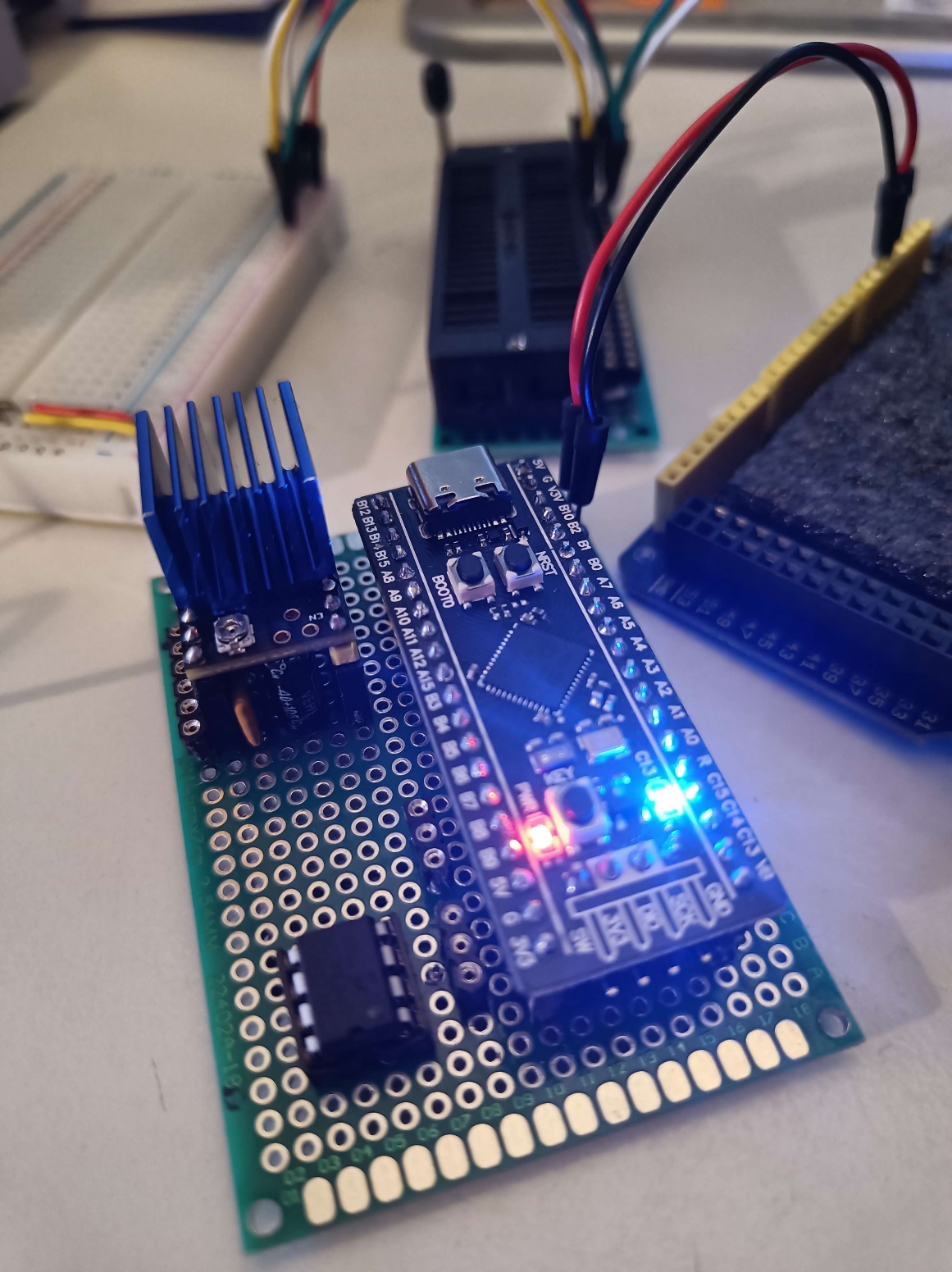
**Robot Arm Project**



**Studentnaam: Marijn Verschuren  
Studentnummer: 510936**

**Klas: PD11  
Vakdocent: Rop Pulles  
  
Versie: V1.0  
Datum: 02-01-2023**

Contents

[1. Design Iteraties: 3](#_Toc124714537)

[2. Huidige Arm\_Computer code: 5](#_Toc124714538)

[3. Huidige Motor\_Control\_Unit code: 7](#_Toc124714539)

[4. Documentatie en Libraries: 11](#_Toc124714540)

[5. Controller code: 12](#_Toc124714541)

# A picture containing text, whiteboard Description automatically generatedDesign Iteraties:

Vanaf het eerste design is het basis idee niet veel veranderd dit idee gaat als volgt:

Op de pc (controller) word een opdracht gemaakt, deze opdracht word vervolgens via de uart doorgestuurd naar de arm computer waarna deze opdracht gechecked en als het nodig is gerepareerd word. Als de opdracht in orde is word deze doorgestuurd naar de correcte motor control unit deze unit heeft in deze versie een sensor (magnetic angle encoder) een stepper motor driver.

Diagram, schematic

Description automatically generated

De volgende disign stap focust zich op de motor control unit en heeft maar één grote verandering en dat is de connectie aan de sensor. De sensor was in de vorige versie alleen met I2C aan de unit aangesloten, in deze versie is de analoge functie van de AS5600 ook aangesloten aan de unit. Om de analoge functie aan te zetten moet de R4 pullup resistor van de AS5600 verwijderd worden. A close-up of a calculator

Description automatically generated with medium confidence

Diagram, schematic

Description automatically generatedDiagram

Description automatically generated with medium confidenceVanaf hier begint een soort tweede versie met de volgende veranderingen:

De arm computer heeft een SPI-bus die shared is over alle motor units met elk een chip select. De enige verandering is dat er een Flag en later een Interupt pin is toegevoegd. De Flag word gebruikt als een verificatie van het ontvangen van de opdracht en de Interupt word gebruikt om de volgende opdracht (in queue) uit te voeren.

En hier is de huidige versie van de motor control unit. Veranderingen:

Rom chip toegevoegd om arm segment vorm (als vector) op te slaan

Bypass capacitors toegevoegd om de power supply voltage consistent te houden

# Huidige Arm\_Computer code:



Er word een UART buffer aangemaakt die door de DMA gevuld word

(de handshake code is nog niet uitgewerkt dus gaat waarschijnlijk veranderen)



Dan word er een handshake aangemaakt en de baud-rate ingesteld (dit word nog niet gebruikt)



De handshake word gereset en de crc op 0xffff zodat deze niet overeen komen.

Text

Description automatically generated

Hier word de handshake uitgevoerd totdat de crc uitkomt. Hier gaan waarschijnlijk wat veranderingen plaatsvinden omdat er nog vanaf een SYNC\_BYTE afgelezen word. Dit vind ik niet zo handig omdat er floating point nummers doorgestuurd worden en de SYNC\_BYTE daar vrij snel tussen zit.



Nadat het ontvangen gelukt is word de handshake aangepast door de motor count toe te voegen. Dan word deze terug gestuurd.

Text

Description automatically generated

Hier word de UART buffer leeg gemaakt en de variables voor: state, instructie en return\_code aangemaakt.

A screenshot of a computer

Description automatically generated with medium confidence

In de mainloop word de instructie opgehaald uit de buffer en gecheckt of deze klopt.



Als deze klopt word een ok signaal terug gestuurd.

A screenshot of a computer

Description automatically generated with medium confidence

Dan word de instructie doorgestuurd naar de motor control unit.

# Huidige Motor\_Control\_Unit code:

Text

Description automatically generated

De sensor word ingesteld.

De stepper driver word uitgezet.



De settings worden geschreven naar de sensor.

A screen shot of a computer

Description automatically generated with low confidence

De state word gereset

A screenshot of a computer

Description automatically generated with medium confidence

De hoek van de motor word afgelezen via I2C (dit word later gedaan via ADC met DMA naar raw\_angle)



Timers 2 en 5 worden aangezet

A screenshot of a computer screen

Description automatically generated with medium confidence

De DMA devices worden aangezet.

A screenshot of a computer

Description automatically generated with medium confidence

De motor driver word aangezet, de timer counter word gereset (start simulatie) en de simulatie timer word aangezet timer 10. Deze interrupt word 100 keer per seconden uitgevoerd.

A screenshot of a computer

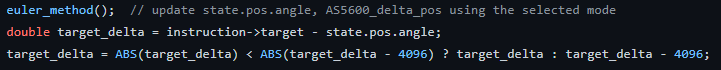
Description automatically generated with medium confidence

In de mainloop worden de stappen gezet. De direction en delay worden vanuit de interrupt aangepast.

A screenshot of a computer

Description automatically generated with medium confidence

Hier begint de interrupt call. Er word gecheckt of deze call komt vanuit de correcte timer en of er een opdracht is.



Hier word de afstand vanaf de target uitgerekend (hier moet nog wat werk gedaan worden). Op de “euler\_method” kom ik later terug.

Text

Description automatically generated

Hier word de stap snelheid uitgerekend (ook hier moet nog aan gewerkt worden)

Text

Description automatically generated

Hier word de volgende instructie geladen. De lock word gereset met de instruction interrupt pin. (hier moet nog gewerkt worden aan het laden van de volgende instructie enz…)

A screenshot of a computer

Description automatically generated with medium confidence

Dit is de SPI callback waarin de instructie word toegevoegd aan de queue

Text

Description automatically generated

De Flag pin word zo gezet dat de arm computer kan zien of de instructie goed is aangekomen (LOW = success).

Text

Description automatically generated

Dit is de exti callback code. Als de NSS (SPI non chip/slave select) de interrupt triggered word de Flag op HIGH gezet zodat de arm computer kan wachten totdat deze gereset is zonder dat er een risico is dat de arm computer de motor control unit inhaalt (race condition). Als de instruction interrupt pin de interrupt triggered word de instructie lock weggehaald.

Graphical user interface, text

Description automatically generated

In de euler method worden de errors die bij analoge signalen komen kijken zo goed mogelijk weggewerkt en de velocity uitgerekend (hier moet nog aan gewerkt worden). Hier zie je ook gebruik van register variables zodat (hopelijk) de compiler de variabelen niet in de ram opslaat wat relatief traag is tegenover registers. (nu weet ik niet wat voor registers op de CORTEX M4 zitten dus dit is niet gegarandeerd)

# Documentatie en Libraries:

Voledige repository: <https://github.com/MarijnVerschuren/Robotic_Arm>

AS5600 library repository: <https://github.com/MarijnVerschuren/STM32F4_AS5600_library>

24LC512IP library: <https://github.com/MarijnVerschuren/EEPROM_Programmer/tree/main/MCU/lib/24LC512IP> Deze moet nog afgemaakt en geisoleerd worden.

Video documentatie (in orde):

<https://youtube.com/shorts/iROxng_Yvek>

<https://youtube.com/shorts/G3Ni2nRQHQA>

<https://youtu.be/2IKLwBE1TLw>

<https://youtu.be/rBEvl1Z75_w>

<https://youtube.com/shorts/V_E33Jzvdlo>

# Controller code:

A screenshot of a computer screen

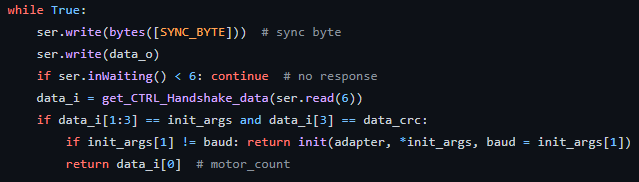
Description automatically generated with medium confidence

Hier word in de init functie de serial port geopend.

Text

Description automatically generated

Hier worden de in- en output buffer gereset. Ook word de handshake en de crc aangemaakt.



In de initloop word de handshake gestuurd totdat er een terug komt.



In de run function word de init functe gecalled om de motor count te krijgen.

Text

Description automatically generated

Onder de run loop worden wat inputs gevraagd om een instructie te maken. (dit gaat natuurlijk veranderd worden)

Text

Description automatically generated

De instructie word gestuurd totdat er een ok return code terug komt