moet worden. Het is gemakkelijker om op het einde een Start frame sturen om terug opnieuw te beginnen waardoor ik zo min mogelijk tijdverlies.

1.4.5 Voltage Level Translator

Om de APA102C's te laten communiceren met de µC zal ik de 3V3 logica moeten omzetten naar 5V logica. Om deze om te zetten maak ik gebruik van een Voltage Level Translator. Ik heb gekozen voor de SN74LVC2T45, hiermee zal ik de SCK (seriële klok) en de SDI (seriële data in) omzetten naar 5V logica.

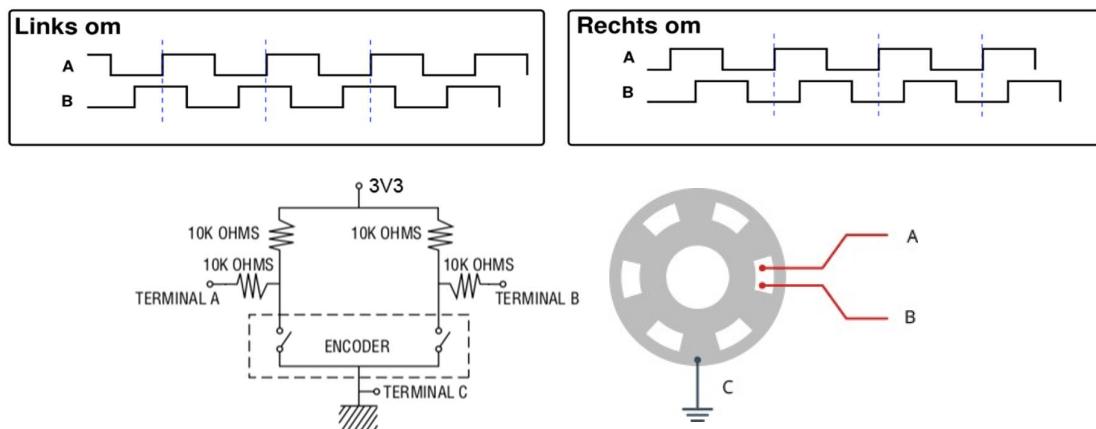


Figuur 9, Schema Voltage Translator

1.4.6 Rotary Encoder

Voor het schaakbord gebruiksvriendelijker te maken bevindt er zich een rotary encoder tussen de 2 drukknoppen op de schaakklok. Hiermee kan voor het spel begint de tijden ingesteld worden en de kleur van de schaakvakken.

Een rotary encoder werkt als volgend:

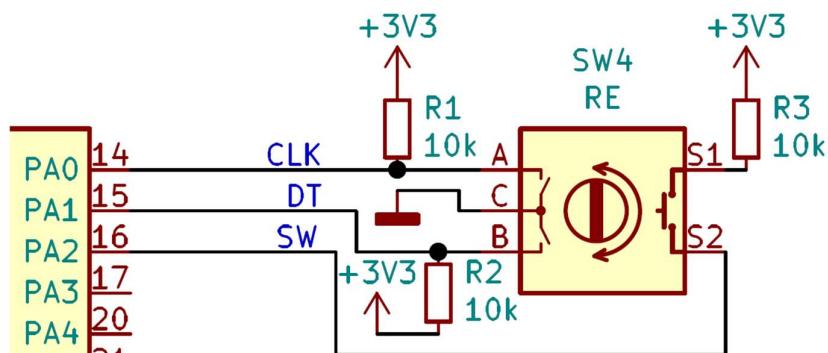


Figuur 10, Werking Rotary Encoder

Wanneer er één stand in wijzerzin (\circlearrowleft) gedraaid wordt zal Terminal A 90° voor-ijlen omdat deze voor Terminal B contact zal maken met de GND of de 3V3. Als daar na gelijk er één stand in wijzerzin (\circlearrowleft) opnieuw gedraaid wordt zal nu Terminal B op hetzelfde contact van Terminal A staan.

Dit is omgekeerd als men één stand in tegenwijzerzin (\circlearrowright) draait zal Terminal B 90° voor-ijlen omdat deze voor Terminal A contact zal maken met de GND of de 3V3. Hetzelfde als er daar na gelijk één stand in tegenwijzerzin (\circlearrowright) opnieuw gedraaid wordt zal nu Terminal A op hetzelfde contact staan als dat van Terminal B.

Deze terminals worden vervolgens zo aangesloten op de STM32:



Figuur 11, Schema Rotary Encoder

In de software zal nu gewoon iedere pin continu uitgelezen worden en als er een verandering plaats vindt zal men te weten komen welke richting er gedraaid werd.

De rotary encoder bevat ook een schakelaar die schakelt wanneer op de knop gedrukt wordt, hiermee is alles wat meer gebruiksvriendelijk zoals om een actie te voltooien.

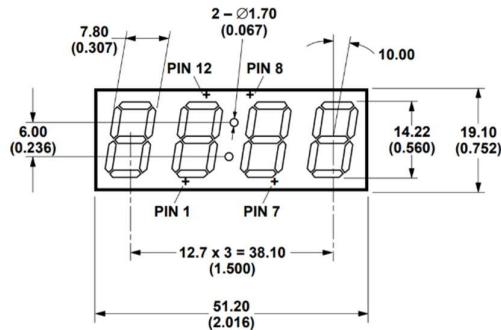
1.4.7 EEPROM

EEPROM of Electrically Erasable Programmable Read-Only Memory is een geheugen IC die data opgeslagen houdt terwijl de IC niet gevoed wordt, dit heet men ook permanent geheugen. EEPROM's kunnen ook hun blokken data (1 byte) individueel her programmeren waardoor er niet alles verwijderd moet worden. Het nadeel is wel dat deze niet oneindig keer herschreven kan worden, maar tegenwoordig kunnen de meeste meer dan 1 miljoen keer herschreven worden.

Ik heb gekozen voor de M24C04 omdat deze via I²C te werk gaat en een geheugen heeft van 4kbits of 512bytes. Hierin zal ik de score zetten van het vorig spel.

1.4.8 Timer Display

Het educatief schaakbord maakt ook gebruik van een timer die zich aan de zijkant bevindt. Om deze weer te geven maak ik gebruik van een 7-segmenten 4-digit display die in het midden een dubbelpunt heeft. Hiervan gebruik ik er 2 omdat bij schaken iedere partij 1 timer heeft. Ik heb gekozen voor een scherm dat groot genoeg en daarmee gemakkelijk af te lezen is, daarnaast besloot ik voor de kleur groen te gaan omdat deze kleur visueel mooier is.



Figuur 12, 7-Segment 4-Digit

Ondanks de grote van de display verbruikt ieder segment maar 20mA en geven ze een minimale lichtsterkte van 3200µcd.

1.4.9 7-segment Driver

Om de timers eenvoudiger aan te sturen besloot ik gebruik te maken van een 7-segment driver. Hierbij moest ik uitzien naar verschillende factoren welk het beste is zoals:

- Communicatieprotocol: Hiervoor koos ik I²C omdat de andere componenten al met SPI werkten.
- Een font map voor 7-segment digits.
- Het aantal segmenten kan aansturen: dat wil zeggen een minimum van

$$N_{\text{segmenten}} = 7 \text{ segmenten} * 8 \text{ digits} + 4 \text{ DP} = 60 \text{ segmenten}$$

Onder deze voorwaarden koos ik voor de MAX6955, deze bevat een font map voor de 7-segmenten eenvoudiger aan te sturen en kan maximaal 16-digits aansturen.

1.4.10 Magneet Sensor

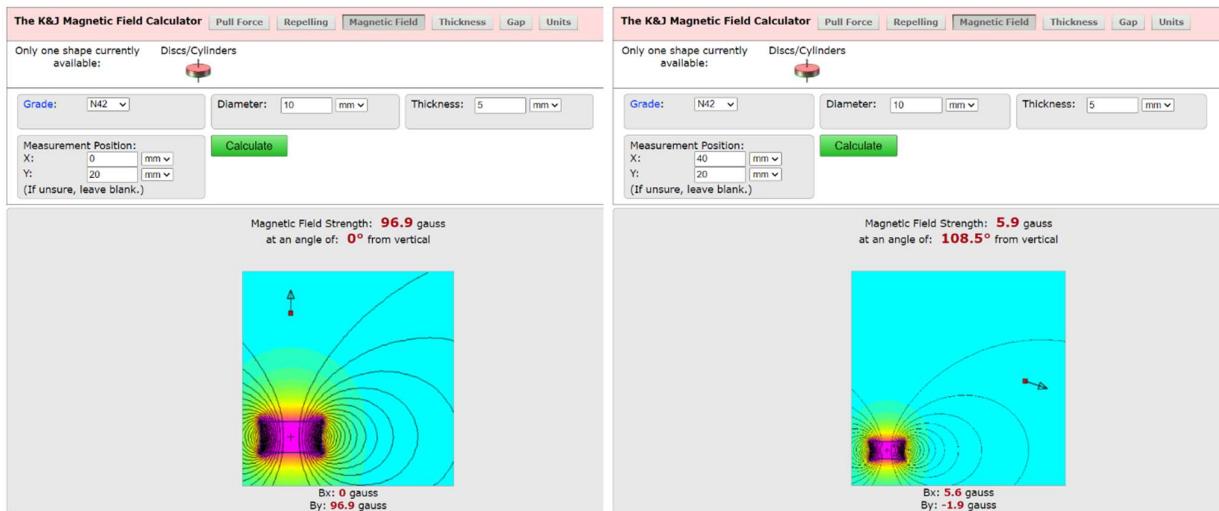
Om de pionnen te detecteren maak ik gebruik van magneet sensoren. Hiermee kan ik on opgemerkt de positie van een pion uitlezen. Doordat ik nog nooit in contact ben gekomen van een magneet sensor heb ik eerst research gedaan achter welke types er bestaan zoals: een reed schakelaar, een hal-sensor, a.d.h.v. spoelen. Omdat de prijs van een reed schakelaar aan de hoge kant lag (>1€) per stuk, heb ik gekozen voor een hal-sensor.

Hierin zijn ook verschillende types die bestaan bijvoorbeeld: omnipolair, unipolair, latch, en nog meer. Ik koos voor omnipolair zonder latch omdat deze zullen schakelen wanneer er een noord of zuid magnetisch veld aangelegd wordt. En zonder latch omdat de pionnen continu verwisseld worden van plaats, daarmee dat de vorige positie niet onthouden moet worden.

Het is ook zeer belangrijk om te weten op welk hoeveelheid magnetische fluxdichtheid er geschakeld wordt. Deze waarden heb ik nagerekend a.d.h.v. een website:

<https://www.kjmagnetics.com/calculator.asp?calcType=disc>

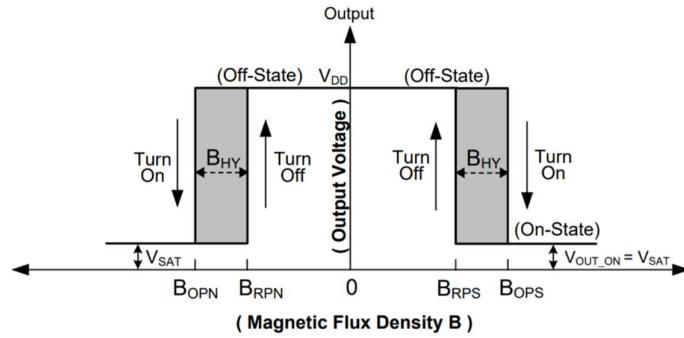
Hiermee gaf ik de volgende waarden ingaf:



Figuur 13, Magnetisch Veld

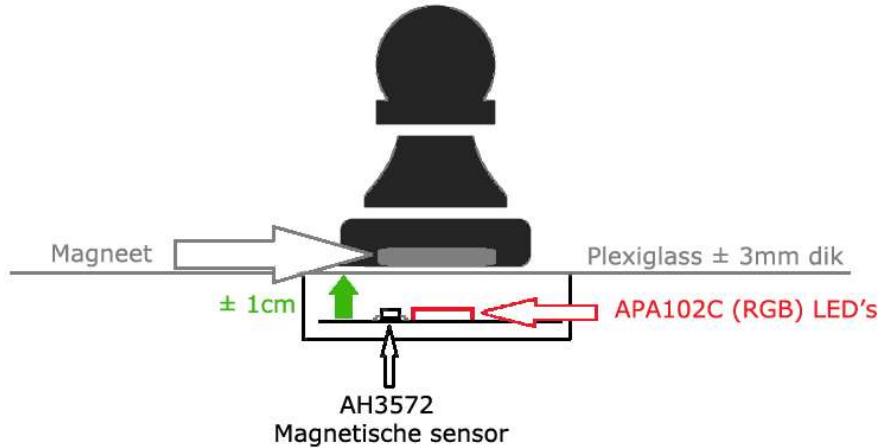
Hierdoor wist ik dus dat een magneet met een diameter van 10mm en dikte van 5mm een magnetische fluxdichtheid van 96,9 gauss op verticale afstand van 2cm. En zoals u ziet aan de rechter berekening zal de magneet geen storing geven aan een ander vak dat minstens 4 cm verwijderd is van de magneet.

Ik koos voor de AH3572 deze aan de goedkopere kant was 0,205€/stuk¹, en schakelde vanaf minimum 20 gauss dat dus geen probleem geeft.



Figuur 14, Schakelpunten

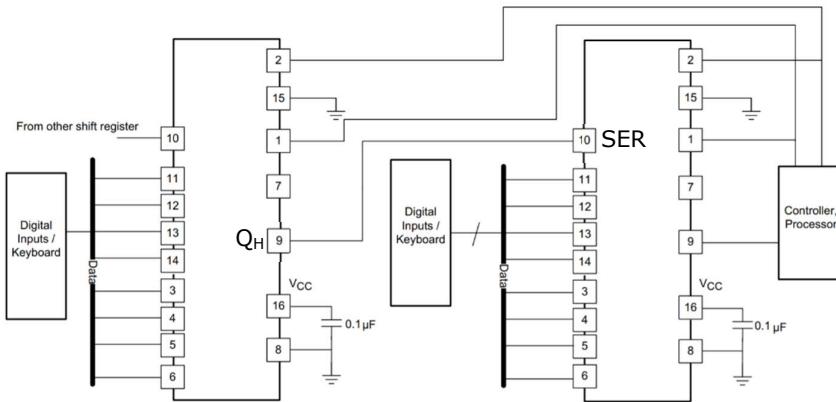
Dit zijn de punten waarop de magneet sensor schakelt. Hier moest ik vooral kijken naar het B_{OPS} punt, deze bedraagt 20 gauss wat dus geen probleem zal geven als de magneet op een afstand van 2cm weg is.



¹ Bij aankoop van 100 stuks.

1.4.11 Shift Register PISO

Voor iedere magneet sensor uit te lezen maak ik gebruik van een shift register omdat de hardware dan meer vereenvoudigd is. Doordat de 64 magneet sensoren (1 magneet sensor per schaak vak) zijn op gedeeld als een matrix koos ik voor een PISO, Parallel In Serieel Out. Ik maak 8 keer gebruik van een shift register om zo iedere rij (8 vakken) uit te lezen, vervolgens zal de output van de shift register aangesloten worden aan de seriële ingang van de volgende rij enzovoort. Ten slotte zal de laatste uitgang aangesloten worden aan de µC.



Figuur 15, Voorbeeld Cascade

Ik heb gekozen voor de SN74HC165 omdat deze zeer courant gebruikt wordt.

2 TESTEN

Om de STM32 te debuggen & programmeren zal ik de boot mode moeten instellen a.d.h.v. de DIP-switchen.

Tabel 2, Boot Modes

Boot mode selection pins		Boot mode	Aliasing
BOOT1	BOOT0		
x	0	Main Flash memory	Main Flash memory is selected as boot space
0	1	System memory	System memory is selected as boot space
1	1	Embedded SRAM	Embedded SRAM is selected as boot space

Ik heb de boot mode ingesteld op systeem geheugen of te wel ROM, hier zal het programma naar geschreven en uitgelezen worden.

2.1 Voltage Translator

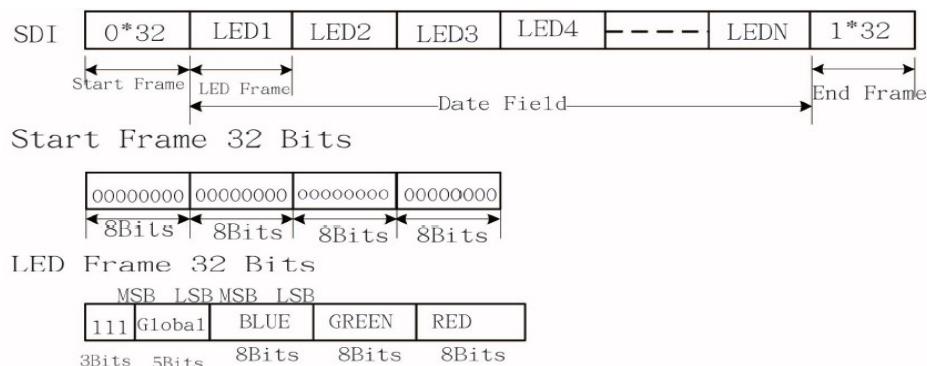
Omdat APA102C's op 5V logica testen ik eerst of de Voltage Translator werkt, deze heb ik getest d.m.v. de pinnen in de software hoog en laag te maken.

Vervolgens heb ik simpelweg de pinnen aan de ingang & de uitgang van de Voltage Translator uit ge meten en kwam ik volgende waarden uit:

	IN	UIT
HOOG	3,293V	4,831V
LAAG	0,009V	0,001V

2.2 RGB LED's

Zoals eerder vermeld werken de APA102C via SPI, dus moet de code er als volgend uitzien:



Figuur 16, APA102C SPI-Protocol

Dit protocol heb ik getest d.m.v. te bitbangen, de test code ziet er als volgend uit:

1. Als eerst een start frame sturen van 32 keer een '0'.

```
void LED_Start(){
    for (i = 0; i < 32; i++) {
        HAL_GPIO_WritePin(SPI_MOSI_GPIO_Port, SPI_MOSI_Pin, GPIO_PIN_RESET);
        SysTickDelayCount2(t);
        HAL_GPIO_WritePin(SPI_SCK_GPIO_Port, SPI_SCK_Pin, GPIO_PIN_SET);
        SysTickDelayCount2(t);
        HAL_GPIO_WritePin(SPI_SCK_GPIO_Port, SPI_SCK_Pin, GPIO_PIN_RESET);
    }
}
```

2. Vervolgens stuur ik de kleuren aan die gewenst zijn. Hiervoor heb ik de voor ingestelde kleuren rood, groen, blauw en RGB die al de kleuren kan combineren. Er wordt ook een kleurintensiteit meegegeven.

```
void LED_Rood(uint8_t brightness){
    for (i = 0; i < 8; i++) {
        HAL_GPIO_WritePin(SPI_MOSI_GPIO_Port, SPI_MOSI_Pin, GPIO_PIN_SET);
        SysTickDelayCount2(t);
        HAL_GPIO_WritePin(SPI_SCK_GPIO_Port, SPI_SCK_Pin, GPIO_PIN_SET);
        SysTickDelayCount2(t);
        HAL_GPIO_WritePin(SPI_SCK_GPIO_Port, SPI_SCK_Pin, GPIO_PIN_RESET);
    }
    for (i = 0; i < 16; i++) {
        HAL_GPIO_WritePin(SPI_MOSI_GPIO_Port, SPI_MOSI_Pin, GPIO_PIN_RESET);
        SysTickDelayCount2(t);
        HAL_GPIO_WritePin(SPI_SCK_GPIO_Port, SPI_SCK_Pin, GPIO_PIN_SET);
        SysTickDelayCount2(t);
        HAL_GPIO_WritePin(SPI_SCK_GPIO_Port, SPI_SCK_Pin, GPIO_PIN_RESET);
    }
    for (i = 7; i >= 0; i--) {
        uint8_t k = brightness >> i;
        if (k & 1){
            HAL_GPIO_WritePin(SPI_MOSI_GPIO_Port, SPI_MOSI_Pin, GPIO_PIN_SET);
        }else{
            HAL_GPIO_WritePin(SPI_MOSI_GPIO_Port, SPI_MOSI_Pin, GPIO_PIN_RESET);
        }
        SysTickDelayCount2(t);
        HAL_GPIO_WritePin(SPI_SCK_GPIO_Port, SPI_SCK_Pin, GPIO_PIN_SET);
        SysTickDelayCount2(t);
        HAL_GPIO_WritePin(SPI_SCK_GPIO_Port, SPI_SCK_Pin, GPIO_PIN_RESET);
    }
}
```

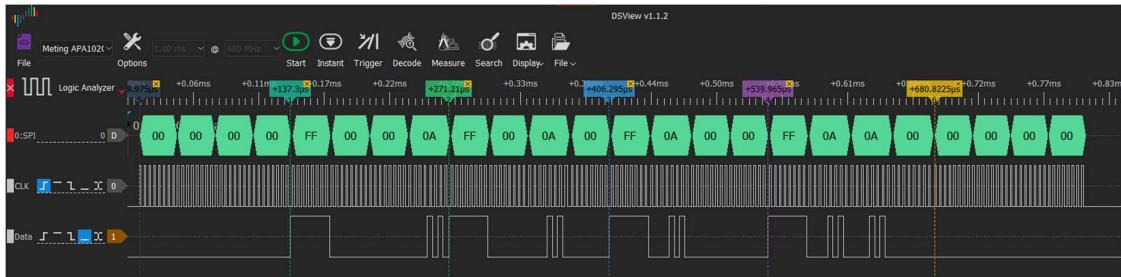
```

void LED_RGB(uint8_t brightnessR, uint8_t brightnessG, uint8_t brightnessB){
    for (i = 0; i < 8; i++) {
        HAL_GPIO_WritePin(SPI_MOSI_GPIO_Port, SPI_MOSI_Pin, GPIO_PIN_SET);
        SysTickDelayCount2(t);
        HAL_GPIO_WritePin(SPI_SCK_GPIO_Port, SPI_SCK_Pin, GPIO_PIN_SET);
        SysTickDelayCount2(t);
        HAL_GPIO_WritePin(SPI_SCK_GPIO_Port, SPI_SCK_Pin, GPIO_PIN_RESET);
    }
    for (i = 7; i >= 0; i--) {
        uint8_t k = brightnessB >> i;
        if (k & 1){
            HAL_GPIO_WritePin(SPI_MOSI_GPIO_Port, SPI_MOSI_Pin, GPIO_PIN_SET);
        }else{
            HAL_GPIO_WritePin(SPI_MOSI_GPIO_Port, SPI_MOSI_Pin, GPIO_PIN_RESET);
        }
        SysTickDelayCount2(t);
        HAL_GPIO_WritePin(SPI_SCK_GPIO_Port, SPI_SCK_Pin, GPIO_PIN_SET);
        SysTickDelayCount2(t);
        HAL_GPIO_WritePin(SPI_SCK_GPIO_Port, SPI_SCK_Pin, GPIO_PIN_RESET);
    }
    for (i = 7; i >= 0; i--) {
        uint8_t k = brightnessG >> i;
        if (k & 1){
            HAL_GPIO_WritePin(SPI_MOSI_GPIO_Port, SPI_MOSI_Pin, GPIO_PIN_SET);
        }else{
            HAL_GPIO_WritePin(SPI_MOSI_GPIO_Port, SPI_MOSI_Pin, GPIO_PIN_RESET);
        }
        SysTickDelayCount2(t);
        HAL_GPIO_WritePin(SPI_SCK_GPIO_Port, SPI_SCK_Pin, GPIO_PIN_SET);
        SysTickDelayCount2(t);
        HAL_GPIO_WritePin(SPI_SCK_GPIO_Port, SPI_SCK_Pin, GPIO_PIN_RESET);
    }
    for (i = 7; i >= 0; i--) {
        uint8_t k = brightnessR >> i;
        if (k & 1){
            HAL_GPIO_WritePin(SPI_MOSI_GPIO_Port, SPI_MOSI_Pin, GPIO_PIN_SET);
        }else{
            HAL_GPIO_WritePin(SPI_MOSI_GPIO_Port, SPI_MOSI_Pin, GPIO_PIN_RESET);
        }
        SysTickDelayCount2(t);
        HAL_GPIO_WritePin(SPI_SCK_GPIO_Port, SPI_SCK_Pin, GPIO_PIN_SET);
        SysTickDelayCount2(t);
        HAL_GPIO_WritePin(SPI_SCK_GPIO_Port, SPI_SCK_Pin, GPIO_PIN_RESET);
    }
}
}

```

3. En als al de kleuren ingesteld zijn stuur ik opnieuw een start frame zodanig dat ik direct kan herbeginnen met de 1^{ste} LED. Dit heb ik gevonden door te testen.

De signalen heb ik ook nagemeten met een Logic Analyzer:

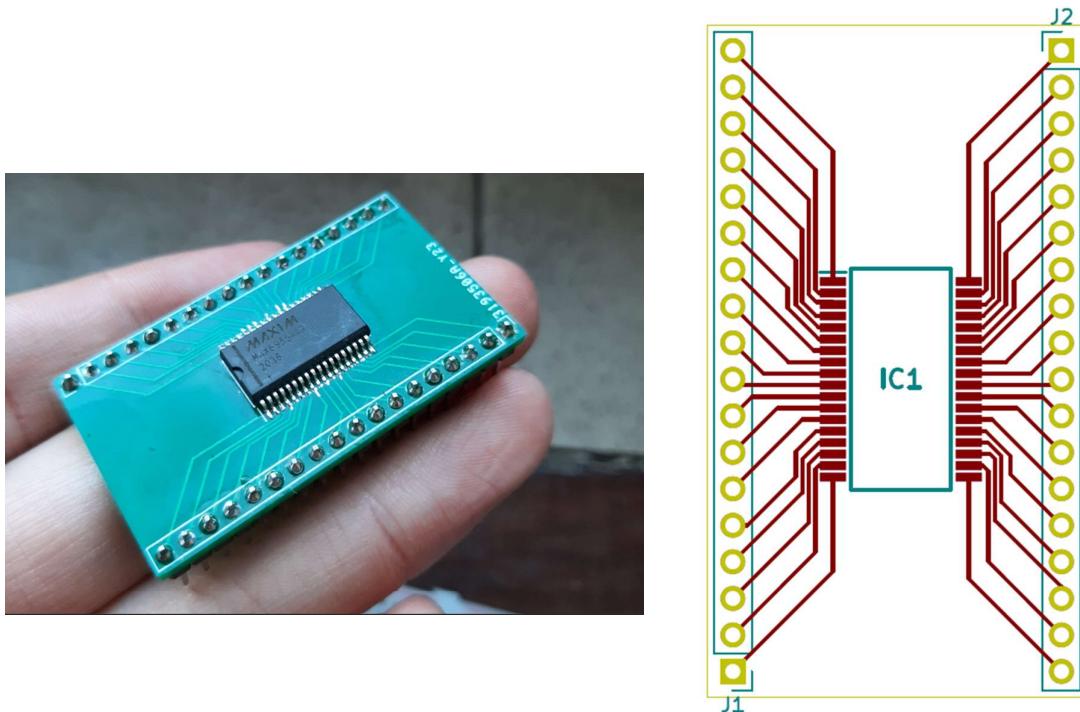


Figuur 17, Meting APA102C

Om de 32 bits staat een lijn om het beter te visualiseren.

2.3 Klok

Omdat de 7-Segment Driver die ik gebruik redelijk complex is heb ik hiervoor een testprint gemaakt zodat ik op voorhand kon bepalen hoe hij werkt.



Figuur 18, Testprint MAX6955

Doordat de MAX6955 werkt op I²C dus zal de code als volgend eruit moeten zien:

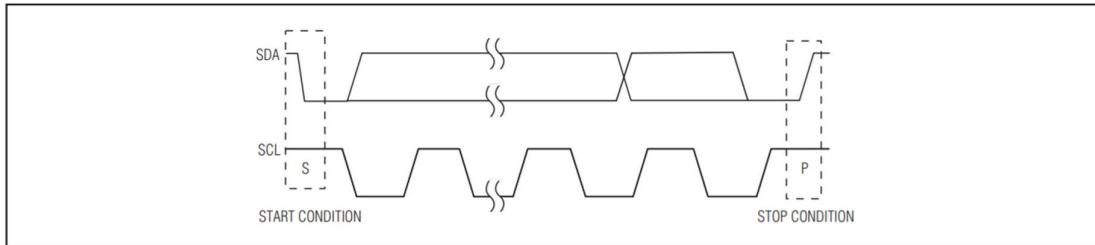


Figure 3. Start and Stop Conditions

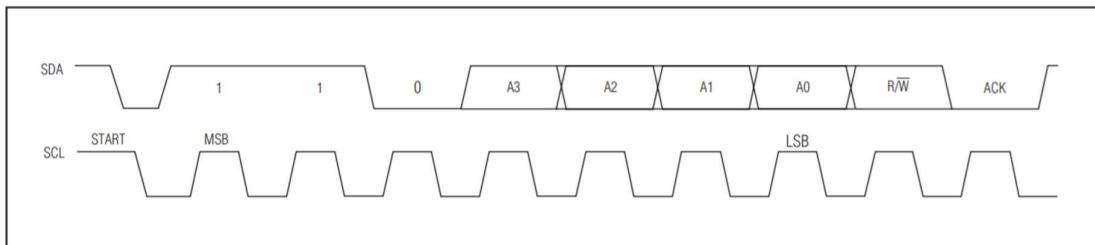


Figure 4. Slave Address

Figuur 19, Voorbeeld I²C

Omdat de STM32 een HAL I²C library zal ik hiervan gebruik maken. De volgende code wordt uitgevoerd in het begin zodanig dat erna vlot geprogrammeerd kan worden:

```
void SS_Start(uint8_t Test){  
  
    // Decode Mode Register (0x01) Table 15  
    // - 0x00 = Geen decoder gebruiken  
    // - 0xFF = Hexadecimale decoder gebruiken voor alle digits  
    REG_CONF = 0x01;  
    buf[0] = 0xFF;  
  
    HAL_I2C_Mem_Write(&hi2c2, MAX6955_ADDR, REG_CONF, 1, buf, 1, HAL_MAX_DELAY);  
  
    // Global Intensity Register (0x02) Table 27  
    // - 0x00 = 1/16 (min on)  
    // - 0x0F = 15/16 (max on)  
    REG_CONF = 0x02;  
    buf[0] = 0x07;  
  
    HAL_I2C_Mem_Write(&hi2c2, MAX6955_ADDR, REG_CONF, 1, buf, 1, HAL_MAX_DELAY);  
  
    // Scan Limit Register (0x03) Table  
    // - 0x00 alleen digit 0  
    // - 0x07 alle digits  
    REG_CONF = 0x03;  
    buf[0] = 0x07;
```

```

    HAL_I2C_Mem_Write(&hi2c2, MAX6955_ADDR, REG_CONF, 1, buf, 1, HAL_MAX_DELAY);

    // Configuration Register (0x04) uitleg blz 11
    // - 0x00 Shutdown
    // - 0x01 Normal operation
    // - ...
    REG_CONF = 0x04;
    buf[0] = 0b00000001;

    HAL_I2C_Mem_Write(&hi2c2, MAX6955_ADDR, REG_CONF, 1, buf, 1, HAL_MAX_DELAY);

    // Digit Type Register (0x0C) Table 13
    // - 0xFF digits 0-7 zijn 14-segment digits
    // - 0x00 digits 0-7 zijn 16- of 7-segment digits
    REG_CONF = 0x0C;
    buf[0] = 0x00;

    HAL_I2C_Mem_Write(&hi2c2, MAX6955_ADDR, REG_CONF, 1, buf, 1, HAL_MAX_DELAY);

    // Display Test Register (0x07) Table 37
    // - 0x00 Display Test Off
    // - 0x01 Display Test On
    REG_CONF = 0x07;
    if (Test == 1){
        buf[0] = 0x01;
    } else {
        buf[0] = 0x00;
    }

    HAL_I2C_Mem_Write(&hi2c2, MAX6955_ADDR, REG_CONF, 1, buf, 1, HAL_MAX_DELAY);

    // Stuur eerst naar planes om te beginnen
    // Hierna moet men enkel schrijven vanaf 0x86
    REG_CONF = 0x60;

    buf[0] = 0x80; // Het getal 0 met DP
    buf[1] = 0x81; // Het getal 1 met DP
    buf[2] = 0x82;
    buf[3] = 0x83;
    buf[4] = 0x84;
    buf[5] = 0x85;
    buf[6] = 0x86;
    buf[7] = 0x87;
    buf[8] = 0x88;
    buf[9] = 0x89;
    buf[10] = 0x8a; // De klinker A met DP
    buf[11] = 0x8b; // De klinker B met DP
    buf[12] = 0x8c;
    buf[13] = 0x8d;
    buf[14] = 0x8e;
    buf[15] = 0x8f;

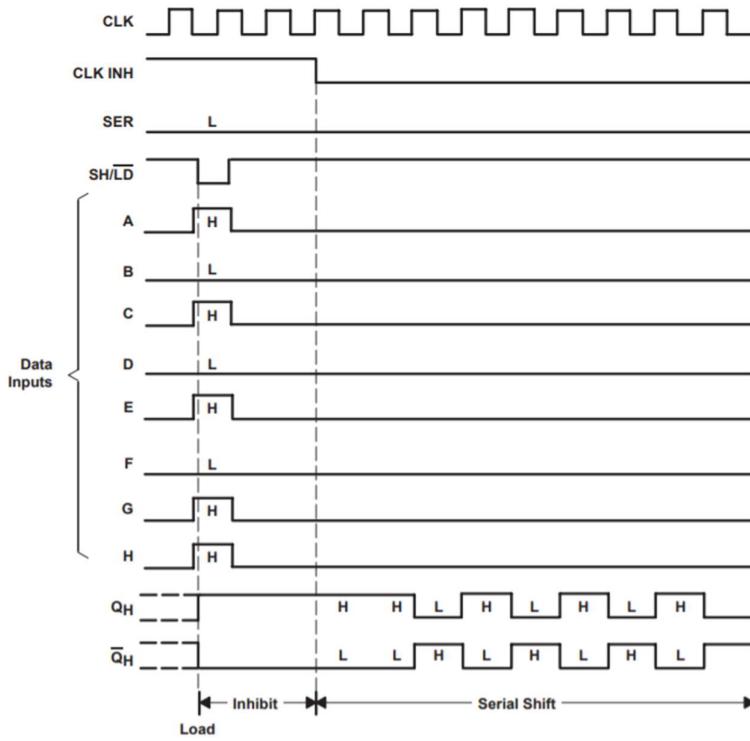
    HAL_I2C_Mem_Write(&hi2c2, MAX6955_ADDR, REG_CONF, 1, buf, 16, HAL_MAX_DELAY);
}

```

Hierna heb ik dan de registers waar ik na geschreven heb terug uit gelezen met de Logic Analyzer en deze kwamen overeen met de waardes die ik doorstuurd.

2.4 Positie uitlezing

Om de positie uit te lezen van de pionnen lees ik magneet sensoren uit in combinatie met shift registers. De shift registers worden zo uitgelezen:



Figuur 20, Uitlezen Shift Register

De volgende code zijn geschreven al bitbangend omdat de HAL SPI library niet vlot verliep en daarmee het zelf doe.

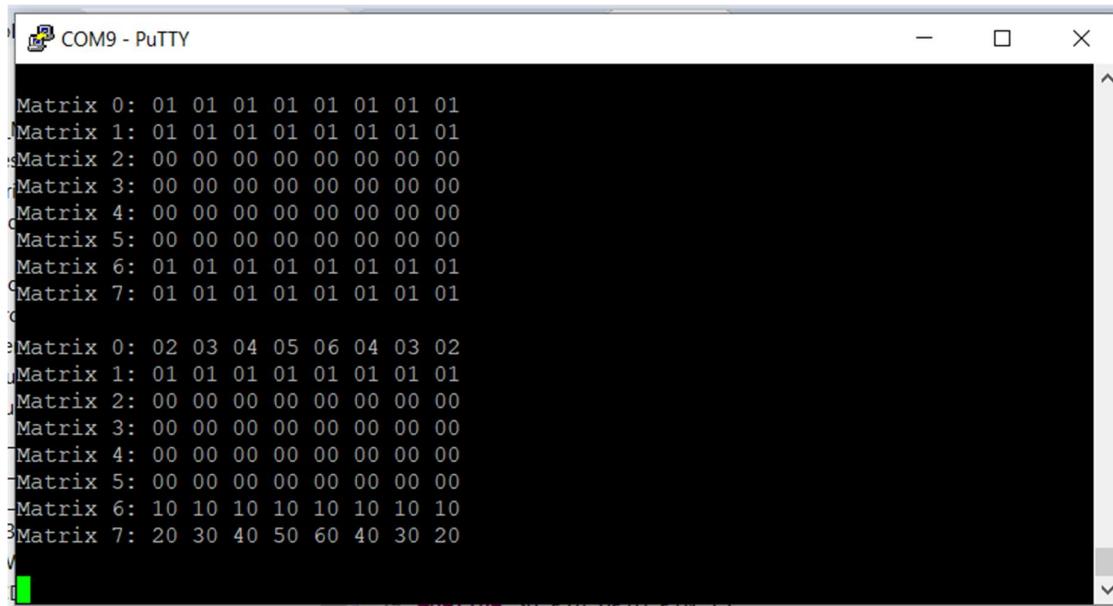
1. De functie "Meten" zal de shift registers uitlezen:

```
void Meten(){
    HAL_GPIO_WritePin(SPI1_CS_GPIO_Port, SPI1_CS_Pin, GPIO_PIN_RESET);
    SysTickDelayCount2(t);
    HAL_GPIO_WritePin(SPI1_CS_GPIO_Port, SPI1_CS_Pin, GPIO_PIN_SET);
    for (int8_t j = 0; j < 1; j++){
        for (int8_t i = 0; i < 8; i++){
            SysTickDelayCount2((t/2));
            // Inverteer de lezing
            buf[j][i] = (HAL_GPIO_ReadPin(SPI1_MISO_GPIO_Port, SPI1_MISO_Pin) ^ 1);
            SysTickDelayCount2((t/2));
            HAL_GPIO_WritePin(SPI1_CLK_GPIO_Port, SPI1_CLK_Pin, GPIO_PIN_SET);
            SysTickDelayCount2(t);
            HAL_GPIO_WritePin(SPI1_CLK_GPIO_Port, SPI1_CLK_Pin, GPIO_PIN_RESET);
        }
    }
}
```

- De functie "Controlere" zal de uitgelezen data van 1 shift register naar doorsturen via USB.

```
void Controle(){
    for (j = 0; j < 8; j++){
        for (i = 0; i < 8; i++){
            if (bufV[j][i] != buf[j][i]){
                coor[0] = j;
                coor[1] = i;
                Bstand[j][i] = 0;
                weg = 1;
                printf("Verandering op positie: %d %d \n\r", coor[0], coor[1]);
            } else {
                Bstand[j][i] = stand[j][i];
            }
        }
    }
}
```

De code hierboven geeft volgende output:



The screenshot shows a terminal window titled "COM9 - PuTTY". The window displays two sets of 8x8 binary matrices. The first set, labeled "Matrix 0" through "Matrix 7", represents the current state of the sensors. The second set, also labeled "Matrix 0" through "Matrix 7", represents the addresses assigned to each pawn type. The matrices are printed row by row, with each row starting with the matrix number and followed by the 8x8 binary grid.

```
Matrix 0: 01 01 01 01 01 01 01 01 01
Matrix 1: 01 01 01 01 01 01 01 01 01
Matrix 2: 00 00 00 00 00 00 00 00 00
Matrix 3: 00 00 00 00 00 00 00 00 00
Matrix 4: 00 00 00 00 00 00 00 00 00
Matrix 5: 00 00 00 00 00 00 00 00 00
Matrix 6: 01 01 01 01 01 01 01 01 01
Matrix 7: 01 01 01 01 01 01 01 01 01

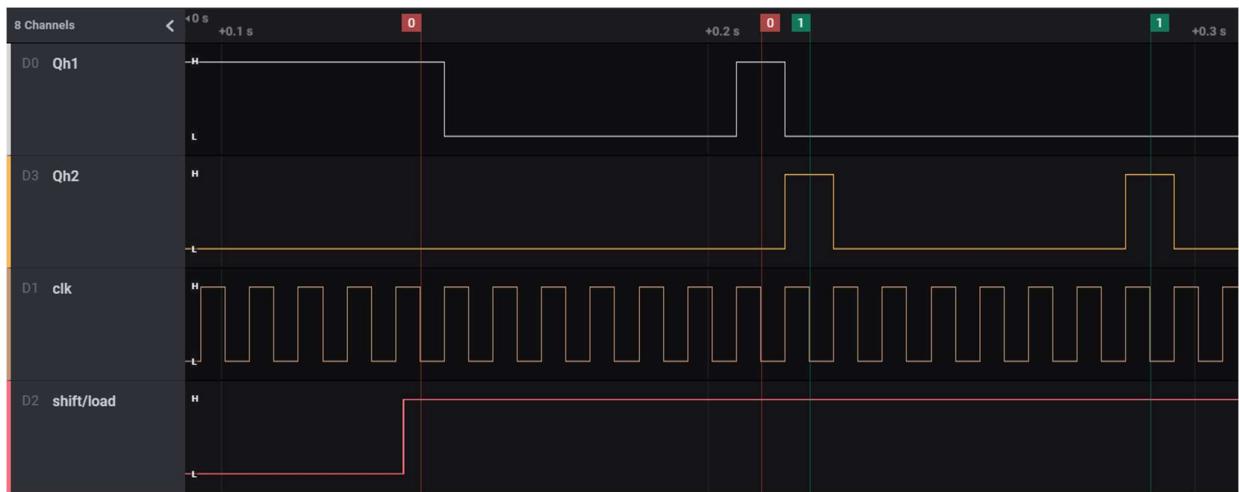
Matrix 0: 02 03 04 05 06 04 03 02
Matrix 1: 01 01 01 01 01 01 01 01 01
Matrix 2: 00 00 00 00 00 00 00 00 00
Matrix 3: 00 00 00 00 00 00 00 00 00
Matrix 4: 00 00 00 00 00 00 00 00 00
Matrix 5: 00 00 00 00 00 00 00 00 00
Matrix 6: 10 10 10 10 10 10 10 10 10
Matrix 7: 20 30 40 50 60 40 30 20
```

De bovenste matrix geeft de uitlezing van de magneet sensoren en de matrix daar onder geeft ieder pionsoort zijn eigen adres. Hierdoor is het eenvoudiger te programmeren welke pion van ieder team of soort is.

Als er een waarde verandert geeft hij de plaats weer:

```
Matrix 0: 00 00 00 00 00 00 00 00 00 00  
Matrix 1: 01 01 01 01 01 01 01 01 01 01  
Matrix 2: 00 00 00 00 00 00 00 00 00 00  
Matrix 3: 00 00 00 00 00 00 00 00 00 00  
Matrix 4: 00 00 00 00 00 00 00 00 00 00  
Matrix 5: 00 00 00 00 00 00 00 00 00 00  
Matrix 6: 01 01 01 01 01 01 01 01 01 01  
Matrix 7: 01 01 01 01 01 01 01 01 01 01  
  
Matrix 0: 02 03 04 05 06 04 03 02  
Matrix 1: 01 01 01 01 01 01 01 01 01 01  
Matrix 2: 00 00 00 00 00 00 00 00 00 00  
Matrix 3: 00 00 00 00 00 00 00 00 00 00  
Matrix 4: 00 00 00 00 00 00 00 00 00 00  
Matrix 5: 00 00 00 00 00 00 00 00 00 00  
Matrix 6: 10 10 10 10 10 10 10 10 10 10  
Matrix 7: 20 30 40 50 60 40 30 20  
  
Verandering op positie: 0 0  
Verandering op positie: 0 1  
Verandering op positie: 0 2  
Verandering op positie: 0 3  
Verandering op positie: 0 4  
Verandering op positie: 0 5  
Verandering op positie: 0 6  
Verandering op positie: 0 7
```

Ik heb 2 shift registers in cascade geplaatst in mijn test periode om alles te controleren. Deze heb ik dan hardware matig aangestuurd d.m.v. knoppen aan de Shift/Load pin en een functie generator. Hieronder een gemeten signaal uitgelezen met de Logic Analyzer:



Het uitgemeten signaal hierboven geeft aan wat de uitgang is van iedere shift register.

De punten tussen de rode strepen is de uitgang van het 1^{ste} shift register en tussen de groene strepen is de meting uit het 2^{de} shift register.

3 BESLUIT

Doorheen het schooljaar heb ik zeer veel bijgeleerd door dit project te maken, hier zijn enkele punten die ik bijgeleerd heb:

- Het ontwerpen van een PCB met SMD-componenten en deze ook zelf op solderen was zeker een uitdaging, maar ben aangenaam verast hoe vlot het solderen ging na wat oefenen.
- Het programmeren van ieder deel ging goed, maar door het tijdsgebrek heb ik het volledige project niet kunnen afwerken. Zoals de EEPROM die ik nog niet gebruikt heb en rotary encoder, deze ga ik later zelf nog implementeren.
- Het ontwerpen van de behuizing is ook niet afgeraakt en is dus enkel een 3D-schets van. Het ontwerp om de behuizing te 3D-printen is ook afgewerkt en kan dus geprint worden.
- Het uitzoeken hoe ik ieder component met elkaar zal communiceren en aangesloten worden was zeer interessant en is ook zeer goed gelukt.

Ten slotte is dus het volgende afgewerkt van het project:

- Er is een PCB ontworpen die bijna volledig werkend is maar ontbreekt 1 pin die niet naar buiten gebracht is (uitgang shift register)
- Er is 1 PCB volledig bestukt en werkend van de 8.
- De PCB van het hoofdbord is volledig bestukt en werkend alleen is de EEPROM nog niet getest en de drukknoppen.
- De 7-segmenten kan ik individueel programmeren, de RGB LED's kan ik apart aansturen en instellen, het shift register van de magneet sensoren kan ik volledig binnen lezen, de STM32F1 gans programmeren & debuggen.
- De magneet sensoren werken, de voltage translator werkt, de boot mode selector werkt, de beide voeding van 5V en 3V3 werkt.

MAX6955.c

```
2 * MAX6955.c
7 #include "main.h"
8
9 I2C_HandleTypeDef hi2c2;
10
11 static const uint8_t MAX6955_ADDR = 0x60 << 1; // Use 8-bit address
12 uint8_t REG_CONF;
13 uint8_t buf[15];
14
15 void SS_Start(uint8_t Test){
16
17     // Decode Mode Register (0x01) Table 15
18     // - 0x00 = Geen decoder gebruiken
19     // - 0xFF = Hexadecimale decoder gebruiken voor alle digits
20     REG_CONF = 0x01;
21     buf[0] = 0xFF;
22
23     HAL_I2C_Mem_Write(&hi2c2, MAX6955_ADDR, REG_CONF, 1, buf, 1, HAL_MAX_DELAY);
24
25     // Global Intensity Register (0x02) Table 27
26     // - 0x00 = 1/16 (min on)
27     // - 0x0F = 15/16 (max on)
28     REG_CONF = 0x02;
29     buf[0] = 0x07;
30
31     HAL_I2C_Mem_Write(&hi2c2, MAX6955_ADDR, REG_CONF, 1, buf, 1, HAL_MAX_DELAY);
32
33     // Scan Limit Register (0x03) Table
34     // - 0x00 alleen digit 0
35     // - 0x07 alle digits
36
37     REG_CONF = 0x03;
38     buf[0] = 0x07;
39
40     HAL_I2C_Mem_Write(&hi2c2, MAX6955_ADDR, REG_CONF, 1, buf, 1, HAL_MAX_DELAY);
41
42     // Configuration Register (0x04) uitleg blz 11
43     // - 0x00 Shutdown
44     // - 0x01 Normal operation
45     // - ...
46
47     REG_CONF = 0x04;
48     buf[0] = 0b00000001;
49
50     HAL_I2C_Mem_Write(&hi2c2, MAX6955_ADDR, REG_CONF, 1, buf, 1, HAL_MAX_DELAY);
51
52     // Digit Type Register (0x0C) Table 13
53     // - 0xFF digits 0-7 zijn 14-segment digits
54     // - 0x00 digits 0-7 zijn 16- of 7-segment digits
55     REG_CONF = 0x0C;
56     buf[0] = 0x00;
57
58     HAL_I2C_Mem_Write(&hi2c2, MAX6955_ADDR, REG_CONF, 1, buf, 1, HAL_MAX_DELAY);
59
60     // Display Test Register (0x07) Table 37
61     // - 0x00 Display Test Off
62     // - 0x01 Display Test On
```

MAX6955.c

```
63     REG_CONF = 0x07;
64     if (Test == 1){
65         buf[0] = 0x01;
66     } else {
67         buf[0] = 0x00;
68     }
69
70     HAL_I2C_Mem_Write(&hi2c2, MAX6955_ADDR, REG_CONF, 1, buf, 1, HAL_MAX_DELAY);
71
72 //  Stuur eerst naar planes om te beginnen
73 //  Hierna moet men enkel schrijven vanaf 0x86
74     REG_CONF = 0x60;
75
76     buf[0] = 0x80; // Het getal 0 met DP
77     buf[1] = 0x81; // Het getal 1 met DP
78     buf[2] = 0x82;
79     buf[3] = 0x83;
80
81     buf[4] = 0x84;
82     buf[5] = 0x85;
83     buf[6] = 0x86;
84     buf[7] = 0x87;
85
86     buf[8] = 0x88;
87     buf[9] = 0x89;
88     buf[10] = 0x8a; // De klinker A met DP
89     buf[11] = 0x8b; // De klinker B met DP
90
91     buf[12] = 0x8c;
92     buf[13] = 0x8d;
93     buf[14] = 0x8e;
94     buf[15] = 0x8f;
95
96     HAL_I2C_Mem_Write(&hi2c2, MAX6955_ADDR, REG_CONF, 1, buf, 16, HAL_MAX_DELAY);
97 }
98
```

MAX6955.h

```
2 * MAX6955.h
7
8 #ifndef INC_MAX6955_H_
9 #define INC_MAX6955_H_
10
11 void SS_Start(uint8_t Test);
12
13#endif /* INC_MAX6955_H_ */
14
```

APA102C.c

```
2 * APA102C.c
7 #include "main.h"
8
9 uint32_t t = 1;
10 int8_t i = 0;
11
12 void LED_Rood(uint8_t brightness){
13     for (i = 0; i < 8; i++) {
14         HAL_GPIO_WritePin(SPI2_MOSI_GPIO_Port, SPI2_MOSI_Pin, GPIO_PIN_SET);
15         SysTickDelayCount2(t);
16         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port, SPI2_SCK_Pin, GPIO_PIN_SET);
17         SysTickDelayCount2(t);
18         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port, SPI2_SCK_Pin, GPIO_PIN_RESET);
19     }
20     for (i = 0; i < 16; i++) {
21         HAL_GPIO_WritePin(SPI2_MOSI_GPIO_Port, SPI2_MOSI_Pin, GPIO_PIN_RESET);
22         SysTickDelayCount2(t);
23         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port, SPI2_SCK_Pin, GPIO_PIN_SET);
24         SysTickDelayCount2(t);
25         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port, SPI2_SCK_Pin, GPIO_PIN_RESET);
26     }
27     for (i = 7; i >= 0; i--) {
28         uint8_t k = brightness >> i;
29         if (k & 1){
30             HAL_GPIO_WritePin(SPI2_MOSI_GPIO_Port, SPI2_MOSI_Pin, GPIO_PIN_SET);
31         }else{
32             HAL_GPIO_WritePin(SPI2_MOSI_GPIO_Port, SPI2_MOSI_Pin, GPIO_PIN_RESET);
33         }
34         SysTickDelayCount2(t);
35         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port, SPI2_SCK_Pin, GPIO_PIN_SET);
36         SysTickDelayCount2(t);
37         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port, SPI2_SCK_Pin, GPIO_PIN_RESET);
38     }
39 }
40
41 void LED_Groen(uint8_t brightness){
42     for (i = 0; i < 8; i++) {
43         HAL_GPIO_WritePin(SPI2_MOSI_GPIO_Port, SPI2_MOSI_Pin, GPIO_PIN_SET);
44         SysTickDelayCount2(t);
45         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port, SPI2_SCK_Pin, GPIO_PIN_SET);
46         SysTickDelayCount2(t);
47         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port, SPI2_SCK_Pin, GPIO_PIN_RESET);
48     }
49     for (i = 0; i < 8; i++) {
50         HAL_GPIO_WritePin(SPI2_MOSI_GPIO_Port, SPI2_MOSI_Pin, GPIO_PIN_RESET);
51         SysTickDelayCount2(t);
52         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port, SPI2_SCK_Pin, GPIO_PIN_SET);
53         SysTickDelayCount2(t);
54         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port, SPI2_SCK_Pin, GPIO_PIN_RESET);
55     }
56     for (i = 7; i >= 0; i--) {
57         uint8_t k = brightness >> i;
58         if (k & 1){
59             HAL_GPIO_WritePin(SPI2_MOSI_GPIO_Port, SPI2_MOSI_Pin, GPIO_PIN_SET);
60         }else{
61             HAL_GPIO_WritePin(SPI2_MOSI_GPIO_Port, SPI2_MOSI_Pin, GPIO_PIN_RESET);
62         }
```

```

63     SysTickDelayCount2(t);
64     HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port,SPI2_SCK_Pin,GPIO_PIN_SET);
65     SysTickDelayCount2(t);
66     HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port,SPI2_SCK_Pin,GPIO_PIN_RESET);
67 }
68 for (i = 0; i < 8; i++) {
69     HAL_GPIO_WritePin(SPI2_MOSI_GPIO_Port,SPI2_MOSI_Pin,GPIO_PIN_RESET);
70     SysTickDelayCount2(t);
71     HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port,SPI2_SCK_Pin,GPIO_PIN_SET);
72     SysTickDelayCount2(t);
73     HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port,SPI2_SCK_Pin,GPIO_PIN_RESET);
74 }
75 }
76
77 void LED_Blauw(uint8_t brightness){
78     for (i = 0; i < 8; i++) {
79         HAL_GPIO_WritePin(SPI2_MOSI_GPIO_Port,SPI2_MOSI_Pin,GPIO_PIN_SET);
80         SysTickDelayCount2(t);
81         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port,SPI2_SCK_Pin,GPIO_PIN_SET);
82         SysTickDelayCount2(t);
83         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port,SPI2_SCK_Pin,GPIO_PIN_RESET);
84     }
85     for (i = 7; i >= 0; i--) {
86         uint8_t k = brightness >> i;
87         if (k & 1){
88             HAL_GPIO_WritePin(SPI2_MOSI_GPIO_Port,SPI2_MOSI_Pin,GPIO_PIN_SET);
89         }else{
90             HAL_GPIO_WritePin(SPI2_MOSI_GPIO_Port,SPI2_MOSI_Pin,GPIO_PIN_RESET);
91         }
92         SysTickDelayCount2(t);
93         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port,SPI2_SCK_Pin,GPIO_PIN_SET);
94         SysTickDelayCount2(t);
95         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port,SPI2_SCK_Pin,GPIO_PIN_RESET);
96     }
97     for (i = 0; i < 16; i++) {
98         HAL_GPIO_WritePin(SPI2_MOSI_GPIO_Port,SPI2_MOSI_Pin,GPIO_PIN_RESET);
99         SysTickDelayCount2(t);
100        HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port,SPI2_SCK_Pin,GPIO_PIN_SET);
101        SysTickDelayCount2(t);
102        HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port,SPI2_SCK_Pin,GPIO_PIN_RESET);
103    }
104 }
105
106 void LED_RGB(uint8_t brightnessR, uint8_t brightnessG, uint8_t brightnessB){
107     for (i = 0; i < 8; i++) {
108         HAL_GPIO_WritePin(SPI2_MOSI_GPIO_Port,SPI2_MOSI_Pin,GPIO_PIN_SET);
109         SysTickDelayCount2(t);
110         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port,SPI2_SCK_Pin,GPIO_PIN_SET);
111         SysTickDelayCount2(t);
112         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port,SPI2_SCK_Pin,GPIO_PIN_RESET);
113     }
114     for (i = 7; i >= 0; i--) {
115         uint8_t k = brightnessB >> i;
116         if (k & 1){
117             HAL_GPIO_WritePin(SPI2_MOSI_GPIO_Port,SPI2_MOSI_Pin,GPIO_PIN_SET);
118         }else{
119             HAL_GPIO_WritePin(SPI2_MOSI_GPIO_Port,SPI2_MOSI_Pin,GPIO_PIN_RESET);

```

```

120         }
121         SysTickDelayCount2(t);
122         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port,SPI2_SCK_Pin,GPIO_PIN_SET);
123         SysTickDelayCount2(t);
124         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port,SPI2_SCK_Pin,GPIO_PIN_RESET);
125     }
126     for (i = 7; i >= 0; i--) {
127         uint8_t k = brightnessG >> i;
128         if (k & 1){
129             HAL_GPIO_WritePin(SPI2_MOSI_GPIO_Port,SPI2_MOSI_Pin,GPIO_PIN_SET);
130         }else{
131             HAL_GPIO_WritePin(SPI2_MOSI_GPIO_Port,SPI2_MOSI_Pin,GPIO_PIN_RESET);
132         }
133         SysTickDelayCount2(t);
134         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port,SPI2_SCK_Pin,GPIO_PIN_SET);
135         SysTickDelayCount2(t);
136         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port,SPI2_SCK_Pin,GPIO_PIN_RESET);
137     }
138     for (i = 7; i >= 0; i--) {
139         uint8_t k = brightnessR >> i;
140         if (k & 1){
141             HAL_GPIO_WritePin(SPI2_MOSI_GPIO_Port,SPI2_MOSI_Pin,GPIO_PIN_SET);
142         }else{
143             HAL_GPIO_WritePin(SPI2_MOSI_GPIO_Port,SPI2_MOSI_Pin,GPIO_PIN_RESET);
144         }
145         SysTickDelayCount2(t);
146         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port,SPI2_SCK_Pin,GPIO_PIN_SET);
147         SysTickDelayCount2(t);
148         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port,SPI2_SCK_Pin,GPIO_PIN_RESET);
149     }
150 }
151
152 void LED_Start(){
153     for (i = 0; i < 32; i++) {
154         HAL_GPIO_WritePin(SPI2_MOSI_GPIO_Port,SPI2_MOSI_Pin,GPIO_PIN_RESET);
155         SysTickDelayCount2(t);
156         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port,SPI2_SCK_Pin,GPIO_PIN_SET);
157         SysTickDelayCount2(t);
158         HAL_GPIO_WritePin(SPI2_SCK_GPIO_Port,SPI2_SCK_Pin,GPIO_PIN_RESET);
159     }
160 }
161

```

APA102C.h

```
2 * APA102C.h
7
8 #ifndef INC_APACHE_H_
9 #define INC_APACHE_H_
10
11 void LED_Rood(uint8_t brightness);
12 void LED_Groen(uint8_t brightness);
13 void LED_Blaauw(uint8_t brightness);
14 void LED_RGB(uint8_t brightnessR, uint8_t brightnessG, uint8_t brightnessB);
15 void LED_Start(void);
16
17 #endif /* INC_APACHE_H_ */
```

```

main.c

1 /* USER CODE BEGIN Header */
4  * @file      : main.c
19 /* USER CODE END Header */
20 /* Includes -----*/
21 #include "main.h"
22
23 /* Private includes -----*/
24 /* USER CODE BEGIN Includes */
25 #include "APA102C.h"
26 #include "MAX6955.h"
27
28 /* USER CODE END Includes */
29
30 /* Private typedef -----*/
31 /* USER CODE BEGIN PTD */
32
33 /* USER CODE END PTD */
34
35 /* Private define -----*/
36 /* USER CODE BEGIN PD */
37
38 /* USER CODE END PD */
39
40 /* Private macro -----*/
41 /* USER CODE BEGIN PM */
42
43 /* USER CODE END PM */
44
45 /* Private variables -----*/
46 I2C_HandleTypeDef hi2c2;
47
48 /* USER CODE BEGIN PV */
49 uint8_t buf[15];
50 /* USER CODE END PV */
51
52 /* Private function prototypes -----*/
53 void SystemClock_Config(void);
54 static void MX_GPIO_Init(void);
55 static void MX_I2C2_Init(void);
56 /* USER CODE BEGIN PFP */
57
58 /* USER CODE END PFP */
59
60 /* Private user code -----*/
61 /* USER CODE BEGIN 0 */
62 void SysTickDelayCount2(unsigned long ulCount){
63     CoreDebug->DEMCR |= CoreDebug_DEMCR_TRCENA_Msk;
64     ITM->LAR = 0xC5ACCE55;
65     DWT->CYCCNT = 0;
66     DWT->CTRL |= DWT_CTRL_CYCCNTENA_Msk;
67
68     while(DWT->CYCCNT < ulCount);
69 }
70 /*
71 void Meten(){
72     HAL_GPIO_WritePin(SPI1_CS_GPIO_Port,SPI1_CS_Pin,GPIO_PIN_RESET);
73     SysTickDelayCount2(t);

```

main.c

```
74     HAL_GPIO_WritePin(SPI1_CS_GPIO_Port,SPI1_CS_Pin,GPIO_PIN_SET);
75     for (int8_t j = 0; j < 1; j++){
76         for (int8_t i = 0; i < 8; i++){
77             SysTickDelayCount2((t/2));
78             buf[j][i] = (HAL_GPIO_ReadPin(SPI1_MISO_GPIO_Port,SPI1_MISO_Pin) ^ 1); //Inverteer
de lezing
79             SysTickDelayCount2((t/2));
80             HAL_GPIO_WritePin(SPI1_CLK_GPIO_Port,SPI1_CLK_Pin,GPIO_PIN_SET);
81             SysTickDelayCount2(t);
82             HAL_GPIO_WritePin(SPI1_CLK_GPIO_Port,SPI1_CLK_Pin,GPIO_PIN_RESET);
83         }
84     }
85 }
86 */
87 /* USER CODE END 0 */
88
89 /**
90 * @brief  The application entry point.
91 * @retval int
92 */
93 int main(void)
94 {
95     /* USER CODE BEGIN 1 */
96
97     /* USER CODE END 1 */
98
99     /* MCU Configuration-----*/
100
101    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
102    HAL_Init();
103
104    /* USER CODE BEGIN Init */
105
106    /* USER CODE END Init */
107
108    /* Configure the system clock */
109    SystemClock_Config();
110
111    /* USER CODE BEGIN SysInit */
112
113    /* USER CODE END SysInit */
114
115    /* Initialize all configured peripherals */
116    MX_GPIO_Init();
117    MX_I2C2_Init();
118    /* USER CODE BEGIN 2 */
119
120    // 7-segment aansturen in test stand
121    SS_Start(1);
122
123    // APA102C's aansturen
124    LED_Start();
125    LED_Rood(10);
126    LED_Groen(10);
127    LED_Blaauw(10);
128    LED_RGB(0, 10, 10);
129    LED_Start();
```

main.c

```
130  /* USER CODE END 2 */
131
132
133 /* Infinite loop */
134 /* USER CODE BEGIN WHILE */
135 while (1)
136 {
137     /* USER CODE END WHILE */
138
139     /* USER CODE BEGIN 3 */
140 }
141 /* USER CODE END 3 */
142 }
143
144 /**
145 * @brief System Clock Configuration
146 * @retval None
147 */
148 void SystemClock_Config(void)
149 {
150     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
151     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
152
153     /** Initializes the RCC Oscillators according to the specified parameters
154     * in the RCC_OscInitTypeDef structure.
155     */
156     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
157     RCC_OscInitStruct.HSIStrate = RCC_HSI_ON;
158     RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
159     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
160     RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI_DIV2;
161     RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
162     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
163     {
164         Error_Handler();
165     }
166     /** Initializes the CPU, AHB and APB buses clocks
167     */
168     RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
169                 |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
170     RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
171     RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
172     RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
173     RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
174
175     if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
176     {
177         Error_Handler();
178     }
179 }
180
181 /**
182 * @brief I2C2 Initialization Function
183 * @param None
184 * @retval None
185 */
186 static void MX_I2C2_Init(void)
```

main.c

```
187 {
188
189 /* USER CODE BEGIN I2C2_Init_0 */
190
191 /* USER CODE END I2C2_Init_0 */
192
193 /* USER CODE BEGIN I2C2_Init_1 */
194
195 /* USER CODE END I2C2_Init_1 */
196 hi2c2.Instance = I2C2;
197 hi2c2.Init.ClockSpeed = 100000;
198 hi2c2.Init.DutyCycle = I2C_DUTYCYCLE_2;
199 hi2c2.Init.OwnAddress1 = 0;
200 hi2c2.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
201 hi2c2.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
202 hi2c2.Init.OwnAddress2 = 0;
203 hi2c2.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
204 hi2c2.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
205 if (HAL_I2C_Init(&hi2c2) != HAL_OK)
206 {
207     Error_Handler();
208 }
209 /* USER CODE BEGIN I2C2_Init_2 */
210
211 /* USER CODE END I2C2_Init_2 */
212
213 }
214
215 /**
216 * @brief GPIO Initialization Function
217 * @param None
218 * @retval None
219 */
220 static void MX_GPIO_Init(void)
221 {
222     GPIO_InitTypeDef GPIO_InitStruct = {0};
223
224     /* GPIO Ports Clock Enable */
225     __HAL_RCC_GPIOC_CLK_ENABLE();
226     __HAL_RCC_GPIOA_CLK_ENABLE();
227     __HAL_RCC_GPIOB_CLK_ENABLE();
228
229     /*Configure GPIO pin Output Level */
230     HAL_GPIO_WritePin(SPI1_CS_GPIO_Port, SPI1_CS_Pin, GPIO_PIN_RESET);
231
232     /*Configure GPIO pin Output Level */
233     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2|SPI1_CLK_Pin, GPIO_PIN_RESET);
234
235     /*Configure GPIO pin Output Level */
236     HAL_GPIO_WritePin(GPIOB, SPI2_SCK_Pin|SPI2_MOSI_Pin, GPIO_PIN_RESET);
237
238     /*Configure GPIO pin : SPI1_CS_Pin */
239     GPIO_InitStruct.Pin = SPI1_CS_Pin;
240     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
241     GPIO_InitStruct.Pull = GPIO_NOPULL;
242     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
243     HAL_GPIO_Init(SPI1_CS_GPIO_Port, &GPIO_InitStruct);
```

main.c

```
244 /*Configure GPIO pins : PA2 SPI1_CLK_Pin */
245 GPIO_InitStruct.Pin = GPIO_PIN_2|SPI1_CLK_Pin;
246 GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
247 GPIO_InitStruct.Pull = GPIO_NOPULL;
248 GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
249 HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
250
251 /*Configure GPIO pin : SPI1_MISO_Pin */
252 GPIO_InitStruct.Pin = SPI1_MISO_Pin;
253 GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
254 GPIO_InitStruct.Pull = GPIO_NOPULL;
255 HAL_GPIO_Init(SPI1_MISO_GPIO_Port, &GPIO_InitStruct);
256
257 /*Configure GPIO pins : SPI2_SCK_Pin SPI2_MOSI_Pin */
258 GPIO_InitStruct.Pin = SPI2_SCK_Pin|SPI2_MOSI_Pin;
259 GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
260 GPIO_InitStruct.Pull = GPIO_NOPULL;
261 GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
262 HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
263
264 }
265
266
267 /* USER CODE BEGIN 4 */
268
269 /* USER CODE END 4 */
270
271 /**
272 * @brief This function is executed in case of error occurrence.
273 * @retval None
274 */
275 void Error_Handler(void)
276 {
277 /* USER CODE BEGIN Error_Handler_Debug */
278 /* User can add his own implementation to report the HAL error return state */
279 __disable_irq();
280 while (1)
281 {
282 }
283 /* USER CODE END Error_Handler_Debug */
284 }
285
286 #ifdef USE_FULL_ASSERT
287 /**
288 * @brief Reports the name of the source file and the source line number
289 * where the assert_param error has occurred.
290 * @param file: pointer to the source file name
291 * @param line: assert_param error line source number
292 * @retval None
293 */
294 void assert_failed(uint8_t *file, uint32_t line)
295 {
296 /* USER CODE BEGIN 6 */
297 /* User can add his own implementation to report the file name and line number,
298 ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
299 /* USER CODE END 6 */
300 }
```

main.c

```
301 #endif /* USE_FULL_ASSERT */  
302  
303 /***** (C) COPYRIGHT STMicroelectronics *****END OF FILE****/  
304
```

```
main.h

1/* USER CODE BEGIN Header */
4 * @file           : main.h
20/* USER CODE END Header */
21
22/* Define to prevent recursive inclusion -----*/
23#ifndef __MAIN_H
24#define __MAIN_H
25
26#endif __cplusplus
27extern "C" {
28#endif
29
30/* Includes -----*/
31#include "stm32f1xx_hal.h"
32
33/* Private includes -----*/
34/* USER CODE BEGIN Includes */
35
36/* USER CODE END Includes */
37
38/* Exported types -----*/
39/* USER CODE BEGIN ET */
40
41/* USER CODE END ET */
42
43/* Exported constants -----*/
44/* USER CODE BEGIN EC */
45
46/* USER CODE END EC */
47
48/* Exported macro -----*/
49/* USER CODE BEGIN EM */
50
51/* USER CODE END EM */
52
53/* Exported functions prototypes -----*/
54void Error_Handler(void);
55
56/* USER CODE BEGIN EFP */
57
58/* USER CODE END EFP */
59
60/* Private defines -----*/
61#define SPI1_CS_Pin GPIO_PIN_1
62#define SPI1_CS_GPIO_Port GPIOC
63#define SPI1_CLK_Pin GPIO_PIN_5
64#define SPI1_CLK_GPIO_Port GPIOA
65#define SPI1_MISO_Pin GPIO_PIN_6
66#define SPI1_MISO_GPIO_Port GPIOA
67#define SPI2_SCK_Pin GPIO_PIN_13
68#define SPI2_SCK_GPIO_Port GPIOB
69#define SPI2_MOSI_Pin GPIO_PIN_15
70#define SPI2_MOSI_GPIO_Port GPIOB
71/* USER CODE BEGIN Private defines */
72
73/* USER CODE END Private defines */
74
```

main.h

```
75 #ifdef __cplusplus
76 }
77#endif
78
79#endif /* __MAIN_H */
80
81 /***** (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
82
```

main.c

```
1 /* USER CODE BEGIN Header */
4  * @file          : main.c
19 /* USER CODE END Header */
20 /* Includes -----*/
21 #include "main.h"
22
23 /* Private includes -----*/
24 /* USER CODE BEGIN Includes */
25
26 /* USER CODE END Includes */
27
28 /* Private typedef -----*/
29 /* USER CODE BEGIN PTD */
30
31 /* USER CODE END PTD */
32
33 /* Private define -----*/
34 /* USER CODE BEGIN PD */
35 #define t 1000
36 /* USER CODE END PD */
37
38 /* Private macro -----*/
39 /* USER CODE BEGIN PM */
40
41 /* USER CODE END PM */
42
43 /* Private variables -----*/
44
45 TIM_HandleTypeDef htim2;
46
47 UART_HandleTypeDef huart1;
48
49 /* USER CODE BEGIN PV */
50 uint8_t stand[8][8] = {
51     {02, 03, 04, 05, 06, 04, 03, 02},
52     {01, 01, 01, 01, 01, 01, 01, 01},
53     {00, 00, 00, 00, 00, 00, 00, 00},
54     {00, 00, 00, 00, 00, 00, 00, 00},
55     {00, 00, 00, 00, 00, 00, 00, 00},
56     {00, 00, 00, 00, 00, 00, 00, 00},
57     {10, 10, 10, 10, 10, 10, 10, 10},
58     {20, 30, 40, 50, 60, 40, 30, 20}
59 };
60 uint8_t Bstand[8][8] = {
61     {02, 03, 04, 05, 06, 04, 03, 02},
62     {01, 01, 01, 01, 01, 01, 01, 01},
63     {00, 00, 00, 00, 00, 00, 00, 00},
64     {00, 00, 00, 00, 00, 00, 00, 00},
65     {00, 00, 00, 00, 00, 00, 00, 00},
66     {00, 00, 00, 00, 00, 00, 00, 00},
67     {10, 10, 10, 10, 10, 10, 10, 10},
68     {20, 30, 40, 50, 60, 40, 30, 20}
69 };
70 uint8_t buf[8][8] = {
71     {00, 00, 00, 00, 00, 00, 00, 01},
72     {01, 01, 01, 01, 01, 01, 01, 01},
73     {00, 00, 00, 00, 00, 00, 00, 00},
```

```

main.c

74     {00, 00, 00, 00, 00, 00, 00, 00},
75     {00, 00, 00, 00, 00, 00, 00, 00},
76     {00, 00, 00, 00, 00, 00, 00, 00},
77     {01, 01, 01, 01, 01, 01, 01, 01},
78     {01, 01, 01, 01, 01, 01, 01, 01}
79 };
80 uint8_t bufV[8][8] = {
81     {00, 00, 00, 00, 00, 00, 00, 01},
82     {01, 01, 01, 01, 01, 01, 01, 01},
83     {00, 00, 00, 00, 00, 00, 00, 00},
84     {00, 00, 00, 00, 00, 00, 00, 00},
85     {00, 00, 00, 00, 00, 00, 00, 00},
86     {00, 00, 00, 00, 00, 00, 00, 00},
87     {01, 01, 01, 01, 01, 01, 01, 01},
88     {01, 01, 01, 01, 01, 01, 01, 01}
89 };
90 uint8_t coor[2];
91 uint8_t i = 0;
92 uint8_t j = 0;
93 uint8_t weg = 0;
94 /* USER CODE END PV */
95
96 /* Private function prototypes -----*/
97 void SystemClock_Config(void);
98 static void MX_GPIO_Init(void);
99 static void MX_TIM2_Init(void);
100 static void MX_USART1_UART_Init(void);
101 /* USER CODE BEGIN PFP */
102
103 /* USER CODE END PFP */
104
105 /* Private user code -----*/
106 /* USER CODE BEGIN 0 */
107 #include <errno.h>
108 #include <sys/stat.h>
109 #include <sys/types.h>
110 #include <sys/unistd.h>
111 int _write(int file, char *ptr, int len) {
112     HAL_StatusTypeDef xStatus;
113     switch (file) {
114     case STDOUT_FILENO: /*stdout*/
115         xStatus = HAL_UART_Transmit(&huart1, (uint8_t*)ptr, len, HAL_MAX_DELAY);
116         if (xStatus != HAL_OK) {
117             errno = EIO;
118             return -1;
119         }
120         break;
121     case STDERR_FILENO: /* stderr */
122         xStatus = HAL_UART_Transmit(&huart1, (uint8_t*)ptr, len, HAL_MAX_DELAY);
123         if (xStatus != HAL_OK) {
124             errno = EIO;
125             return -1;
126         }
127         break;
128     default:
129         errno = EBADF;
130         return -1;

```

main.c

```
131     }
132     return len;
133 }
134
135 void SysTickDelayCount2(unsigned long ulCount){
136     CoreDebug->DEMCR |= CoreDebug_DEMCR_TRCENA_Msk;
137     DWT->LAR = 0xC5ACCE55;
138     DWT->CYCCNT = 0;
139     DWT->CTRL |= DWT_CTRL_CYCCNTENA_Msk;
140
141     while(DWT->CYCCNT < ulCount);
142 }
143
144 void Meten(){
145     HAL_GPIO_WritePin(SH_Port,SH_Pin,GPIO_PIN_RESET);
146     SysTickDelayCount2(t);
147     HAL_GPIO_WritePin(SH_Port,SH_Pin,GPIO_PIN_SET);
148     for (j = 0; j < 1; j++){
149         for (i = 0; i < 8; i++){
150             SysTickDelayCount2((t/2));
151             buf[j][i] = (HAL_GPIO_ReadPin(SPI_MISO_GPIO_Port,SPI_MISO_Pin) ^ 1); //Inverteer
de lezing
152             SysTickDelayCount2((t/2));
153             HAL_GPIO_WritePin(SPI_CLK_GPIO_Port,SPI_CLK_Pin,GPIO_PIN_SET);
154             SysTickDelayCount2(t);
155             HAL_GPIO_WritePin(SPI_CLK_GPIO_Port,SPI_CLK_Pin,GPIO_PIN_RESET);
156         }
157     }
158 }
159
160 void Controle(){
161     for (j = 0; j < 8; j++){
162         for (i = 0; i < 8; i++){
163             if (bufV[j][i] != buf[j][i]){
164                 coor[0] = j;
165                 coor[1] = i;
166                 Bstand[j][i] = 0;
167                 weg = 1;
168                 printf("Verandering op positie: %d %d \n\r", coor[0], coor[1]);
169             } else {
170                 Bstand[j][i] = stand[j][i];
171             }
172         }
173     }
174 }
175 /* USER CODE END 0 */
176
177 /**
178 * @brief The application entry point.
179 * @retval int
180 */
181 int main(void)
182 {
183     /* USER CODE BEGIN 1 */
184
185     /* USER CODE END 1 */
186 }
```

main.c

```
187 /* MCU Configuration-----*/
188
189 /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
190 HAL_Init();
191
192 /* USER CODE BEGIN Init */
193
194 /* USER CODE END Init */
195
196 /* Configure the system clock */
197 SystemClock_Config();
198
199 /* USER CODE BEGIN SysInit */
200
201 /* USER CODE END SysInit */
202
203 /* Initialize all configured peripherals */
204 MX_GPIO_Init();
205 MX_TIM2_Init();
206 MX_USART1_UART_Init();
207 /* USER CODE BEGIN 2 */
208 /* USER CODE END 2 */
209
210 /* Infinite loop */
211 /* USER CODE BEGIN WHILE */
212 while (1)
213 {
214     /* USER CODE END WHILE */
215
216     /* USER CODE BEGIN 3 */
217     Meten();
218     for (j = 0; j < 8; j++){
219         printf("Matrix %d: ", j);
220         for (i = 0; i < 8; i++){
221             printf("%02d ", buf[j][i]);
222         }
223         printf("\n\r");
224     }
225     printf("\n\r");
226     for (j = 0; j < 8; j++){
227         printf("Matrix %d: ", j);
228         for (i = 0; i < 8; i++){
229             printf("%02d ", Bstand[j][i]);
230         }
231         printf("\n\r");
232     }
233     printf("\n\r");
234
235     Controle();
236
237     for (j = 0; j < 8; j++){
238         for (i = 0; i < 8; i++){
239             bufV[j][i] = buf[j][i];
240         }
241     }
242
243     HAL_Delay(1000);
```

main.c

```
244
245 }
246 /* USER CODE END 3 */
247 }
248
249 /**
250 * @brief System Clock Configuration
251 * @retval None
252 */
253 void SystemClock_Config(void)
254 {
255     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
256     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
257     RCC_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};
258
259     /** Configure LSE Drive Capability
260     */
261     HAL_PWR_EnableBkUpAccess();
262     /** Configure the main internal regulator output voltage
263     */
264     __HAL_RCC_PWR_CLK_ENABLE();
265     __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
266     /** Initializes the RCC Oscillators according to the specified parameters
267     * in the RCC_OscInitTypeDef structure.
268     */
269     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
270     RCC_OscInitStruct.HSISState = RCC_HSI_ON;
271     RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
272     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
273     RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
274     RCC_OscInitStruct.PLL.PLLM = 8;
275     RCC_OscInitStruct.PLL.PLLN = 200;
276     RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
277     RCC_OscInitStruct.PLL.PLLQ = 2;
278     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
279     {
280         Error_Handler();
281     }
282     /** Activate the Over-Drive mode
283     */
284     if (HAL_PWREx_EnableOverDrive() != HAL_OK)
285     {
286         Error_Handler();
287     }
288     /** Initializes the CPU, AHB and APB buses clocks
289     */
290     RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
291                     |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
292     RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
293     RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
294     RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
295     RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
296
297     if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_6) != HAL_OK)
298     {
299         Error_Handler();
300     }
```

main.c

```
301 PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_USART1;
302 PeriphClkInitStruct.Usart1ClockSelection = RCC_USART1CLKSOURCE_PCLK2;
303 if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct) != HAL_OK)
304 {
305     Error_Handler();
306 }
307 }
308
309 /**
310 * @brief TIM2 Initialization Function
311 * @param None
312 * @retval None
313 */
314 static void MX_TIM2_Init(void)
315 {
316
317     /* USER CODE BEGIN TIM2_Init 0 */
318
319     /* USER CODE END TIM2_Init 0 */
320
321     TIM_ClockConfigTypeDef sClockSourceConfig = {0};
322     TIM_MasterConfigTypeDef sMasterConfig = {0};
323
324     /* USER CODE BEGIN TIM2_Init 1 */
325
326     /* USER CODE END TIM2_Init 1 */
327     htim2.Instance = TIM2;
328     htim2.Init.Prescaler = 0;
329     htim2.Init.CounterMode = TIM_COUNTERMODE_DOWN;
330     htim2.Init.Period = 650;
331     htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
332     htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
333     if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
334     {
335         Error_Handler();
336     }
337     sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
338     if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
339     {
340         Error_Handler();
341     }
342     sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
343     sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
344     if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
345     {
346         Error_Handler();
347     }
348     /* USER CODE BEGIN TIM2_Init 2 */
349
350     /* USER CODE END TIM2_Init 2 */
351
352 }
353
354 /**
355 * @brief USART1 Initialization Function
356 * @param None
357 * @retval None
```

main.c

```
358 */
359 static void MX_USART1_UART_Init(void)
360 {
361
362 /* USER CODE BEGIN USART1_Init 0 */
363
364 /* USER CODE END USART1_Init 0 */
365
366 /* USER CODE BEGIN USART1_Init 1 */
367
368 /* USER CODE END USART1_Init 1 */
369 huart1.Instance = USART1;
370 huart1.Init.BaudRate = 115200;
371 huart1.Init.WordLength = UART_WORDLENGTH_8B;
372 huart1.Init.StopBits = UART_STOPBITS_1;
373 huart1.Init.Parity = UART_PARITY_NONE;
374 huart1.Init.Mode = UART_MODE_TX_RX;
375 huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
376 huart1.Init.OverSampling = UART_OVERSAMPLING_16;
377 huart1.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
378 huart1.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
379 if (HAL_UART_Init(&huart1) != HAL_OK)
380 {
381     Error_Handler();
382 }
383 /* USER CODE BEGIN USART1_Init 2 */
384
385 /* USER CODE END USART1_Init 2 */
386
387 }
388
389 /**
390 * @brief GPIO Initialization Function
391 * @param None
392 * @retval None
393 */
394 static void MX_GPIO_Init(void)
395 {
396     GPIO_InitTypeDef GPIO_InitStruct = {0};
397
398     /* GPIO Ports Clock Enable */
399     __HAL_RCC_GPIOI_CLK_ENABLE();
400     __HAL_RCC_GPIOA_CLK_ENABLE();
401     __HAL_RCC_GPIOC_CLK_ENABLE();
402     __HAL_RCC_GPIOH_CLK_ENABLE();
403     __HAL_RCC_GPIOB_CLK_ENABLE();
404
405     /*Configure GPIO pin Output Level */
406     HAL_GPIO_WritePin(SPI_CLK_PIN_GPIO_Port, SPI_CLK_PIN_Pin, GPIO_PIN_SET);
407
408     /*Configure GPIO pin Output Level */
409     HAL_GPIO_WritePin(SH_GPIO_Port, SH_Pin, GPIO_PIN_RESET);
410
411     /*Configure GPIO pin : SPI_CLK_PIN_Pin */
412     GPIO_InitStruct.Pin = SPI_CLK_PIN_Pin;
413     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
414     GPIO_InitStruct.Pull = GPIO_PULLUP;
```

main.c

```
415 GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
416 HAL_GPIO_Init(SPI_CLK_PIN_GPIO_Port, &GPIO_InitStruct);
417
418 /*Configure GPIO pin : SPI_MISO_PIN_Pin */
419 GPIO_InitStruct.Pin = SPI_MISO_PIN_Pin;
420 GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
421 GPIO_InitStruct.Pull = GPIO_NOPULL;
422 HAL_GPIO_Init(SPI_MISO_PIN_GPIO_Port, &GPIO_InitStruct);
423
424 /*Configure GPIO pin : SH_Pin */
425 GPIO_InitStruct.Pin = SH_Pin;
426 GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
427 GPIO_InitStruct.Pull = GPIO_PULLUP;
428 GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
429 HAL_GPIO_Init(SH_GPIO_Port, &GPIO_InitStruct);
430
431 }
432
433 /* USER CODE BEGIN 4 */
434
435 /* USER CODE END 4 */
436
437 /**
438 * @brief This function is executed in case of error occurrence.
439 * @retval None
440 */
441 void Error_Handler(void)
442 {
443 /* USER CODE BEGIN Error_Handler_Debug */
444 /* User can add his own implementation to report the HAL error return state */
445 __disable_irq();
446 while (1)
447 {
448 }
449 /* USER CODE END Error_Handler_Debug */
450 }
451
452 #ifdef USE_FULL_ASSERT
453 /**
454 * @brief Reports the name of the source file and the source line number
455 * where the assert_param error has occurred.
456 * @param file: pointer to the source file name
457 * @param line: assert_param error line source number
458 * @retval None
459 */
460 void assert_failed(uint8_t *file, uint32_t line)
461 {
462 /* USER CODE BEGIN 6 */
463 /* User can add his own implementation to report the file name and line number,
464 ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
465 /* USER CODE END 6 */
466 }
467 #endif /* USE_FULL_ASSERT */
468
469 /***** (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
470
```

```
main.h

1 /* USER CODE BEGIN Header */
4  * @file          : main.h
20 /* USER CODE END Header */
21
22 /* Define to prevent recursive inclusion -----*/
23 #ifndef __MAIN_H
24 #define __MAIN_H
25
26 #ifdef __cplusplus
27 extern "C" {
28#endif
29
30 /* Includes -----*/
31 #include "stm32f7xx_hal.h"
32
33 /* Private includes -----*/
34 /* USER CODE BEGIN Includes */
35 #define SPI_CLK_Pin GPIO_PIN_1
36 #define SPI_CLK_GPIO_Port GPIOI
37 #define SPI_MISO_Pin GPIO_PIN_14
38 #define SPI_MISO_GPIO_Port GPIOB
39 #define SH_Pin GPIO_PIN_15
40 #define SH_Port GPIOB
41 /* USER CODE END Includes */
42
43 /* Exported types -----*/
44 /* USER CODE BEGIN ET */
45
46 /* USER CODE END ET */
47
48 /* Exported constants -----*/
49 /* USER CODE BEGIN EC */
50
51 /* USER CODE END EC */
52
53 /* Exported macro -----*/
54 /* USER CODE BEGIN EM */
55
56 /* USER CODE END EM */
57
58 /* Exported functions prototypes -----*/
59 void Error_Handler(void);
60
61 /* USER CODE BEGIN EFP */
62
63 /* USER CODE END EFP */
64
65 /* Private defines -----*/
66 #define SPI_CLK_PIN_Pin GPIO_PIN_1
67 #define SPI_CLK_PIN_GPIO_Port GPIOI
68 #define SPI_MISO_PIN_Pin GPIO_PIN_14
69 #define SPI_MISO_PIN_GPIO_Port GPIOB
70 #define SH_Pin GPIO_PIN_15
71 #define SH_GPIO_Port GPIOB
72 /* USER CODE BEGIN Private defines */
73
74 /* USER CODE END Private defines */
```

main.h

```
75
76 #ifdef __cplusplus
77 }
78#endif
79
80#endif /* __MAIN_H */
81
82 /***** (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
83
```