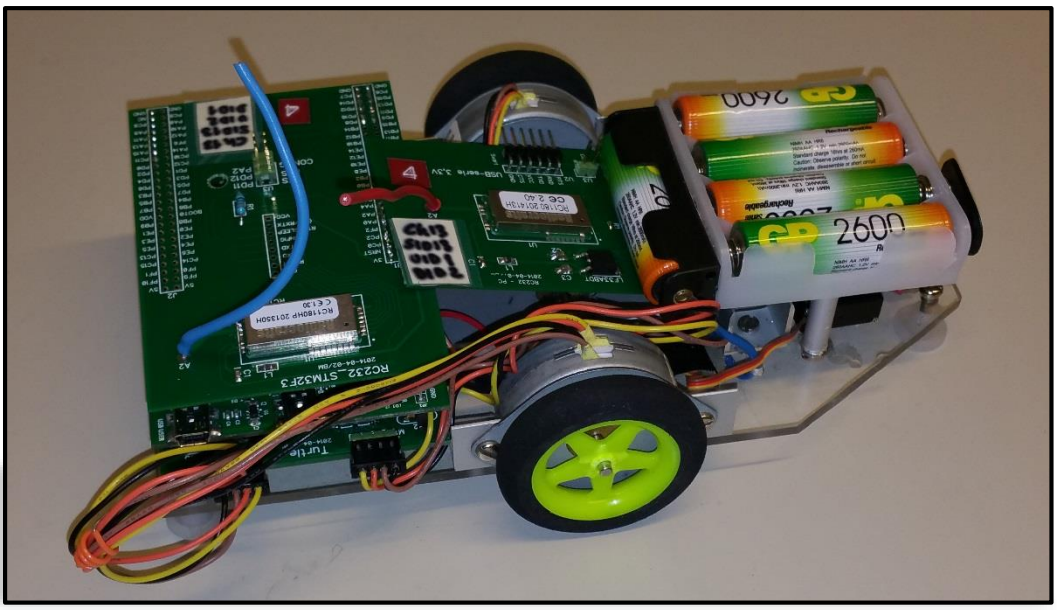
Turtle robot

Projektrapport för ett utvecklat inbyggd system



Tanvir Saif Ahmed

2016-03-18

Inbyggda system, 7.5 HP
Examinator: Bengt Molin

KTH
Skolan för informantions- och kommunikationsteknik
164 40 Kista, Sverige

Sammanfattning

Turtle Graphics är en term som används för att beskriva programmering som utförs av en robot genom att rita linjer eller mönster.

I denna rapport beskrivs uppbyggnaden av roboten Turtle gällande dess hårdvaruarkitektur och mjukvaruarkitektur.

Hårdvaran av roboten består av två stegmotorer, drivkrets, servomotor samt en kommunikationskrets, som är alla sammankopplade till mikrokontrollerkortet STM32F303VC. Dessa elektronikenheter har testats och beskrivs mer detaljerat i kopplingsscheman.

Mjukvaran av roboten grundar på att roboten kan ta emot kommandon trådlöst och utföra grundinstruktioner som t.ex. att åka framåt, rotera höger eller vänster samt lyfta upp eller ned pennan. Den har även möjlighet att lära sig nya ord som definieras av användaren, vilket innebär att roboten kan rita komplexa mönster, som t.ex. fyrkant, triangel, osv. Ett testdriven program har även utvecklats, för att säkerhetsställa att programmodulerna uppfyller de kraven som ställts för roboten.

Nyckelord

Turtle Graphics, mikrokontroller, mönster, linjer, programmering, moduler, testdriven, hårdvara, mjukvara, motor, trådlös, kommandon.

Förord

Jag vill tacka examinatorn Bengt Molin, för att ha gett oss tillgång till material och robot för att utföra denna uppgift.

Innehåll

1 Inledning

I detta kapitel, presenteras en introduktion till roboten samt målet med uppgiften.

1.1 Introduktion

Roboten är skapad för att ge den implementationen att kunna kommunicera trådlöst och utföra några instruktioner, som t.ex. att åka framåt, rotera höger eller vänster, samt lyfta upp eller ned en penna.

För att utveckla roboten, gavs det några seminarier där hårdvaran samt mjukvaran diskuterades. Hårdvaran diskuterades genom att ge varandra idéer på hur kopplingsschemat kan se ut för enheterna som bygger upp roboten samt vilka anslutningar på STM32F3 Discovery mikrokontrollerkortet, som ska konfigureras till ingångar eller utgångar. Detta gjordes för att kunna få en uppfattning av hur hårdvaruenheterna ska kommunicera med varandra för att klara av de instruktionerna som roboten ska utföra. Kopplingsschemat för hårdvaran gavs sedan ut, där det angavs hur enheterna ska kopplas samt vilka anslutningar som ska sättas till utgångar och ingångar på STM32F3 Discovery mikrokontrollerkortet.

Mjukvaran var höjdpunkten för roboten och diskuterades utförligt under seminariet, där idéer gavs angående hur mjukvaruarkitekturen ska se ut, både hierarkiskt och lagermässigt, hur programmodulerna ska utvecklas samt hur dessa programmoduler ska testas för att kunna uppfylla kraven.

1.2 Målsättning

Målet för uppgiften, är att designa och utveckla programmoduler för att kunna styra roboten. Följande delmål ska realiseras:

- Roboten ska kunna ta emot instruktioner trådlöst via ett terminalfönster, där användaren matar in instruktionen som kommandon.
- Roboten ska kunna utföra grundinstruktioner som t.ex. åka framåt, rotera höger eller vänster, samt lyfta upp eller ned pennan.
- Roboten ska kunna lära sig ett nytt ord och spara detta ord i en databas. Ordet ska kunna användas som ett kommando av användaren.
    - Ord som skapas för roboten, ska kunna utföra flera instruktioner ett antal gånger för att t.ex. rita en figur.
- Roboten ska kunna ge feedback samt felmeddelanden till användaren om kommandon som den inte kan tolka.

1.3 Avgränsningar

Robotens avgränsningar:

- Då radiokretsen som tillhandhålls i hårdvaran, kräver en viss tidsfördröjning för att nästa sträng ska skickas, kommer kommunikationen med roboten att vara långsam.

- Stegmotorerna som tillhandhålls i hårdvaran, drar väldigt mycket ström, vilket innebär att batterierna som ska spänningsförsörja roboten, kommer att behöva laddas kontinuerligt.

Uppgiftens avgränsningar:

- En tidsbudget som omfattar schemalagda laborationspass samt egen tid.

2 Teori och bakgrund

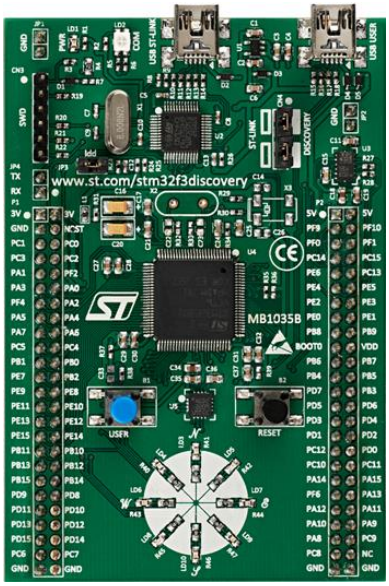
I detta kapitel, presenteras teorin för de delarna som används i roboten i form av mjukvara och hårdvara.

2.1 Hårdvara

2.1.1 Mikrokontrollerkort

Det existerar olika typer av mikrokontrollerkort ute i marknaden. Mikrokontrollerkort är en liten dator som består av en CPU, RAM och ROM. Dessa är kärnan för ett inbyggt system och används främst till för att styra yttre enheter samt finns integrerat i dagens elektroniksystem, som t.ex. tvättmaskin, bilar, mobiltelefoner, osv.

Mikrokontrollerkortet som användes för att styra hela roboten är STM32F303VC Discovery (se figur 1), tillverkat av STMicroelectronics. Denna består av 100 anslutningar, där anslutningarna kan konfigureras som utgång eller ingång. Den bygger på ARM Cortex-M4 RISC kärnan, har totalt 256 kB minne, innehåller en flyttalsprocessor på en frekvens upp till 72 MHz, samt arbetar på en spänning med 5 V. Övriga specifikationer är fyra 12 bitars A/D omvandlare, två UART, 16- och 32-bitars timers, USB anslutning, etc [1].



Figur 1: Figuren visar mikrokontrollerkortet STM32F303VC Discovery [21].

2.1.2 Stegmotor

Stegmotor är en elektrisk motor som omvandlar elektriska pulser till diskreta motorkraft. Pulserna skickas till stegmotorn i en viss sekvens och leder till att stegmotorn roterar ett visst steg. Rotationen av motorn är baserad på antal pulser, vilket gör det lättare att kontrollera längden av rotationen genom att skicka en sekvens av digitala pulser [2].

Stegmotorer existerar i två varianter, unipolära och bipolära. Skillnaden mellan dessa är uppbyggnaden samt vilka krav som ställs för att använda ett av dessa stegmotorn. En unipolär stegmotor har en simplare konstruktion och kräver en enkel drivkrets, medan en bipolär har en mer komplex konstruktion och kräver oftast mer avancerad drivkrets [3].

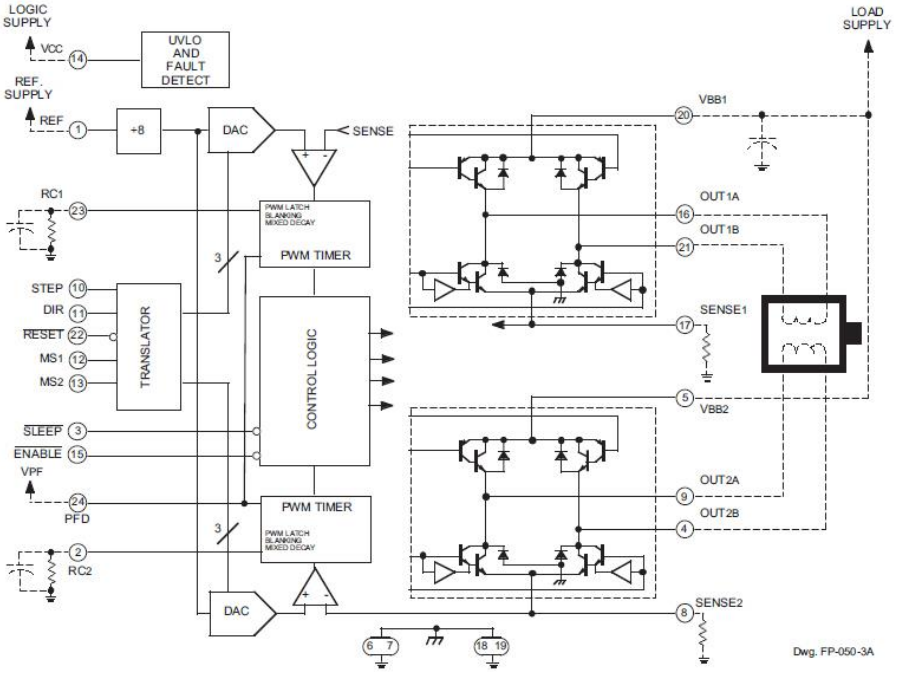
Figur 2: Figuren visar stegmotorn 42PM100S01-02B [22].

Fördelarna med en stegmotor är att dessa har ett inbyggt öppen slingsystem, som innebär att navigeringen kring stegmotorns position blir mer optimalare. Detta gör att felplaceringar kan detekteras och åtgärdas snabbare [2]. Nackdelen med en stegmotor är att dessa drar väldigt mycket ström och är oftast dyra.

Stegmotorn som användes i roboten för att styra hjulen är den bipolära stegmotorn 42PM100S01-02B (se figur 2), tillverkat av Moons. Den har en stegvinkel på 3.6° och arbetar på en spänning med 3 V och kräver en ström på 0,6 A [4].

2.1.3 Drivkrets

Drivkretsar används oftast till för att driva enheter som kräver en hög ström. En typisk drivkrets som används i de flesta elektroniktillämpningar (t.ex. motorer och reläer) är ULN2003A, som är en integrerad krets bestående av sju stycken Darlington-transistorpar. För varje par, så finns det en ström och spänningskapacitet på 500 mA respektive 50 V. Dessa drivkretsar kan även parallellkopplas för att uppnå en högre ström [5].



Figur 3: Figuren visar blockdiagrammet för drivkretsen A3967 [6].

Drivkretsen som användes i roboten för att driva stegmotorerna är A3967 (se figur 3), tillverkat av Allegro. Denna har en ström och spänningskapacitet på ±750 mA respektive 30 V och används mest för att driva bipolära stegmotorer. Regulatorn inbyggt i drivkretsen ger möjligheten till att bestämma ifall drivkretsen ska arbeta i en långsam, snabb eller blandad läge. Detta gör att störningar av motorn kan minskas samt öka noggrannheten av de stegen som stegmotorn utför [6].

Drivkretsen ger möjligheten till stegmotorn att utföra steg i fyra olika lägen, som är: fullsteg, halvsteg, kvartsteg och mikrosteg.

När drivkretsen ställs in så att stegmotorn stegar i fullsteg eller halvsteg, innebär det att det krävs 100 respektive 200 pulser för att ett varv ska utföras. För steglägena kvartsteg eller mikrosteg, krävs det 400 respektive 800 pulser för att ett varv ska utföras. Fördelen med dessa steglägen som kräver mer pulser, är att öka noggrannheten av positionen samt vinkeln när stegmotorn ska rotera en viss grad [2].

2.1.4 Servomotor

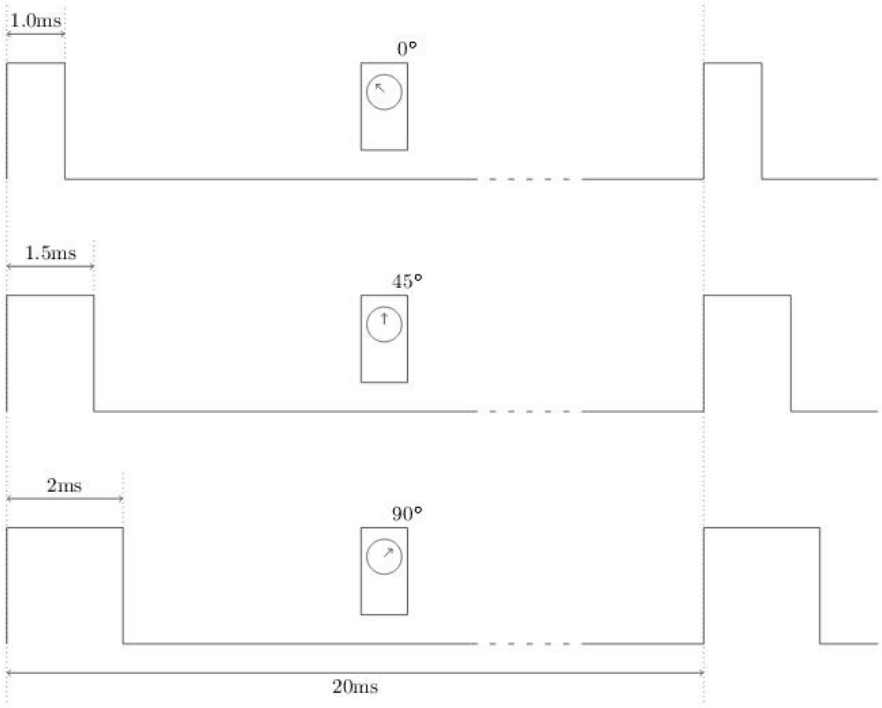
En servo motor (se figur 4) är en typ av elektrisk motor, som består av två DC motorer, potentiometer, en integrerad krets samt ett skaft. Denna typ av motor används mest i bilar, flygplan, robotar, etc. och är effektiva när det gäller hastighet och vridmoment.

Figur 4: Figuren visar en servomotor [23].

Genom att skicka kodade signaler från servomotorn, så kan skaften av servomotorn ändras till en viss vinkelposition och behåller dess vinkelposition tills en sekvens av kodade signaler skickas. Om en ny sekvens av kodade signaler skickas, så kommer skaften att ändra sin vinkelposition beroende på hur signalerna är kodade [7].

Servomotorn som användes i roboten för att styra ett metallskaft som ska lyfta upp en penna, består av tre anslutningar, vilket är 5V, GND samt kontrollsignal. Genom att använda timer, så kan denna servo motor kontrolleras med hjälp av pulser under en maximal periodtid av 20 ms [8].

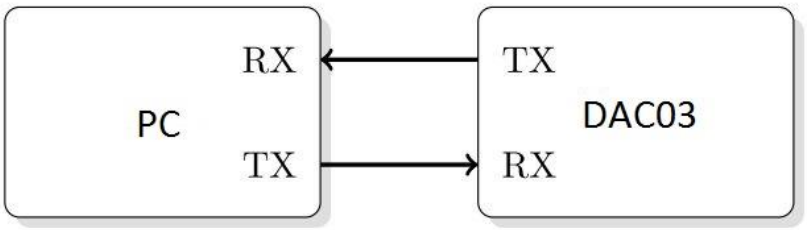
Rotationen av servomotorn i grader baserat på en given pulsbredd kan åskådliggöras i följande figur (se figur 5). Denna visar att med en pulsbredd på 1 ms, så kommer rotationen av servomotorn att vara 0 grader. Om pulsbredden ökar med hälften av 1 ms, så kommer servomotorn att rotera 45 grader och om pulsbredden ökar dubbelt så mycket av 1 ms, så kommer servomotorn att rotera 90 grader [9].



Figur 5: Figuren visar hur servomotorns rotation beror av en given pulsbredd [9].

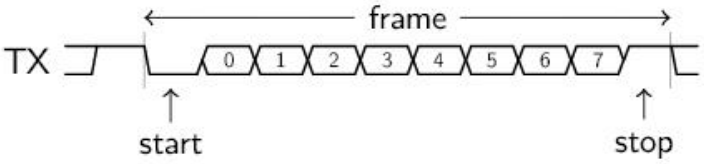
2.1.5 UART

UART, förkortning av *Universal Asynchronous Receiver/Transmitter* är en hårdvaruenhet som används för att kommunicera mellan två noder, nämligen en källa och en destination (se figur 6). Källan agerar som en sändare och sänder data i form av bytes, som sedan görs om till bitar. Dessa bitar omvandlas sedan till bytes vid destinationen för att informationen ska kunna tolkas [9].



Figur 6: Figuren visar kommunikationen mellan noderna källa (PC) och destination (FAC03) via UART [24].

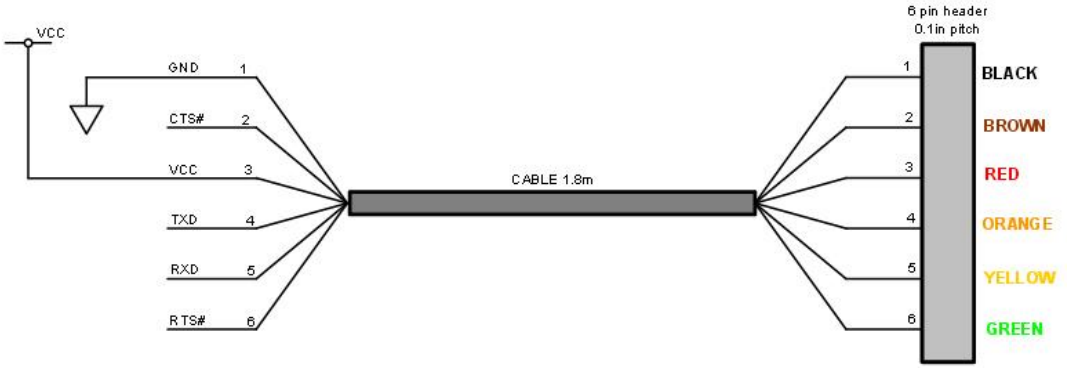
Denna typ av kommunikation kallas för Asynchronous eftersom varje byte som skickas i form av ram, består av 10 bitar (se figur 7). Den första biten representerar en start bit som är låg, följd av åtta bitar som representerar byte av ett tecken. Dessa bitar kan antingen vara låga eller höga, beroende på vilket tecken som har skickats. Ramen avslutas med en stopp bit, som är låg [9]. För att upptäcka fel i data som skickas, så läggs en paritetsbit oftast i slutet av ramen.



Figur 7: Figuren visar ramen för ett tecken, som sänds via UART [9].

UART som användes för att kommunicera med roboten genom en USB seriell kabel trådlöst, hade en baud rate inställt på 19200 Bd, som är en mått på hur många gånger en signal ändras per sekund.

2.1.6 USB seriell konverterare



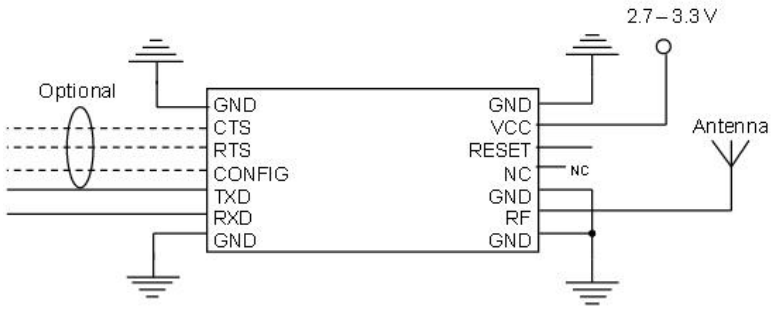
Figur 8: Figuren visar FTDI USB TTL Serial samt dess anslutningar [10].

För att skicka data seriellt till destinationen, så krävdes en USB till seriell adapter. USB till seriell adapter som användes i roboten för att kommunicera seriellt mellan en UART gränsnitt och en USB anslutning är en FTDI USB TTL Serial (se figur 8). Denna består av anslutningarna TX (Transmit), RX (Receive), RTS (Request To Send), CTS (Clear To Send), 5V samt GND [10].

2.1.7 Kommunikationskrets

Radiomoduler är små elektronikenheter som kan användas för att sända och ta emot information i form av radiovågor. Dessa typer av enheter används mest i elektroniksystem i syfte med att kommunicera trådlöst mellan två noder genom att skicka och ta emot information som bärs av radiovågor [11].

Kommunikationskretsen som används i roboten för att skicka och ta emot information, är RC1180HP-RC232 (se figur 9), tillverkat av Radiocrafts. Denna består av en inbyggt RC232 protokoll, som används för kommunikationen, högeffektiv RF transceiver, intern spänningsregulator samt en effektförstärkare. Arbetsspänningen för kommunikationskretsen ligger på 3.3 V och datahastigheten ligger på 76,8 kbit/s [12].



Figur 9: Figuren visar kommunikationskretsen RC1180HP-RC232 och dess anslutningar [12].

En kommunikationskontrollerare finns även inbyggt i kommunikationskretsen, som sköter hanteringen av protokollen för att skicka radio paket, UART gränssnittet samt RF transceivern. Information som skickas från källan i form av datapaket, tas emot av RXD anslutningen. Denna datapaket buffras sedan i kommunikationskontrolleraren, där själva paketet ges en inledning, start av ram begränsning, adressinformation samt en kontrollsumma för felupptäckt. Därefter så skickas datapaketet vidare till RF transceivern [12].

2.1.8 Oscilloskop

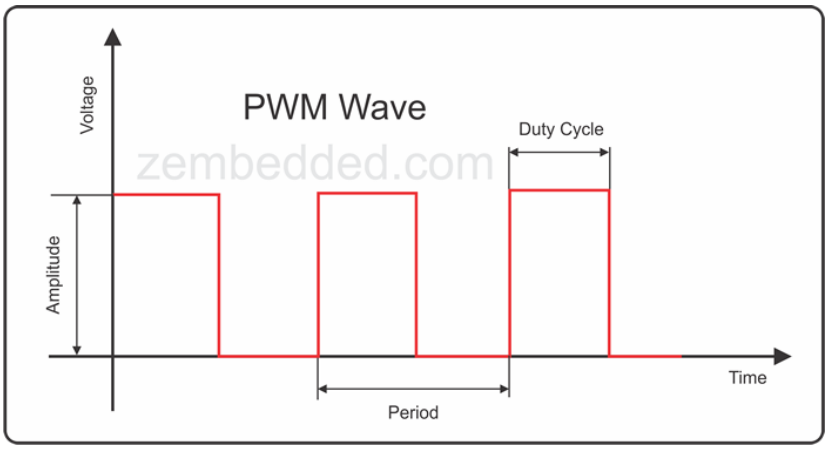
För att kunna säkerhetsställa att hårdvaran för roboten fungerar felfritt och inte orsakar problem, som kan leda till att de yttre enheterna eller mikrokontrollerkortet blir ur funktion, krävs ett oscilloskop. Oscilloskop är ett mätinstrument som används för att kontrollera kretsar samt mäta spänningsnivån [13].

Figur 10: Figuren visar oscilloskopet DSOX2014 [25].

Oscilloskopet som användes för att kontrollera robotens hårdvara, är DSOX2014, tillverkat av Agilent (se figur 10).

2.1.9 Pulsbreddsmodulering (PWM)

PWM, förkortning av *Pulse Width Modulation* är en metod som används för att skapa en varierbar spänning som kan slås på och av mycket snabbt genom att reglera den (se figur 11). En PWM signal har en period som representeras som tiden och en arbetscykel (duty cycle) som är själva pulsbredden [14].



Figur 11: Figuren visar en illustration av PWM [26].

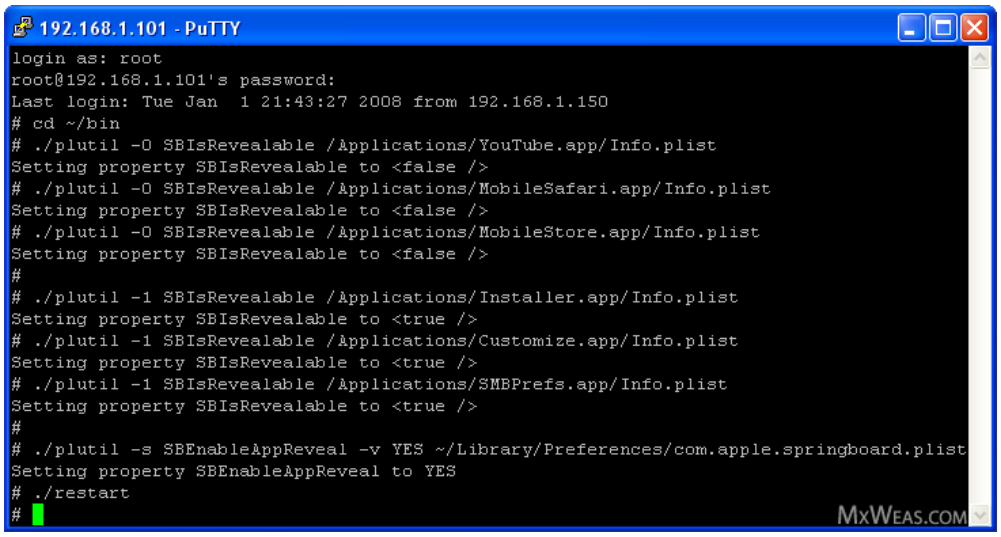
PWM signalen som skapas, representeras som en fyrkantsvåg, vilket kan antingen vara låg eller hög. Genom att skicka ett antal pulser, så kommer PWM signalen att variera mellan spänningsnivåerna 5 V eller 0 V. Signalen är aktiv då spänningen är 5 V och signalen är inaktiv då spänningen är 0 V. PWM används oftast i elektroniksystem där spänningen behöver regleras [14].

PWM användes i roboten för att signalera servomotorn om att lyfta upp och ned pennan med hjälp av en periodtid och två arbetscykler.

2.2 Mjukvara

2.2.1 Terminalprogram

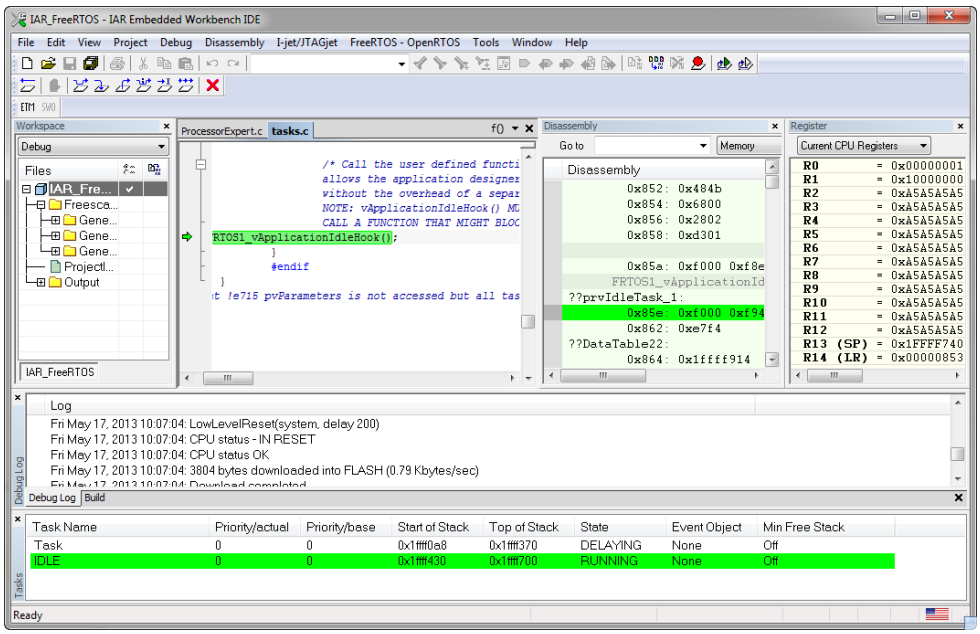
Terminalprogram används oftast för att starta igång program, mata in kommandon, kommunicera med annan dator, osv. De är även användbara för att kunna kommunicera med ett system som kan tolka kommandon och utföra något utifrån det, t.ex. styra en yttre enhet via mikrokontrollerkortet.



Figur 12: Figuren visar terminalprogrammet PuTTY [27].

Terminalprogrammet som användes för att skicka kommandon samt ta emot feedback från roboten, är PuTTY (se figur 12). Detta är ett klientprogram som används främst för kommunikation via olika typer av protokoll som t.ex. SSH, Telnet, etc. PuTTY ger möjligheten att kommunicera via en terminal, där det är möjligt att sända och ta emot information i form av ASCII tecken. Kommunikationen med en annan enhet via terminalen sker oftast genom en USB anslutning [15].

2.2.2 Utvecklingsmiljö



Figur 13: Figuren visar utvecklingsmiljön IAR Embedded Workbench [28].

Utvecklingsmiljö är datorprogram som allmänt består av en textredigerare, kompilator samt en debugger. Dessa används främst för att utveckla både mjukvaruprogram och inbyggda system.

Utvecklingsmiljön som användes för att utveckla roboten gällande mjukvaran, är IAR Embedded Workbench (se figur 13). Denna utvecklingsmiljö består av en textredigerare, C/C++ kompilator samt en debugger verktyg. IAR används främst för att utveckla inbyggda system och innehåller funktioner som gör det möjligt att söka och korrigera fel i program på optimala sätt [16].

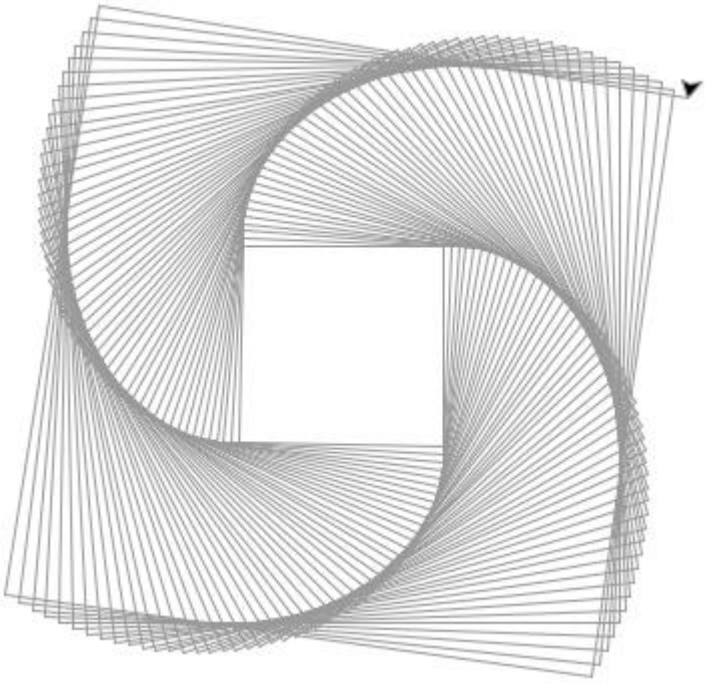
2.2.3 Programmeringsspråk

Programmeringsspråk är språk som används för att skapa datorprogram. Datorprogram kan vara textbaserade eller grafiska. Det finns olika typer av programmeringsspråk som används för att utveckla mjukvara, där de populära är C, C++, Java, C#, Python, etc.

Programmeringsspråket som användes för att utveckla roboten, är det imperativa programspråket C. C är en av de mest populära samt inflytelserika programspråket som skapades av Dennis Ritchie år 1972 på Bell Laboratories. Språk som har fått sin inspiration från C, är t.ex. C++ och Java [17].

2.2.4 Turtle Graphics

Turtle Graphics (se figur 14) är en term som förknippas med vektor programmering. Principen grundar på att en sköldpadda byggt som en robot, använder sig av en markör och ritar ut ett mönster på ett plan. Turtle Graphics var en del av programmeringsspråket Logo, som utvecklades av Wally Feurzig and Seymour Papert år 1966 och användes till syfte på att ge en introduktion till programmering [18].



Figur 14: Figuren visar ett mönster som har ritats upp med Turtle Graphics [29].

3 Metod

I detta kapitel, redogörs för hur roboten har utvecklats samt de tester som har gjorts i form av mjukvara och hårdvara.

3.1 Hårdvara

3.1.1 Testning av elektronikenheterna

3.1.1.1 Testning av UART

För att säkerhetsställa att information kan sändas från terminalen, testades UART:en genom att analysera informationen i oscilloskopet. FTDI kabeln kopplades till STM32F3 Discovery mikrokontrollerkortet och PD9 anslutningen av mikrokontrollerkortet mättes med hjälp av en prob i oscilloskopet för att få en bild av hur ett tecken ser ut i bitar. Följande tabell (se tabell 1) visar de anslutningarna av FTDI kabeln som ska vara kopplade till STM32F3 Discovery mikrokontrollerkortet:

FTDI USB TTL Serial	STM32F3 Discovery
RXD	PD8
TXD	PD9
GND	GND

Tabell 1: Tabellen visar anslutningarna mellan FTDI USB TTL Serial och STM32 Discovery [8].

Verktygsprogrammet STM32CubeMX som genererar en konfigurering av anslutningar på STM32 Discovery mikrokontrollerkortet, användes för att konfigurera anslutningarna PD8 och PD9 till USART3 RX respektive USART3 TX.

Ett testdriven program användes för att sända tecken via terminalprogrammet PuTTY, som sedan togs emot av STM32F3 Discovery mikrokontrollerkortet. Följande figur (se figur 15) visar hur ett tecken representeras av ett ram, som består av en start bit, följd av åtta databitar och avslutas med en stopp bit.

Figur 15: Figuren visar ramen för tecknet 'Z' i form av spänningsnivåer i oscilloskopet.

3.1.1.2 Testning av stegmotorn

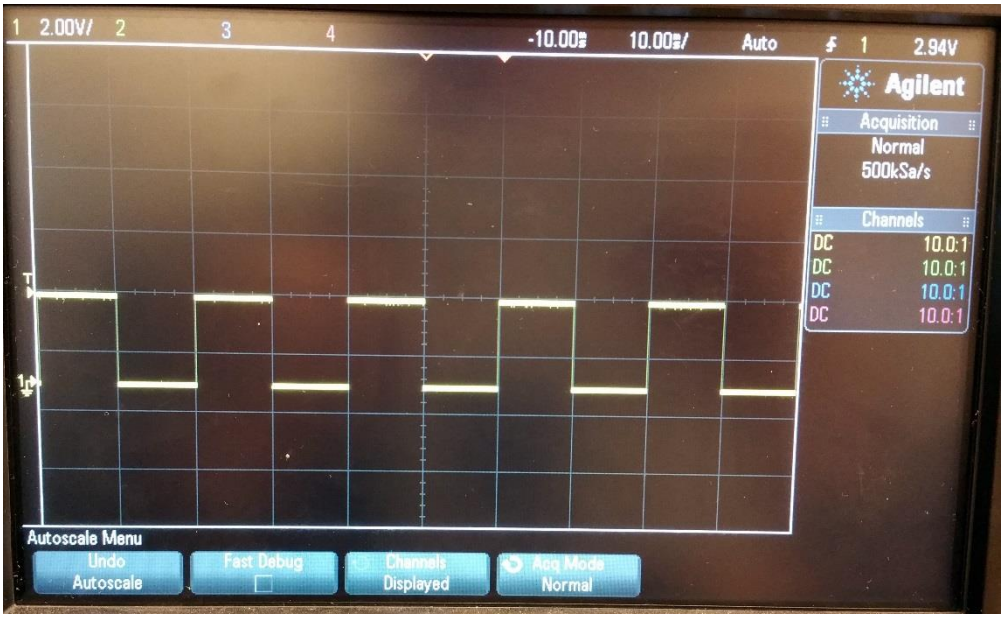
För att kontrollera ifall en sekvens av pulser genereras till stegmotorerna, testades GPIO anslutningarna av mikrokontrollerkortet som styrde stegmotorerna genom att mäta en av dessa GPIO anslutningar i oscilloskopet. STM32F3 Discovery mikronkontrollerkortet kopplades till stegmotorkortet och PC6 anslutningen av mikrokontrollerkortet mättes med hjälp av en prob i oscilloskopet för att få en bild av hur pulserna ser ut som drivkretsen ska generera till stegmotorerna. Följande tabell (se tabell 2) visar de anslutningarna av drivkretsen på stegmotorkortet som ska vara kopplade till STM32F3 Discovery mikrokontrollerkortet:

A3967	STM32F3 Discovery
STEP1	PC6 / vänster
DIR1	PC7 / vänster
RESET	NC (ingen anslutning)
MS1	PC8
MS2	PC9
/SLEEP	NC (ingen anslutning)
ENABLE	PC12 (aktivering av stegmotorerna)
STEP2	PC10 / höger
DIR2	PC11 / höger

Tabell 2: Tabellen visar anslutningarna mellan A3967 och STM32 Discovery [8].

Verktygsprogrammet STM32CubeMX som genererar en konfigurering av anslutningar på STM32 Discovery mikrokontrollerkortet, användes för att konfigurera anslutningarna PC6, PC7, PC8, PC9, P10, P11 samt P12 till GPIO Output.

En testdriven program användes för att skicka ett antal pulser till GPIO anslutningen, som var kopplad till PC6. Pulserna hade en fördröjning på 10 ms, som går att åskådliggöra på följande figur (se figur 16).



Figur 16: Figuren visar en sekvens av pulser som har generats för stegmotorn i oscilloskopet.

3.1.1.3 Testning av servomotorn

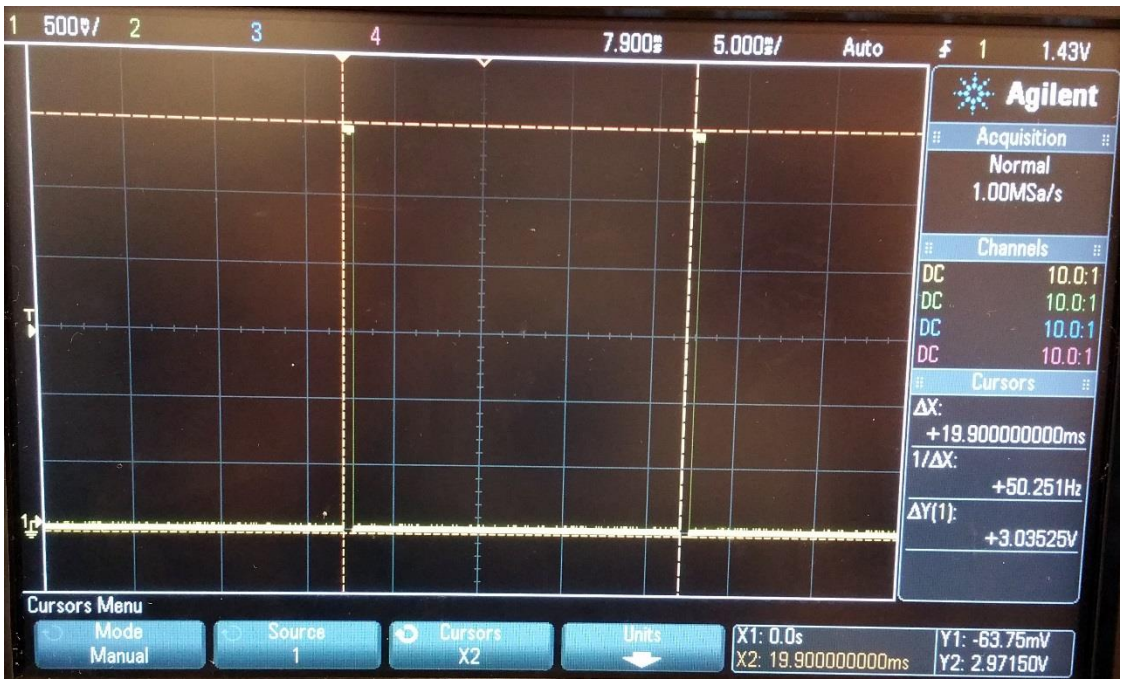
För att kontrollera att rätt puls generas till servomotorn, testades TIMER anslutningen av mikrokontrollerkortet som styrde servomotorn genom att mäta en TIM anslutning i oscilloskopet. Då det finns en risk att servomotorn kan förstöras, behövde rätt typ av pulslängd skickas till servomotorn [8]. Servomotorn kunde ta emot två typer av pulslängder, som varade antingen 1 ms eller 0.6 ms. Båda pulslängderna har en maximum period på 20 ms. PD13 anslutningen på STM32F3 Discovery mikrokontrollerkortet mättes med hjälp av en prob i oscilloskopet för att få en bild av hur pulslängderna såg ut som TIMER anslutningen ska generera till servomotorn. Följande tabell (se tabell 3) visar de anslutningarna av servomotorn som ska vara kopplade till STM32F3 Discovery mikrokontrollerkortet:

Servomotor	STM32F3 Discovery
Kontrollsignal (PWM)	PD13
VCC	5V
GND	GND

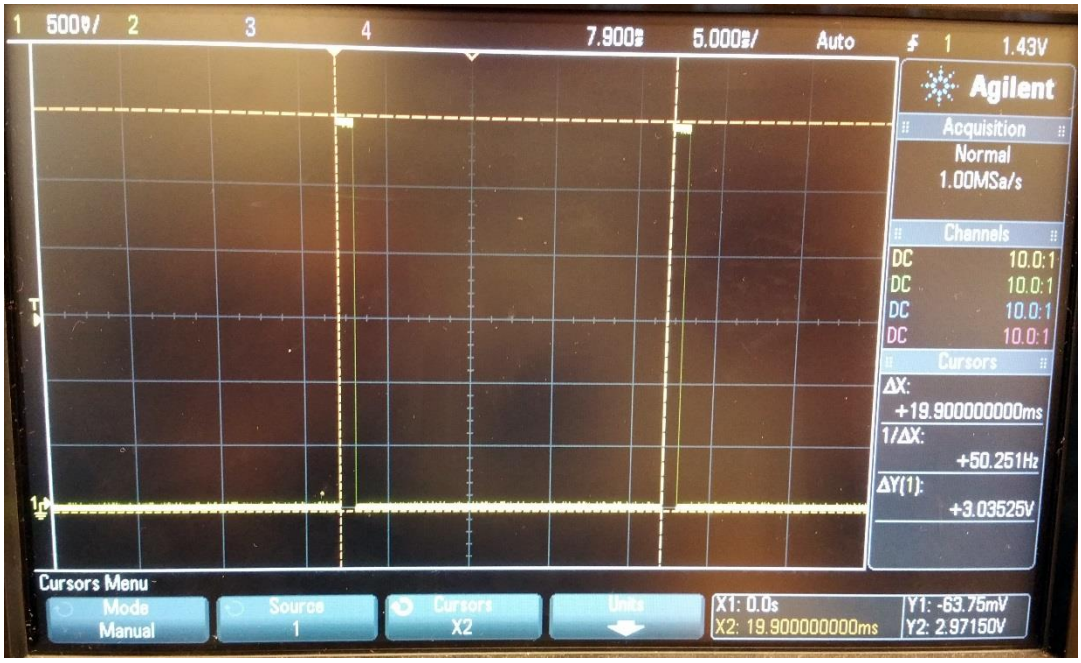
Tabell 3: Tabellen visar anslutningarna mellan servomotorn och STM32 Discovery [8].

Verktygsprogrammet STM32CubeMX som genererar en konfigurering av anslutningar på STM32 Discovery mikrokontrollerkortet, användes för att konfigurera anslutningen PD13 till TIM4.

En testdriven program användes för att generera en PWM signal till TIMER anslutningen, som var kopplad till PD13. Två PWM signaler generades genom att ändra på pulsbredden. Följande figurer (se figur 17 och 18) visar två olika pulsbredd (0,6 ms och 1 ms) med en periodtid på 20 ms.



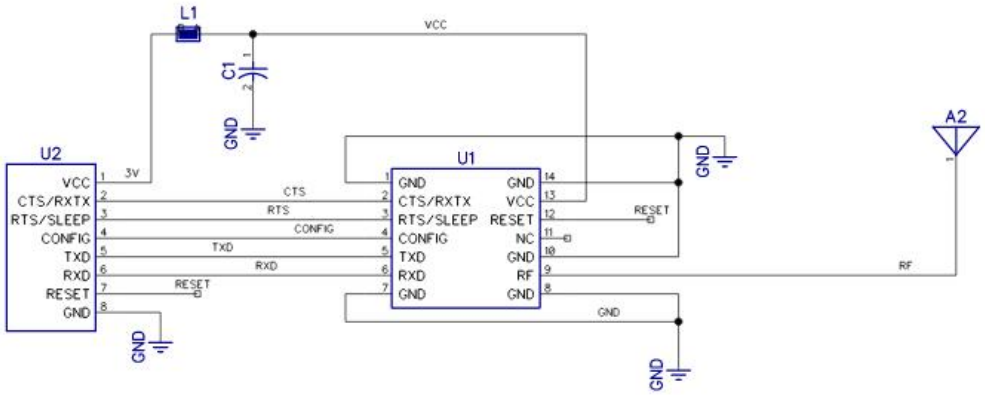
Figur 17: Figuren visar pulsbredden 0,6 ms för servomotorn i oscilloskopet.



Figur 18: Figuren visar pulsbredden 1 ms för servomotorn i oscilloskopet.

3.1.2 Anslutningar av elektronikenheterna

3.1.2.1 Anslutning av kommunikationskretsen

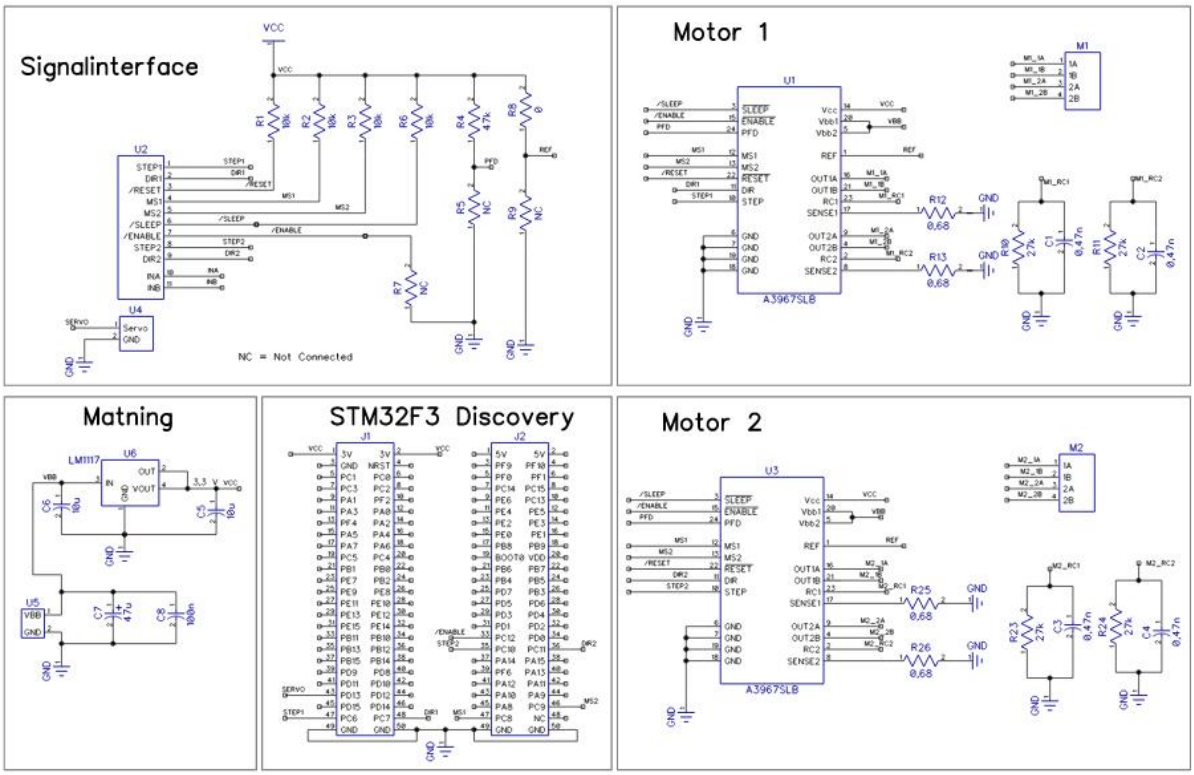


Figur 19: Figuren visar kopplingsschemat för kommunikationskretsen [19].

Anslutningen av kommunikationskretsen visas i ovanstående figur (se figur 19). Figuren visar två kretsar, där den markerad med U2 är mottagare och den markerad med U1 är drivkretsen. Båda dessa är kopplade till STM32F3 Discovery mikrokontrollerkortet i form av ett integrerad radiokort, för att kunna ta emot data samt sända feedback. En ytterligare krets, som ska agera som en sändare, kopplades till FTDI kabeln för att kunna sända data samt ta emot feedback [19].

De viktigaste anslutningarna i kretsarna för att få igång kommunikationen var VCC, TXD samt RXD, som existerar i både sändarkretsen och mottagarkretsen. Anslutningarna CTS, RTS samt CONFIG var frivilliga.

3.1.2.2 Anslutning av stegmotorerna



Figur 20: Figuren visar kopplingsschemat för stegmotor 1 och 2, matning av stegmotorerna, drivkretsen samt STM32F3 Discovery mikrokontrollerkortet [20].

Anslutningen av stegmotorerna visas i ovanstående figur (se figur 20). Figuren visar fem kretskopplingar, där två av dessa är stegmotorerna, en är drivkretsen A3967 samt kretsen för STM32F3 Discovery kortet och spänningsmatning för stegmotorerna. Stegmotor 1 och 2 var kopplade till drivkretsen A3967 i form av ett stegmotorkort, för att kunna ta emot pulser som ska driva stegmotorerna [20].

De viktigaste anslutningarna i drivkretsen är STEP1, DIR1, MS1, MS2, STEP2, DIR2 och ENABLE. Dessa används för att slå på strömmen för stegmotorerna, sätta igång båda stegmotorerna, ändra rotationsriktning samt ändra stegläge till fullsteg, halvsteg, kvartssteg eller mikrosteg.

För att kalibrera avståndet och roteringen som roboten skulle göra i mm respektive grader, användes följande data om roboten [8]:

$Hjuldiameter$: $47\ mm$

$Avståndet\ mellan\ hjulen$: $120\ mm$

Steglägen för stegmotorerna ställdes in till fullsteg så att ett varv skulle motsvara 100 pulser.

Följande beräkningar användes för att ta reda på hur många pulser som krävdes för ett visst avstånd:

$Omkretsen\ av\ hjulen$: $47\ mm * \pi = 147{,}69\ mm$

$Ett\ varv$: $100\ pulser = 147{,}69\ mm$

$$1\ puls = \frac{147{,}69\ mm}{100\ pulser} = 1{,}4769\ mm$$

För att roboten ska åka fram 100 mm, så behöver stegmotorerna så många pulser som beräknas nedan:

$$\frac{100\ mm}{1{,}4769} = 67{,}7\ \approx 68\ pulser$$

Följande beräkningar användes för att ta reda på hur många pulser som krävdes för en viss rotation:

$Omkretsen\ av\ hjulens\ avstånd$: $120\ mm * \pi = 376{,}99\ mm$

$90\ graders\ motsvarighet\ i\ längd$: $\frac{377\ mm}{4} = 94{,}25\ mm$

$1\ graders\ motsvarighet\ i\ längd$: $\frac{94{,}25\ mm}{90\ grader} = 1{,}05\ mm$

För att roboten ska rotera 90 grader, så behöver stegmotorerna så många pulser som beräknas nedan:

$90 * 1{,}05\ mm\ =\ 94{,}25\ mm$

$$\frac{94{,}25\ mm}{1{,}4769} = 63{,}8\ \approx 64\ pulser$$

3.1.2.3 Anslutning av servomotorn

Anslutningen av servomotorn görs med hjälp av anslutningarna VCC, GND samt kontrollsignal. Dessa är kopplade till 5V, GND respektive PD13 på STM32F3 Discovery mikronkontrollerkortet. Kontrollsignalen är den anslutningen som genererar en pulsbreddsmodulerad signal till servomotorn, så att denna ska justera om läget för metallskaften [8].

För att generera en pulsbredd för servomotorn, används följande data för roboten [8]:

$$Periodtid: T = 20\ ms$$

$$Antalet\ klockningar: 40000\ klockningar$$

$$Tid\ per\ klockning: \frac{20\ ms}{40000\ klockningar} = 0{,}5\ \mu s$$

$$Frekvens: \frac{1}{0{,}5\ \mu s} = 2\ MHz$$

Då STM32F3 Discovery mikrokontrollerkortet har en kristalloscillator som genererar en klockfrekvens upp till 72 M Hz, innebär det att TIMER anslutningens prescaler måste vara enligt beräkningarna nedan för att få ned den till 2 MHz:

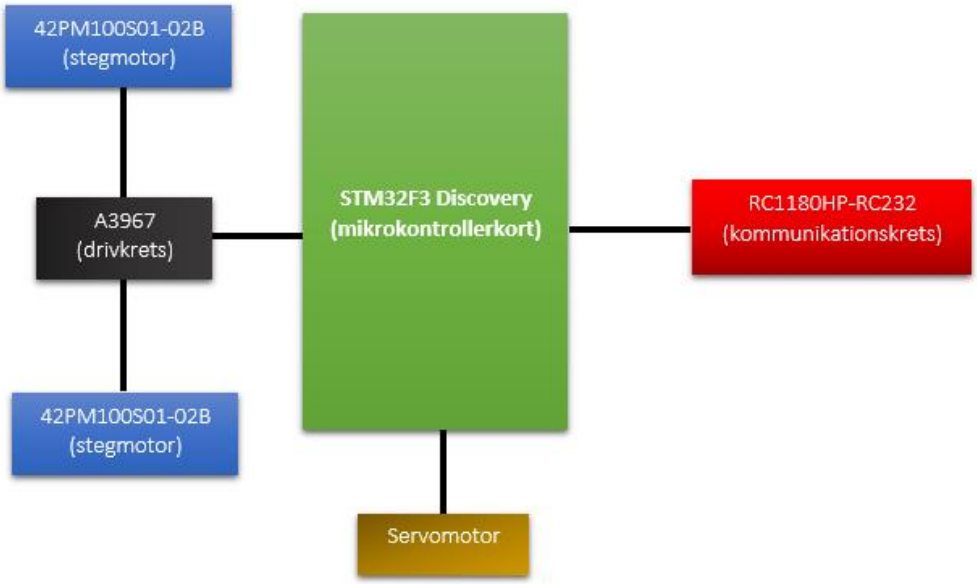
$$\frac{72\ MHz}{2\ MHz} = 36$$

Med hjälp av denna prescaler, genererades två olika pulsbredder med hjälp av följande formel:

$$\frac{(40000 * x)}{100}$$

För att lyfta ned pennan, ska x ha värdet x för att generera en pulsbredd på 0,6 ms. För att lyfta upp pennan, ska x ha värdet 5 för att generera en pulsbredd på 1 ms.

3.1.3 Blockschema

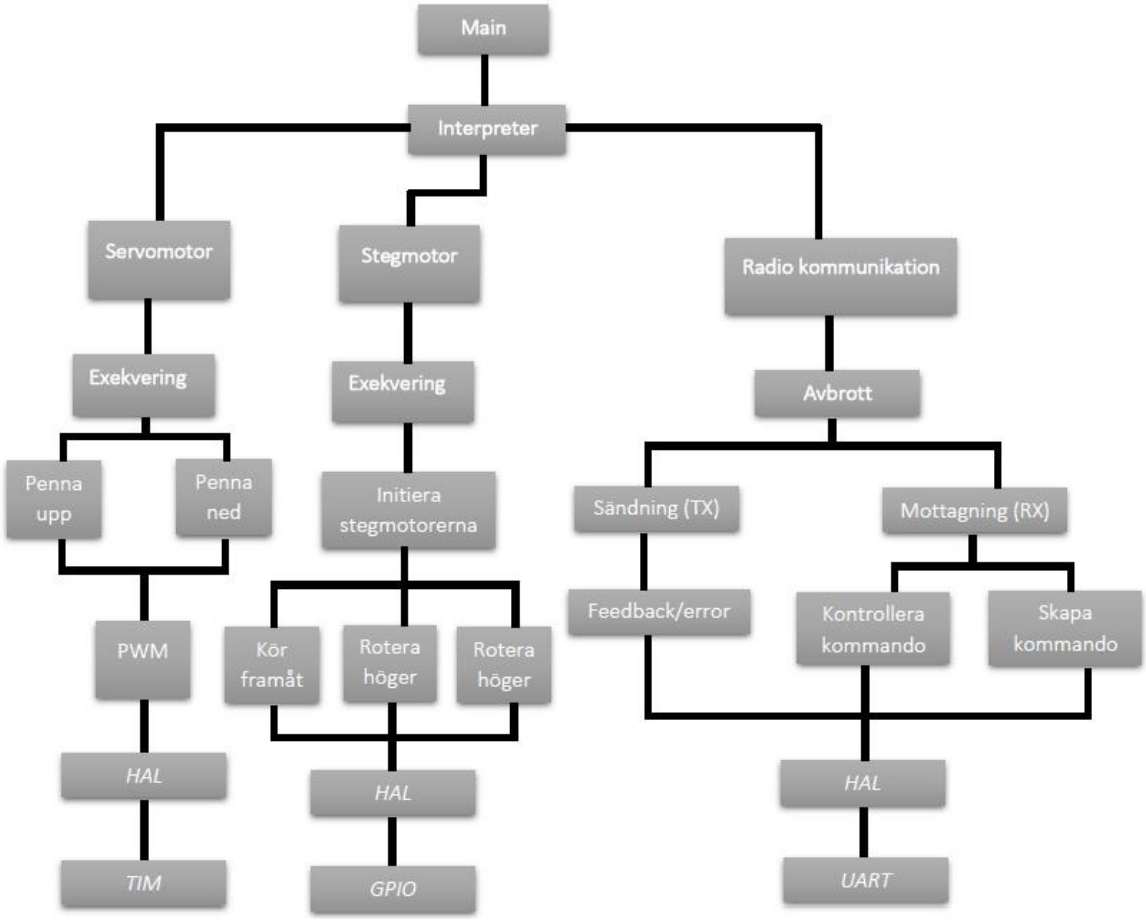


Figur 21: Figuren visar blockschemat som beskriver robotens hårdvara.

Ovanstående figur (se figur 21) visar robotens uppbyggnad och interaktionen mellan blocken. Den vänstra delen av blockschemat visar två stegmotorer som är kopplade till en drivkrets. Denna drivkrets är kopplad till STM32F3 Discovery mikrokontrollerkortet. Nedersta samt högra delen av blockschemat visar servomotorn respektive kommunikationskretsen, som är direktkopplade till STM32F3 Discovery mikrokontrollerkortet.

3.2 Mjukvara

3.2.1 Hierarkiskt arkitektur

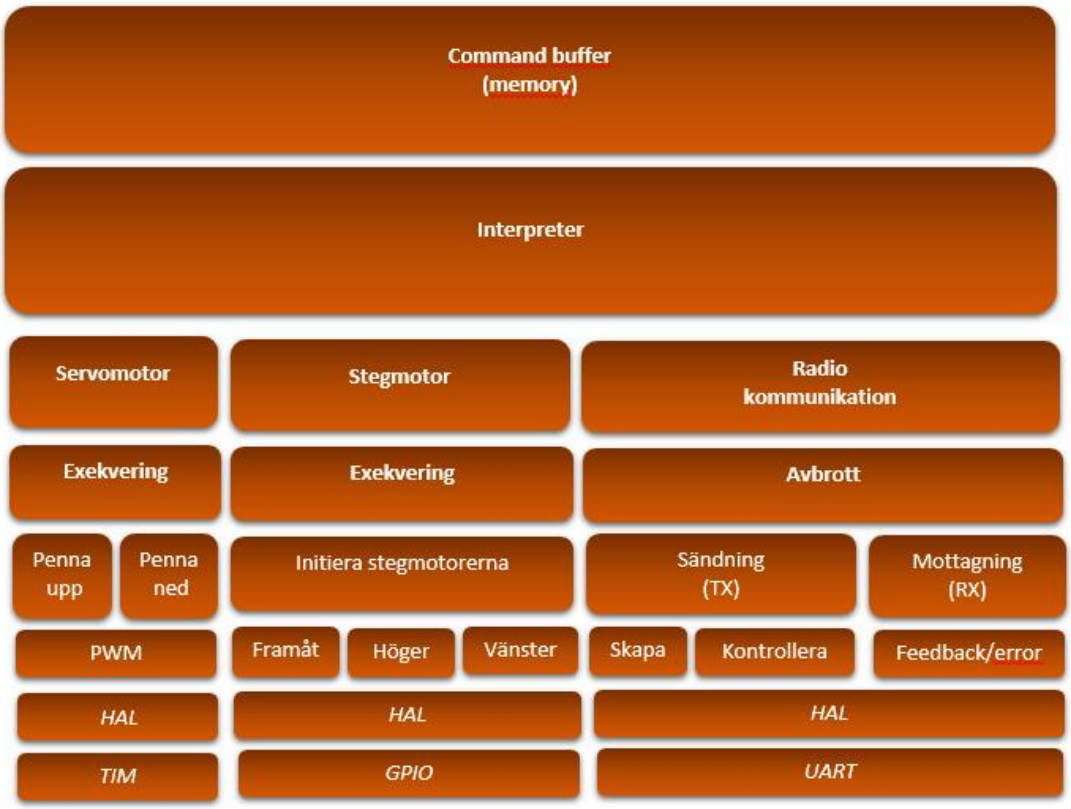


Figur 22: Figuren visar en hierarkisk arkitektur av mjukvaran.

Figuren ovan (se figur 22) visar en hierarkisk arkitektur av mjukvaran, där de viktigaste blocken är inkluderade. Arkitekturen visar de diskreta blocken samt de block som anropas. Övre delen av arkitekturen visar blocken main som är kärnan och styr hela programmet. Blocket interpreter beror av vad användaren har matat in för instruktion till roboten och denna i sin tur kontrollerar vilken diskret block som ska utföras. Dessa diskreta block består av servomotor, stegmotor eller radiokommunikation. Varje diskret block är självständig och har sin egen funktion. Ett exempel är diskreta blocket servomotor, som anropar blocken för PWM, dess HAL-bibliotek samt anslutningen TIM.

Dessa diskreta block måste vara inkluderade under main och interpreter blocket för att hela roboten ska fungera som den ska.

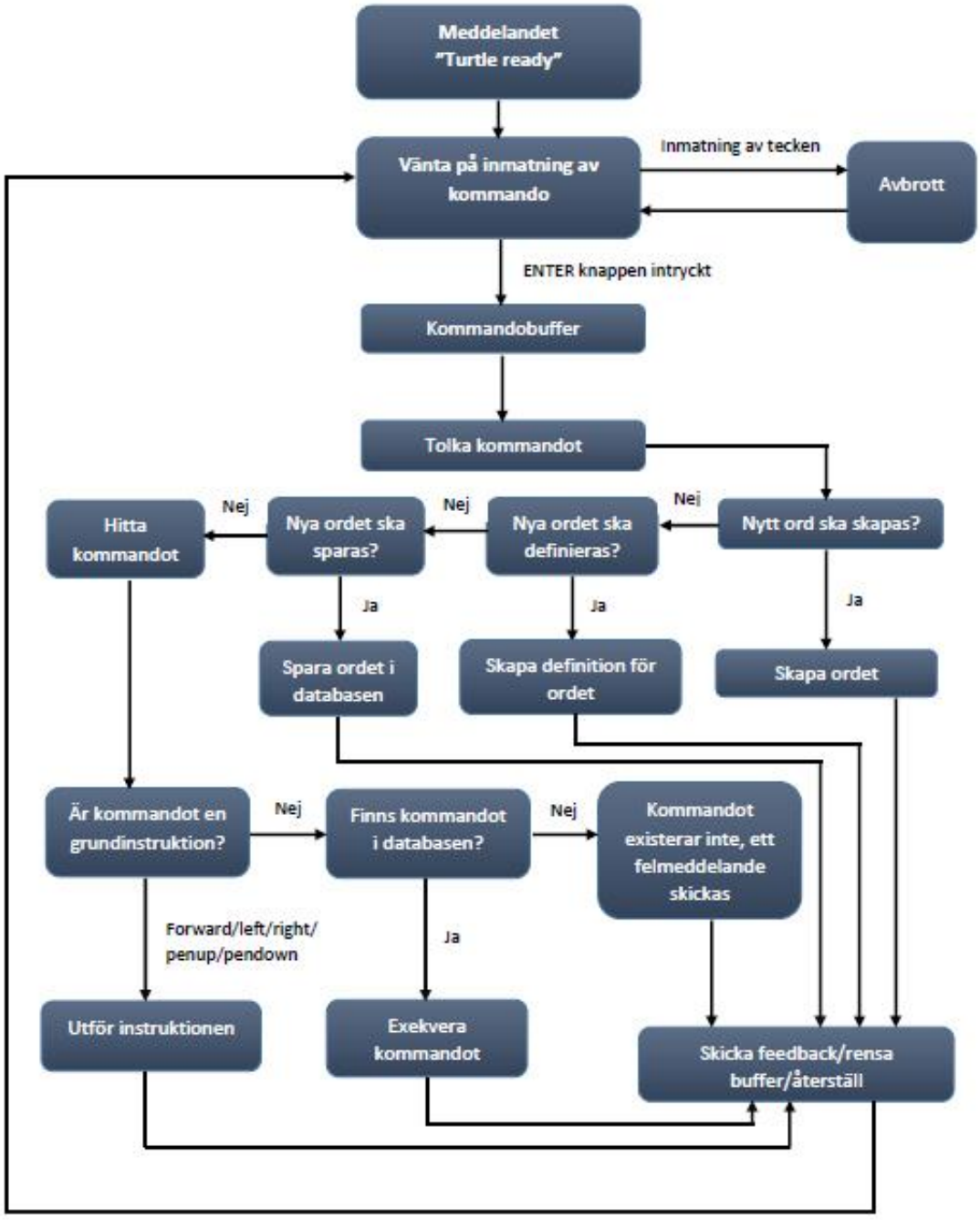
3.2.2 Lagermässig arkitektur



Figur 23: Figuren visar en lagermässig arkitektur av mjukvaran.

Figuren ovan (se figur 23) visar en lagermässig arkitektur av mjukvaran, där de största blocken visas högst upp. De stora blocken i arkitekturen beskriver hur komplex dessa delar av mjukvaran är. Ett exempel är blocken interpreter som tar emot en instruktion och behandlar detta genom att kontrollera instruktionen. Detta innebär att ett utav de undre blocken, som är beroende av interpreter blocket, kan utföras. De undre blocken är inte beroende av varandra och har sin egen funktion.

3.2.3 Flödesschema



Figur 24: Figuren visar ett flödesschema för mjukvaran.

Figuren ovan (se figur 24) visar flödesschemat för den fullständiga mjukvaran. Programmet börjar med att ett meddelande ”Turtle ready” skickas till terminalfönstret. Därefter så kommer programmet att invänta ett kommando från användaren i en loop.

När användaren matar in ett kommando, så kommer ett avbrott att ske. Efter att användaren har matat in kommandot och tryckt på ENTER, så kommer kommandot att sparas i ett buffervariabel, som bearbetas. Med bearbetningen innebär då att strängen delas upp i olika delar, där delarna sparas i separata variabler. Detta kommando ska då tolkas av programmet och följande fall kan ske:

1. Om användaren har matat in att den ska skapa ett nytt ord för roboten (t.ex. ’to square’), som sedan används som ett kommando, kommer programmet att spara ordet i en variabel.

2. Om användaren har matat in en definition samt antalet repetitioner av definitionen för ordet som ska skapas för roboten (t.ex. ’repeat 4 [forward 90 right 90]’), kommer programmet att spara definitionen och antalet repetitioner i två variabler.

3. Om användaren har matat in att den vill spara det nya ordet tillsammans med definitionen och antalet repetitioner av definitionen (i det här fallet, ’end’), så kommer det nya ordet att sparas som en struct i en array. Arrayet agerar som en databas och får plats med egendefinierade ord som användaren skapar för roboten.

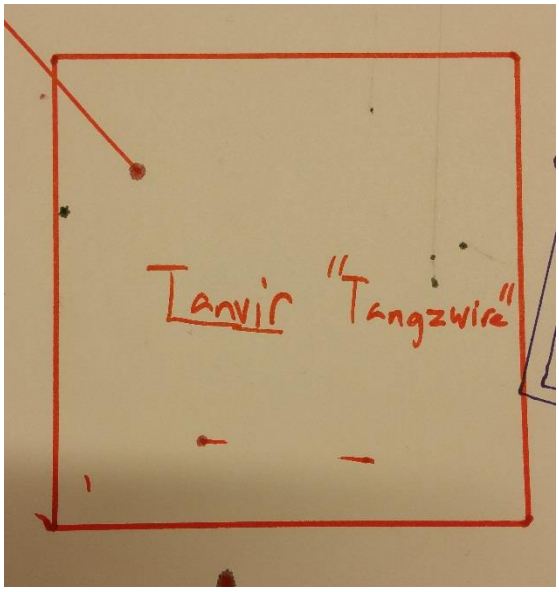
4. Om användaren har matat in ett kommando som ska leda till att roboten ska utföra en instruktion, kommer detta att kontrolleras med en av de grundinstruktionerna som existerar för roboten. Dessa grundinstruktioner är att köra framåt, rotera höger, rotera vänster, lyfta upp pennan eller att lyfta ned pennan. Om kommandot stämmer överens med en av grundinstruktionerna, kommer denna grundinstruktion att ske och en feedback kommer att skickas till terminalfönstret. Ifall det visar sig att användaren har matat in ett kommando som är egendefinierad, kommer kommandot att letas upp i databasen och utföras. Ett exempel är att användaren har matat in kommandot ’square’. Då kommer dess definition samt antalet repetitioner av definitionen att exekveras, vilket leder till att roboten ritar en fyrkant. Annars så kommer ett felmeddelande att skickas till terminalfönstret, som talar om att ordet inte existerar.

Vid alla dessa fall, kommer buffervariabeln samt övriga variabler att rensas för återställning och programmet går tillbaka till början för att invänta ett kommando från användaren i en loop.

4 Resultat

Roboten kan förstå grundkommandon som användaren skickar via ett terminalprogram och bearbeta dessa för att utföra instruktioner. Instruktionerna var att styra roboten så att den kan åka framåt, rotera höger eller vänster och lyfta upp eller ned pennan.

Roboten kan även lära sig ett nytt ord som användaren skapar, definierar samt sparar via ett terminalprogram. Orden sparas i en databas, där användaren sedan kan mata in ordet som ett kommando. Kommandot kan då exekvera dess definition och antalet repetitioner av definitionen så att roboten kan rita ett mönster (se figur 25).



Figur 25: Figuren visar fyrkanten som roboten ritade under körning.

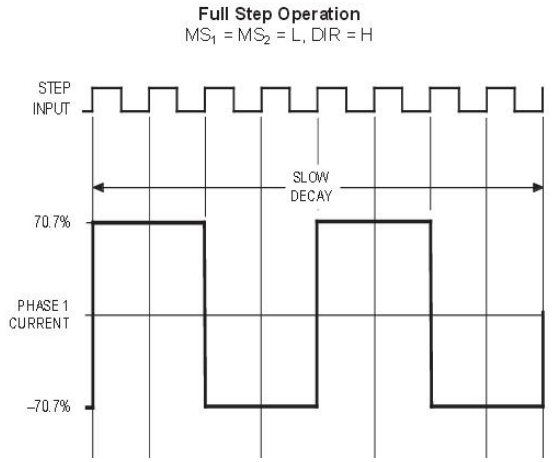
Linjer samt mönstren som ritas av roboten, har visat sig vara noggranna och skiljer inte sig alls från det som användaren matar in gällande avstånd samt rotation när en instruktion eller en definition av instruktioner ska utföras.

Även feedback och felmeddelanden rapporteras av roboten när en instruktion har körts klart respektive när ett fel kommando har matats in av användaren.

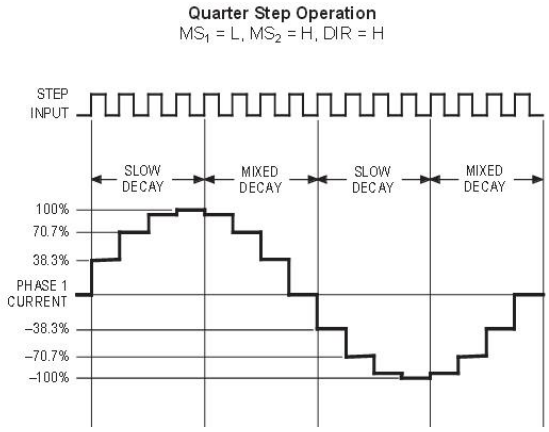
Ett testdriven har även skapats för att säkerhetsställa att programmodulerna som har utvecklats, testas och uppfyller de kraven som ställts. Programmodulerna bestod av att sända/ta emot strängar, stränghantering (interpreter), aktivering av elektronikenheterna samt att spara ett egen definierat kommando.

5 Analys och diskussion

Under tiden roboten utvecklades, så uppstod det ett par problem som gjorde att roboten inte fungerade som den skulle. Ett av dem var att roboten inte åkte fram eller roterade höger eller vänster ordentligt. Stegmotorerna hakade upp sig vid de flesta tillfällen, vilket ledde till att antingen den ena eller den andra stegmotorn stannade upp roboten vid körning. Det visade sig att när stegläget på stegmotorerna ställdes in till fullsteg, så tappade roboten oftast pulser på vägen när den skulle åka framåt eller rotera vänster eller höger. Stegmotorerna ställdes in i stegläget kvartsteg för att eliminera problemet. Detta ledde dock till att roboten åkte sakta men gav en mer noggrannhet gällande avstånd och rotation. Nedanstående figurer (se figur 26 och 27), visar noggrannheten av steglägena fullsteg och kvartsteg.



Figur 26: Figuren visar diagrammet för fullsteg [6].



Figur 27: Figuren visar diagrammet för kvartsteg [6].

Det testdrivna programmet var svårt att utveckla, då det var vissa delar av programmoduler som inte kunde testas. Ett exempel var att skapa ett test där en sträng som innehåller flera ord med mellanslag mellan varje ord, skulle delas upp så att ett ord i taget skulle skrivas ut på terminal I/O. Denna del av programmodulen var avancerat integrerat och kunde tyvärr inte testas via testdrivna programmet.

På grund av den begränsade tiden, så kunde utvecklingen av roboten ha påbörjats lite tidigare, för att få lite mer information och förståelse kring hårdvaran för roboten.

6 Slutsats

Roboten har uppnått alla mål och fungerar efter de krav som har ställts. Roboten utvecklades efter kopplingsschema som var tillgänglig och flödesschema som togs fram.

Testning av elektronikenheterna som bygger upp roboten samt det framtagna testdrivna programmet som grundar sig på programmodulerna, har gett en bra översikt över hur ett inbyggt system ska utvecklas, både mjukvarumässigt och hårdvarumässigt.

7 Rekommendation

Det kan vara bra att läsa datablad för de elektronikenheterna som ska användas i ett system. Det ger en bra översikt om vad elektronikenheterna klarar samt hur dessa ska hanteras så att risken för att dessa blir ur funktion, minskar.

En bra start på utvecklingen av ett system är att rita upp ett flödesschema och tänka ut vad varje block i flödesschemat ska göra. Detta underlättar och ger även mer tid till att felsöka problem som kan uppstå, samt även optimera koden till att göra den kortare. En vältänkt testdriven program, som kan testa alla delar av programmodulerna, leder till att programmet är felfri och är säker att redigera.

Referenser

Faktakällor

[1] Specifikationer för STM32F303VC Discovery
http://www.st.com/web/en/catalog/mmc/FM141/SC1169/SS1576/LN1531/PF252054
Besökt datum: 2016-01-29

[2] Stegmotor
http://www.omega.com/prodinfo/stepper_motors.html
Besökt datum: 2016-02-28

[3] Unipolära och bipolära stegmotorer
http://mechatronics.mech.northwestern.edu/design_ref/actuators/stepper_intro.html
Besökt datum: 2016-02-28

[4] Datablad för Moons 42PM100S01-02B
http://shpat.com/docs/elfa/05446620.pdf
Besökt datum: 2016-01-28

[5] Drivkrets
http://www.hobbytronics.co.uk/uln2003a-darlington-array
Besökt datum: 2016-01-28

[6] Datablad för Allegro A3967
https://www.kth.se/social/upload/52e8f0eef27654272409c84f/A3967-Datasheet.pdf
Besökt datum: 2016-03-01

[7] Servomotor
http://www.servocity.com/html/what_is_a_servo_.html
Besökt datum: 2016-03-03

[8] Turtle robot KTH (Inbyggda system, IS1300)
https://www.kth.se/social/course/IS1300/page/turtle-hardvara-2/
Besökt datum: 2016-03-15

[9] Discovering the STM32 Microcontroller
http://www.cs.indiana.edu/~geobrown/book.pdf
Besökt datum: 2016-03-04

[10] Datablad för FTDI USB TTL Serial kabel
http://www.ftdichip.com/Support/Documents/DataSheets/Cables/DS_TTL-232R_CABLES.pdf
Besökt datum: 2016-03-05

[11] Radio kommunikation
http://www.futureelectronics.com/en/wireless-rf-radio-frequency/rf-modules-solutions.aspx
Besökt datum: 2016-03-08

[12] Datablad för Radiocrafts RC1180HP-RC232
https://www.kth.se/social/upload/52e8f114f276542846218f71/Radiocrafts_RC232.pdf
Besökt datum: 2016-03-10

[13] Oscilloskop
http://uenics.evansville.edu/~amr63/equipment/scope/oscilloscope.html
Besökt datum: 2016-03-12

[14] Pulse Width Modulation (PWM)
https://www.arduino.cc/en/Tutorial/PWM
Besökt datum: 2016-03-06

[15] PuTTY terminal
http://www.chiark.greenend.org.uk/~sgtatham/putty/faq.html
Besökt datum: 2016-03-06

[16] IAR Systems
https://www.iar.com/iar-embedded-workbench/
Besökt datum: 2016-03-07

[17] Programmeringspråket C
http://groups.engin.umd.umich.edu/CIS/course.des/cis400/c/c.html
Besökt datum: 2016-02-17

[18] Turtle Graphics
https://docs.python.org/2/library/turtle.html
Besökt datum: 2016-03-01

[19] Kopplingsschema för kommunikationskretsen
https://www.kth.se/social/files/54d0f99df2765475ccb40612/Schema_RC232_STM32F3.pdf
Besökt datum: 2016-03-16

[20] Kopplingsschemat för stegmotor, spänningsmatning, drivkretsen och STM32F3 Discovery.
https://www.kth.se/social/upload/52e8f0eef27654272409c84f/A3967-Datasheet.pdf
Besökt datum: 2016-03-16

Bildkällor

[21] STM32F303VC Discovery
http://www.st.com/st-web-ui/static/active/en/fragment/product_related/rpn_information/board_photo/stm32f3discovery.jpg

[22] Moons 42PM100S01-02B
http://www.ittgroup.ee/962/stepper-motor-bipolar-42pm100s01-02b.jpg

[23] Servomotor
http://www.robotshop.com/media/files/images/hitec-hs-5585mh-servo-motor-large.jpg

[24] Source (PC) and destination (DAC03)
http://armprogramming.com/wp-content/uploads/2014/12/host-target.jpg

[25] Agilent DSOX2014
https://www.kth.se/social/files/5513bf60f276547ce0536711/agilent2000_large.png

[26] Pulse Width Modulation (PWM)
http://www.zembedded.com/wp-content/uploads/2012/12/AVR_PWM.png

[27] PuTTY terminal
http://mxweas.com/blog/wp-content/uploads/2008/01/sb6.png

[28] IAR Embedded Workbench
https://mcuoneclipse.files.wordpress.com/2013/05/iar-embedded-workbench-ide-debugging-with-cmsis-dap-and-frdm-kl25z.png

[29] Turtle Graphics
http://blog.ziyeliu.com/uploads/9/9/8/3/9983584/7899092_orig.png