CHALMERS
UNIVERSITY OF TECHNOLOGY

Utveckling av autoleveleringssystem för specialtillverkade trailrar

Examensarbete inom Högskoleingenjörsprogrammet Elektroteknik

Oskar Nilzén

Carl-Johan Vickström

Institutionen för Elektroteknik

CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2023
www.chalmers.se

Examensarbete vid institutionen för Elektroteknik 2023

Utveckling av autoleveleringssystem för specialtillverkade trailrar

Oskar Nilzén, Carl-Johan Vickström

CHALMERS

Institutionen för Elektroteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2023

Utveckling av autoleveleringssystem för specialtillverkade trailrar
Design och konstruktion tillämnat för Groth Kaross AB
Oskar Nilzén
Carl-Johan Vickström

Handledare: Erik Magndal, Groth Kaross AB
Examinator: Veronica Olesen, Institutionen för Elektroteknik

Högskoleexamensarbete 2023
Institutionen för Elektroteknik
Chalmers tekniska högskola
SE-412 96 Göteborg, Sverige
Telefon: +46 (0)31-772 1000

Omslagsbild: Bild tagen av Erik Magndal hos Groth Kaross AB.

Sammanfattning

Detta projekt syftade till att automatisera en manuell process för att autolevelera en lastbilstrailer i samarbete med Groths Kaross AB. Autolevelering innebär att ett system automatisk justerar sig själv för att upprätthålla en horisontell position och behövs för att skapa en stabil grund för trailern. Tidigare har stödbenen manövrerats manuellt med fjärrkontroll och en lösning i form av ett elektronsikt styrsystem önskades.

Tillsammans med handledare planerades arbetet utifrån komponenter, laborationsmodell och mål. Arbetet strukturerades in i totalt fyra delar där den första delen var att ta fram ett kretsschema. Därefter fortsatte arbetet med att montera samtliga komponenter och koppla samman dem på en nedskalad 1:10-modell av en verklig trailer. Avslutningsvis programmerades systemet utefter de mål som sattes upp tidigare.

Systemet konstruerades genom användning av en Arduino MEGA som programmerades med funktioner för att styra samtliga processer. Därefter integrerades olika elektriska komponenter såsom tryckknappar, reläer, MPU-enheter och linjära ställdon i systemet. Resultatet blev en komplett modell som fungerade enligt samtliga krav som företaget hade satt upp och kommer användas i demonstrationssyfte för Groths kunder.

Keywords: Trailers, Automation, Autolevelering , Smarta system , Arduino , Avståndsmätare, Filtrering, MPU6050, Relays.

Abstract

This project aimed to automate a manual process of auto-leveling a truck trailer in collaboration with Groths Kaross AB. Auto-leveling involves a system automatically adjusting itself to maintain a horizontal position and is necessary to create a stable foundation for the trailer. Previously, the support legs were maneuvered manually using a remote control, and a solution in the form of an electronic control system was desired.

Together with the supervisor, the work was planned based on components, a laboratory model, and goals. The work was structured into a total of four parts, where the first part involved creating a circuit diagram. The work then continued with the assembly of all components and their connection on a scaled-down 1:10 model of a real trailer. Finally, the system was programmed according to the goals set earlier.

The system was constructed using an Arduino MEGA, which was programmed with functions to control all processes. Various electrical components such as push buttons, relays, MPU devices, and linear actuators were then integrated into the system. The result was a complete model that functioned according to all the requirements set by the company and will be used for demonstration purposes for Groths' customers.

Keywords: Trailers, Automation, Autolevelering , Smart system , Arduino , Distance Sensor, Filtering, MPU6050, Relays.

Förord

Detta examensarbete har varit en spännande och lärorik resa för oss och vi vill därför ta tillfället i akt att tacka alla som har bidragit till att göra detta möjligt.

Först och främst vill vi tacka företaget Groth Kaross som har gett oss möjligheten att genomföra detta examensarbete. Vi är tacksamma för företagets företroende och för att ha fått chansen att komma in i deras bransch, förstå hur de arbetar och lösa ett verkligt problem.

Vi vill rikta ett stort tack till Erik Magndal, vår fantastiska handledare på Groth Kaross, för all hjälp och stöd som han har bistått med under hela projektet. Hans feedback, hjälp med att bygga grundmodellen och all tid och energi han har lagt ner har varit ovärderlig för att vi skulle kunna lyckas med arbetet.

Till sist vill vi även tacka vår examinator på Chalmers, Veronica Olesen, för all vägledning för både den tekniska biten och med rapport skrivningen.

Vi hoppas att detta arbete kommer vara till god hjälp för Groth Kaross i deras strävan att underlätta för sina kunder.

Oskar Nilzén och Carl-Johan Vickström, Göteborg, 2023

Terminologi

Nedan är en lista över förkortningar och terminologi som används i rapporten i alfabetisk ordning:

A/D	Analog to digital
GND	Referenspunkt som erhåller spänningen 0V.
I2C	Ett seriellt kommunkationsprotokoll som överför data mellan elektroniska enheter.
IMU	Inertial Measurement Unit, Elektronisk enhet som består av flera sensorer som accelerometer, gyroskop för att mäta rörelser i 3 dimensioner.
LSB	Least Significant Bit. Det minst signifikanta biten i en binär kod
MPU	Motion Processing Unit
NC	Normally Closed, En kontakttyp som är stängd (ström flyter igenom) då brytaren eller knappen ej är aktiverad.
NO	Normally Open, En kontakttyp som är öppen då brytaren eller knappen ej är aktiverad.
Pitch Axis	En enhets rotation kring dess sidledsaxel.
PWM	Pulse Width Modulation, Signaltyp där bredden på en periodisk pulssignal varieras för att styra spänningen.
Roll Axis	En enhets rotation kring dess längdaxel.
SPDT	Single Pole, Double Throw, En typ av strömbrytare som har en gemensam styringång och två separata utgångar.
VCC	Positiv spänning som driver en krets.

Innehållsförteckning

1
Introduktion

1.1 Bakgrund

SKAB-Gruppen, som har sitt huvudkontor i Halland, består av ett antal olika bolag där Groth Kaross AB ingår. Groths Kaross AB tillverkar specialanpassade fordon och påbyggnader inom lastbilsbranchen. Exempelvis på tidigare byggen är blodgivartrailer till Karolinska sjukhuset, mobil arrest till polisen samt en vallatrailer till svenska skidskyttelandslaget. Dessa trailers är mobila, det vill säga att de körs ut och sedan lämnas på plats med hjälp av en dragbil. För att trailern sedan ska kunna stå stadigt på plats utan dragbilen, behövs därför stödben för att upprätthålla positionen. Stödbenens huvuduppgifter blir därmed att möjliggöra frånkoppling av dragbilen samt att planutjämna trailern.

Idag manövereras stödbenen manuellt med hjälp av en fjärrkontroll vilket är både tidskrävande och försvårar processen mellan stationärt och mobilt läge. En lösning i form av ett elektronisk styrsystem önskades.

Idag finns liknande färdiga system för just autoleveleringsändamål, men i Groths fall är de inte användbara eftersom de är skapade för andra syften. De system som är riktade för starkare stödben anses inte ha tillräckligt med möjligheter till styrning. De system som har denna möjlighet är endast riktade till applikationer som baseras på svagare stödben.

1.2 Syfte

Syftet med examensarbetet var att designa och konstruera ett styrsystem som går att implementera på trailernas befintliga hydralsystem. För att styra det befintliga hydralsystemet används magnetventiler som baseras på ett 24 V system. Det här arbetet kommer att tillhandahålla en lösning som kommer göra det enklare för Groths kunder att parkera sina trailers.

1.3 Avgränsningar

Eftersom det inte fanns möjlighet att laborera på ett befintligt hydralsystem så kommer samtliga tester basera sig på linjära ställdon istället för magnetventiler. Utbytet medförde skillnader i testerna och i verkligheten i form av tider, mekaniska påfrestningar samt delar av styrsystem som kommer behövdes anpassas efter längd

och andra fysiska egenskaper.

Systemet och samtliga komponenter kommer enligt produktblad klara av vanliga väderförhållanden. Dock testades inte uthålligheten vare sig det gäller långvariga perioder eller extrema väderförhållanden.

Projektet baserades endast på en modell då tidsbegränsningarna inte tillät möjligheten att köra systemet på en verklig trailer.

1.4 Mål

Tillsammans med handledaren från Groths Kaross AB, Erik Magndal, togs diverse krav fram som systemet ska klara av. Modellen som konstrueras ska kunna göra detsamma som en verklig trailer vilket ledde till beslutet av komponenterna som har valts samt uppbyggnaden av modellen. Dessa funktioner ska vara marksättning: där benen går ner och håller trailern i ett stilla läge så att dragbilen kan frånkopplas. Systemet ska också komma ihåg denna positionen för att när dragbilen ska kopplas på igen måste trailern ha samma position som när den frånkopplades. Autonom planutjämning: där systemet ska lyfta upp trailern och använda de fyra olika stödbenen för att på så sätt planutjämna trailern så att det går att arbeta i den.

Utöver dessa automatiska system ska det även gå att manuellt styra alla benen både upp och ner. Det ska även finnas minst två säkerhetssystem i form av en pausfunktion, som stannar den processen som körs ifall användaren vill det, och en startsignal som måste vara aktiv för att systemet ska kunna flytta benen.

1.5 Precisering av frågeställningen

I systemet ska följande sekvenser automatiseras:

Från mobilt till stationärt läge:
- Marksättning med stödben i markparallellt läge - Touch-Down
- Autonom planutjämning av trailern - Autolevel

Stationärt till mobilt läge:
- Återgå till markparallellt läge för dragbilsanslutning - Trailer Connect
- Hissa upp stödben - Raise

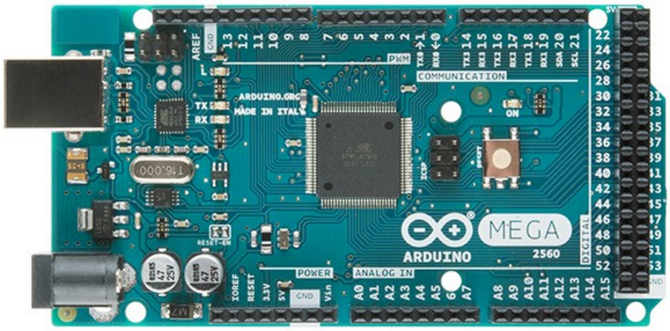
Frågeställningar:
- Vilka säkerhetsfunktioner krävs av systemet?
- Är modellen en lämplig representation av verkligheten?
- Hur kan systemet vidareutvecklas?

2
Teori

I detta kapitel presenteras teorin och funktionaliteten av de olika komponenterna och styrsystemen som använts i projektet.

2.1 Arduino MEGA mikrokontroller

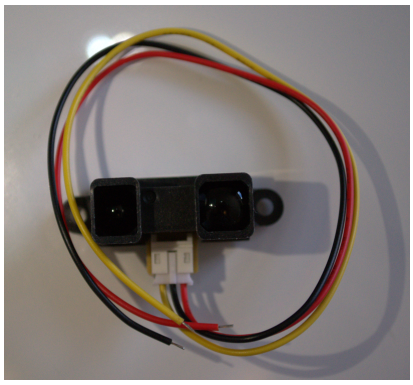
Arduino Mega är en öppen källkod mikrokontroller som erbjuder en enkel lösning för utveckling av elektroniska projekt [1]. En mikrokontroller är en enhet som består av en processor och minne. För versionen som användes i detta projekt, MEGA2560 versionen som visas i figur 2.1, finns det 54 digitala ingångar/utgångar, 16 analoga ingångar, jord, 5 V och 3.3 V utmatning[2]. Arduino Mega 2560 samlar då allt på ett och samma kretskort vilket förenklar många kretsar. Arduinon använder sig av sin egna IDE, en programvarumiljö som baserar sig på en förenklad version av C++ [1].



Figur 2.1: Designlayout för en Arduino Mega. [3] CC-BY

2.2 Avståndssensor IR

En infraröd (IR) avståndssensor, som visas i figur 2.2, är en komponent som används för avståndsmätning till en yta eller ett föremål. Grundprincipen baserar sig på att den utstrålar en IR stråle från sin sändardel och sedan mäter vinkeln som strålen kommer tillbaka in i mottagardelen [4]. På så sätt kan sensorn uppskatta och beräkna avstånden till närmsta föremålet. Avståndsmätaren har tre anslutningar: GND (jord), VCC (inmatning) samt V_o (analog utmatning). Avståndssensorn skickar informationen genom att ändra spännningsnivån där en högre spänning innebär större avstånd och lägre spänning innebär närmare avstånd. För att veta exakt vilken spänning som representerar vilket avstånd används sensorns datablad [5].

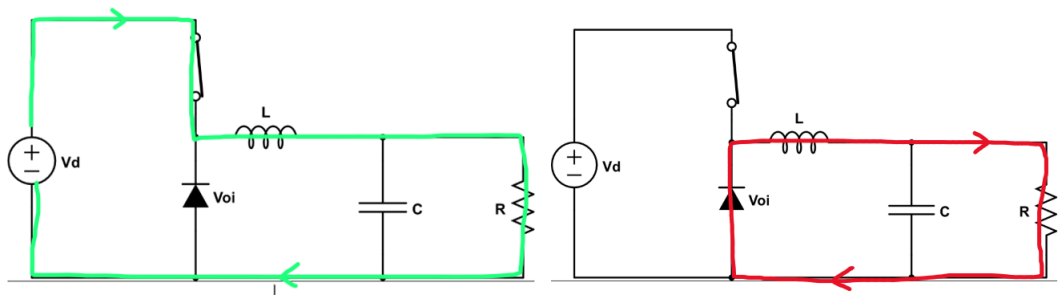


Figur 2.2: Avståndssensor IR 4-30cm.

2.3 Buck/Step-down-omvandlare

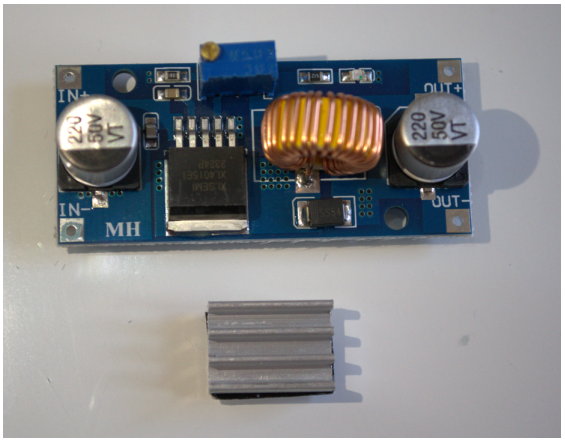
En switchregulator, även känd som step-down omvandlare, är en DC-DC komponent som är designad för att minska spänningen hos en insignal till en lägre nivå. En switchregulator består vanligtvis av en ingång, en switch (oftast en transistor), en induktor, en kondensator, en diod och en utgång [6]. När switchen är på slås ingångsspänningen på och strömmen flödar genom induktorn, vilket lagrar energi. Under på läget laddas även kondensatorn upp för att utjämna eventuella variationer i spänningen. När switchen sedan slås av, kopplas induktorn om så att den överför den lagrade energin till utgången. Kondensatorn används för att utjämna och stabilisera spänningen, vilket resulterar i en stabiliserad utspänning.

För att justera utspänningen reglerar switchregulatorn switchens till- och från-tid. Genom att minska på-tiden ökar den genomsnittliga utspänningen, medan en längre på-tid minskar utspänningen. Denna teknik kallas pulsbreddsmodulering (PWM) och är vanligt förekommande i switchregulatorer. I figur 2.3 kan du se en illustration av kretsdiagrammet. Den gröna linjen representerar strömmen när switchen är på och den röda linjen visar strömmen när switchen är av. Linjerna visar hur strömmen flödar genom kretsen under både på- och av-läget. Denna process resulterar i en genomsnittlig utspänning som är stabil och lägre än inspänningen [6].



Figur 2.3: Strömmar i switch regulator.

Frekvensen av switch-cykeln, dvs tiden som omvandlaren är i på-tid, styrs genom att skruva på PWM kontrollern i mörk blå färg som syns i fig. 2.4.

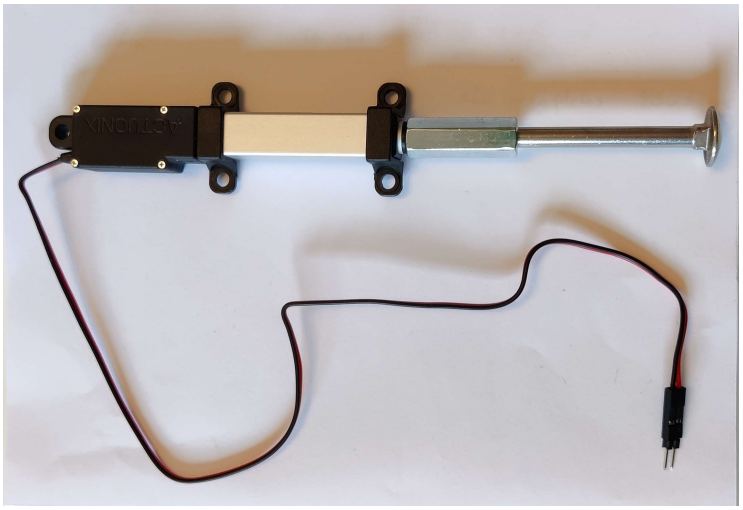


Figur 2.4: Buck omvandlare 1.25-35 V5A.

2.4 Linjära ställdon

Ett linjärt ställdon är en komponent som används för att få en mekanisk rörelse med hjälp av en elektrisk signal. De ställdon som används i projektet består av en enkel DC-motor som kan rotera medurs eller moturs vilket bestäms av spänningspolariteten. Denna rotation omvandlas sedan till en linjär rörelse vilket gör att det linjära ställdonet antingen kan utvidgas eller dras tillbaka.

Det finns flera versioner av ställdon som kan ge mer information gällande exempelvis nuvarande position. Dock för att efterlikna det hydrauliska systemet som inte har dessa funktioner, så kommer denna version med endast VCC och GND att användas (se figur 2.5 nedan). Dessa ska ersätta de hydrauliska stödbenen för projektet.



Figur 2.5: Simpelt DC linjärt ställdon.

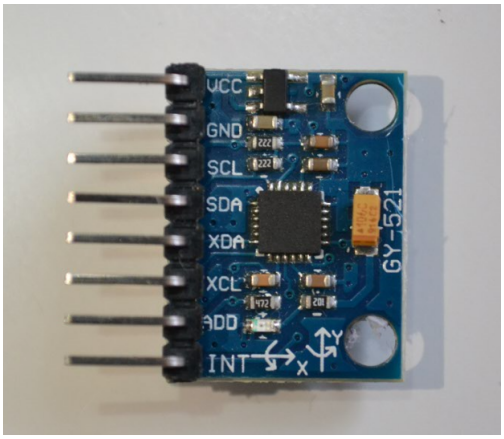
2.5 MPU-6050 accelerometer

MPU-6050 består av en accelerometer och ett gyroskop [7] som kommunicerar via I2C protokollet vilket beskrivs senare i avsnitt 2.6. Accelerometern används för att mäta linjär acceleration genom tre axlar (X,Y,Z). Det den bestämmer är accelerationen av en enhet relativ till jordens gravitation. Gyroskopet är en sensor som används för att mäta vinkelhastigheten runt tre ortogonala axlar (X,Y och Z). Gyroskopet och accelerometern tillsammans fastställer därmed orientationen av enheten.

Som tidigare nämnt använder MPU:n I2C kommunikation och enheten kan ha två olika adresser, antingen 0x68 eller 0x69 [8]. Adressen bestäms av AD0 porten och beror på om den är kopplad till GND eller 5 V. Då kommunikationen till flera olika enheter används på en och samma buss så behövs adresser för att informationen ska veta vart den ska. Exempelvis kan adresserna och informationen liknas som ett brev. Om ett brev ska skickas måste en adress finnas med så brevbäraren vet vart brevet ska.

En accelerometer mäter ett analogt värde vilket avser en signal som kan anta alla värden inom ett visst intervall. Exempel på analoga signaler inkluderar ljudvågor eller mätningar av fysiska storheter så som acceleration och tryck. Den andra varianten är digitala värden som är diskreta signaler och kan endast anta ett begränsat antal diskreta värden, representerade av binära siffror 0 och 1. Exempel på digitala signaler inkluderar dataöverföring via datorer och elektroniska enheter, där informationen kodas som en sekvens av binära siffror.

En analog-till-digital (A/D)-omvandlare omvandlar en analog signal till en digital signal. Den analoga ingångssignalen delas upp i mindre delar, eller kvantiseringsnivåer [9]. Varje kvantiseringsnivå motsvarar en binär kod vilket erhåller ett visst värde där Least Significant byte (LSB) representerar det minst signifikanta biten i denna binära kod.



Figur 2.6: MPU-6050 accelerometer 3-axel & gyro.

LSB representerar alltså den minsta förändringen i den digitala signalen från A/D-omvandlaren. Beroende på antalet bitar som används i omvandlaren bestämmer LSB:ens precisionen. Detta innebär då att LSB att utgöra den minsta möjliga förändringen i det analoga värdet som det digitala värdet kan registrera.

2.6 I2C kommunikation

I2C, eller Inter-Integrated Circuit, är ett seriellt kommunikationsprotokoll och kommer att användas i systemet för kommunikationen med MPU:erna. Denna typen av kommunikation har fördelen att den endast använder två kablar, Serial Clock (SCL) och Serial Data (SDA) där SCL används för att synkronisera dataöverföringen på SDA bussen [10]. Kommunikationen sker mellan en masterenhet och en eller flera slaveenheter. Kommunikationen sker genom att masterenheten skickar ut en adress dit den vill kommunicera och då svarar enheten på adressen med ett bekräftelsemeddelande. Därefter kan kommunikationen mellan enheterna påbörjas. Masterenheten kan därefter antingen skicka eller ta emot data från slavenheten och varje meddelande som skickas är en byte, eller åtta bitar. Efter varje byte som skickas eller tas emot skickas också ett bekräftelsemeddelande. När kommunikationen är färdig skickar masterenheten en stoppsignal ut på bussen vilket gör att slavenheten slutar lyssna på instruktioner och lyssnar endast efter adressen igen.

I detta projektet användes Arduinos inbyggda I2C bibliotek Wire.h [11]. Kommunikationshastigheten kommer vara 400 kHz enligt instruktionerna för MPU:n [8]. Det kommer även, som tidigare nämnts i kapitel 2.5, att användas två stycken MPU:er. Därför kommer båda adresserna för MPU:erna att användas, 0x68 och 0x69.

2.7 Kalmanfilter

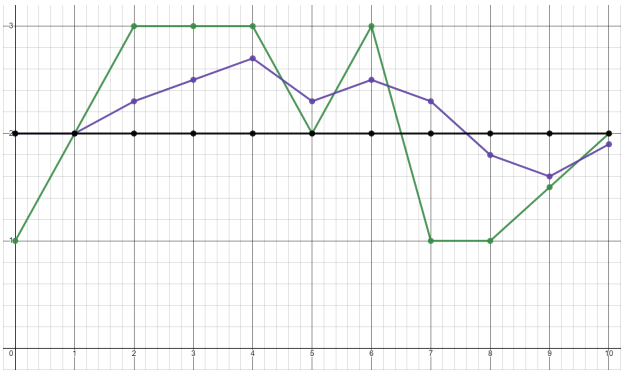
Ett Kalmanfilter är en metod för att uppskatta en okänd variabels värde baserat på mätningar och gissningar för att filtrera ut störningar. I detta system användes en

variant av ett kalmanfilter att användas som kallas för 1-dimensionellt kalmanfilter [12]. Denna typ av filtrer används vid endimisionella variabler så som avstånd eller vinklar.

Filtret använder sig av två olika informationspunkter för att uppnå en mer noggrann och pålitlig uppskattning av det faktiska värdet. Den första informationspunkten är en gissning eller en uppskattning av hur värdet kommer att förändra sig baserat på tidigare observationer och kunskap om systemet. Denna gissning kan vara baserad på olika algoritmer och modeller som tar hänsyn till tidigare mätvärden, trender eller andra relevanta faktorer. Den andra informationspunkten är ett faktiskt mätvärde som erhålls från mätutrustningen eller sensorerna. Mätvärdet representerar den verkliga observationen av systemet vid en given tidpunkt. Genom att kombinera den tidigare gissningen med aktuella mätvärden kan filtret förbättra sin förmåga att förutsäga och approximera de framtida mätpunkterna.

Uppdateringen i gissningarna resulterar i en gissning som liknar det faktiska värdet i verkligheten. Filtret viktar också hur mycket av det uppskattade värdet i förhållande till det faktiska som ska gå igenom till det filtrerade värdet. Denna viktning kallas i detta projekt för Kalman_Q, se kapitel 6.2.8 för hur denna variabel justeras i mjukvaran.

Ett exempel visas nedan i figur 2.7 där den gröna linjen är mätvärden som sensorn ger ifrån sig. Den svarta är den faktiska värdet och den lila är det filtrerade värdet.

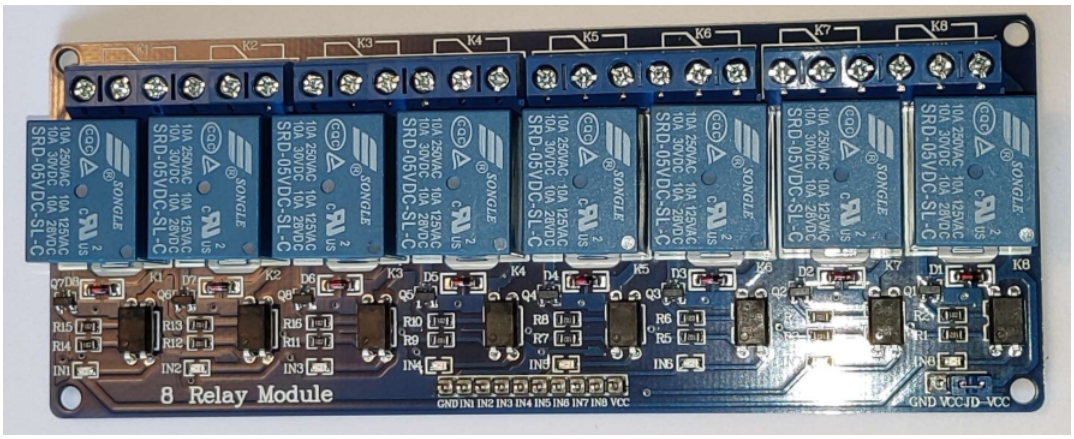


Figur 2.7: 1D Kalmanfiltrering.

2.8 Relä

Ett relä är i grunden en elektrisk strömbrytare som styrs av en yttre insignal. Det fungerar genom att använda en elektromagnet för att antingen öppna eller stänga en kontakt. När strömmen flödar från in-signalen så bildas ett magnetfält som antingen repellerar eller attraherar en metallarmatur som rör sig mellan de två kontakterna normalt öppen (NO) eller normalt stängd (NC). Dessa kontakter är de två olika stadierna som relät kan ha. NC kontakten är standardläget för relät och dit kretsen är kopplat, NO kontakten är dit relät kopplas när elektromagneten attraherar metallarmaturen. Genom att styra magneten så är det då möjligt att styra denna strömbrytare rent elektrisk samtidigt som in-signalen är isolerad från lasten.

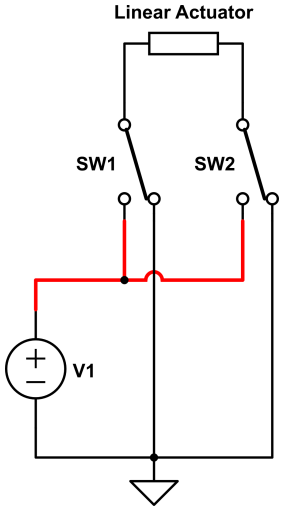
I figur 2.8 nedan ses den relämodul, som kommer användas i arbetet, som består av 8 reläer. Dessa är fastmonterade på ett och samma kort för att förenkla arbetet. Varje relä har 3 stycken ingångar och en utgång: styrsignalen från Arduino IN_X där x symboliserar vilket relä på kortet signalen går till, GND, VCC samt utsignalen V_o. I figuren ser vi att "ovanför" varje relä (de stora blåa komponenterna) sitter tre kontakter med dragna vita linjer. Dessa indikerar vilken av kontakterna som är NC eller NO. I en NC-kontakt är kontakten normalt stängd vilket innebär att kretsen är sluten via den kontakten och kan leda ström då styrsignalen ej är aktiverad. För NO-kontakten gäller motsatsen, kretsen är bruten via denna kontakt tills att styrsignalen aktiveras och magnetfältet attraherar metallarmaturen.



Figur 2.8: x8 kanal DC 5 V relämodul.

2.9 H-brygga

En H-brygga är en elektronisk krets för att styra riktningen hos en ström genom en belastning, vanligtvis en motor. Det finns olika konfigurationer av H-bryggor och en uppsättning är att använda två relän vilket visas i figur 2.9. Reläets NC kontakter är kopplade till GND och NO är kopplade till VCC. I bilden är båda reläer anslutna till GND och motorn används ej.



Figur 2.9: Exempelkrets H-Brygga.

I det första scenariot är SW1 är stängd och SW2 öppen. Det innebär att VCC flödar genom SW1, motorn, SW2 och sist till jord.

I det andra scenariot är SW1 öppen och SW2 stängd. Vilket innebär att VCC flödar genom SW2, motorn, SW1 och till sist jord. På så sätt har polariteten växlats.

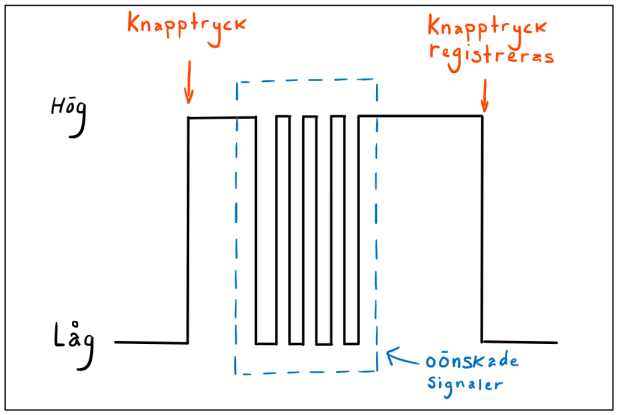
2.10 Tryckknapp

För att styra elektroniska system med Arduino är ett vanligt alternativ tryckknappar. Tryckknappar fungerar genom att vanligtvis vara en bruten krets, när knappen sedan trycks ner sluts kretsen så att ström kan flöda. Exempelvis kan en tryckknapp se ut som i figur 2.10.

Figur 2.10: Framsida/baksida av tryckknapp.

2.11 Debounce

Vid användning av knappar i projektet med Arduino så är det viktigt att en algoritm som kallas ”Debounce” implementeras för att undvika falska tryckningar [13]. I figur 2.11 visas signal registeringen hur en mekanisk knapp kan studsa mellan kontaktytorna VCC och GND när den aktiveras. Studsandet innebär att en knapptrycking kan generera flera signaler som registreras som knapptryck till Arduinon även om användaren endast har tryckt en gång.



Figur 2.11: Signalregistrering vid ett knapptryck.

Debounce är därmed en algoritm som har i uppgift att säkerställa att de oönskade signaler som registreras blir ignorerade av Arduinon. Algoritmen lägger till en fördröjning hos insignaler från knappar till Arduino:n efter att ett första knapptryck registrerats. Under denna tiden så filtreras därmed alla ytterligare signaler så att Arduinon endast registrerar en stabil och korrekt signal.

3
Metod

För att på ett enklare sätt genomföra projektet så delades det upp i två delar. Dessa delar var hårdvara och mjukvara där mjukvarudelen i sig var uppdelad i fem delar.

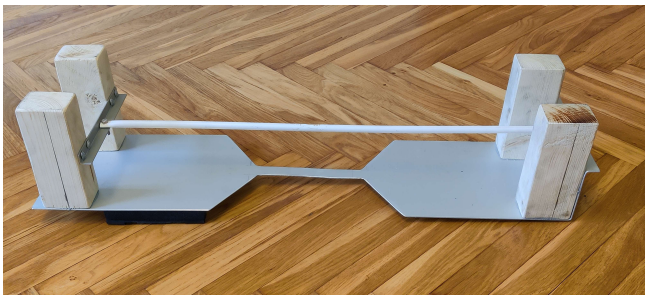
Hårdvaran bestod av de komponenter som beskrevs i kapitel 2. Utefter detta indelades arbetet i följande delar:

- Kretsschema
- Montering av komponenter
- Koppling av krets

Mjukvaran delades upp, som tidigare nämnt, i fem olika delar. Dessa delar består utav de större funktionerna som programmet kommer använda sig av och är följande:

- Inställningar / Förenklingar
- Huvud-Loop
- Automatisk Styrning
- Manuell Styrning
- Sensoravläsning / datahantering

Projektet delades upp i dessa delmålen med intentionen att det delvis skulle bli lättare att bygga upp mjukvaran men också att det skulle underlätta att arbeta parallellt på programmeringen. Då varje funktion kan fungera självständigt så delades projektet upp på detta sättet. Det här var tankesättet genom hela planeringen av arbetet, dock behövde hårdvaran färdigställas innan mjukvaran kan påbörjas. För att kunna säkerställa att systemet fungerar som det ska tillverkade också projektets handledare från Groth Kaross AB en modell, se figur 3.1 nedan, som ska imitera den verkliga trailern i skala 1:10.



Figur 3.1: En nedskalad modell av en trailer utan några komponenter.

3.1 Val av komponenter

I denna sektionen motiveras valet och funktionen för de komponenter som användes i projektet.

3.1.1 Arduino MEGA

Arduino Mega kommer att fungera som systemets centrala enhet, med ansvar för att kontrollera och reglera positionen för stödbenen, hantera indikationssystemet med hjälp av lysdioderna i tryckknapparna, samt läsa av lutningsgivare och tryckknappar. I projektet användes en nätadapter för att förse Arduinon med ström, men i praktiken kommer istället en switchregulator användas.

3.1.2 Avståndsmätaren

Avståndsmätaren kommer att användas för att uppskatta avståndet till marken då dragbilen har kopplats loss. Avståndet kommer sedan bli ett börvärde för trailern då den ska återgå till markparallelt läge för dragbilsanslutning. För att stabilisera strömförsörjningen till avståndssensorn sattes en kondensator på 10 µF enligt rekomendation från tillverkaren [14].

3.1.3 Switchregulatorn

Switchregulatorn kommer användas för att spänningsmata båda reläbräden samt de linjära ställdon från en 24 V matning till 5 V.

3.1.4 Linjära ställdon

Det finns flera versioner av linjära ställdon som kan ge mer information gällande nuvarande position och andra egenskaper. Dock för att efterlikna det hydrauliska systemet som inte har dessa funktioner, så kommer denna version med endast VCC och GND att användas. Dessa ska ersätta de hydrauliska stödbenen för detta projekt.

3.1.5 MPU-6050

MPU-6050 har används brett inom många olika applikationer exempelvis inom drönarteknik, kontrollsystem och VR. Den erbjuder en väldigt kompakt storlek, låg energiförbrukning, stor tillgänglighet och anses enkelt att använda vilket motiverar valet. I detta projekt kommer MPU-6050 användas för att förse Arduino med information om trailerns orientering.

3.1.6 Relä

I projektet kommer två stycken x8 relämoduler användas. Den ena modulen kommer möjliggöra styrning av de linjära ställdon, samt den andra kommer användas för att styra LED belysningen i tryckknapparna.

3.1.7 Knappar

För att kunden ska kunna styra systemet behövs någon form av manövrering vilket i detta fallet är fem stycken tryckknappar. I figuren 2.10 visas det på knappens baksida att det finns tre ingångar samt tre utgångar. Tryckknappen följer samma logik för NO samt NC, som för reläerna i sektion 2.8. Kontakterna i mittten styr LED belysningen med en matningsspänning på 24 V DC.

3.1.8 Falling Edge

När tryckknappar används kan systemet registrera en nedtryckning på två olika sätt, kända som "Falling Edge" och "Rising Edge", vilka faller under samlingsnamnet Signal Edge Detection [15]. Skillnaden mellan dem är att knapptrycket registreras antingen när knappen trycks ner (rising edge) eller när knappen släpps (falling edge).

Både falling edge och rising edge har sina fördelar och nackdelar, men i systemet valdes falling edge. Nackdelen med denna variant är att den kan upplevas som mindre intuitiv för användaren eftersom ingenting händer förrän användaren släpper knappen.

4
Säkerhetsfunktioner

För att systemet ska kunna köras utan att bedömmas som en fara så har det implementerats en del säkerhetsfunktion i mjukvaran.

4.1 Tid

I samtliga processer som involverar stödbenen så har det implementerats en tidsfunktion. Denna tidsfunktion ska säkerställa att stödbenen stängs av inom en säker tidsram ifall en sensor skulle gå sönder eller om andra fel skulle uppstå.

4.2 Enablesignal

Spänningsmatningen till reläbrädet som styr stödbenen styrs av en så kallad enable signal. Enable signalen medför att för att stödbensreläet ska ha någon spänning och därmed kunna styra stödbenen, så måste enable signalen vara hög (5 V). Ifall signalen är låg eller saknas kommer reläbrädet inte ha någon spänningsmatning vilket sätter stopp för oavsiktlig användning av stödbenen.

4.3 Feltryck

Systemet har en inbyggd sekvens som kräver att varje steg utförs i rätt ordning för att undvika oavsiktliga eller felaktiga knapptryck. Sekvensen är enligt följande som visas i tabell 4.1:

Tabell 4.1: Felttryckssekvens.

Process som får köras	**Om senaste processen var**
Touch-Down	Raise, Manuellt eller Touch-Down
Auto-Level	Raise, Manuellt eller Trailer Connect
Trailer Connect	Auto-Level
Raise	Raise, Manuellt eller Trailer Connect

4.4 Pausfunktion och nödstopp

När en process har startats finns det möjlighet till en pausfunktion som låter användaren pausa processen genom att trycka på samma knapp igen. Om knappen sedan

trycks ned ytterligare en gång så återupptas processen därifrån den avbröts. Det är viktigt att poängtera att denna pausfunktionen inte ska användas vid en nödsituation. Ifall en farlig situation uppstår så ska nödstoppet brukas istället eftersom nödstoppet bryter spänningen till Arduinon vilket garanterar att alla funktioner omedelbart stoppas.

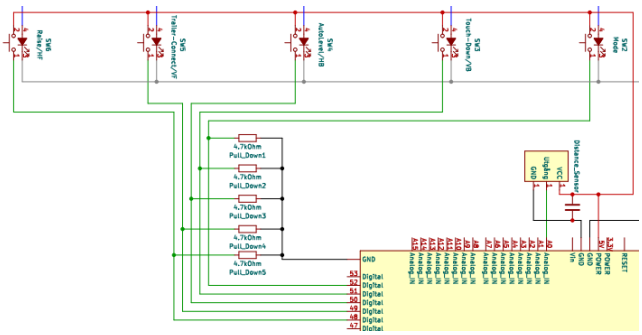
En nackdel med nödstoppet är att Arduinons minne raderas då spänningen bryts. En omstart innebär att användaren kommer behöva köras stödbenen manuellt för att sedan återuppta processen från ett säkert läge.

5
Elektriska kopplingar

I detta kapitel kommer de viktigaste delarna av kretsschemat att förklaras i mer detalj. För den fulla bilden av kretsschemat se bilaga A.1.

5.1 Tryckknappar

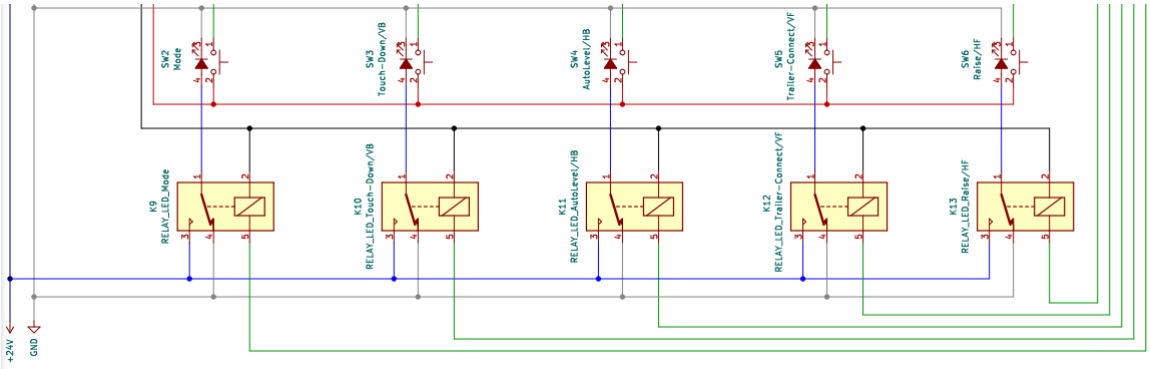
För att läsa av ifall en knapp har tryckts ned eller inte så kopplas reläets NC port till Arduinons GND och NO till 5 V. Denna koppling innebär att Arduinon läser av GND när knappen inte är nedtryckt och 5 V när knappen är nedtryckt. Systemet fungerar i en ideal värld men i verkligheten så kan spänningen vara någonstans mellan 0 och 5 V. Osäkerheten kan göra så att Arduinon inte vet ifall knappen är nedtryckt eller inte och kan därför registrera felsignaler. För att undvika detta så används en pull-down resistor, som visas i figuren 5.1 nedan. En pull-down resistor används för att säkerställa att en ingång på en krets hålls vid en stabil logisk nivå som är låg eller noll när det inte finns någon ingångssignal. Resistorn kopplas därför mellan ingången och kretsens jord. Ifall kretsen utsätts för störningar så kan ingången hamna i ett osäkert mellanläge mellan hög och låg signal. Detta kan leda till att Arduinon felaktigt tolkar signaler eller inte läser av dem alls. I projektet används därmed pull-down resistorn för att se till att Arduinon läser en låg signal så länge som knappen inte är nedtryckt. Valet av resistansvärde för pull-down resistorn påverkar hur kretsen fungerar. Om resistansen är för hög, kan det bli svårt för strömmen att flyta genom resistorn och då kan signalläget bli ostabilt. Om resistansen är för låg, kan det orsaka överbelastning och skador på kretsen. Då spänningen är 5 V är ett bra startvärde på resistorn 4,7 kΩ[16].



Figur 5.1: Kretsschema för tryckknapparna.

5.2 LED

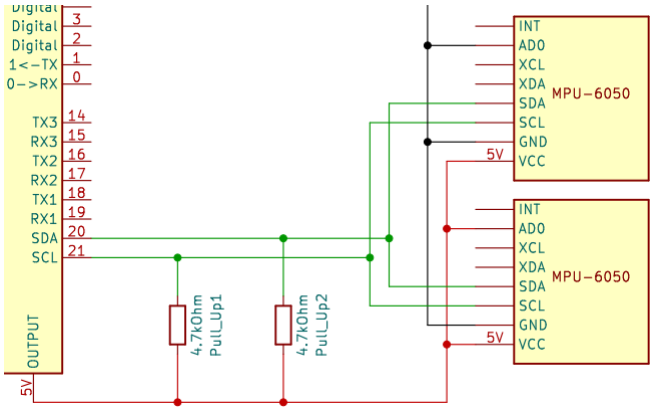
Det sitter lysdioder inbyggt i tryckknapparna, de kommer att användas för att indikera vilken senaste knapptryckningen var, vilken process som körs, om programmet är pausat och vilket läge sytemet är i. Dessa behöver styras via 24 V för modellen av tryckknapp som används i detta projekt. Kontrollenheten, Arduinon, kan endast arbeta med upp till 5 V, vilket innebär att Arduinon inte kan styra LED:erna direkt. Därför används reläer som möjliggör att styrenheten kan kontrollera lysdioderna. Arduinons styrsignal kopplas till relät, som i sin tur har sin NC kontakt till GND och NO kontakt till 24 V. Lysdioderna styrs därmed genom att skicka en 5 V signal till reläet som då kör igång lysdioden. För att undvika en potentialskillnad, att strömmar går mellan olika jordningspunkter, har lysdioderna en skild jord från Arduinon, då 24 V matningen kommer från en annan källa, se figur 5.2 nedan.



Figur 5.2: Kretsschema för LED:erna.

5.3 MPU

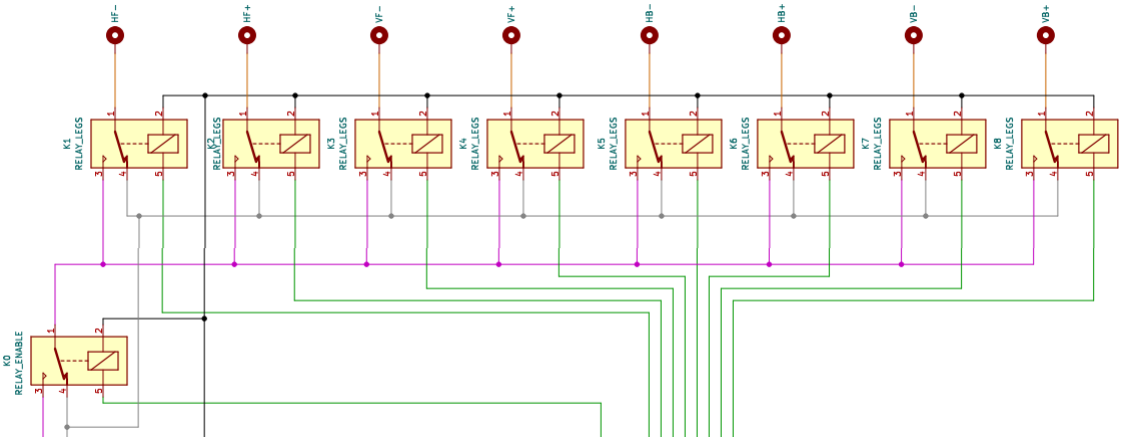
Eftersom I2C-kommunikation används måste SCL och SDA kontakterna på MPU:erna kopplas ihop med Arduinons motsvarande kontakter. Utöver kopplingen måste också enheterna få spänningsmatning via Arduinons VCC samt jordning via GND. För att möjliggöra I2C kommunikationen mellan tre enheter måste också varje enhet ha varsin adress. Som tidigare nämnt, också i kapitel 2.6, kan MPU:n ha en adress på antingen 0x68 eller 0x69. Detta bestäms av kontakten AD0 på enheten, ifall denna kontakt kopplas till VCC blir adressen 0x69 och ifall den kopplas till GND blir adressen 0x68. Därför har dessa kontakter kopplats olika på båda enheterna. Av samma anledning som med Pull-Down resistorerna för tryckknapparna i kapitel 5.1 behövs även här en liknande resistor. Dock eftersom I2C kommunikationen är mestadels 5 V så behövs en pull-up resistor istället som ser till att nivån är 5 V, se figur 5.3 nedan.



Figur 5.3: Kretsschema för MPU:erna.

5.4 Linjära ställdon

För att möjliggöra styrningen av de linjära ställdonen, eller benen, i denna krets behöver polariteten kunna växlas om vid matningen till benen. Då benen styrs genom att kontrollera polariteten på spänningen kan kopplingen avgöra om en positiv spänning ska dra in benen eller utvidga dem. I detta projekt används en positiv spänning för att utvidga benen och en negativ spänning för att dra in dem igen. Styrningen av benen kräver mer ström än vad en Arduino kan förse och därför kommer benen att styras av reläer som sedan styrs av Arduinon. För att möjliggöra polaritetsväxlingen kopplas ett relä till en kontakt till ställdonen. Då kretsen inte har en negativ spänning kommer varje ställdon att behöva två stycken reläer som bestämmer polariteten. Utöver detta finns även säkerhetsfunktion enable signal, se kapitel 4, i form av ett relä som måste vara aktiverat för att de linjära stödbenen ska få någon spänning. Även denna jordning måste vara skild från Arduinons av samma anledning som för lysdioderna 5.2, se figur 5.4 nedan.



Figur 5.4: Kretsschema för kontroll av ställdon.

6
Beskrivning av mjukvara

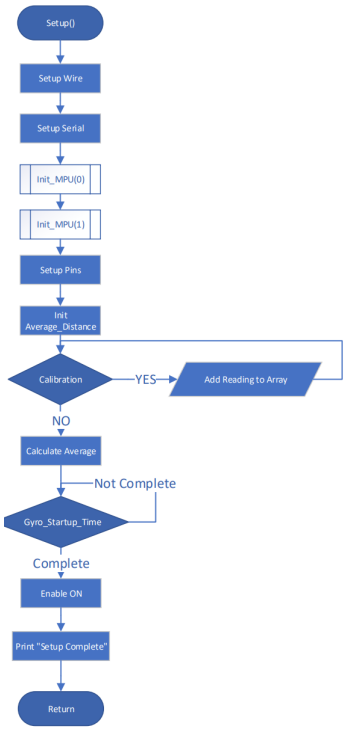
I detta kapitel beskrivs mjukvaran i detalj gällande samtliga funktioner och sedan justerbara inställningar.

6.1 Funktioner

I denna delen beskrivs samtliga funktioner som använts i systemet.

6.1.1 Setup

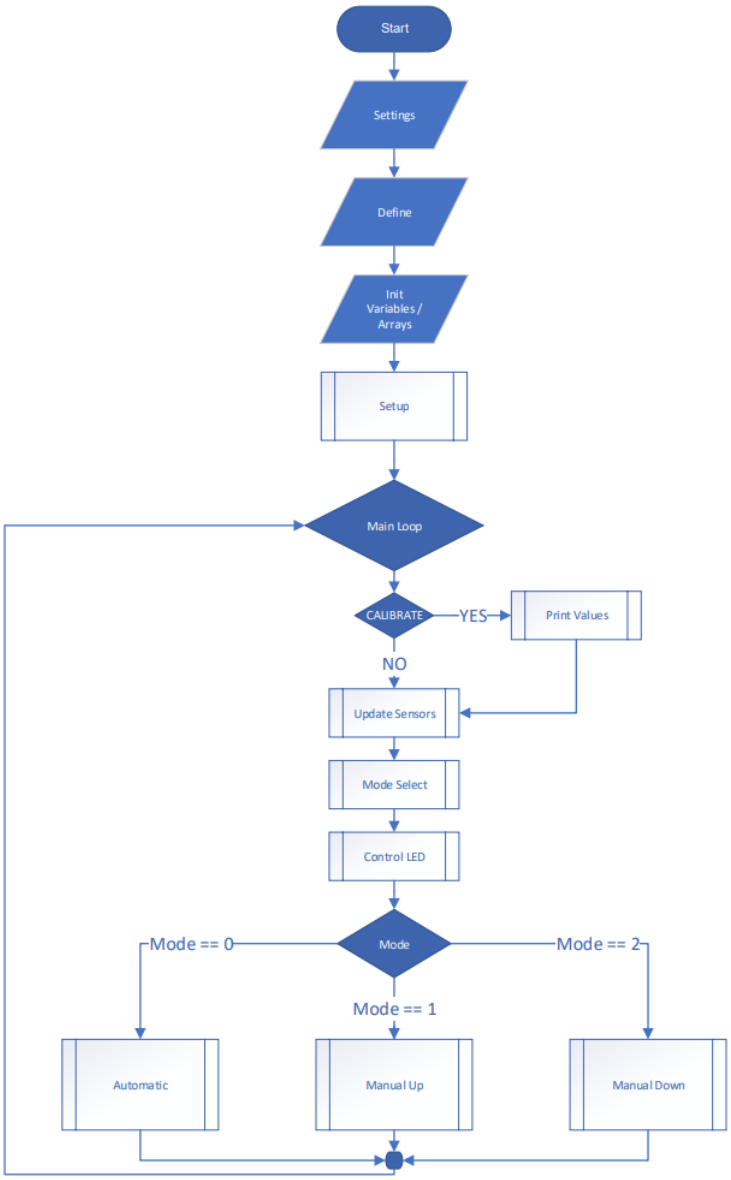
Funktionen i figur 6.1 är en av två som är en del av strukturen i Arduino språket [17] och kallas endast en gång, när Arduinon startas up. Funktionen används, bland annat, i detta system för att initiera de olika in och utgångarna på Arduino Mega. Funktionen startar också upp både Serial och I2C kommunikationen med MPU:erna och därefter startas avståndssensorn. Innan Setup släpper programmet vidare inväntar den även att MPU:erna ska stabilisera sig enligt Gyro_Startup_Time 6.2.6.



Figur 6.1: Flödesschemat för Setup funktionen.

6.1.2 Loop

Figur 6.2 är den andra funktionen som är en del av strukturen i Arduino språket [18]. Efter att Setup har körts, går programmet in i denna funktion som innehåller en oändlig loop som upprepas om och om igen.

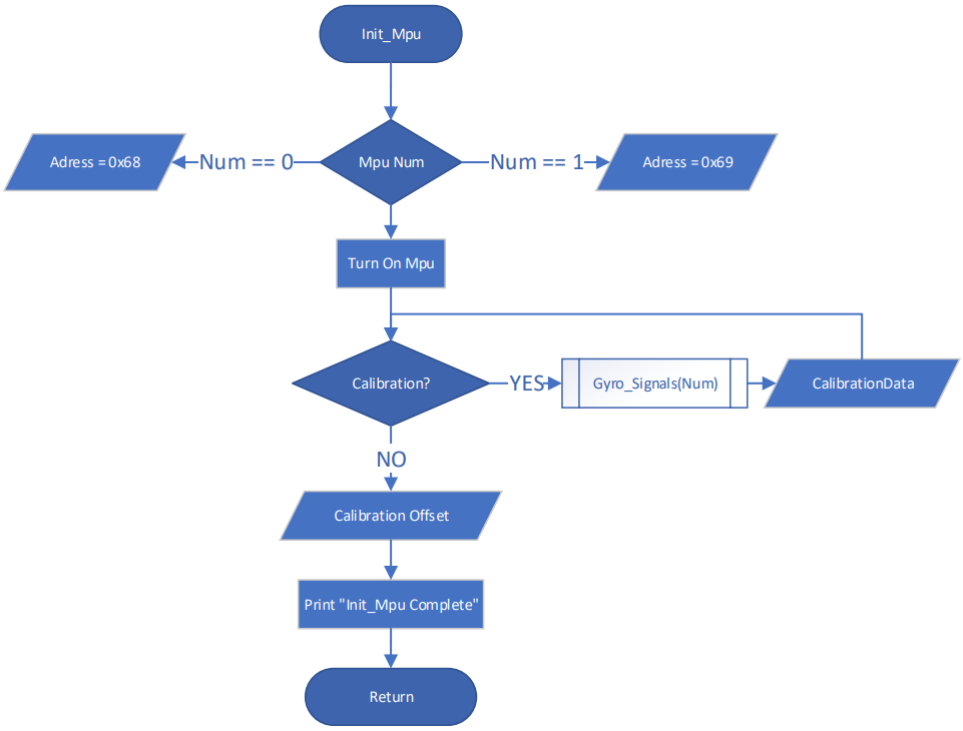


Figur 6.2: Flödesscehma för mjukvarans huvud-loop.

6.1.3 Init_MPU

Funktionen i figur 6.3 används för att starta upp kommunikationen med MPU:erna samt att kalibrera felmarginalerna i gyroskopen. När funktionen kallas så skickar den ett kommando till MPU:n som startar upp enheten. Sedan läser funktionen av 2000 värden i alla axlar från MPU:n för att få ett medelvärde av felmarginalen då

MPU:n ligger stilla. Dessa världen sparas och kommer sedan att användas för att få ett mer exakt värde.



Figur 6.3: Flödesschema för Init_MPU funktionen.

6.1.4 Gyro_Signals

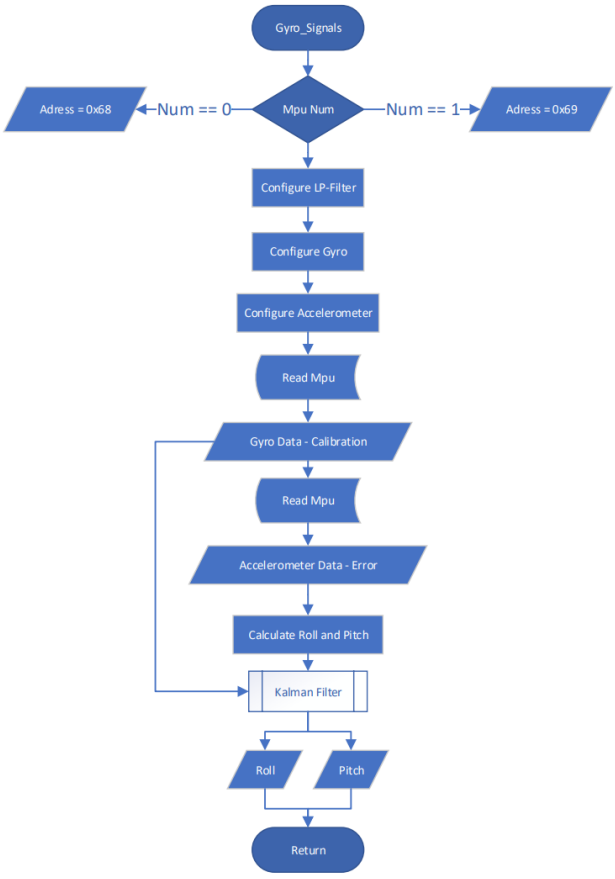
Det är denna funktion som sköter all kommunikation med systemets MPU:er. Funktionen kallar på MPU:erna för att få data skickad till sig och därmed läsa av vinklarna som trailern just nu har. Denna funktion bör därför kontinuerligt anropas. Funktionen fungerar genom att först skriva inställningar till MPU:erna. Till exempel antalet grader per sekund som gyrot ska läsa av och hur känslig accelerometern ska vara. I detta system valdes 500°/sekund för gyrot och 16.4 LSB/°/sekund.

För att läsa av datan måste programmet skriva vilken adress MPU:n ska hämta datan ifrån och för att ta reda på dessa adresser används MPU:ns Register Map [8]. Register mappen innehåller alla adresser för var informationen finns sparad på MPU:en. Informationen, alltså exempelvis gyrodatan i x-axeln, som finns på MPU:erna består av 2 bytes var. Därför måste Arduinon göra två avläsningar från MPU:n och sedan lägga ihop dessa. Det betyder att det första värdet som kommer in vänstershiftas åtta platser för att göra rum för de nästa åtta och sedan läggs nästa åtta bitar in på de lediga platserna. Denna process görs sex gånger för varje MPU då det finns tre axlar och i varje axel finns gyroskopets data och accelerometerdata.

Efter processerna räknas vinklarna ut från accelerometern genom trigonometriformlerna (6.1) och (6.2) nedan. Det sista som denna funktionen gör är att skicka gyrodatan och accelerometervinklarna till kalmanfiltret och filtret lägger ihop denna data för att filtrera ut störningar, se kapitel 6.1.5 nedan.

$$\theta_{roll} = \arctan(\frac{Acc_Y}{\sqrt{Acc_X^2 + Acc_Z^2}}) \tag{6.1}$$

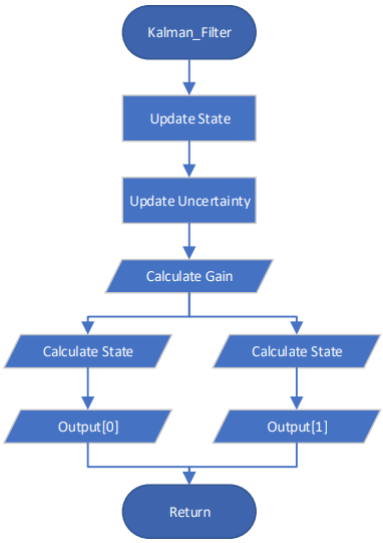
$$\theta_{pitch} = \arctan(\frac{-Acc_X}{\sqrt{Acc_Y^2 + Acc_Z^2}}) \tag{6.2}$$



Figur 6.4: Flödesschema för Gyro_Signals funktionen.

6.1.5 Kalmanfilter

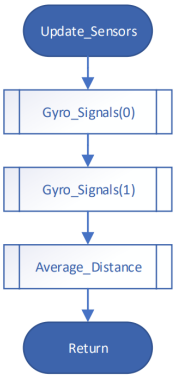
Som tidigare nämnt i kapitel 2.7 används kalmanfiltret för att göra en uppskattning om vad värdet bör vara i förhållande till faktiska mätvärdet. Funktionen har fyra ingångar, det uppskattade värdet, osäkerheten i uppskattade värdet, gissningen av hur värdet bör förändras och det faktiska mätvärdet. Mätvärdet som skickas in i filtret är både pitch och roll-vinklarna, en i taget, och skickas då tillbaka till Gyro_Signals, se kapitel 6.1.4.



Figur 6.5: Flödesschema för kalmanfiltret.

6.1.6 Update_Sensors

Denna funktion i figur 6.6 har i uppgift att uppdatera samtliga sensorvärden. I de olika processerna behöver flera sensorer uppdateras samtidigt, därför skapades denna funktion som underlättar användandet och förståelsen för koden.



Figur 6.6: Flödesschema för Update_sensor funktionen.

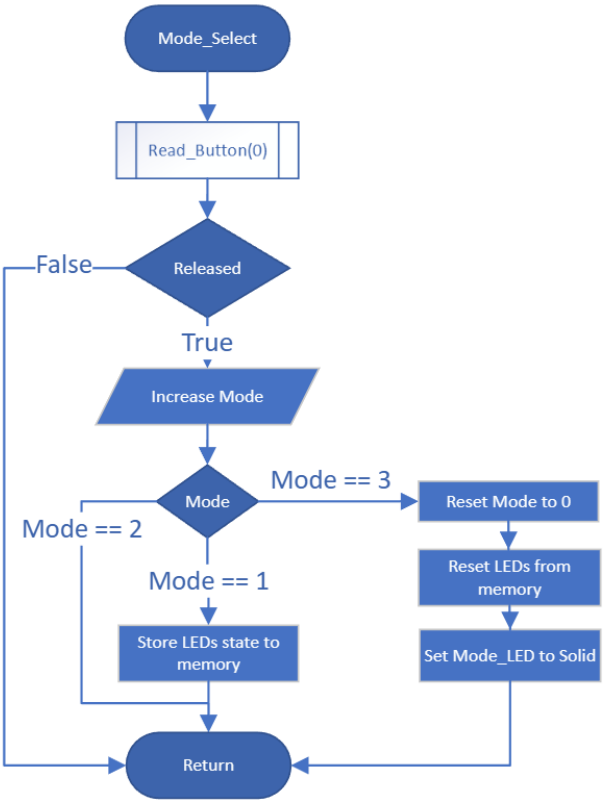
6.1.7 Mode_Select

Den här funktionen ansvarar för att hantera Mode-variabeln vilken bestämmer det aktuella läget i programmet. Programmet kan befinna sig i ett av de tre olika lägen: Automatiskt (Mode-variabel är lika med 0), manuellt ned (Mode-variabel är lika med 1) eller manuellt upp (Mode-variabel är lika med 2).

I figur 6.7 visas flödesschemat för funktionen. Först så kontrolleras om Mode-knappen har släppts via Read_Button(0) funktionen. Ifall knappen nyligen har

släppts så inkrementeras värdet på Mode-Variablen med 1. De olika moderna är enligt följande:

- När Mode-variabeln är lika med 1 så sparas flaggorna för lysdioderna i ett minne. Detta görs eftersom när programmet sedan återgår till det automatiska läget så ska programmet memorera vilken som var den senaste processen och indikera det via lysdioderna.
- Då Mode-variabeln är lika med 2 så behövs inte några ytterligare funktioner göras.
- Ifall Mode-variabeln överskrider 2, så återställs Mode-variabeln till 0. Därmed återgår programmet till automatiskt läge och lysdiodernas flaggor uppdateras från minnet. Tillslut sätts även flaggan för lysdioden till 1 vilket innebär att den ska lysa konstant.



Figur 6.7: Flödesschema för Mode_Select.

6.1.8 Control_LED

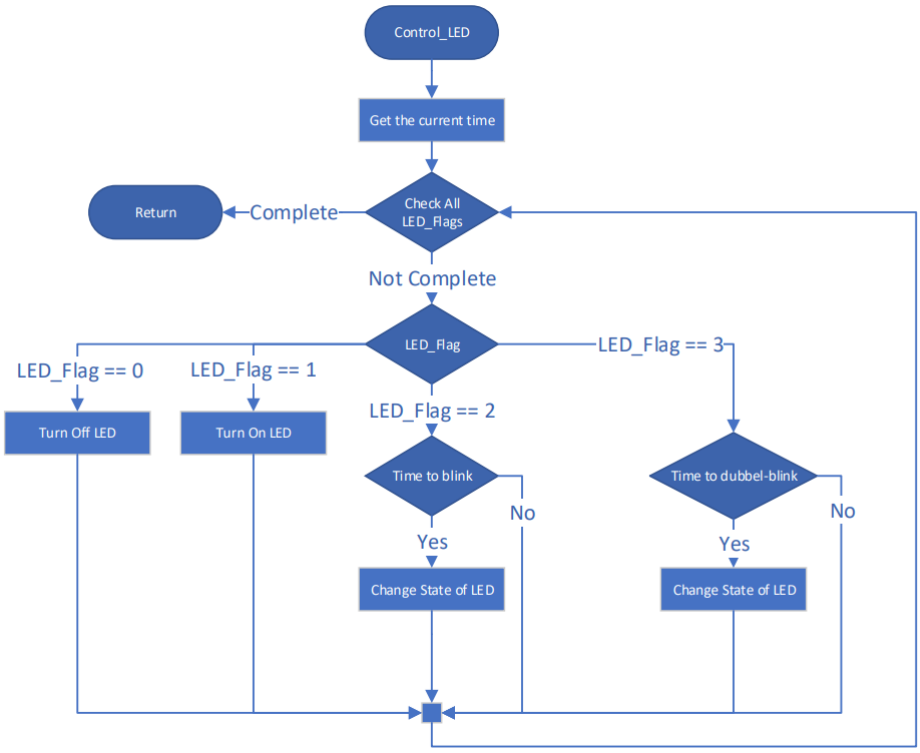
Den här funktionen styr flaggsystemet för lysdioderna i detta system. Varje lysdiod representeras av ett element i arrayen LED[] som har valts att kallas för LED-flagga, utifrån värdet av respektive LED-flagga så agerar lysdioderna enligt följande:

- Om LED-flaggan är satt till 0 (Avstängd), kommer lysdioden att stängas av.

- Om LED-flaggan är satt till 1 (Tänd), kommer lysdioden lysa konstant.
- Om LED-flaggan är satt till 2 (blinkande), kommer lysdioden blinka med ett inställd tidsintervall.
- Om Led-flaggan är satt till 3 (Dubbelblinkande), kommer lysdioden att blinka två gånger per tidsintervall.

I figur 6.8 visas flödesschemat för funktionen. En for-loop går igenom samtliga LED-flaggor och hanterar dessa därefter enskilt. För att blinka lampan med ett givet intervall (LED_Flag == 2) används funktionen millis vilken är en inbyggd funktion i Arduinobiblioteket. Funktionen hämtar tiden för när funktionen körs, därefter jämförs värdet med den nuvarande tiden för att kontrollera om tillräckligt med tid har gått. Tidsintervallet justeras då av inställningen Blink_Interval vilket nämndes i 6.2.5 och ifall tillräckligt med tid har gått så ändras LED-statusen.

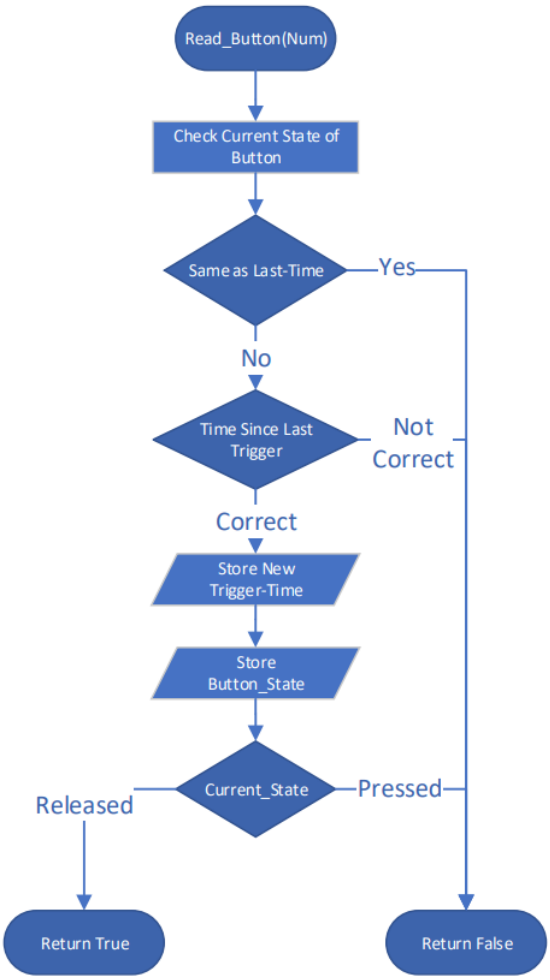
För dubbelblinkning (LED_Flag == 3) så används även millis, här krävs dock hjälp från en räknarvariabel vid namn Blink_Count. Variabeln används för att räkna antalet iterationer och där LED-lampan slås på eller av beroende på i vilken iteration som räknaren befinner sig i. Om Blink_Count är 1 eller 3 så sätts lysdioderna igång och i fall Blink_Count är 2 eller 4 så släcks lysdioderna. Efter Blink_Count har nått 4 så väntar programmet tills det har gått 8 iterationer där den sedan återställs till 0 och blinkandet börjar om från början. Denna tid mellan iteration 4-8 blir då tiden mellan varje dubbelblink.



Figur 6.8: Flödesschema för Control_LED.

6.1.9 Read_Button

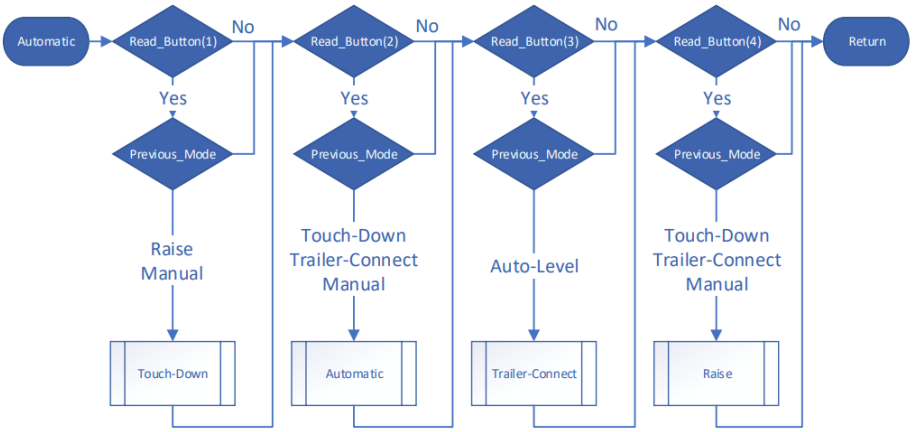
Denna funktion används av programmet för att avgöra om en falling edge har registrerats. Flödet av funktionen kan följas i figur 6.9 nedan. Funktionen fungerar genom att när programmet kallar på den så anges också argumentet som symboliserar vilken knapp funktionen ska avläsa. Funktionen börjar med att läsa av värdet på knappen för att avgöra om den är nedtryckt eller inte, och sparar sedan värdet. Sedan jämförs det nuvarande läget med vilket läge knappen hade förra gången den lästes av och ifall dessa två är olika betyder det att det har skett en statusändring i knappen. Efter statuskontrollen måste också programmet kontrollera så att det faktiskt är en statusändring eller ifall det är debounce, som tidigare nämnt i kapitel 2.11. I det fall statusändringen är efter tidsgränsen så sparas den nya tiden ner och den nya statusen registreras för knappen. Därefter då den nya statusen är hög betyder det att knappen precis tryckts ner och då returnerar funktionen false. Däremot ifall den nya statusen är falsk så betyder det att knappen precis har släppts och funktionen returnerar sant. Alltså, ifall knappen endast hålls ner kommer inget att hända utan systemet väntar på att knappen släpps.



Figur 6.9: Flödesschema för Read_Button.

6.1.10 Automatic

Automatic funktionen består av de fyra olika autonoma processerna Touch-down, Autolevel, Trailer-Connect och Raise. Funktionens uppgift är att välja en av de fyra funktionerna baserat på knapptryck och det senaste läget i minnet. I flödesschemat i fig 6.10 så kontrolleras först om en knapp har tryckts ned. Ifall en knapp har tryckts ned så kontrolleras det först att det senaste läget är godkänt enligt sekvensen som beskrevs i 4.3. Om sekvensordningen är godkänd fortsätter programmet i respektive funktion.



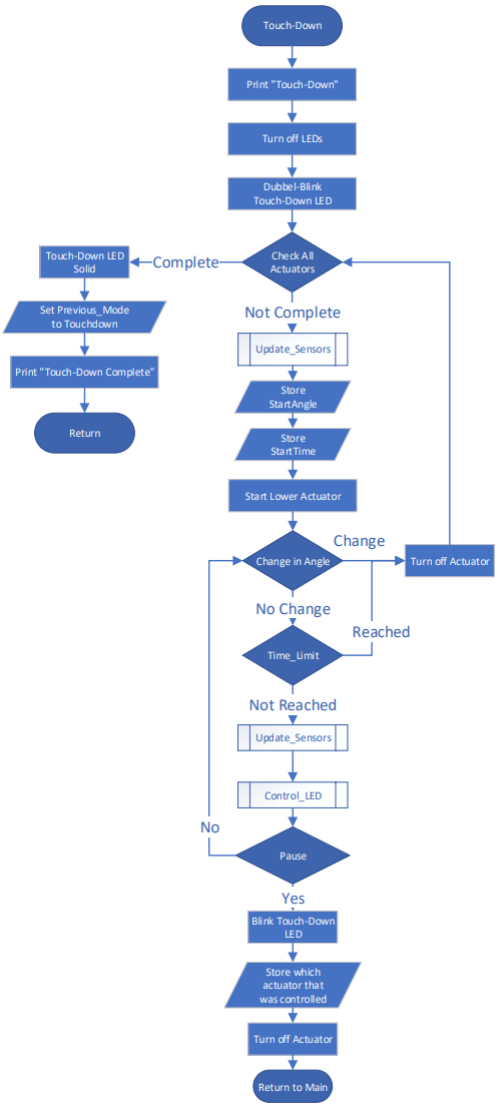
Figur 6.10: Flödesschema för Automatic funktionen.

6.1.11 Touch-Down

I flödeschemat i figur 6.11 börjar funktionen Touch-Down med att stänga av samtliga LEDs och för att visa att en process pågår så flaggas lysdioden för dubbelblinkning. Funktionen går sedan igenom samma följande procedur för samtliga stödben tills att de är klara.

Variabeln StartAngle används för att memorera MPU:ernas lutning innan stödbenen börjar köras ut. När stödbenet sedan körs ut och tar emot marken så kommer MPU:erna känna av en viss skillnad i lutning. När denna skillnaden uppstår, där känsligheten bestäms av inställningen 6.2.2, så är det dags att stanna stödbenet. Touchdown sekvensen repeteras då i ordningen: Vänster bak (VB), höger bak (HB), vänster fram (VF), höger fram (HF). När samtliga stödben är på plats så flaggas det korrekt för lysdioden och samtliga ben stängs av.

När pausfunktionen aktiveras i programmet så sparas vilket stödben som senast körde i en variabel som kallas ”Previous_Index”. Variabeln gör det möjligt för användaren att återuppta funktionen där den senast avbröts, istället för att behöva börja om från början.



Figur 6.11: Flödesschema för Touch-Down funktionen.

6.1.12 Autolevel

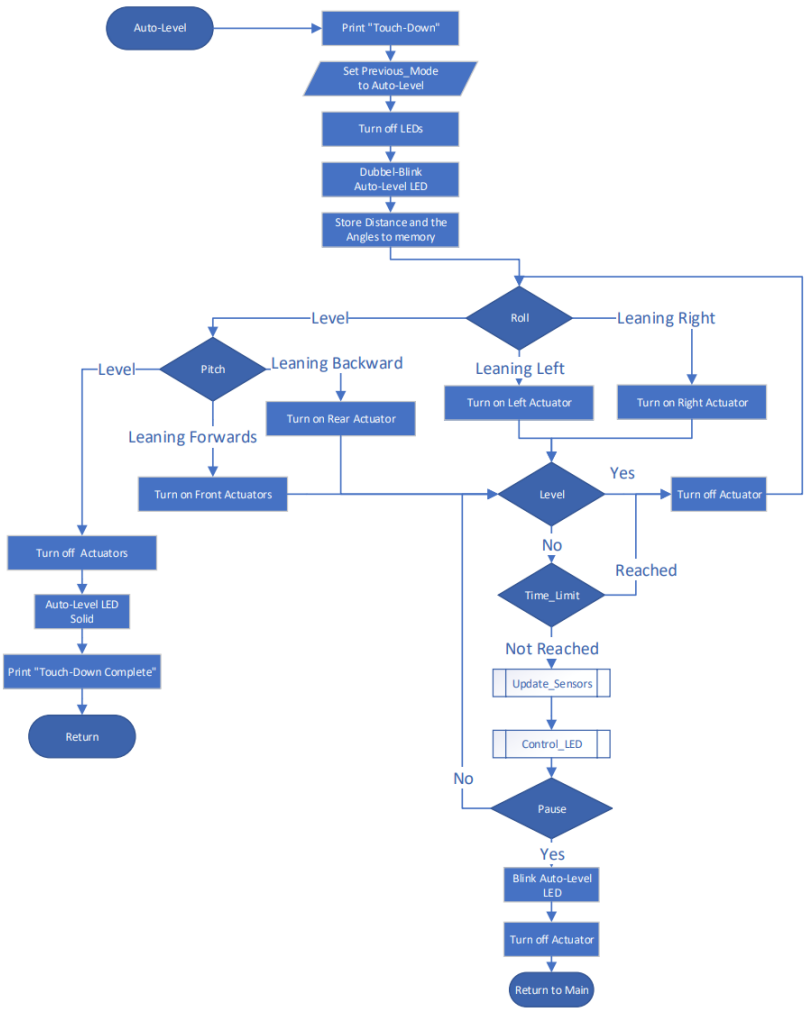
Auto-level i figur 6.12 är den andra processen i automationssekvensen. Den börjar med att flagga lysdioderna för dubbelblinkning och sedan sätter det senaste läget till Auto-Level. Lägesänringen görs eftersom ifall användaren senare i processen vill pausa så blir det tillåtet att autoleveleringen körs igen. Eftersom trailern ska återgå till samma läge som den befinner sig i innan autoleveleringen så sparas de olika sensorvärderna från MPU:erna samt avståndsmätaren i minnet. Sedan är det dags för att justera nivån.

Först kontrolleras rollen, det vill säga om trailern lutar åt vänster eller höger. Ifall trailern låter åt vänster så aktiveras stödbenen på vänster sida av trailern och vice versa för lutning åt höger. Medan stödbenen körs utåt så uppdateras MPU:erna kontinuerligt och körs till att vinkeln blir noll vilket då anses plant.

Efter rollen har justerats, fortsätter Auto-Level processen med en kontroll av pitch, vilket utvärderar om trailern lutar uppåt eller nedåt. Processen hanteras i följande sekvens:

- Om den bakre pitch är negativ, vilket indikerar en lutning uppåt, aktiveras de bakre stödbenen att köras ned tills MPU:n anses plan.
- Om den främre pitch är negativ, vilket indikerar en lutning uppåt, aktiveras de främre stödbenen att köras upp tills MPU:n anses plan.
- Om den främre pitch är positiv, vilket indikerar en lutning nedåt, aktiveras de främre stödbenen att köras ned tills MPU:n anses plan.

I en uppförsbacke är det då önskvärt att först höja den bakre delen och sedan sänka den främre delen ifall det inte är tillräckligt plant. Däremot är det inte möjligt att sänka den bakre delen i en nedförsbacke eftersom hjulen då tar emot. I nedförsbacke blir det då endast möjligt att höja den främre delen för att uppnå en plan yta.

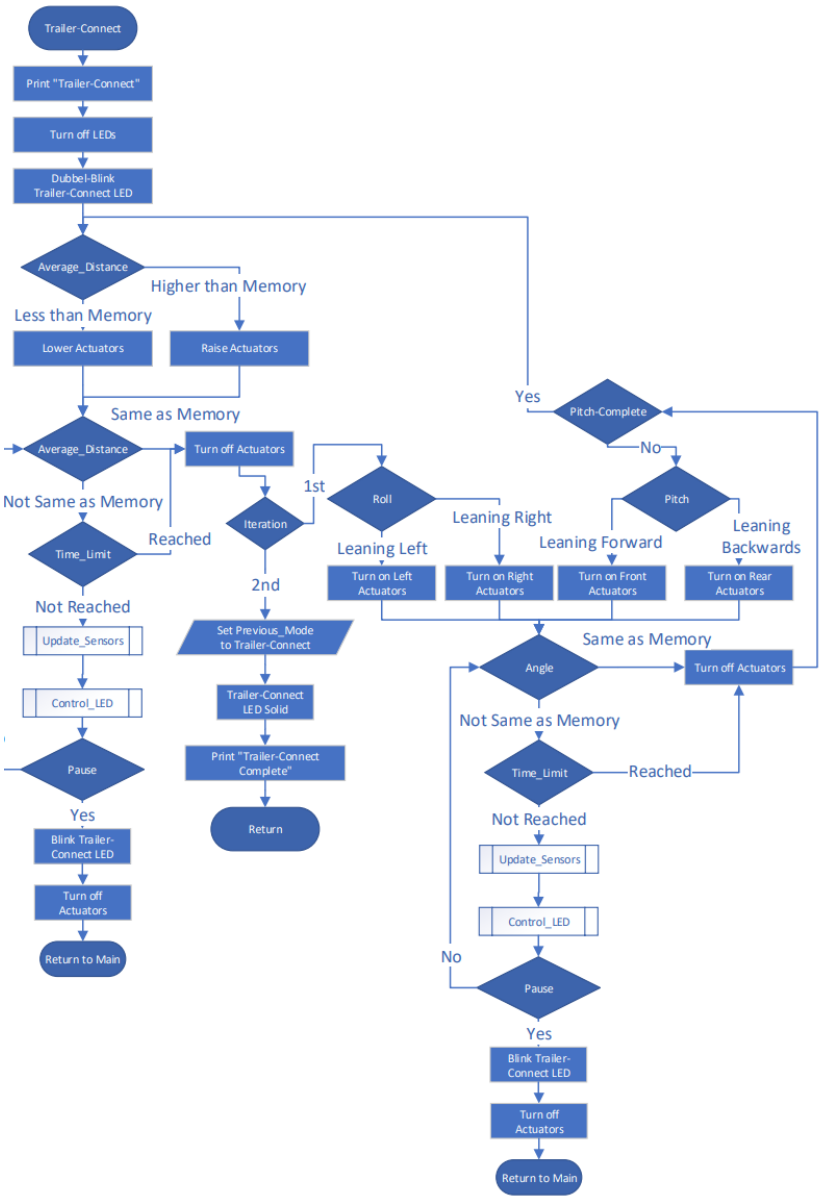


Figur 6.12: Flödesschema för Auto-Level funktionen.

6.1.13 Trailer Connect

Funktionen initieras på samma vis som tidigare processer med flaggning för lysdioderna. Först körs Height_Return funktionen vilket returnerar trailern till rätt höjd. Detta görs eftersom all justering kommer ske genom att köra ned stödbenen och för att då undvika att de når sin maximala längd så körs funktionen.

Därefter justeras rollen enligt MPU-enheterna med hjälp av roll_Return funktionen och efter det justeras pitch i fören med pitch_Return. pitchen korrigeras efter roll_Return funktionen eftersom roll funktionen påverkar pitch:en, men inte tvärtom. Tillslut när pitch:en är klar har höjden förändrats och behöver då återigen justeras till rätt höjd en sista gång via Height_Return.

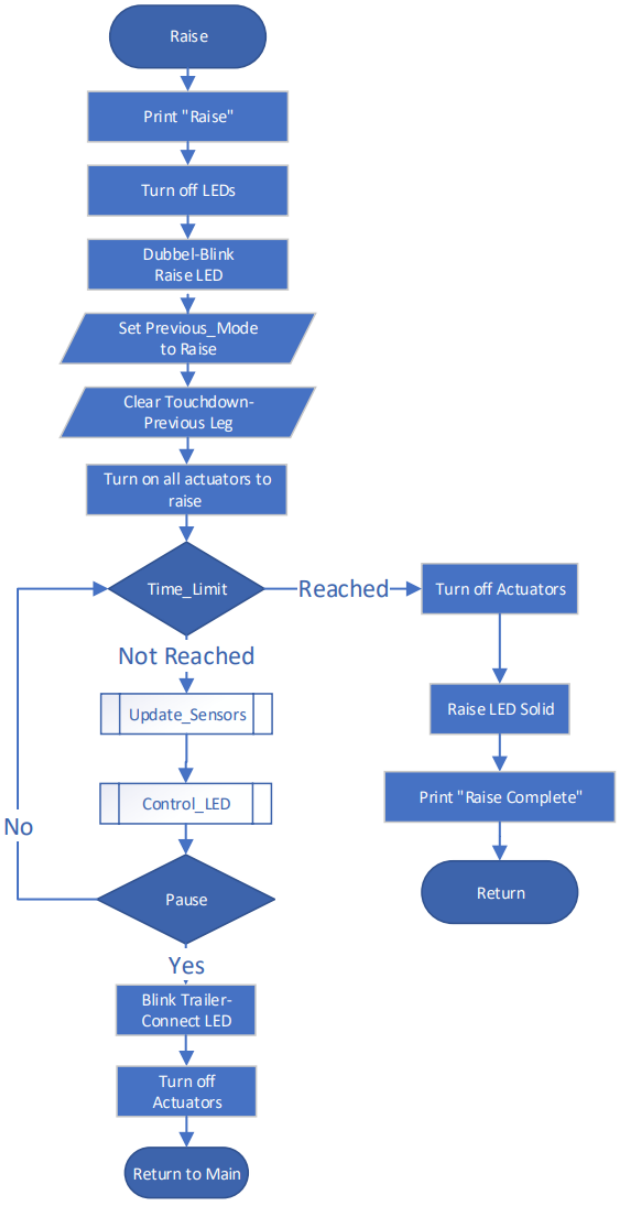


Figur 6.13: Flödesschema för Trailer-Connect funktionen.

6.1.14 Raise

Raise-funktionen, se figur 6.14, är det sista steget i automationssekvensen och har till syfte att säkerställa att stödbenen är indragna innan trailern körs iväg. Indragningen åstadkoms enkelt genom att dra in samtliga stödben.

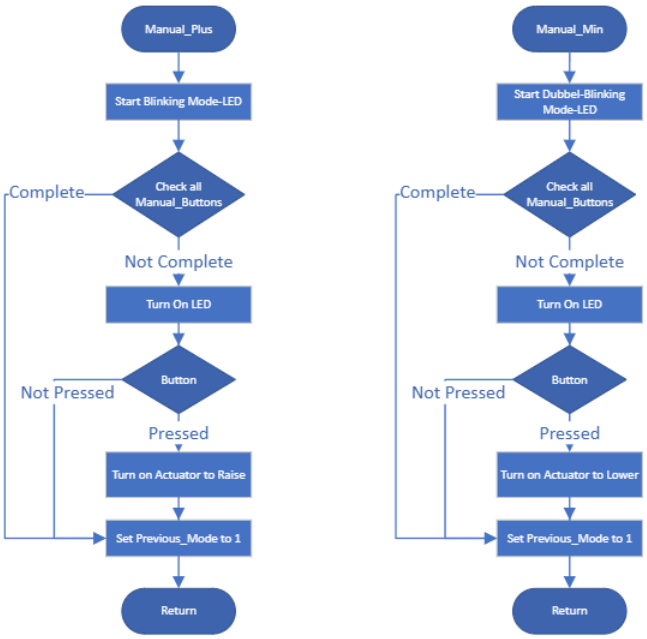
I verkligheten, hos Groth, kommer funktionen att avbrytas på annat sätt än vad som sker i denna laboration. Eftersom det saknas övertrycksgivare, en sensor som mäter och övervakar trycket i ett hydralsystem, så kommer stödbenen att endast baseras utefter en viss tid (6.2.3). Medan i verkligheten så kommer det kompletteras ytterligare med kod för att stänga av respektive stödben då övertrycksgivaren ger en signal.



Figur 6.14: Flödesschema för Raise funktionen.

6.1.15 Manual_Plus & Manual_Min

I figur 6.15 återfinns flödesschemat för funktionerna som sköter manuell styrning. Beroende på vilken av de två modes som programmet befinner sig i så flaggas lysdioderna med antingen enkelblink eller dubbelblink. Efter det loopar funktionen igenom en sekvens som aktivt kontrollerar om funktionsknapparna har blivit nedtrycka. Om knappen är nedtryckt körs respektive stödben ned eller upp.

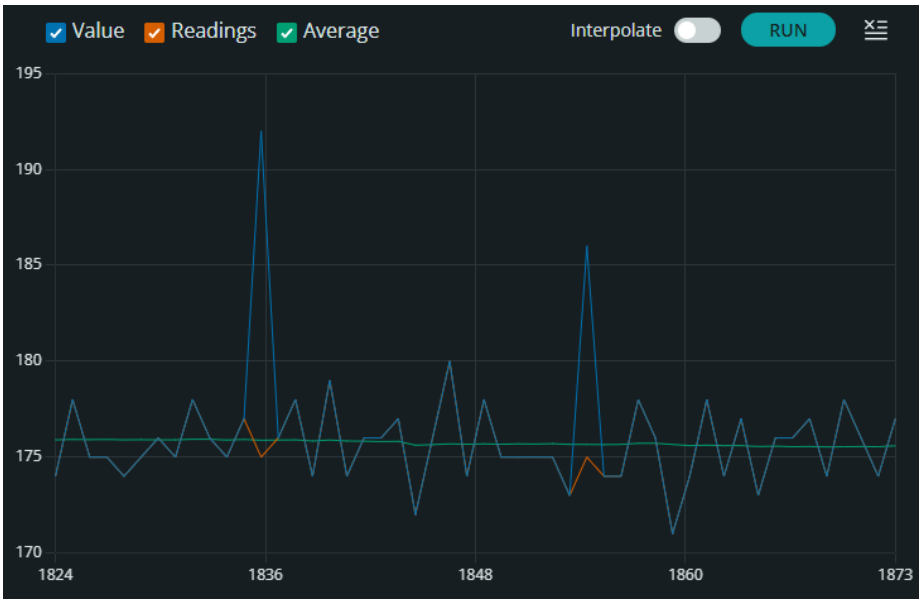


Figur 6.15: Flödesschema Manuellt läge.

6.1.16 Average_Distance

I detta projekt är avståndsmätaren en viktigt komponent eftersom den genererar en utdata som blir avgörande för systemets funktion. Eftersom sensorer ofta är känsliga för olika typer av störningar så krävs det en skyddande åtgärd för att säkerställa att avståndsmätaren fungerar korrekt.

En vanlig typ av störning som kan uppstå hos avståndssensorer är skarpa hopp i sensorns utdata som visas i figur 6.16.



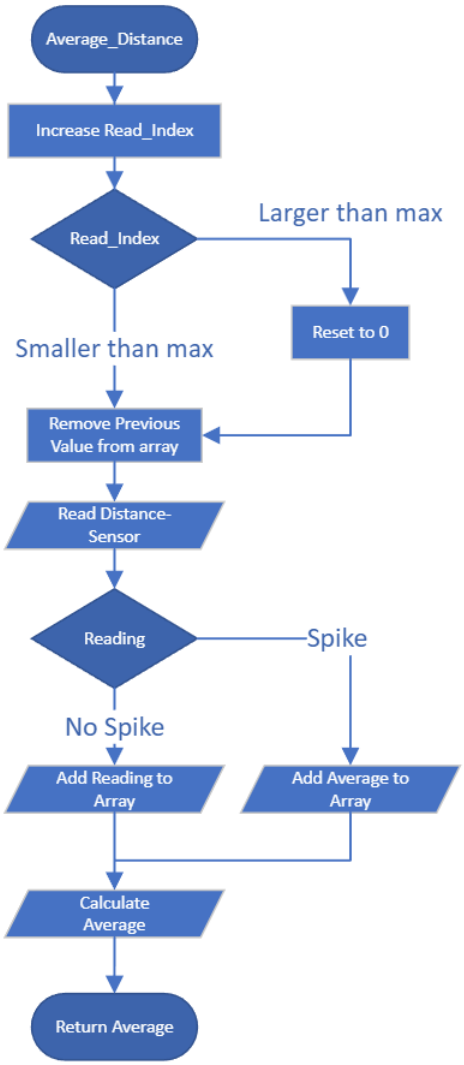
Figur 6.16: Blå färg representerar rådatan från sensorn, röd färg representerar den filtrerade datan och grön färg är genomsnittet av den röda filtrerade datan.

Dessa hopp kan uppstå på grund av flera olika anledningar och vanligtvis beror det på att sensorns utgående IR-ljus träffar ett föremål som reflekterar ljuset som sedan studsar några gånger innan den kommer tillbaka till sensorn. Därför kan det tolkas som ett längre avstånd.

För att hantera störningar som dessa så används en algoritm som kallas ”Moving Average Filter”. Denna algoritm fungerar genom att ta ett medelvärde för ett antal mätvärden och sedan använda medelvärdet som en representativ bild för sensors värden.

För att kunna beräkna medelvärdet i filtret så skapas en array som används för att lagra mätvärderna. Storleken på arrayen påverkar även filtrets egenskaper till stor del. En större array med fler platser resulterar till att filtret får en längre minnesperiod. En större array leder till att mätningen blir mer stabil men samtidigt långsammare på att reagera på förändringar i sensorns utdata. Å andra sidan så bidrar en mindre array med snabbare reaktionsförmåga på förändringar där mätvärdet dock blir mindre stabilt. Baserat på egna tester ansågs 100 element en lämplig mängd då det gav en bra balans mellan stabilitet och snabbhet.

Dock räcker det inte med att endast ta genomsnittet för arrayen då dessa skarpa hopp fortfarande har en påverkan på utdatan. Därför implementerades en ytterligare funktion som visas i flödesschemat i figur 6.17 där varje värde som sensorn utger utvärderas först ifall det är ett hopp eller inte. Om programmet känner av att det är ett hopp genom att jämföra skillnaden i den nuvarande och förra värdet så sätts värdet på det befintliga elementet till genomsnittet. Felkorrigeringen visas även 6.16 där den röda linjen följer den blå linjen tills ett hopp sker och då sätts värdet till så nära genomsnittet som möjligt istället.



Figur 6.17: Flödesschema för Average_distance funktionen.

6.2 Inställningar

Detta kapitel kommer att förklara mjukvaruinställningarna och deras påverkan på olika delars egenskaper. Här beskrivs vad varje inställning innebär och hur de påverkar de olika komponenternas egenskaper.

6.2.1 CALIBRATE

I programmet finns det en möjlighet att skriva ut värden som användaren behöver för att kalibrera MPU:erna. Funktionen kan dock stängas av genom att kommentera bort den ur koden. Att kommentera bort är rekommenderat då den inte används eftersom kalibreringen gör att programmet körs långsammare.

6.2.2 Limit

Denna variablen definerar hur stor skillnad i vinkel (grader) relativt till startvinklen som krävs för att Touch-Down funktionen ska tolka det som att benet har gått ner i marken. Ett högre värde för Limit variabeln innebär att MPU:n måste känna av en större skillnad i vinkel innan den accepterar att benet är nere. Ett högre värde minskar känsligheten för vibrationer som kan annars vilseleda systemet att tro att benet är på marken fastän det inte är det. För modellen som har använts i detta projekt gav 0.5° en bra balans.

6.2.3 Time_Limit

Time_Limit ska vara den maximala tiden, i millisekunder, som det tar för benen att gå från fullt utsträckta till helt indragna. På modellen har denna tid uppmätts till 10 sekunder alltså 10 000 millisekunder. Denna används för att ange den maximala tiden som en process bör köras innan den automatisk går vidare. Till exempel, om mjukvaran kör Touch-Down processen och tror att ett ben är på väg ner, men i själva verket händer inget på grund av någon felaktighet, så kan programmet fastna. För att undvika fel används därför denna tidsgräns för att automatiskt gå vidare.

6.2.4 Debounce_Delay

I kapitel 2.11 nämndes att fysiska knappar kan leda till falska tryckningar. För att motverka dessa används en tidsgräns, i millisekunder, som bestämmer hur lång tid det måste ha gått sen senast knapptryckning. Ett högre värde på denna inställning gör så att systemet måste vänta längre innan en knapp kan tryckas ner igen. För modellen så testades ett värde fram på 50 millisekunder vilket har visat goda resultat.

6.2.5 Blink_Interval

Denna inställning avgör hur snabbt, eller långsamt, lysdioderna ska blinka. Denna tid är i millisekunder och valdes på modellen till 1000 millisekunder, vilket innebär att den byter status 1 gång i sekunden. Inställningen påverkar också att när den dubbelblinkar så blinkar lamporna 2 gånger per sekund följt av en kort paus.

6.2.6 Gyro_Startup_Time

Det denna variabeln gör är att den bestämmer hur länge programmet ska vänta på att MPU:erna ska stabiliseras. När programmet först startas upp måste ett antal värden gå igenom filtret för att värdena ska bli så korrekta som möjligt. Därför används denna inställningen för att bestämma hur länge programmet ska vänta, i millisekunder. Vid testning var en bra tid 15 sekunder alltså 15 000 millisekunder. Med denna inställning är det bättre att vänta lite för länge då funktionen endast påverkar uppstarten av Arduinon. Ifall värdet är för litet och ett program startar snabbt inpå Arduinons uppstart så kan MPU:erna vara ostabila och vinklarna kan ändras utan att MPU:erna rör på sig.

6.2.7 Num_Readings

Denna inställning avgör hur många värden, styck, som distanssensorn ska använda för att räkna ut ett medelvärde på avståndet. Ett högre värde på denna ger att medelvärdet blir mer exakt mot vad avståndet är i verkligheten och gör systemet mindre känslig mot störningar. Däremot tar det också längre tid för medelvärdet att uppdateras. Enligt tester är en bra nivå 100st värden för att få en bra balans mellan hastighet och ett bra mätvärde.

6.2.8 Kalman_Q

Detta värde tillhör Kalman filtret som används för att filtrera ut vibrationer ur MPU:ernas data. Q-värdet bestämmer hur stor felmarginal som filtret har vilket innebär att ett högre värde gör att värdet blir mer exakt men långsammare medan ett lägre värde gör filtret snabbare men ger det också mer felmarginal.

6.2.9 Kalman_Time_Constant

Ett lägre värde på denna konstant gör så att MPU:n blir mer precis och snabbare men innebär också att den är mer känslig för vibrationer. Fördelen med att öka värdet innebär då att den blir mindre känslig för vibrationer och störningar men nackdelen är att den blir långsammare. En bra balans på denna är ett värde av 0.005 enligt testerna som gjordes på modellen.

6.2.10 MPU_Acc_Error

För nästan samtliga elektroniska komponenter är det nödvändigt att kalibrera dem innan de används i ett projekt då de kan skilja individuellt från fabrik. Detta gäller även för MPU:erna som används i projektet, där det behövs kalibrering av accelerometern i X-,Y- och Z-led. När datan omvandlas från LSB till gravitationskraft (g) så adderas en feljustering till de tre variablerna för de olika leden.

För att kalibrera MPU:n placeras komponenten på en stabil plan yta. Därefter skrivs ut de värden som MPU:n läser från accelerometerns tre axlar, värdena kan exempelvis se ut såhär:

X: 0.13 Y: 0.09 Z: 0.92

Dessa värden justeras därefter manuellt genom att addera eller subtrahera värden till MPU_Acc_Error så att de blir:

X: 0 Y: 0 Z: 1

Justeringen innebär att accelerometern ska vid en plan yta känna av 1 G kraft nedåt och 0 G kraft i X och Y led. Kalibreringen görs en gång när allt är färdigmonterat och användaren har justerat trailern manuellt till ett läge som anses plant.

6.2.11 Kalman_Angle_Error

Kalman_Angle_Error är en variabel som används för att korrigera eventuella fel i pitch- och rollavläsningen hos en MPU-enhet. Programmet utgår ifrån definitionen av en plan nivå är en yta där både pitch och roll är lika med noll. Även om trailern är placerad i perfekt plan nivå så kan MPU:n fortfarande visa fel. Exempelvis kan dessa fel bero på monteringen av MPU:n att den inte helt i nivå horisontellt. Såhär kan det se ut innan kalibrering:

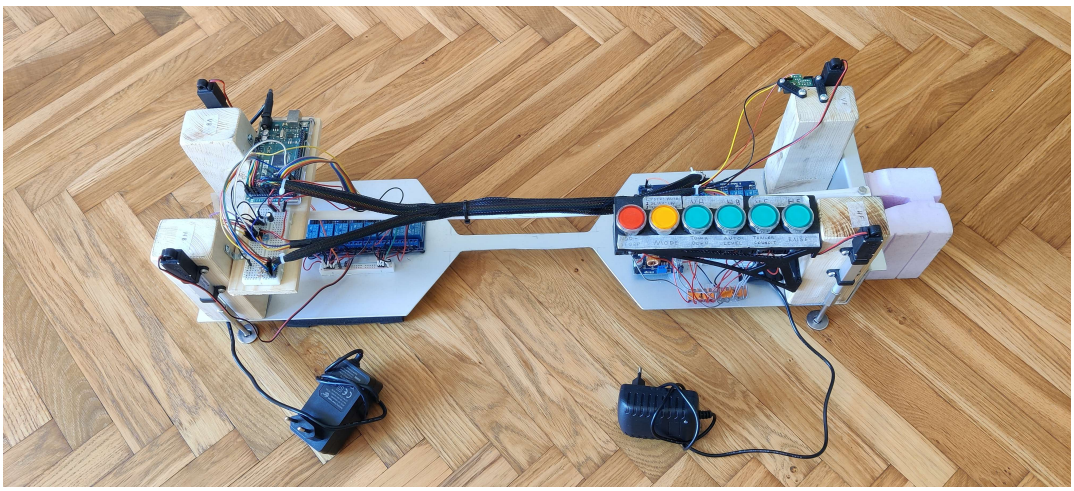
roll[0]: 2.17 roll[1]: 3.12 pitch[0]: -1.4 pitch[1]: -1.7

Dessa värden justeras därefter på samma sätt som för accelerometern genom att addera eller subtrahera värden till Kalman_Angle_Error:

roll[0]: 0 roll[1]: 0 pitch[0]: 0 pitch[1]: 0

7
Resultat

Syftet med detta projekt var att utveckla ett mjukvarusystem som skulle underlätta för Groths kunder att parkera och autolevelera sina trailers. Projektet omfattade val av komponenter, konstruktion av en testrigg (se figur 7.1), och huvudsakligen programmering, vilket resulterade i en god prototyp för autoleveleringssystemet.



Figur 7.1: Färdig byggnation av testrigg-Trailern.

Under utvecklingen av systemet genomfördes olika tester för att utvärdera dess prestanda och funktionalitet i olika scenarier som en verklig trailer kan möta. Dessa tester inkluderade uppförsbacke, nedförsbacke och sidledslutningar. För att simulera dessa scenarier höjdes en sida av modellen i taget för att skapa olika lutningar. Dessutom genomfördes vibrationstester genom att skaka bordet där modellen placerades, vilket efterliknade de vibrationer som kan uppstå med en verkligtrailer när exempelvis motorn körs. På så sätt kunde man bedöma systemets prestanda och stabilitet under dessa förhållanden. Ytterligare tester utfördes även för att verifiera de implementerade säkerhetsfunktionerna. Dessa tester inkluderade scenarier där säkerhetsfunktionerna aktiverades och svarade på olika utmaningar och risker. Resultaten av dessa tester hjälpte till att bedöma systemets förmåga att upptäcka och förhindra eventuella faror och säkerhetsincidenter. Testerna genomfördes kontinuerligt på modellen samtidigt som programvaran utvecklades för att säkerställa att inget slutade fungera under utvecklingen. Genom att utföra tester i realtid säkerställdes snabb identifiering av eventuella problem och därefter genomföra nödvändiga justeringar och förbättringar i systemet.

Resultaten av testerna visade att de valda komponenterna och konstruktionen funger-

ade väl under de olika scenarierna. Trots att det finns mer avancerade och kostsamma komponenter tillgängliga, valdes de nuvarande komponenterna för att underlätta implementeringen av systemet och hålla kostnaderna nere. Systemet visade sig vara mycket konsekvent och pålitligt under de olika testerna som gjordes och kunde hantera de utmaningar och scenarier som uppstod. Dessa tester bidrog till att verifiera systemets funktionalitet och prestanda samt ge förtroende för dess kapabiliteter inför vidare användning och drift.

8

Diskussion & Slutsats

Detta kapitel kommer att diskutera fördelar och nackdelar med systemet, utvärdera vad som fungerar bra och vad som kan förbättras.

8.1 Frågeställning

- Vilka säkerhetsfunktioner krävs av systemet?

Besvaras i kapitel 4. Sammanfattningsvis behövs en tidsgräns, "Enable signal", feltrycks kontroll, pausfunktion och ett nödstopp.

- Är modellen en lämplig representation av verkligheten?

Modellen som har utvecklats är en lämplig representation av verkligen för de huvudfunktioner som systemet är avsett att utföra. Det bör dock påpekas att det inte har gjorts någon utvärdering av hur mjukvaran är särskilt anpassad för en riktig trailer med avseende på fysiska påfrestningar och nötning över tid. Anpassningen för en verklig trailer är något som Groth kommer behöva undersöka vidare.

- Hur kan systemet vidareutvecklas?

Det finns flera olika sätt som systemet kan vidareutvecklas. Exempelvis finns det andra styrenheter som är vanligare i industriella miljöer [19]. Enligt denna jämförelsen kan en PLC vara ett mer lämpligt alternativ att använda i systemet då tanken är att den ska vara utomhus på en lastbilstrailer. Systemet kunde också vidareutvecklats genom att lägga till fler avståndssensorer vid varje ben för att ha en mer exakt positionering av trailern. En annan förbättringar hade varit en alternativ styrningsenhet som exempelvis en mobilapp eller liknande för att underlätta styrningen av trailern.

8.2 Förbättringar i modellen

En förbättring kunde ha varit att konstruera en PCB för att eliminera de lösa kontakter och komponenter som finns på modellen.

Utöver detta har det uppstått problem med MPU-sensorerna att de får en timeout, vilket innebär att Arduinon inte kan kommunicera med sensorerna. Enligt Arduinos officiella hemsida [20] är dessa timeouts nästan alltid ett tecken på ett

underliggande problem, exempelvis felaktigt beteende från sensorn, brus eller andra elektriska störningar. I koden har problemet dock kringgåtts med hjälp av den inbygga funktionen setWireTimeout() vilket agerar som ett skyddsnät. Ifall sensoren inte svarar inom en viss tidsram så avbryts I2C-kommunikationen och försöker sedan igen. Genom att sätta denna tidsram kan programmet då undvika att fastna i oändliga loopar. I projektet är tidsramen satt till 1000 µs vilket är en väldigt kort tid, det vill säga att det är ingenting som påverkar användaren. Men det är rekommenderat att det underliggande problemet bör tas i hänsyn och fixas.

När det kommer till förlusten av Arduinos minne vid nödstopp, skulle en möjlig lösning kunna vara att införa ett externt minnesalternativ som fungerar med en separat spänningsmatning. Genom att göra detta skulle informationen sparas och användaren skulle slippa att manuellt styra trailern efter ett nödstopp.

8.3 Slutsats

För att sammanfatta projektet så har alla krav från Groths sida uppfyllts. Modellen har säkerhetssystem implementerade och de fyra huvudfunktionerna: Touch-Down, Autolevel, Trailer-Connect och Raise fungerar utan problem på modellen. Som nämnt tidigare finns det förbättringar och utvecklingsmöjligheter för systemet. Dock var detta projektet menat som ett proof of concept, alltså att systemet skulle bevisa att det gick att utföra samtliga steg med komponenterna som valts. Avslutningsvis kommer modellen att behållas av Groth och den kommer att användas i demonstrationssyfte för att på en enkelt sätt visa sina kunder hur systemet kontrolleras och används. Groths ska även försöka implementera systemet på en verklig trailer.

Litteraturförteckning

[1] SparkFun. *What is an Arduino?* URL: https://learn.sparkfun.com/tutorials/what-is-an-arduino/all. (hämtad: 21-02-2023).

[2] Arduino.CC. *Mega 2560 Rev3*. URL: https://docs.arduino.cc/hardware/mega-2560. (hämtad: 02-05-2023).

[3] SparkFunElectronics. *Arduino Mega 2560 R3*. URL: https://openverse.org/image/a7ee9fd8-8f16-46e1-aa39-cfe11de546de?q=Arduino+2560. (hämtad: 02-05-2023).

[4] MakerGuides. *How to use a SHARP GP2Y0A21YK0F IR Distance Sensor with Arduino?* URL: https://www.makerguides.com/sharp-gp2y0a21yk0f-ir-distance-sensor-arduino-tutorial/. (hämtad: 15-02-2023).

[5] SHARP. *Analog Output Type Distance Measuring Sensor*. URL: http://www.socle-tech.com/doc/IC%5C%20Channel%5C%20Product/Sensors/Distance%5C%20Measuring%5C%20Sensor/Analog%5C%20Output/GP2Y0A21YK0F_spec.pdf. (hämtad: 19-03-2023).

[6] Components101. *Buck Converter: Basics, Working, Design and Operation*. URL: https://components101.com/articles/buck-converter-basics-working-design-and-operation. (hämtad: 4-04-2023).

[7] HowToMechatronics. *Arduino and MPU6050 Accelerometer and Gyroscope Tutorial*. URL: https://howtomechatronics.com/tutorials/arduino/arduino-and-mpu6050-accelerometer-and-gyroscope-tutorial/. (hämtad: 24-01-2023).

[8] InvenSense. *MPU-6000 and MPU-6050 Register Map and Descriptions Revision 4.2*. URL: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf. (hämtad: 12-04-2023).

[9] ElectronicsTutorials. *Analogue to Digital Converter*. URL: https://www.electronics-tutorials.ws/combination/analogue-to-digital-converter.html. (hämtad: 27-03-2023).

[10] Jared Becker Jonathan Valdez. *Understanding the I2C Bus*. URL: https://www.ti.com/lit/an/slva704/slva704.pdf. (hämtad: 03-04-2023).

[11] Arduino. *Wire*. URL: https://www.arduino.cc/reference/en/language/functions/communication/wire/. (hämtad: 01-04-2023).

[12] Alex Becker. *ONE-DIMENSIONAL KALMAN FILTER WITHOUT PROCESS NOISE*. URL: https://www.kalmanfilter.net/kalman1d.html. (hämtad: 26-02-2023).

[13] Circuit-Basics. *HOW TO DE-BOUNCE SWITCHES ON THE ARDUINO*. URL: https://www.circuitbasics.com/how-to-use-switch-debouncing-on-the-arduino/. (hämtad: 25-02-2023).

[14] SHARP. *Analog Output Type Distance Measuring Sensor GP2Y0A41SK0F*. URL: https://www.electrokit.com/uploads/productfile/41013/GP2Y0A41SK0F_ED05G101A.pdf. (hämtad: 29-05-2023).

[15] X-Engineering. *Xcos tutorial – signal edge detection*. URL: https://x-engineer.org/xcos-signal-edge/. (hämtad: 027-04-2023).

[16] EEPower. *Pull-up and Pull-down Resistors*. URL: https://eepower.com/resistor-guide/resistor-applications/pull-up-resistor-pull-down-resistor/. (hämtad: 02-05-2023).

[17] Arduino. *setup()*. URL: https://www.arduino.cc/reference/en/language/structure/sketch/setup/. (hämtad: 24-04-2023).

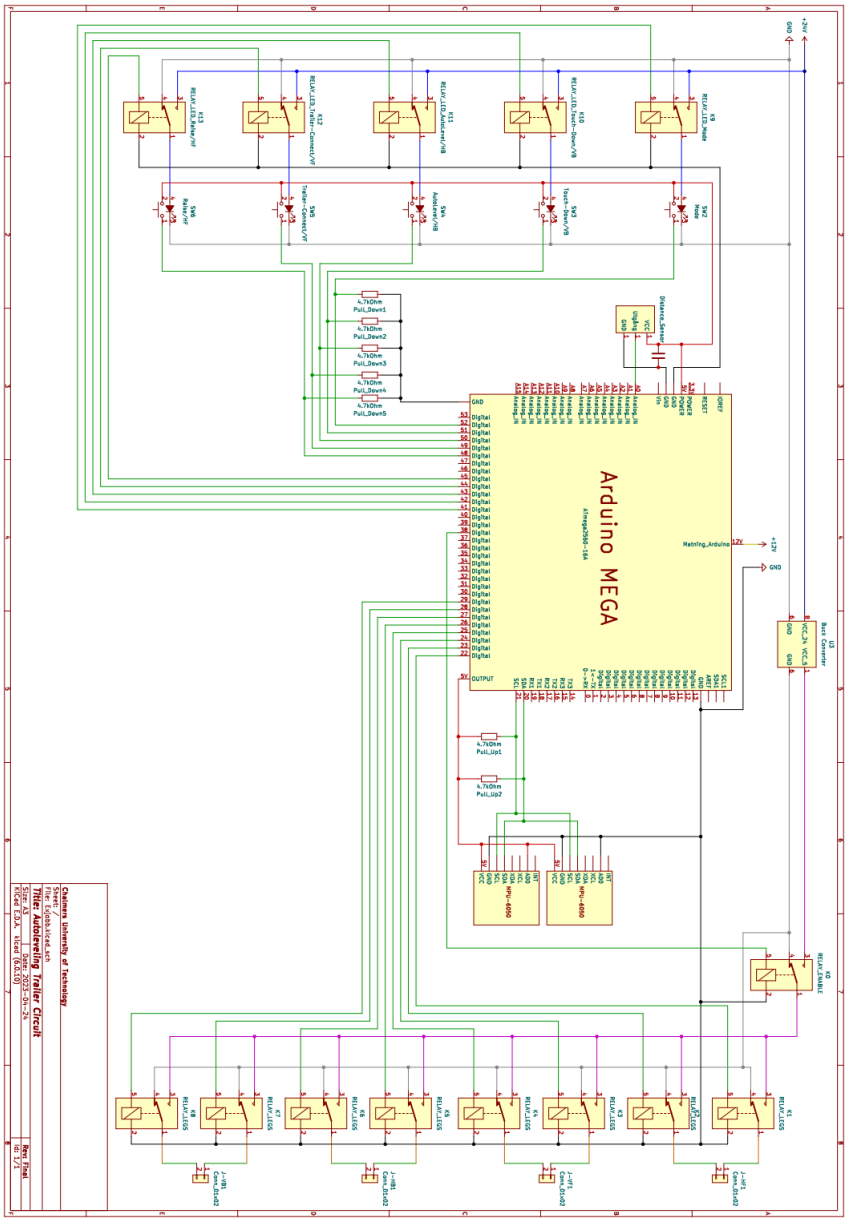
[18] Arduino. *loop()*. URL: https://www.arduino.cc/reference/en/language/structure/sketch/loop/. (hämtad: 01-05-2023).

[19] Circuit-digest. *Microcontroller vs PLC: A Detailed Comparison*. URL: https://circuitdigest.com/article/microcontroller-vs-plc-detailed-comparison-and-difference-between-plc-and-microcontroller. (hämtad: 25-03-2023).

[20] Arduino. *setWireTimeout()*. URL: https://reference.arduino.cc/reference/en/language/functions/communication/wire/setwiretimeout/. (hämtad: 05-05-2023).

A

Appendix 1



Figur A.1: Kretsschemat för modellen.

Institutionen för Elektroteknik
Chalmers Tekniska Högskola
Göteborg, Sverige
www.chalmers.se

CHALMERS