



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 moeilijker 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 moeilijker SMD-componenten.

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

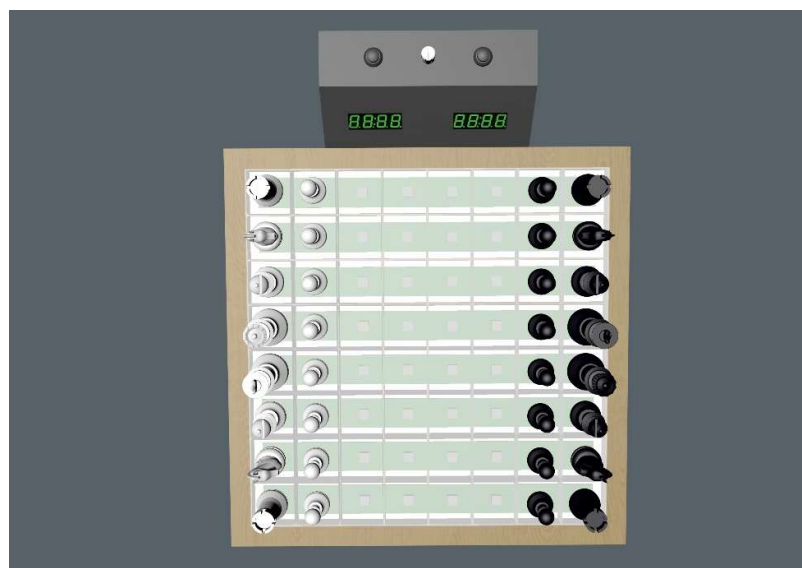
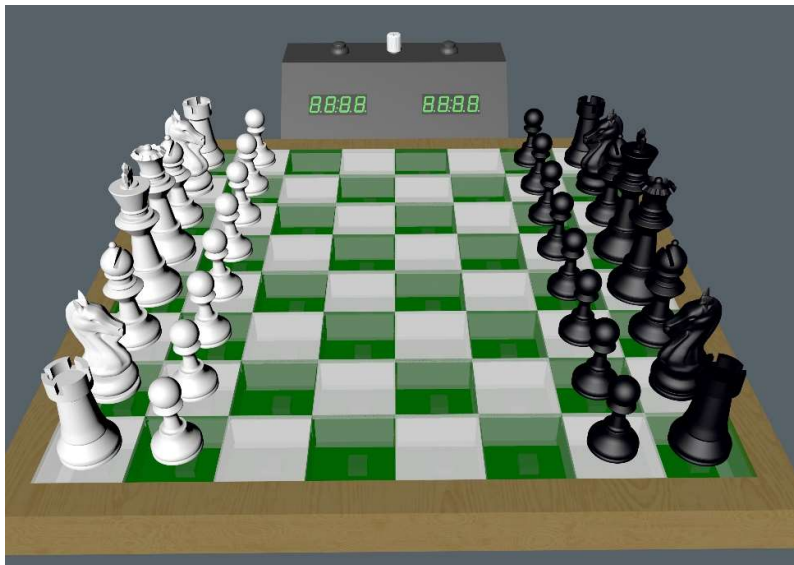
En uiteraard kon ik ook altijd op de steun van mijn vrienden rekenen, waar ik altijd aan iets kon vragen of om mijn ideeën te kunnen realiseren.

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 μ Controller) 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 zou 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.

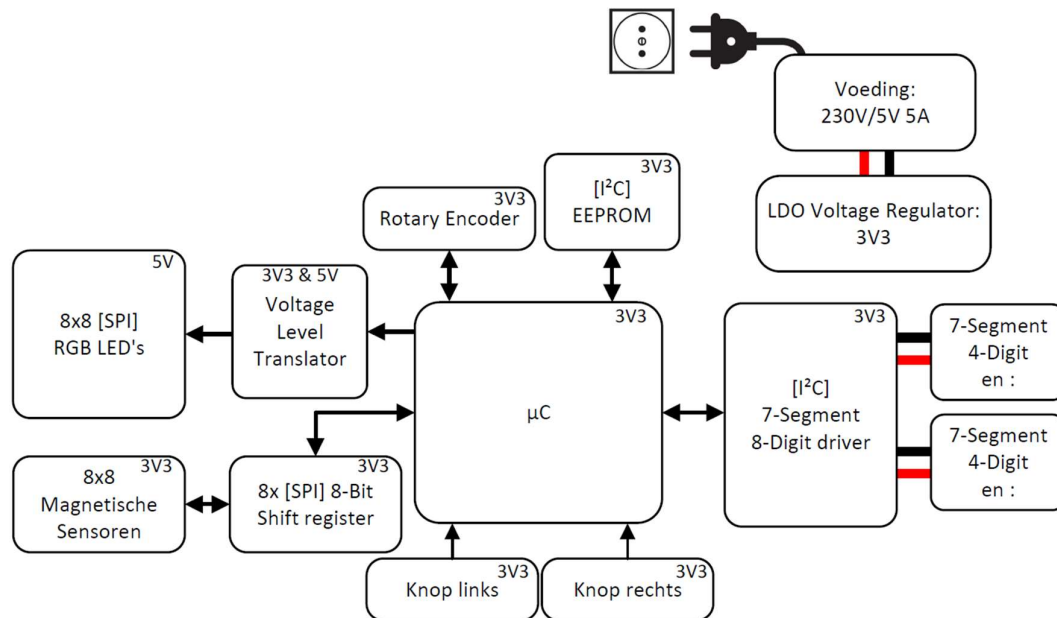
Er is ook een behuizing die zeer nauwkeurig moet zijn wegens de standaard van het schaken, hier onder is een 3D-schets van hoe het ganse project er zal uit zien:



1 HARDWARE

1.1 Blokschema

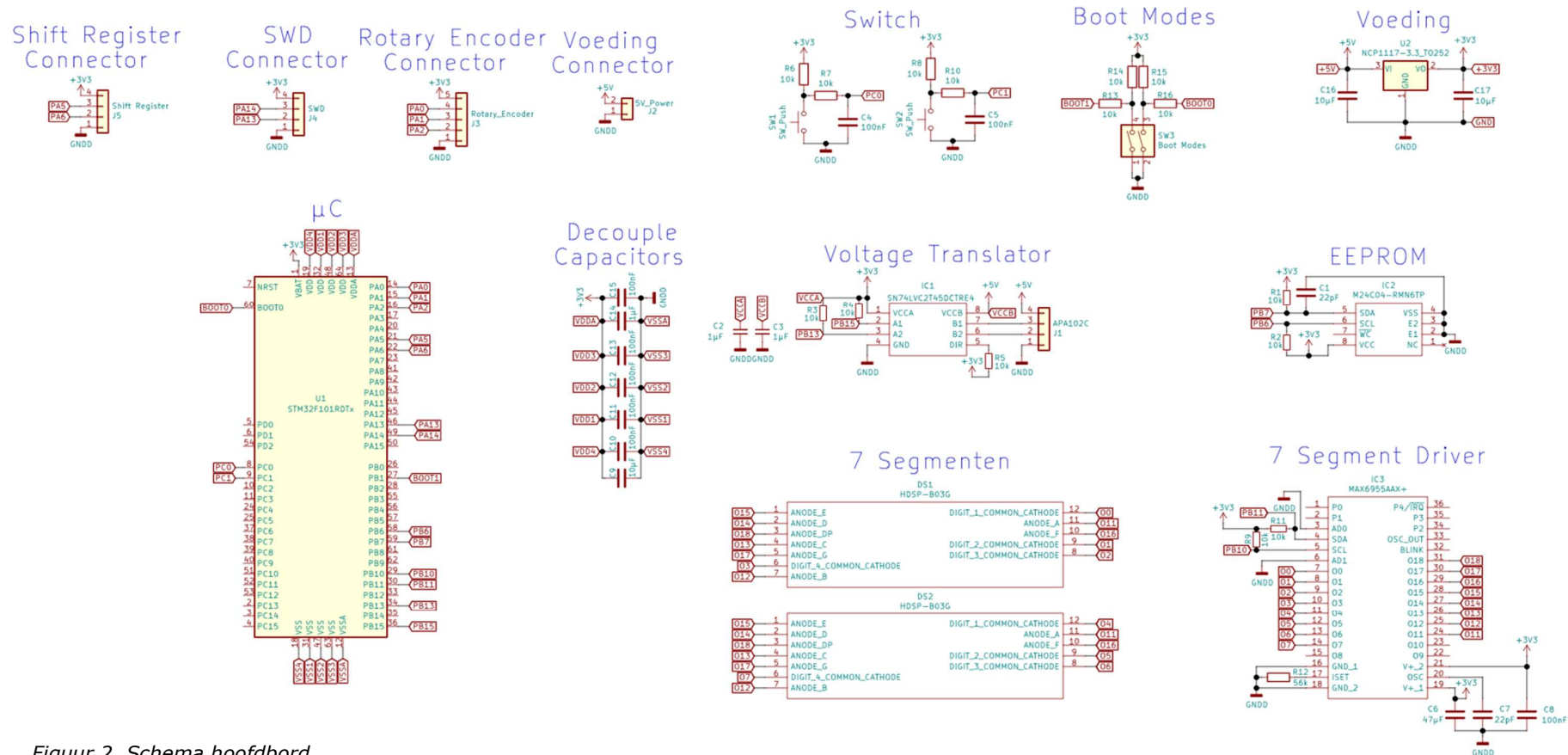
In het blokschema zien we hoe de verschillende blokken verbonden zijn met elkaar maar dit in een vereenvoudigde weergave. Hierbij kan je zien hoe alles logisch met elkaar verbonden is d.m.v. voedingsaansluitingen en communicatielijnen.



Figuur 1, Blokschema

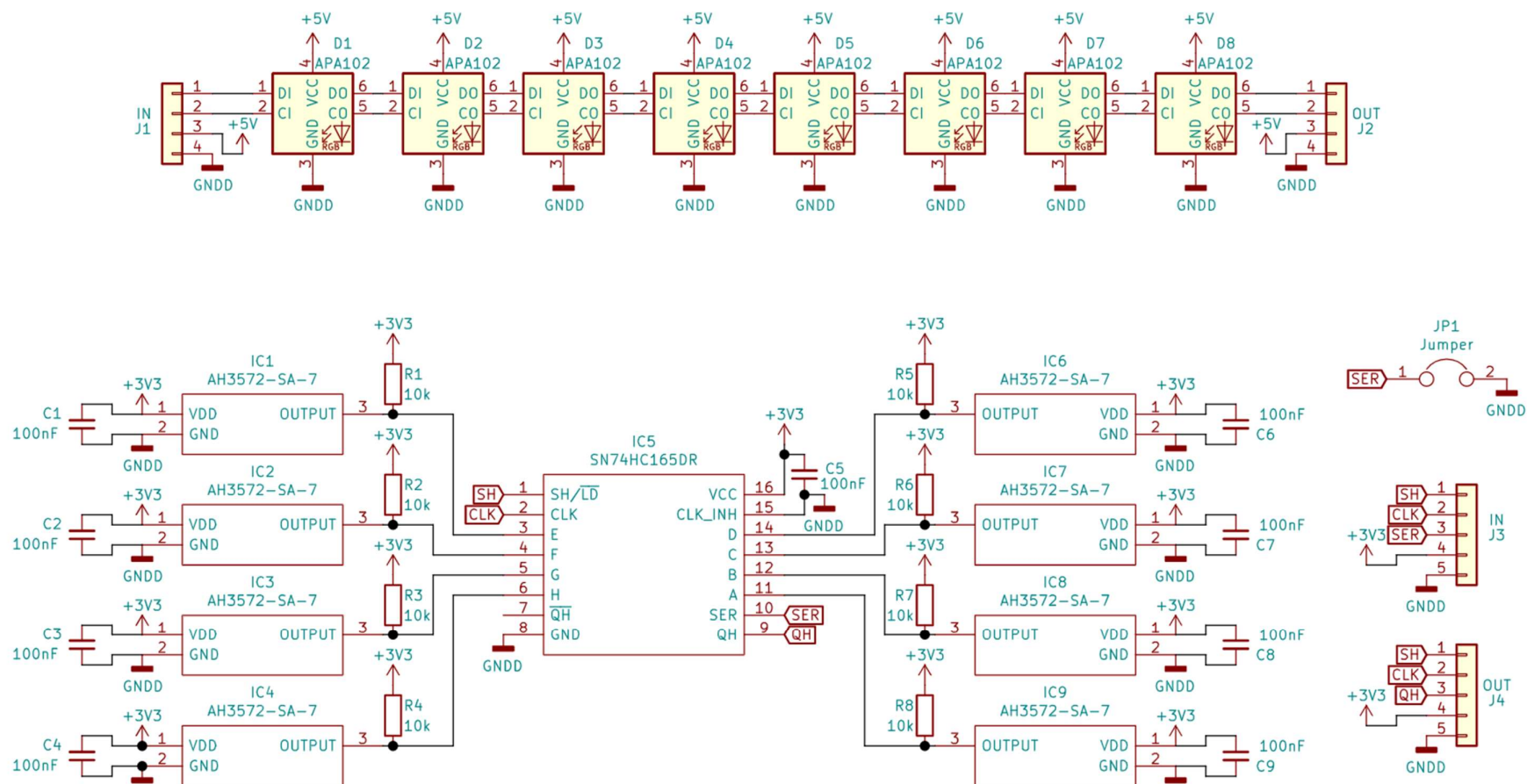
1.2 Schema's

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 bordes die zich onderaan de pionnen liggen, Later wordt ieder deel besproken.

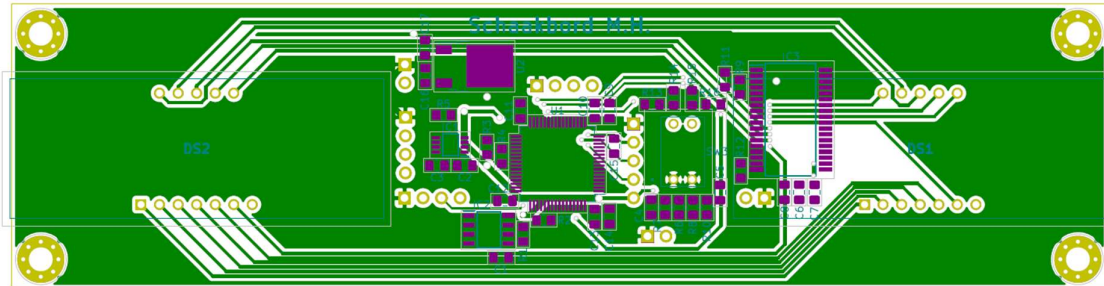


Figuur 3, Schema pionnen

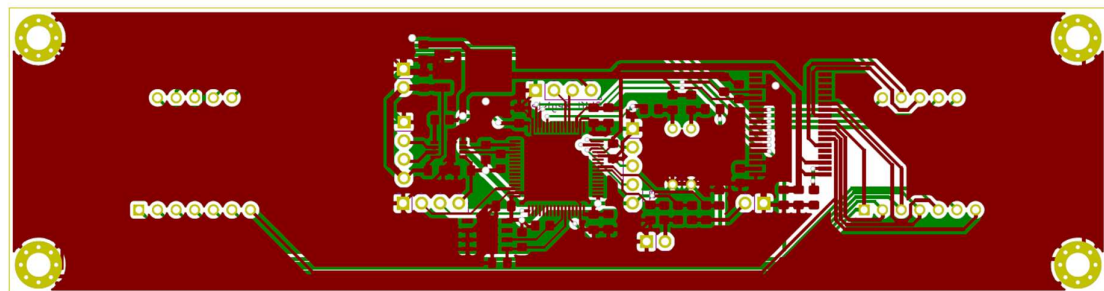
1.3 PCB's

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.

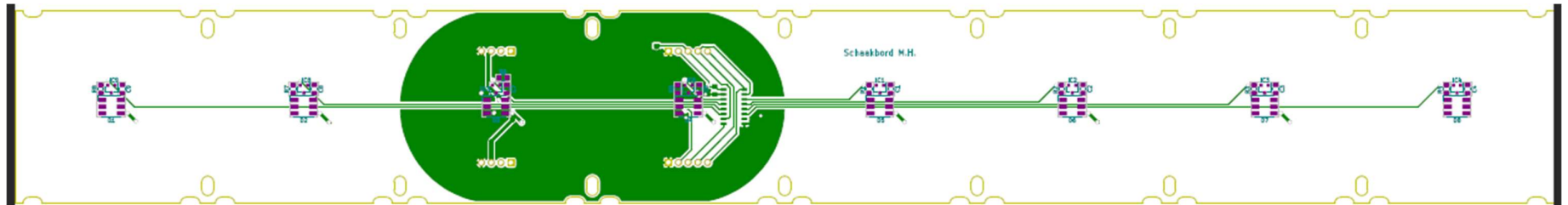
Bovenkant Hoofdbord:



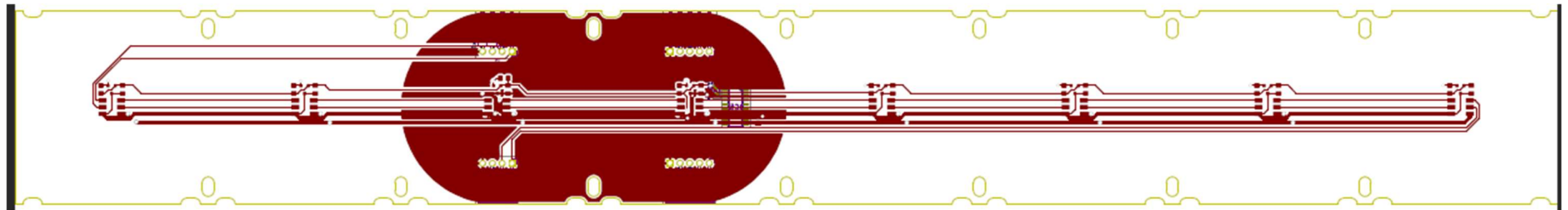
Onderkant Hoofdbord:



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)	5V voor Schakelende voeding		
5	73,5	64	4,704	23,52			
STM32F101RDTx (µC)							
Uin(V)	Iin(mA)			Itot(mA)	Ptot(mW)	I _{max} (mA)	
3,3	20			20	66		
AH3572 (Magnetic Sensor)							
Uin(V)	Iin(mA)	Aantal	Itot(mA)	Ptot(mW)	3V3 voor LDO		
3,3	3	64	192	633,6			
NCP1117 (LDO)							
Uin(V)	Uuit(V)	Iq(mA)	Itot(mA)	Ptot(mW)	I _{max} (mA)	P _{max} (mW)	
5,2	3,2	10	20	40	305,899	983,466	

(*) De totale stromen onder 1mA wordt niet meegerekent. (EEPROM, Shift Register en Voltage Translator)

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 (µC):

Power/Memory		Enabled Peripherals	
Power Mode	RUN	GPIOA GPIOB GPIOC	
Power Scale	No Scale	I2C1 I2C2 SPI1 SPI2	
Memory Fetch Type	FLASH	TIM2	
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		

Itot 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 schakbord: 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:

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

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

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:

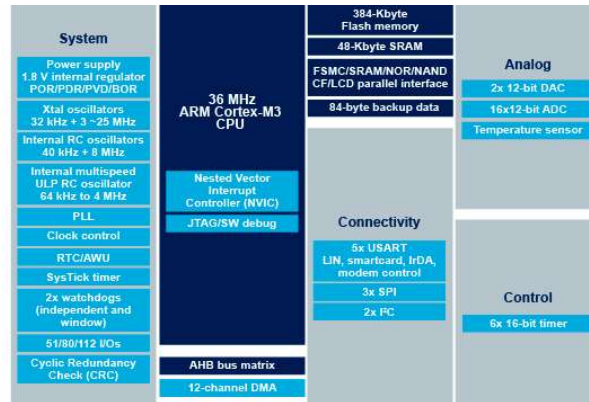
- Communicatieprotocollen zoals SPI & I²C zijn eenvoudiger te implementeren.
- Om lengte/grootte van software te beperken.
- Eenvoudiger te debuggen.

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 gekend 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.



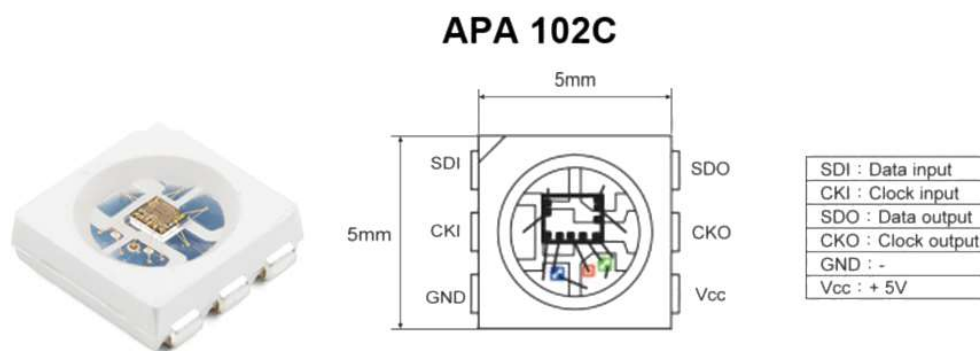
Figuur 6, Circuit Diagram

1.4.4 RGB LED's

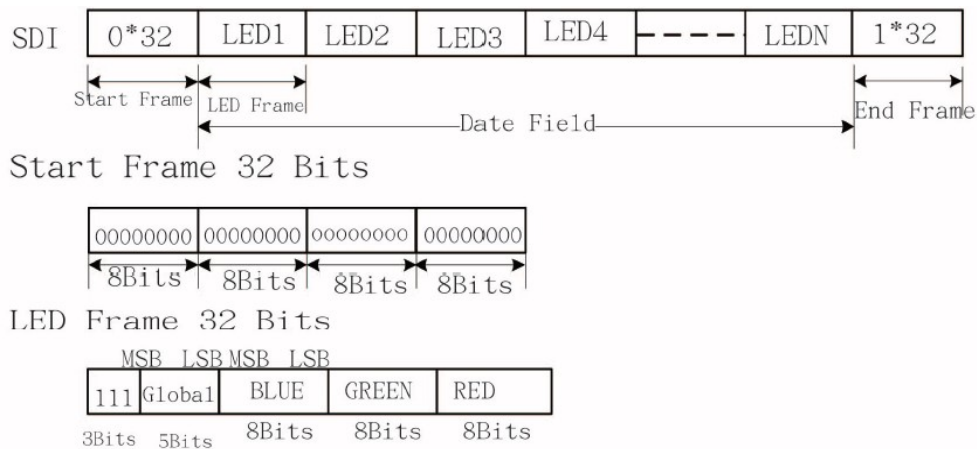
Om de schaakvakken een kleur te geven maak ik gebruik van adresseerbare RGB LED's, hierdoor kan ik elk vak individueel een andere kleur geven wat dat belangrijk is voor aan te duiden wat de pion hun mogelijke zetten zijn en meer.

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.

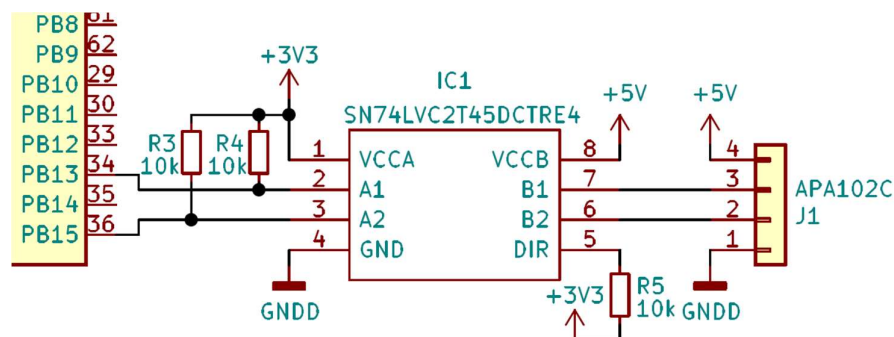
Vervolgens stuurt men de data door van de tweede LED.

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 schakbord bestaat uit 64 RGB LED's, dat wil zeggen dat er 32 keer 4 bytes die '1' zijn (eind frame) verstuurd 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 SN74LVC2T45DCTRE4, hiermee zal ik de SCK (seriële klok) en de SDI (seriële data in) omzetten naar 5V logica.

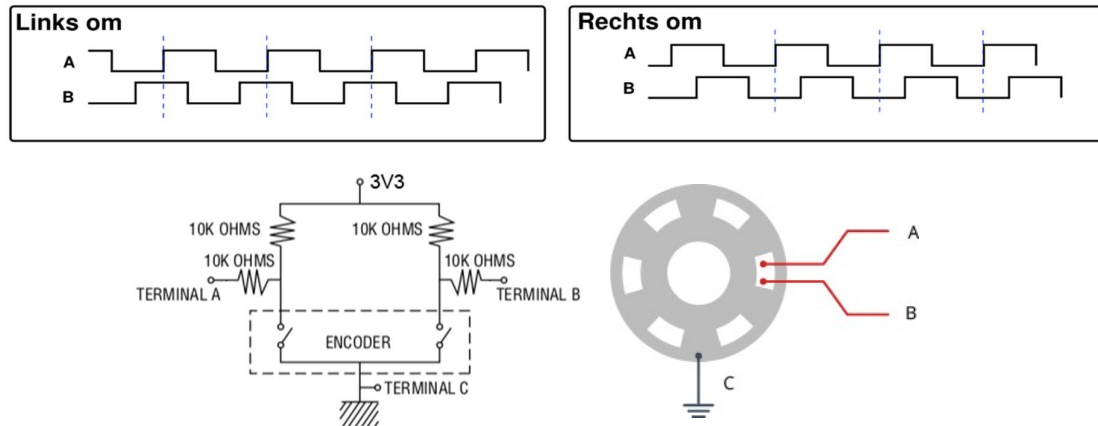


Figuur 9, Schema Voltage Translator

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 volgt:

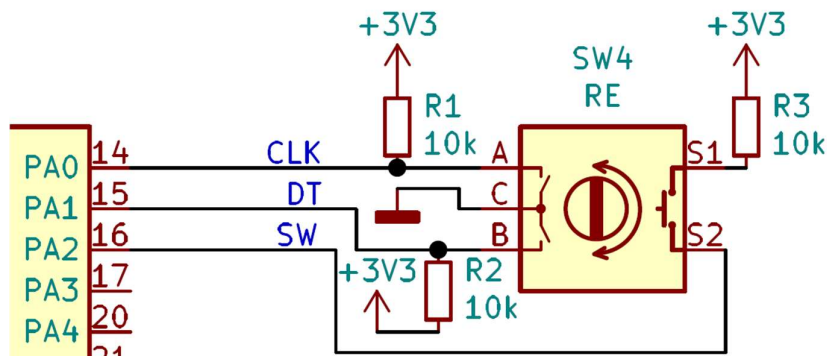


Figuur 10, Werking Rotary Encoder

Wanneer er één stand in wijzerzin (↻) 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 (↻) opnieuw gedraaid wordt zal nu Terminal B op hetzelfde contact van Terminal A staan.

Dit is omgekeerd als men één stand in tegenwijzerzin (↻) 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 (↻) 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.

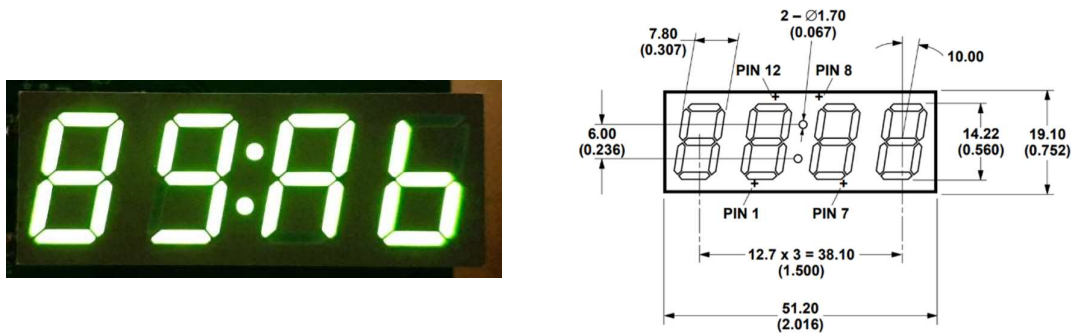
1.4.7 EEPROM

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 herprogrammeren 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 uitkijken 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

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

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

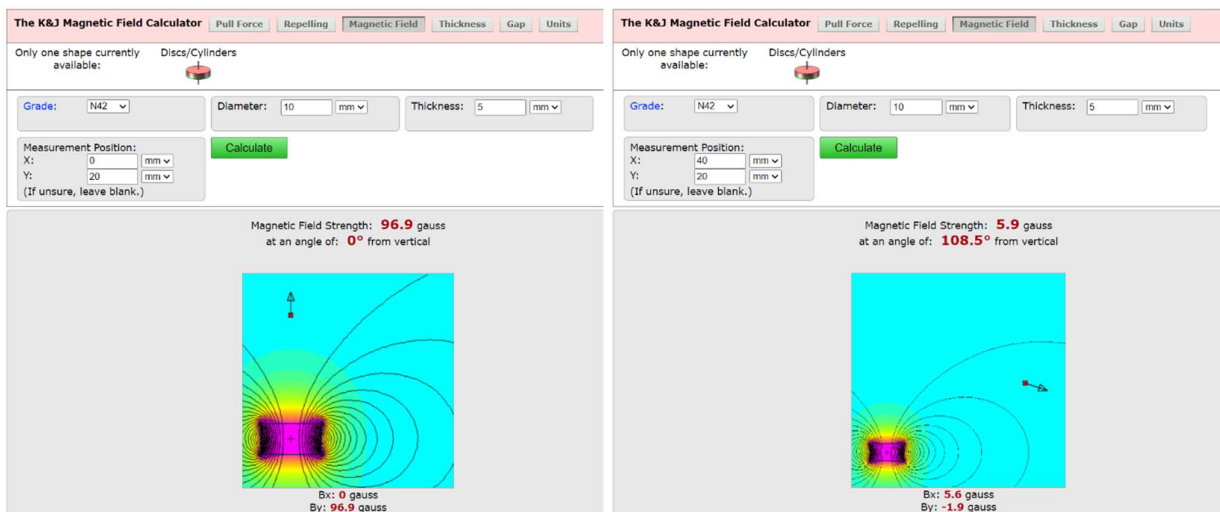
Om de pionnen te detecteren maak ik gebruik van magneet sensoren. Hiermee kan ik on opgemerkt de positie van een pion uitlezen. Doordat ik nog nooit in contact ben gekomen van een magneet sensor heb ik eerst research gedaan achter welke types er bestaan zoals: een reed schakelaar, een hal-sensor, a.d.h.v. 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>

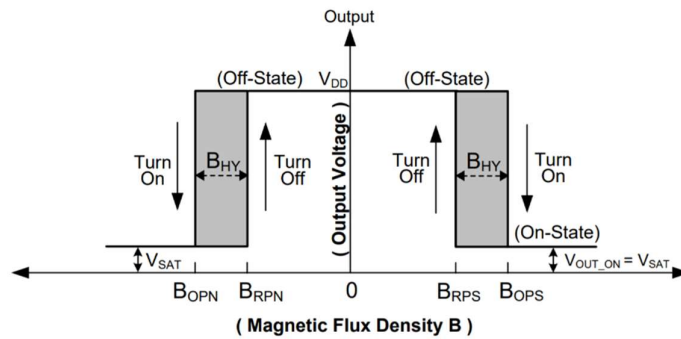
Hiermee gaf ik de volgende waarden ingaf:



Figuur 13, Magnetisch Veld

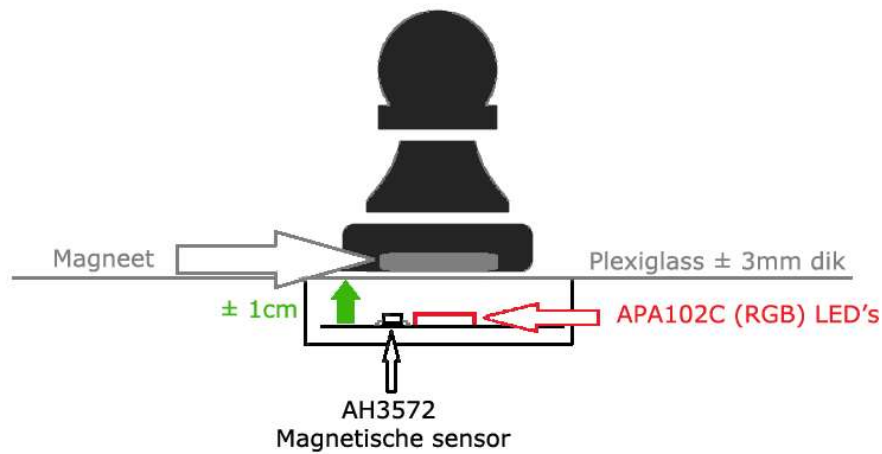
Hierdoor wist ik dus dat een magneet met een diameter van 10mm en dikte van 5mm een magnetische fluxdichtheid van 96,9 gauss op verticale afstand van 2cm. En zoals u ziet aan de rechter berekening zal de magneet geen storing geeft aan een ander vak dat minstens 4 cm verwijderd is van de magneet.

Ik koos voor de AH3572 deze aan de goedkopere kant was 0,205€/stk¹, en schakelde vanaf minimum 20 gauss dat dus geen probleem geeft.



Figuur 14, Schakelpunten

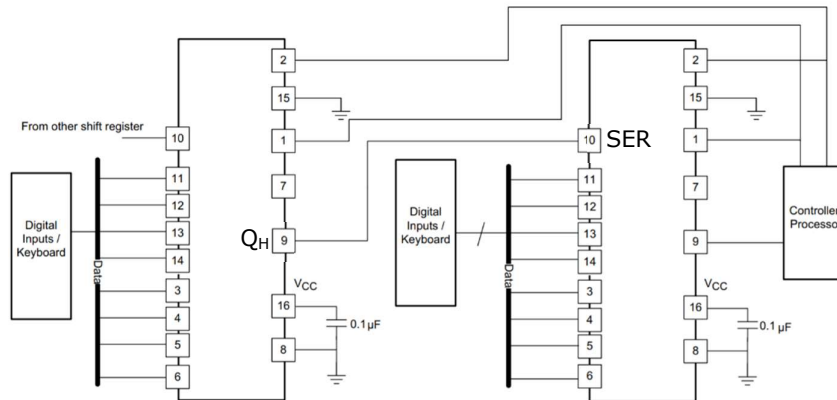
Dit zijn de punten waarop de magneet sensor schakelt. Hier moest ik vooral kijken naar het B_{OPS} punt, deze bedraagt 20 gauss wat dus geen probleem zal geven als de magneet op een afstand van 2cm weg is.



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 schak 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

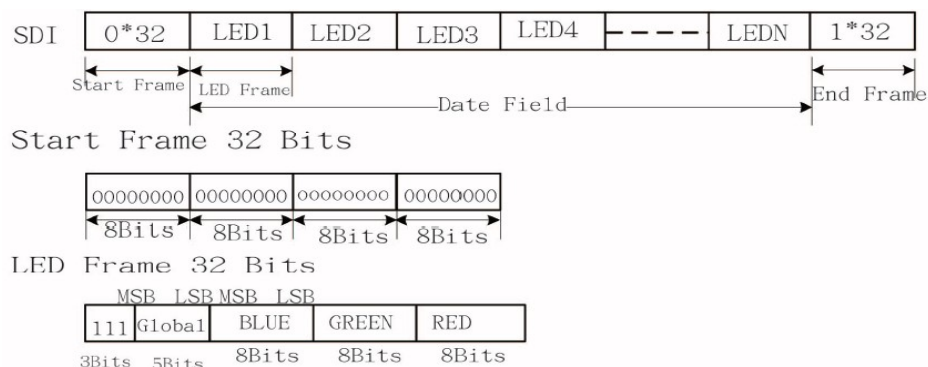
Omdat APA102C's op 5V logica testen ik eerst of de Voltage Translator werkt, deze heb ik getest d.m.v. de pinnen in de software hoog en laag te maken.

Vervolgens heb ik simpelweg de pinnen aan de ingang & de uitgang van de Voltage Translator uit ge meten en kwam ik volgende waardes 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);
    }
}
```

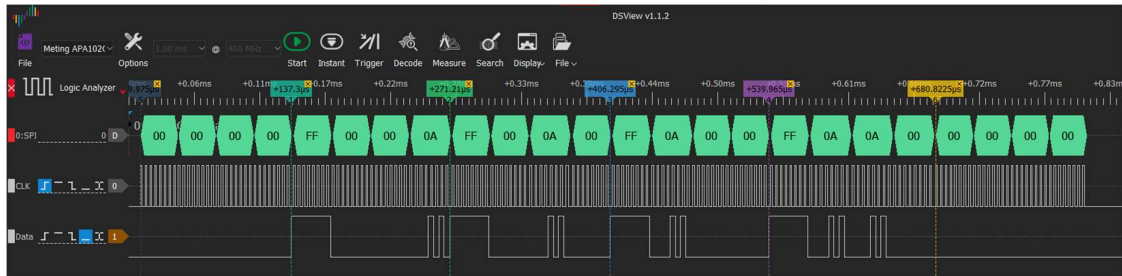
```

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:

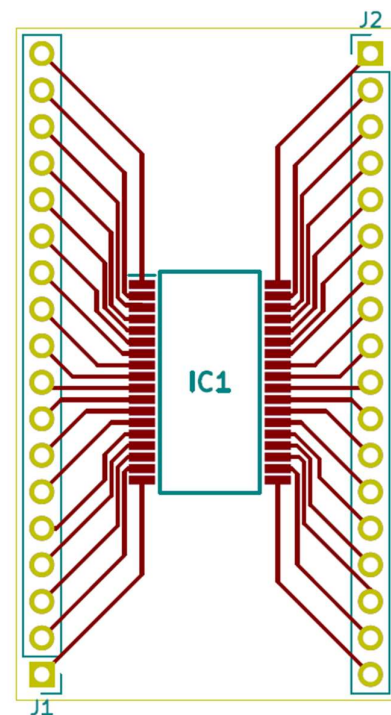
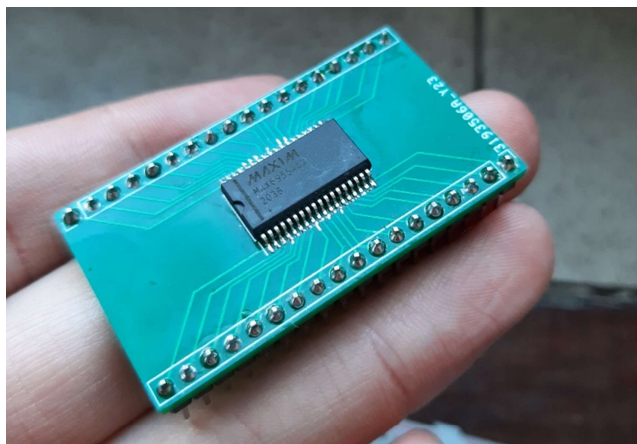


Figuur 17, Meting APA102C

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:

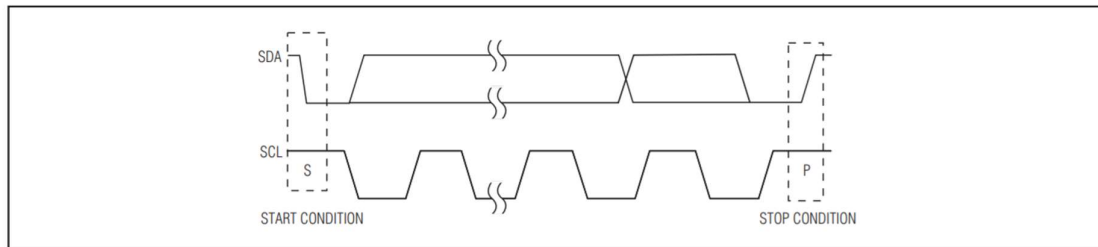


Figure 3. Start and Stop Conditions

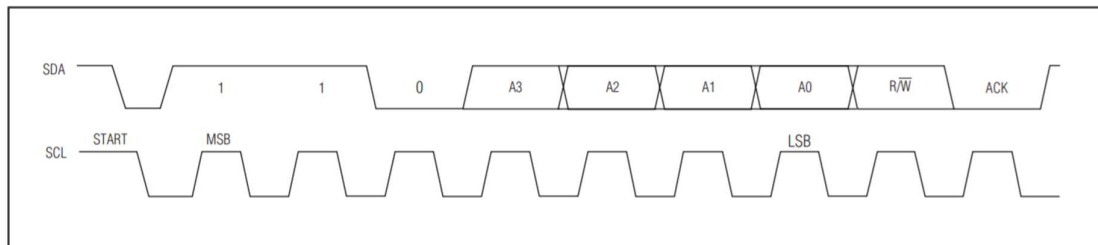


Figure 4. Slave Address

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;

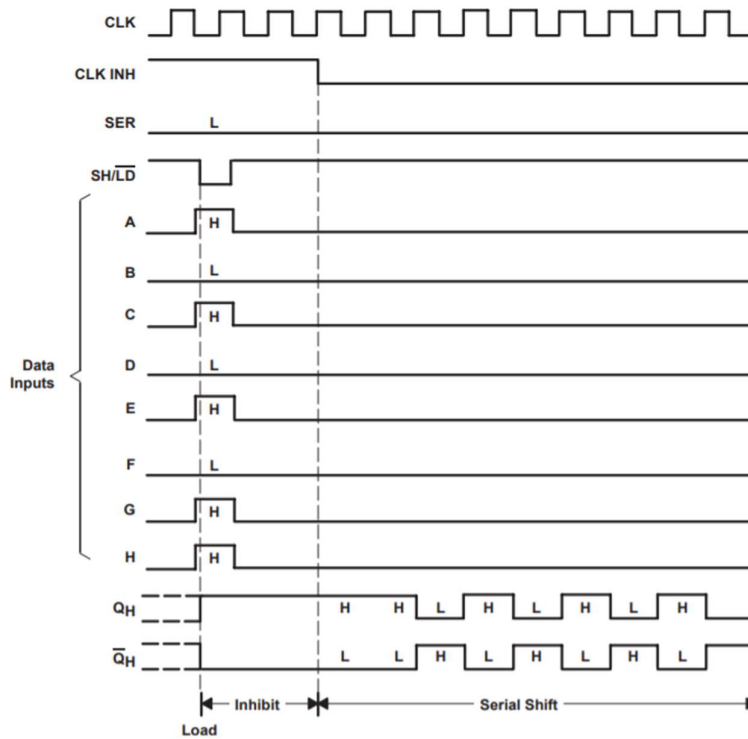
    HAL_I2C_Mem_Write(&hi2c2, MAX6955_ADDR, REG_CONF, 1, buf, 16, HAL_MAX_DELAY);
}

```

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 doorstuurde.

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.

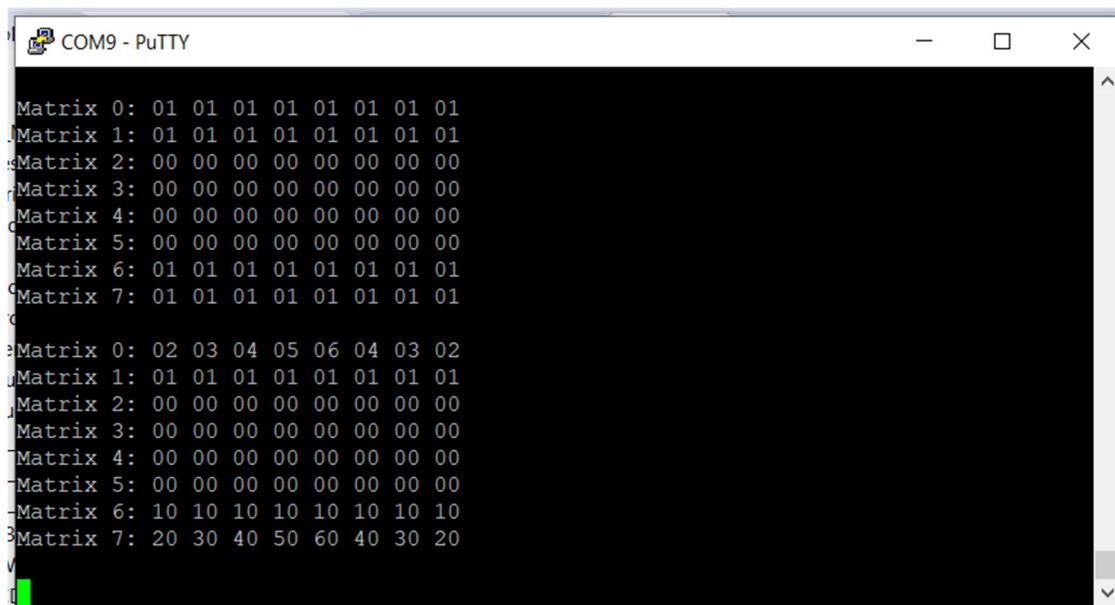
1. De functie "Meten" zal de shift registers uitlezen:

```
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);
        }
    }
}
```

2. De functie "Controlere" zal de uitgelezen data van 1 shift register naar doorsturen via de 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\n", coor[0], coor[1]);
            } else {
                Bstand[j][i] = stand[j][i];
            }
        }
    }
}
```

De code hierboven geeft volgende output:



```
COM9 - PuTTY

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

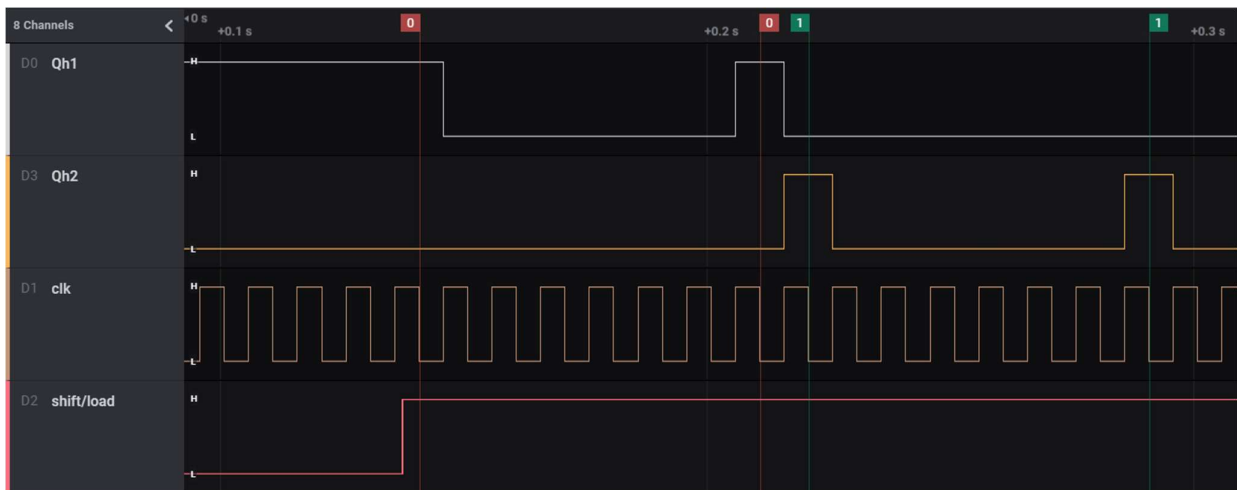
Als er een waarde verandert geeft hij de plaats weer:

```
COM9 - PuTTY
Matrix 0: 00 00 00 00 00 00 00 00
Matrix 1: 01 01 01 01 01 01 01 01
Matrix 2: 00 00 00 00 00 00 00 00
Matrix 3: 00 00 00 00 00 00 00 00
Matrix 4: 00 00 00 00 00 00 00 00
Matrix 5: 00 00 00 00 00 00 00 00
Matrix 6: 01 01 01 01 01 01 01 01
Matrix 7: 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
Matrix 2: 00 00 00 00 00 00 00 00
Matrix 3: 00 00 00 00 00 00 00 00
Matrix 4: 00 00 00 00 00 00 00 00
Matrix 5: 00 00 00 00 00 00 00 00
Matrix 6: 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 roden 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.