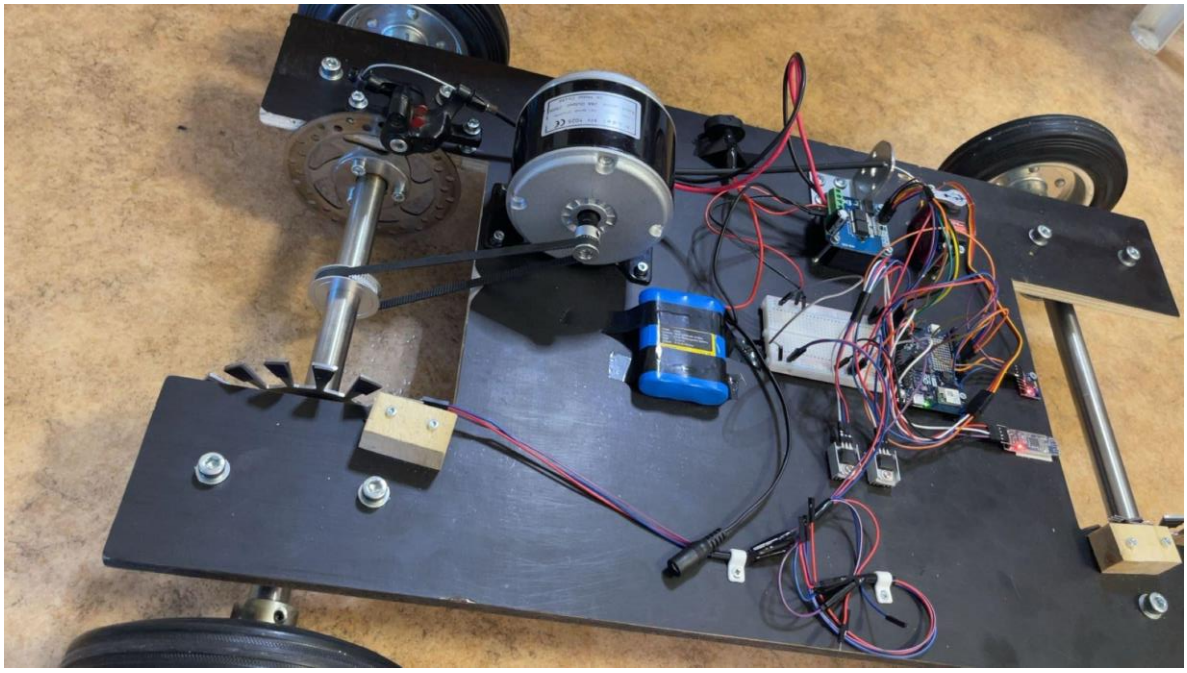


Utveckling av fordonsprototyp för analys av låghastighetskörning

Examensarbete inom högskoleingenjörsprogrammet i maskinteknik.

Amjad Alnachawati
Hanna Shamoun

INSTITUTIONEN FÖR MEKANIK OCH MARITIMA VETENSKAPER

CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2026
www.chalmers.se

Abstract

This thesis presents the design and development of a vehicle prototype with an integrated measurement system for analyzing speed, acceleration and jerk at low velocities. The purpose of the project was to create a low-cost platform that can be used to study vehicle dynamics at low speeds, with focus on measuring jerk during acceleration from standstill and braking.

The prototype was powered by a 12 V DC motor, with power transmitted to the rear wheels through a belt drive. Braking was achieved using a bicycle brake system controlled by a servo motor. The measurement system was based on an Arduino UNO Q, which was used as the control unit. Hall sensors were used to measure speed, while an MPU-6050 accelerometer was used to measure acceleration.

To improve the reliability of the acceleration values, sensor fusion was used by combining data from the hall sensor and the accelerometer. The acceleration value was weighted with 70% from the hall sensor and 30% from the accelerometer. Wireless control and data collection were done using an HM-10 Bluetooth Low Energy module and a web application developed in HTML5 and JavaScript.

Two test runs were conducted, and the results showed that jerk was highest during braking and acceleration from standstill, with a maximum value of 4.83 m/s^3 measured during braking. Overall, the system shows that cost-effective hardware can be used to study vehicle dynamics at low velocities and provides a foundation for further development with different loads, speeds and road surfaces.

Tackbrev

Vi vill rikta ett varmt tack till våra handledare Samira Deylaghian och Mats Jonasson för allt stöd, rådgivning och engagemang under arbetets gång. Er erfarenhet och kunskaper har bidragit till att forma detta arbete.

Vi vill även tacka vår examinator Petri Piiroinen för hans närvaro och engagemang under mötena. Hans förståelse och vägledning underlättade arbetsprocessen och hjälpte oss att se slutmålet tydligare.

INNEHÅLLSSIDOR

Innehåll

INNEHÅLLSSIDOR V
1. Introduktion 1
1.1 Bakgrund 1
1.2 Syfte 1
1.3 Avgränsningar 1
1.4 Precisering av frågeställningen 2
2. Systemmodellering och teori 3
2.1 Användning av en fordonsprototyp för ryckmätning 3
2.1.1 Ändring av acceleration 3
2.1.2 Människa under ryck 3
2.2 Hall sensor 4
2.3 Internal Measurement Unit 4
2.3.1 Accelerometer 4
2.3.2 Gyroskop 5
2.3.3 Magnetometer 5
2.4 Borstlös DC motor 5
2.5 Servomotor 6
2.6 Motorstyrning H-brygga 6
2.7 Arduino 6
2.8 Pulsbreddsmodulering 7
3. Metod 9
3.1 Informationsinsamling 9
3.2 Artificiell intelligens 10
3.3 Konstruktion 11
3.4 Programmering 12
3.5 Elektrisk koppling 14
3.6 Beräkning 15
3.6.1 Hall-effekt skiva periodberäkning 15
3.6.2 Hastighet beräkning av tidsperiod 16
3.6.3 Timeout för $\boldsymbol{vmin}$ 16
3.6.4 Förbättring av hastighetsberäkning 17
3.6.5 Acceleration och ryck 17
3.6.6 Effektiv spänning och PWM-nivå 18

3.6.7 LM7805 regulator dimensionering ... 18
4. Resultat ... 19
4.1 Fordonets hastighetsanalys ... 19
4.2 Analys av acceleration och retardation ... 20
4.3 Ryckanalys ... 21
4.4 Bromssystemets funktion ... 22
4.5 Human-machine interface (HMI) ... 22
4.6 Felkällor ... 23
5. Diskussion ... 25
6. Slutsats: ... 27
7. Referenser ... 28
8. Bilagor ... 31
Bilaga A: Kravspecifikation ... 31
Bilaga B: Hall-effekt skiva ... 32
Bilaga C: C++ programkod ... 33
Bilaga D: HMI layout ... 40
Bilaga E: Kopplingsschema ... 41
Bilaga F: Datalogg CSV-file ... 42

1. Introduktion

Det här stycket innehåller bakgrund, syfte och avgränsningar.

1.1 Bakgrund

Vid körning i låga hastigheter, särskilt vid start och stopp, upplever föraren ofta en obehaglig ryckning, som kallas *ryck*, vilket innebär en snabb förändring i accelerationen. Detta beteende uppstår till följd av två olinjära fenomen. Det ena orsakas av friktionskrafter mellan däck och mark och det andra av friktion i drivlinans mekanism. Mer specifikt uppstår fenomenet mellan två ytor när det sker en förändring i riktningen på den relativa hastigheten mellan ytorna, vilket i sin tur innebär ett steg i de pålagda krafterna och/eller de involverade vridmomenten.

En annan konsekvens av ryckfenomenet vid låga hastigheter är svårigheten att mäta rörelseegenskaper, som hjulhastighet och hjulmoment, vilket kan påverka fordonets styrbarhet.

1.2 Syfte

Syftet med detta examensarbete är att designa och bygga en modell som underlättar analysen av acceleration och ryck vid höghastighetskörning, framför allt vid start från stillastående och vid stopp.

Arbetet ska resultera i en fungerande fordonsprototyp tillsammans med en modellbeskrivning samt preliminära testresultat, vilka senare kan ligga till grund för vidare utveckling inom fordonsdynamik och reglerteknik.

1.3 Avgränsningar

Arbetet avgränsas till följande:

- Analys av låghastighetsbeteende vid start och stopp.
- Tillverkning av en fordonsprototyp, cirka 0.5 m lång, och inte ett fullskaligt fordon.
- Grundläggande mätning och analys av acceleration och ryck.
- Implementering av någon slags friktionsbromsningsmekanism.

1.4 Precisering av frågeställningen

Utifrån syftet preciseras arbetet genom följande frågeställningar:

- På vilket sätt kan en skalad fordonsprototyp utformas för att på ett tillförlitligt sätt återskapa och mäta ryck vid start och stopp?
- Hur ska mätningen genomföras och vilka sensorer är lämpliga för att mäta acceleration, hastighet och ryck vid låga hastigheter?
- Överensstämmer de experimentella mätdata med den teoretiska modellen av fordonets dynamik?

2. Systemmodellering och teori

I detta stycke kommer systemförståelsesteori att presenteras samt en beskrivning av hur modellen och systemets delar byggs samman för att vidare bilda en funktionell prototyp.

2.1 Användning av en fordonsprototyp för ryckmätning

För att undvika ryckproblem som kan uppstå kan implementeringen av en fordonsprototyp vara ett användbart alternativ för validering och optimering av drivlinans prestanda, komfort och hållbarhet. En sensor är installerad på en fordonsprototyp som läser in ryckkrafternas förändringar, vilket kan ge ett värde på hur stora ryck föraren upplever under sådana omständigheter. Med hjälp av det kan olika tester av scenarier som föraren upplever under färd.

2.1.1 Ändring av acceleration

Ryck representeras på ett matematiskt sätt som en vektorstorhet och har beteckningen “j” med enheten meter per kubiksekund (m/s³). Den beskriver hur snabbt en förändring av accelerationen sker. Den har stor betydelse inom fordonskonstruktion, maskinelement och konstruktion av olika berg- och dalbanor där påfrestningar och komfort är viktiga (Hayati et al., 2020).

Ryck definieras som den tredjederivatan av positionen med avseende på tiden och kan matematiskt skrivas som

$\vec{J} = \frac{d\vec{a}(t)}{dt} = \frac{d^2\vec{v}(t)}{dt^2} = \frac{d^3\vec{r}(t)}{dt^3}$.

2.1.2 Människa under ryck

Människans kropp påverkas av ryckkrafterna på ett märkvärdigt sätt. Människans muskler är begränsade och reagerar inom vissa gränser när ryck inträffar kraftiga eller plötsliga, vilket orsakar att människokroppen tappar kontroll tillfälligt eftersom musklerna inte hinner anpassa sig tillräckligt snabbt. Det förklarar fenomenet varför oerfarna förare kan orsaka en ryckig resa för sina passagerare (Hayati et al., 2020).

2.2 Hall sensor

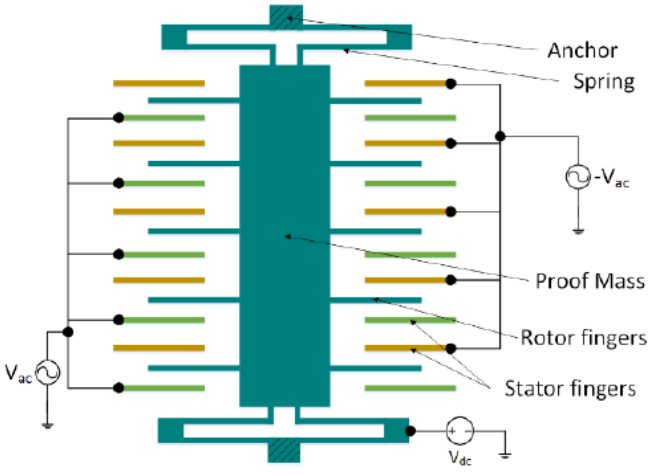
En hallsensor mäter spänningsändringar när ett magnetfält kommer i närheten av den. Magnetfältet skjuter elektronerna i sensorn åt ena sidan, vilket skapar spänningsförändringar i sensorn. Dessa förändringar läses av för att mäta hastigheten på likströmsmotorn (DC-motor) genom att en roterande magnet i rotorn slår på och av hallens sensor när den roterar. Med avseende på hur snabbt sensorn sätts på och av kan hastigheten på rotorn beräknas (EngThink, 2023).

2.3 Internal Measurement Unit

IMU fungerar genom att detektera rotationsrörelsen i tre axlar, oftast kända som pitch, roll och yaw. För att uppnå dessa använder sig en IMU av inbyggd accelerometer, gyroskop och magnetometer (Shawn, 2020).

2.3.1 Accelerometer

En accelerometer är en typ av sensor som mäter acceleration vid rörelse. Kraften som orsakas av vibrationer leder till att den inre massan komprimerar ett piezoelektriskt material, vilket i sin tur producerar en elektrisk laddning som är proportionell mot kraften. Accelerometern plockar endast upp pitch och roll men ger ingen information om yaw. Inom bilindustrin används accelerometern för att mäta vibrationer, acceleration samt övervaka prestandan hos bilar och andra fordon (DwyerOmega, u.å & LaboTest AB, u.å.).



Figur 1: Visualiserad överblick på en accelerometer

2.3.2 Gyroskop

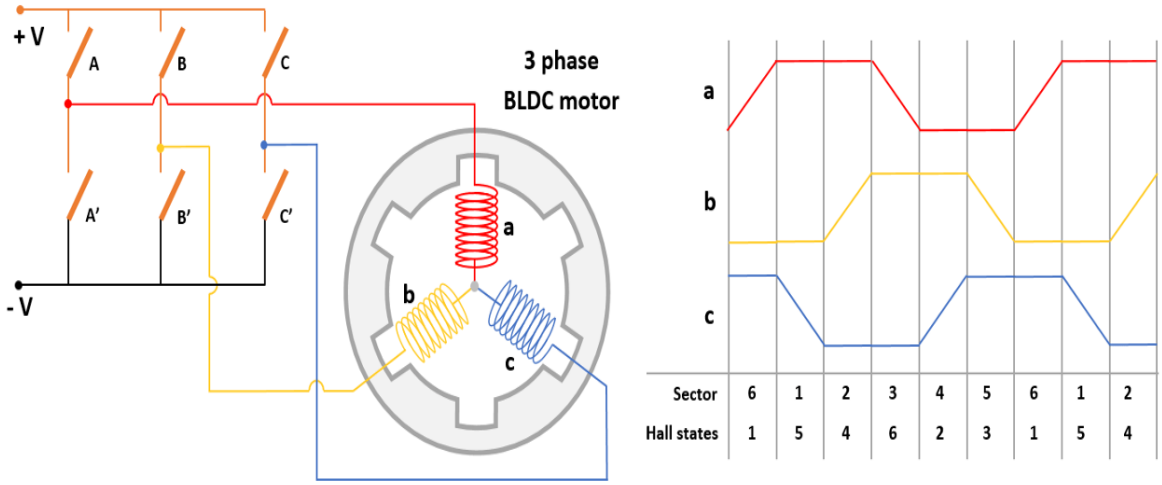
Gyroskop är ett hjul eller en skiva monterad på en kardan så att det kan rotera snabbt runt en axel som är fri att ändra riktning. Den roterande rotorns vinkelmoment gör att den bibehåller sin position även när den enhet som den är monterad på byter lutning. Gyron bibehåller sin ursprungliga position i förhållande till referenspunkten i rymden, utan att påverkas av jordens rotation.
Gyroskopen i en IMU-sensor används för att mäta rotation, förbättra prestandan och noggrannheten hos navigeringssystem eftersom den inte kräver någon form av GPS-signal (Dingman, u.å.).

2.3.3 Magnetometer

Det är sensorn som används vid mätning av magnetfält, särskilt jordens magnetfält. I IMU-systemet är magnetometern en digital kompressor som innehåller kursinformation med avseende på jordens magnetpol. Den mäter styrkan och riktningen på ett magnetfält med hjälp av varierande tekniska metoder. Den vanligaste typen inom konsumentelektronik är känd som Hall-effektsensor eller magnetoresistiv sensor (Daisch Sensor, u.å.).

2.4 Borstlös DC motor

Likströmsmotorer används ofta inom motorstyrningsapplikationer. De två vanligast förekommande versionerna av likströmsmotorer är med borstar eller utan. Båda versionerna fungerar genom att attraktion och repulsion uppstår mellan spolarna och de permanenta magneterna. Borstlösa likströmsmotorer använder sig av elektronisk styrning för att få motoraxeln att rotera, medan borstar används för att styra strömriktningen och få motoraxeln att rotera. På en borstlös likströmsmotor sitter permanenta magneter fästa vid rotorn. Vanligtvis har statorn tre kopplade lindningar antingen i ”delta”- eller ”stjärn”-konfiguration. Den fungerar genom elektronisk kommutering för att snurra statorns magnetfält och för att åstadkomma detta används aktiv styrningselektronik. (Monolithic Power Systems, u.å.)



Figur 2: Kommuteringssekvens för en trefas DC-motor.

2.5 Servomotor

En servomotor är en motorvariant som möjliggör styrning av motorns axelns position, hastighet samt acceleration. Med hjälp av sensorer och avancerade styrsystem kan positionen styras elektroniskt eller hastigheten regleras. En servomotor klarar väldigt höga krav när det gäller antingen dynamik, inställningsmöjligheter eller precisa rörelser. Servomotorer kan bestå av asynkronmotorer, synkronmotorer eller DC-motorer, enligt (TYHE Motors, u.å & SEW-EURODRIVE, u.å.).

2.6 Motorstyrning H-brygga

En H-brygga är en krets med komprimerade ingångar och utgångar för styrning av en motor som har högre ström än vad Arduino kan styra. Arduino kan skicka en 5 V-signal, men motorn drivs med 12 V. För att lösa spänningsskillnaden kan en H-brygga användas. H-bryggan kan även skydda motorn från att gå sönder när den sätts på och stängs av vid användning av PWM, för tydlig förkortning beskrivning se avsnitt 2.8. H-bryggan som används i projektet är av typen BTS7960, enligt (Handson Technology, u.å.).

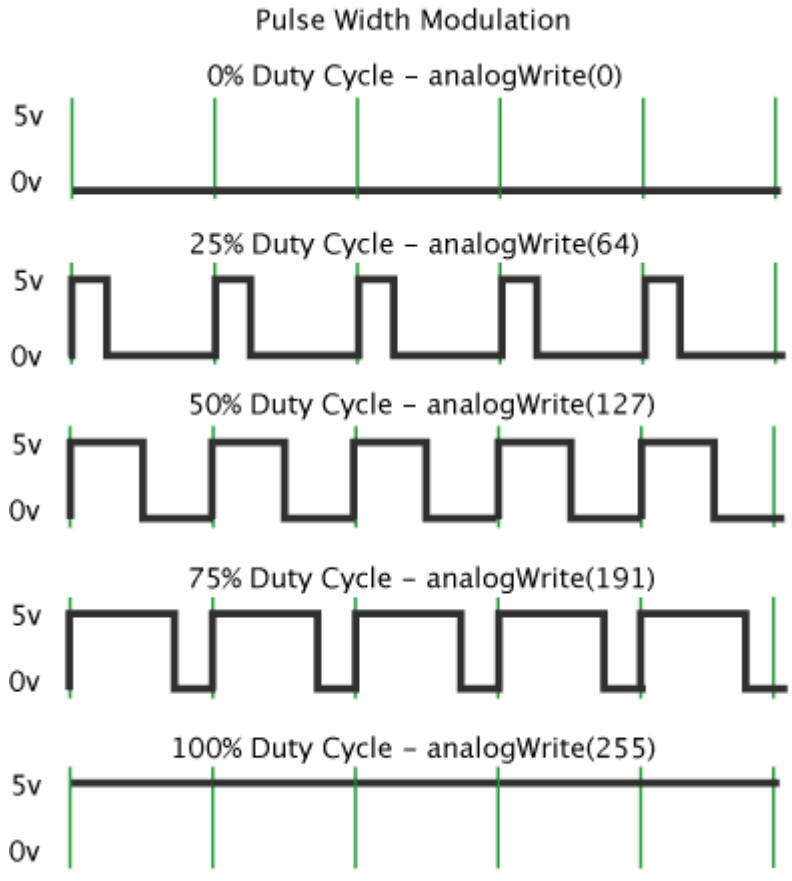
2.7 Arduino

Arduino UNO Qualcomm är en liten mikrokrets med dubbla kraftfulla processorer och 16 digitala ingångar eller utgångar som kallas PIN. Denna PIN har 6 utgångar som kan

hantera PWM. Arduino har även en USB-typ C-port och en resetknapp. Med andra ord är Arduino hjärnan som möjliggör styrning av fordonsprototypen, eftersom den kan användas antingen som en liten dator eller som en styrenhet. Den här Arduino typen är väldigt snabb då den stödjer 2 GB i RAM minne och även 16 GB eMMC för lagring av filer.
Den här kapaciteten öppnar möjligheten till olika tilläggsfunktioner vid vidare utveckling eller vid behov av annat än att bara mäta IMU i en fordonsprototyp (Arduino, u.å.).

2.8 Pulsbreddsmodulering

PWM är en teknik som omvandlar digitala metoder till analoga resultat genom att ställa in en komponent i ett läge mellan av och på. Den här metoden möjliggör digital styrning av komponenter genom att skicka en 5 V-signal till Arduino UNO q och en 0 V-signal för att få varierande analoga värden. En ändring av pulsbredden görs vid repeterade på och av mönster med tillräckligt hög hastighet. Detta medför en styrningsmekanism som kan styra exempelvis hastigheten på en motor eller ljusstyrkan på en LED-lampa.
Figur 3 längre ner visar hur en period skulle se ut, där de gröna linjerna markerar periodens gräns. Invers till en period kan vara PWM-frekvensen som mäts i Hz. En Arduino har en frekvens på 500 Hz och en tid på 2 millisekunder under den gröna linjen. Med denna information kan en anropsparameter “analogWrite” beräknas enligt Figur 3.



Figur 3: Pulse Width Modulation

Under projektets gång kommer PWM att användas för att styra olika hastighetsnivåer på DC-motorn. Projektteamet siktar på en hastighet mellan 0 och 1 m/s, men hantering av högre hastighet skulle kunna vara möjlig enligt önskemål genom att ändra det på en simulerad plattform där styrningen för fordonsprototypen sker, enligt (Arduino, u.å.).

3. Metod

Det här kapitlet handlar om vilka metoder som används i projektet för att bygga fordonsprototypen. Den beskriver även tillvägagångssättet och programvaran som användes.

3.1 Informationsinsamling

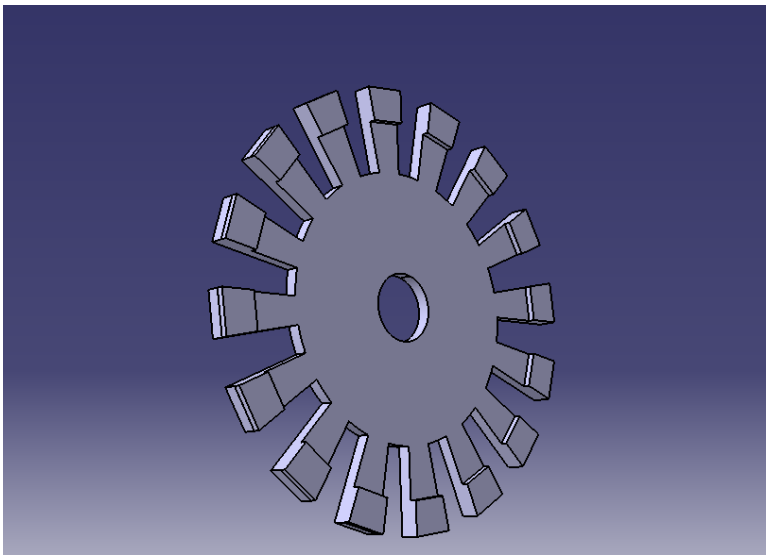
Informationen började samlas först genom ansökningsaffischen där det framgick att en fordonsprototyp som mäter ryck vid låg hastighet ska framställas. Vidare ställdes frågor till intressenter om vad som behövdes för att kunna påbörja projektet. För att få en klarare bild och en mer fördjupad analys av vad ryck exakt betyder inleddes en omfattande sökning efter artiklar och studier som handlar om det. Även användningen av AI har varit en betydande faktor som har bidragit till nytta genom att diskutera fenomenet och på ett enkelt sätt beskriva problemet.

När informationen var tillräcklig för att påbörja arbetet. En kravspecifikation skapades enligt (Bilaga A), samt en inköpslista med alla komponenter, med hjälp av informationen som examinatorn efterfrågade, samt några tilläggsfunktioner som tycktes vara intressanta för projektteamet. Dessa tilläggsfunktioner kommer att bidra till att förbättra funktionaliteten i helhetskonceptet.

Ett intressant problem som även krävde informationssökning är att körningen kommer att ske vid väldigt låga hastigheter, och vid mätning av hastigheten kommer mätaren att visa värdet noll på grund av tidsdifferensen, vilket tar väldigt lång tid för att klara en cykel. Efter sökningen visade det sig att problemet är ganska känt inom bilindustrin, men att ingen relevant lösning ännu har presenterats. För att lösa problemet, en diskussion med Claude AI implementerades för att komma på en relevant lösning som kan bidra med att minska effekter av fenomenet eller lösa det helt.

Lösningen bygger på att ställa 16 magneter på en skiva som har samma antal mellanrum mellan magneterna och en Hallsensor som mäter tiden under mindre avstånd enligt (Figur 4). Metoden används på båda drivaxlarna för att jämföra hastigheten från två olika källor. Magneterna är magnettejp som klipptes men dimensionerna på de magneterna är inte identiska och har små toleranser som kan påverka beräkningen av hastighet. Dimensioner enligt Bilaga B

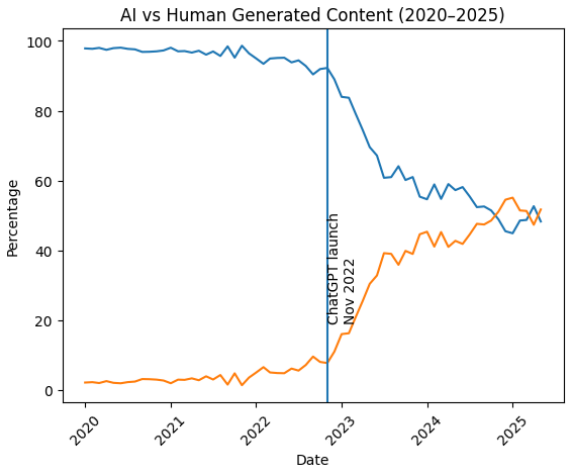
En annan lösning är att använda en programkod som kallas “micros()”. Den här koden möjliggör beräkning av en mer noggrann hastighet med ett litet procentuellt fel. Kombinationen av både en fysisk lösning och programkod ger en mer exakt hastighetsmätning.



Figur 4: Hall sensors skiva med magneter

3.2 Artificiell intelligens

Artificiell intelligens (AI) har utvecklats enormt de senaste åren, och många företag och skolor har börjat använda AI för att underlätta förståelsen av arbetet. En smart idé är att kunna utnyttja teknologin på ett korrekt sätt. AI är en bra start för att få bollen i rullning, men inte för att förlita sig helt på AI eller lita på allt utan att dubbelkolla om informationen är trovärdig genom källor och relatera det till hur AI användes under arbetets gång. AI har använts i det här projektet i syfte att rätta stavfel och förbättra grammatiken med verktyget Grammarly samt som hjälpmedel vid programkods-skrivningen med Claude. Enligt Figur 5 har mängden AI-genererade artiklar ökat kraftigt sedan ChatGPT släpptes, vilket visar att många artiklar som anses vara trovärdiga har AI-skrivet innehåll bakom sig.



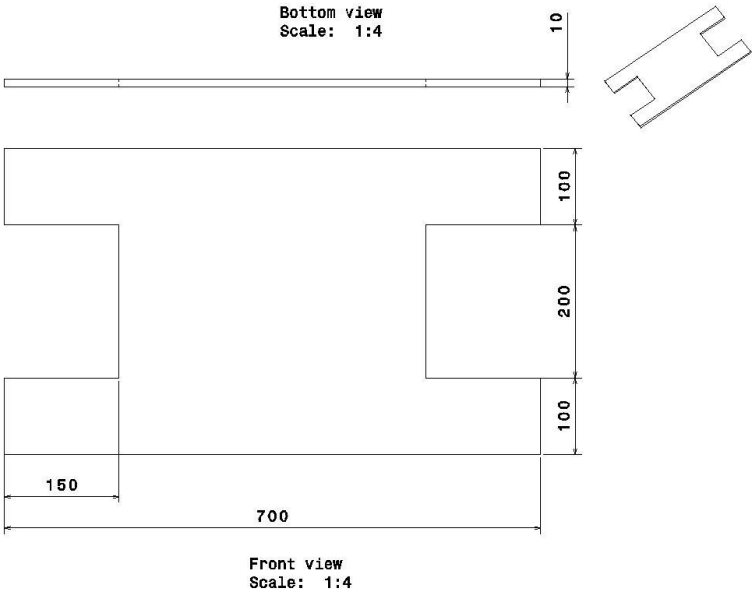
Figur 5: Procentuell ändring i AI-genererade artiklar och mänskligt genererade artiklar.

Egen visualisering baserad på data från Graphite (2024), “More Articles Are Now Created by AI Than Humans”.

3.3 Konstruktion

För ett fordon med totalvikt på 10–15 kg konstruerades ett chassi med dimensionerna 700 mm × 400 mm. Valet av Formplywood som material uppfyller kraven på vikt och hållfasthet, där målet var ett stabilt chassi som håller komponenterna utan att öka den onödiga vikten av den totala massan.

Ramen utformades som ett ”H” för att minimera vikten och frigöra utrymme för remskivan som sitter på axeln. Komponenterna placeras med hänsyn till tyngden för att uppnå en centrerad tyngdpunkt.



Figur 6: Dimensionering av fordonprototyp chassi, alla mått i mm

Implementeringen av en 12 V DC-motor ledde till att motorn måste vara fast indragen på chassidelen som överför drivkraften genom en rem vidare till drivaxeln. Rembandet måste vara väldigt hårt spänt för att utnyttja det mesta av motorkraften.

Konstruktionen utformas för att på ett enkelt sätt definiera fordonets främre eller bakre del med hjälp av position och riktning. Fordonet har två axlar med en justerbar hastighetsmätarskiva och en skiva på en av axlarna för att inkludera bromssystemet, där bromsen är fast monterad i chassidelen. Bromsningen sker genom en servomotor som drar ett kabelrep för att vidare klämma fast bromsskivan.

Elektronikmonteringen i samband med bestämning av motorns placering omfattades även av bestämning för de andra elektronikkomponenterna där batteriet och motor-drivaren satt väldigt nära motorn, men samtidigt nära de andra elektronikdelarna som får spänningsförsörjning från batteriet.

Servomotorn försörjs med 5 V genom två parallellkopplade spänningsregulatorer, snarare än direkt från Arduino. Anledningen till det är att Arduino UNO Q:s inbyggda 5 V-Pin är begränsad till 0.2-0.5 A kontinuerlig ström. Vid full belastning kräver servomotorn 2 A, vilket i sin tur inte går att få via 5 V-Pin. Genom parallellkopplingen av spänningsregulatorer kan spänningen uppgå till 3 A, vilket ger en god säkerhetsmarginal och säkerställer att kraven uppfylls. Regulatorerna monterades med kylflänsar för att kyla värmen som uppstår vid omvandling av spänning från 12 V till 5 V.

3.4 Programmering

Programmeringen utförs i en programvara som kallas Arduino IDE. Programvaran möjliggör programmering av olika typer av Arduino. Genom att lägga ner olika biblioteksfiler kan olika funktioner inkluderas i programmet, och med hjälp av AI kan test och programinformation planeras och diskuteras i förväg utan tillgång till fysiska komponenter. Koden har genererats med hjälp av Claude, men ändringar och tester har utförts för att anpassa den till projektets mål. Koden är skriven i C++ programmeringsspråket, se Bilaga C

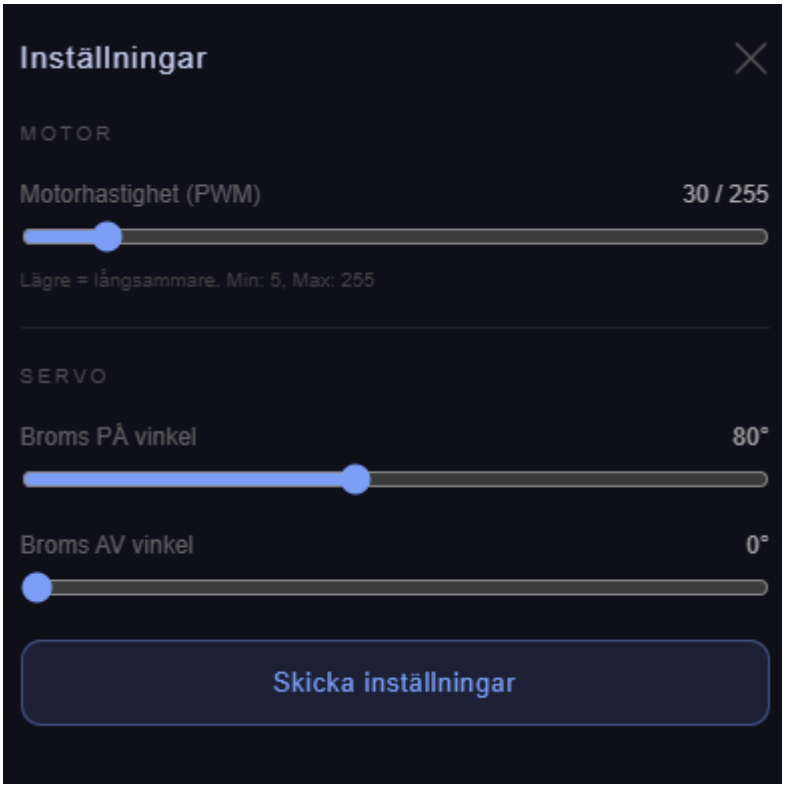
En webbapplikation utvecklades för att möjliggöra trådlös styrning samt datainsamling av värden under testkörningarna. Webbapplikationen är programmerad med hjälp av HTML5, CSS och JavaScript, där Claude användes som stöd i form av en instruktör. Arduino UNO Q kommunicerar med webbapplikationen med hjälp av HM-10 Bluetooth och Web Bluetooth API, vilket gör att telefonen och Arduinon kan kommunicera tvåvägs med varandra utan att behöva installera några externa program.

Webbapplikationen innehåller olika knappar som möjliggör styrning av motorn samt bromssystem, och även en realtidsvisning för olika mätdata som hastighet, acceleration, ryck, en räknare för signaler från fram och bak hall-sensorer, motorns status visas längst upp på applikationen och uppdateras varje 200 ms och två grafer för hastighet och acceleration kommer upp när man avslutar loggen, se Bilaga D.

Webbapplikationen har även en dataloggfunktion där data sparas vid varje inkommande data som Arduino skickar istället för att spara data istället för en separat timer, vilket ger ett mer noggrant värde mellan mätningen och loggningen och eliminerar onödiga fördröjningar. Som visas i Figur 7 där varje data som tas emot från Arduinon registreras direkt i loggen. Två grafer för hastighet och acceleration dyker upp efter att loggen har avslutats. Data som har samlats kan sedan laddas ner och sparas som en Excel-fil. Med hjälp av värdena kan grafer för hastighet och acceleration skapas för att vidare utforska ryckförändringar.

Figur 7: En överblick över dataloggen.

För att underlätta hanteringen av motorn, PWM och bromssystemet skapades en inställningspanel. Vid ändring av motorns hastighet via PWM ändras servomotorns vridvinkel för en mjukare eller hårdare inbromsning.



Figur 8: En överblick över inställningspanelen

3.5 Elektrisk koppling

Elkretsen och programkoden hänger samman med information på grund av signalsändning från Arduino pin-portar där varje pin har en siffra som sedan definieras i programkoden för signalsstyrning. För att på ett tydligt och enkelt sätt beskriva hur kretsen har kopplats ihop har ett kopplingsschema gjorts med hjälp av Cirkit Designer, se Bilaga E. Och även tabellen nedan som visar hur strömförsörjningen för varje delkomponent har kopplats samt Arduino-kopplingen.

Från-komponent	Från-Pin	Till-komponent	Till-Pin	Kabelfärg
Batteri 12V	Positiv-pol	Arduino UNO Q	VIN	Röd
Batteri 12V	Positiv-pol	BTS7960	B+	Röd
Batteri 12V	Positiv-pol	LM7805	IN(vänster ben)	Röd
Batteri 12V	Negativ-pol	GND-Buss	Gemensam	Blå
LM7805	OUT(höger ben)	Servo	Röd Ledning	Vitt
LM7806	GND(mitten ben)	GND-Buss	Brun Ledning	Blå
BTS7960	M+	DC-motor	Terminal 1	Röd
BTS7960	M-	DC-motor	Terminal 2	Svart

BTS7960	GND	GND-Buss	Gemensam	Blå
BTS7960	VCC	Arduino UNO Q	5V	Röd
BTS7960	R_IS	Arduino UNO Q	Pin 1	Svart
BTS7960	R_EN	Arduino UNO Q	Pin 2	Orange
BTS7960	RPWM	Arduino UNO Q	Pin 3	Lila
BTS7960	L_IS	Arduino UNO Q	Pin 4	Brun
BTS7960	L_EN	Arduino UNO Q	Pin 5	Gul
BTS7960	LPWM	Arduino UNO Q	Pin 6	Grön
HM-10	TXD	Arduino UNO Q	Pin 7	Vitt
HM-10	RXD	Arduino UNO Q	Pin 8	Lila
HM-10	VCC	Arduino UNO Q	5V	Röd
HM-10	GND	GND-Buss	Gemensam	Svart
Servo	Orange Ledning	Arduino UNO Q	Pin 9	Gul
Servo	Brun Ledning	GND-Buss	Gemensam	Blå
Hall-sensor 1-2	S (signal)	Arduino UNO Q	Pin 10-11	Lila
Hall-sensor 1-2	VCC	Arduino UNO Q	5V	Röd
Hall-sensor 1-2	GND	GND-Buss	Gemensam	Blå
MPU-6050	SDA	Arduino UNO Q	SDA	Brun
MPU-6050	SCL	Arduino UNO Q	SCL	Lila
MPU-6050	VCC	Arduino UNO Q	3,3V	Orange
MPU-6050	GND	GND-Buss	Gemensam	Blå

Tabell 1: elektronikkoppling för fordonets komponenter

3.6 Beräkning

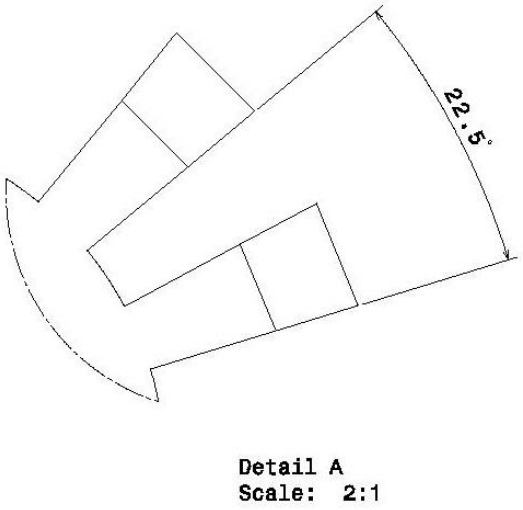
Det här avsnittet handlar om beräkningar som ingår i projektet för framtagning av olika parametrar som är relevanta för fordonets konstruktionsframgång.

3.6.1 Hall-effekt skiva periodberäkning

Båglängd beräkning per puls från en hallsensor

$$L_{Puls} = \frac{D * \pi}{n} = \frac{0.130 * \pi}{16} = 0.02553\,m = 25.5\,mm$$

där D är diametern på hel hall-effektskivan och n är antalet blad den har.
Det innebär att varje puls är då en förflyttning av fordonet 25.5 mm, vilket ger en bättre upplösning för hastighetsmätning. En hel period motsvara då $\frac{360°}{16} = 22{,}5°$ som visas i Figur 9.



Figur 9: Hall-effekt skiva signal period

3.6.2 Hastighet beräkning av tidsperiod

Hastigheten beräknas med hjälp av periodmätning, då man mäter tiden mellan två pulser efter varandra:

$$v = \frac{L_{puls}}{\Delta t} = \frac{0.02553}{\Delta t} \text{ m/s}$$

där Δt är tiden mellan två pulser i sekunder. Periodmätning valdes som lösning eftersom den ger ett mer korrekt hastighetsvärde jämfört med att räkna pulser inom ett intervall, särskilt vid väldigt låg hastighet när det inte kommer in många pulser under ett kort intervall.

3.6.3 Timeout för v_{min}

En timeout i systemet implementerades för att identifiera när fordonet är stillastående. En timeout på 0.4 sekunder startar när inga pulser läses in av hall-sensorerna. Minsta hastigheten då fordonet kan ha utan att timeouten sätts på beräknas enligt

$$v_{min} = \frac{L_{puls}}{t_{timeout}} = \frac{0.02553}{2.0} = 0.013\, m/s.$$

3.6.4 Förbättring av hastighetsberäkning

Tester och beräkningar utfördes för att jämföra hall-effektsskiva med 16 blad och 8 blad för att utreda hur stor skillnad i upplösningen den påverkar.

$$L_8 = \frac{D*\pi}{n} = \frac{0.130*\pi}{8} = 51.0\,mm,$$

$$L_{16} = \frac{D*\pi}{n} = \frac{0.130*\pi}{16} = 25.5\,mm,$$

$$Förbättring = \frac{51.0}{25.5} = 2 = 200\%\ bättre\ upplösning.$$

En fördubbling av upplösningen med en 16-bladig hall-effekt skiva, vilket är förväntat eftersom dubbelt så många signaler samlas under samma period som med en 8-bladig hall-effekt skiva. Det är en fördubbling av noggrannheten vid låg hastighet.

3.6.5 Acceleration och ryck

MPU6050 jobbar på flera känsliga mätområden där skalan är ±2 g, ±4 g, ±8 g, ±16 g där ±2 g är känsligast. ±2 g med 16-bitarsupplösning valdes där g är upplösningen. Accelerometern returnerar rådata som sedan divideras med 16 384 LSB (minst signifikanta bit) för att få fram g:

$$= 2^{16} = \pm 32768\,LSB,$$

$$g = \frac{32768}{2} = 16384\,LSB,$$

$$a_{MPU} = \frac{a_{rådata}\,LSB}{g} * 9.81\,m/s^2 \Rightarrow a_{MPU} = \frac{a_{rådata}\,LSB}{16384\,LSB} * 9.81\,m/s^2$$

Genom att vikta accelerationerna från hall-sensorn och MPU-6050, där a_{Hall} är 70% och a_{MPU} är 30% beräknas accelerationfusionen

$$a_{Hall} = \frac{v_{nu} - v_{föregående}}{\Delta t},$$

$$a_{fusion} = 0.7 * a_{Hall} + 0.3 * a_{MPU}.$$

Det är ett verkligt värde och lämnades utan accelerationfilter. Accelerometern jobbar på en frekvens på 100 Hz, vilket motsvarar 10 millisekunder per mätvärde.

Genom att derivera accelerationen kan ryck beräknas då

$$j = \frac{\Delta a}{\Delta t} = \frac{a_{nu} - a_{föregående}}{\Delta t}.$$

Genom att alltid spara den föregående accelerationen innan registrering av den nya kan ryck räknas ut genom att ta skillnaden dividerat med skillnaden i tiden mellan de två värdena.

Ryck-värdet går igenom ett lågpassfilter där brus som stör ryck-värdet elimineras

$j_{filtrerad} = 0.8 * j_{föregående} + 0.2 * j$.

Ryckfiltret och viktfördelningen för sensorfusion valdes utifrån tester och observationer där MPU-6050 bidrar med 30% av sin vikt för en snabbare respons vid hastighetsförändringar. Hall-sensorn bidrar med 70% av sin vikt eftersom den mäter hjulrörelse direkt utan brus eller elektromagnetiska störningar från omgivningen. Där används en kombination av accelerometerns snabbhet och Hall-sensorns noggrannhet *(Joy-IT, u.å.)*.

Ryckfiltret valdes genom att utföra flera tester där det visade sig att ett lägre filter på 50% på föregående och 50% för nya gav högt brus på värdena på grund av olika små vibrationer från omgivningen, och ett högre värde som 90% och 10% gav nästan inget ryckvärde där ryck värdet var noll även vid broms, vilket resulterade i att 80% och 20% gav ett jämnt värde.

3.6.6 Effektiv spänning och PWM-nivå

Motorns effektiva spänning beräknas utifrån valt PWM-nivå:

$$U_{eff} = \frac{PWM}{255} * U_{batteri} = \frac{30}{255} * 12 = 1.41\,V$$

Där 255 är på grund av att Arduino kör på 8-bitars PWM ($2^8 = 256$) där räkningen börjar från 0. Tester har genomförts för att bestämma vilket PWM-värde som ger fordonet en mjuk, låg och kontrollerad starthastighet. PWM 30 gav bra resultat och valdes som standardvärde. 30 PWM motsvarar 12% av maximal driftström, se Figur 8.

3.6.7 LM7805 regulator dimensionering

LM7805 valdes för att försörja DS3218-servomotorn med 5V från ett batteri som ger 12V spänning:

$U_{ut} = 5\,V, \quad U_{in} = 12\,V, \quad I_{max} = 1.5\,A\ per\ regulator, \quad I_{tot} = 3\,A\,,\ I_{servo} = 2\,A$

Eftersom servomotorn kräver 2 A för att kunna försörja sig vid maximal last. Två regulatorer kopplades för att försörja servomotorn med en total ström på 3 A. Där den har en säkerhetsmarginal på 1 A över sitt maximala strömuttag, vilket säkerställer en stabil drift vid full belastning.

4. Resultat

Avsnittet redovisar resultatet efter testkörning av fordonet genom beräkningar av olika relevanta mätvärden, som bland annat fås via Hallsensor för hastighetsmätning.

4.1 Fordonets hastighetsanalys

Resultatet av fordonet beskrivs vara relevant och stämmer överens med förväntade data genom att ta ett medelvärde av två olika resultat från Hallsensorerna som sitter på fordonets axlar. Resultatet visar i princip en konstant hastighet under lång körsträcka, men några värden diffar lite ibland på grund av magneternas storlek, som fördelas över 16 blad. Det betyder att pulsen registrerar en något kortare eller längre båglängd med några decimalers noggrannhet. Vid högre hastighet kan värdet bli mer exakt, eftersom hastigheten försummar den lilla skillnaden.

Figur 10: En överblick över dataloggen

Tabell 2 och datalogen ovan visar hur hastigheten ser ut vid testkörning där data under 12 sekunder med 60 mätvärden registreras. Hastigheten varierade mellan 0.120 och 0.509 m/s. Vid jämn körning stabiliserades hastigheten kring 0.204-0.243 m/s. Detta tyder på att det konstanta värdet inte är särskilt noggrant, men är ganska nära konstant. Den maximala hastigheten registrerades vid 0,509 m/s under tidspunkten 8.870 sekunder precis innan bromsaktiveringen. Resultatet närmade sig det konstanta läget då underlaget är jämnt och har körts utan störningar, som bland annat upp- eller nedkurvor, men inte lika exakt för de andra värdena, som acceleration eller ryck. Detta beror på att IMU-sensorn är känslig och kan upptäcka små störningar mycket snabbare än en Hallsensor. Mer

resultatanalys och detaljer om acceleration och ryck redovisas senare i rapporten. Fordonet visar 0 m/s i hastighet när fordonet har stannat och detta beror på implementeringen av 0.4 sekunders timeout där hastigheten automatiskt nollställs.

4.2 Analys av acceleration och retardation

Resultatet av accelerationen blev rimligt uppskattat på grund av kombinationen av mätdata från både MPU-sensor och Hallsensor. Genom att derivera hastighetsvärdet och sedan jämföra det med resultatet från MPU-sensorn kan ett mer tillförlitligt resultat uppnås. MPU-sensorn ses vara väldigt känslig och kan reagera på den minsta lilla rörelsen. Detta gav flera orimliga svar i början. När kombinationen implementerades blev resultatet bättre vid fordonets rörelseförändring. Implementeringen fungerar som ett filter där 70% av datan från hastigheten har tagits och 30% från MPU-sensorn. Med hjälp av den här metoden jämnade ut störningen som fås från servomotorn. Dataloggen kan inkludera både acceleration och retardation och detta kan upptäckas när resultatet får ett negativt tecken.

I början av körningen visades hög acceleration som minskade när hastigheten närmade sig konstant driftläge. Motsatt reaktion visades vid bromsning som speglade retardation.

Enligt Tabell 2 visar att vid stabil hastighet på 0.204-0.243 m/s registrerades accelerationsdata i ett intervall mellan -0.047 till 0.0139 m/s², vilket är nära noll. Vid bromsning ökade accelerationssvärdet gradvis till -0.367 m/s² i takt med att bromsvajer-systemet spändes.

Tid (s)	Hastighet (m/s)	Acceleration (m/s²)
8.870	0.509	-0.103
9.095	0.293	-0.051
9.283	0.272	-0.199
9.470	0.197	-0.238
9.695	0.160	-0.367

9.883	0.123	-0.040
11.874	0.000	-0.062

Tabell 2: Accelerationsmätdata

4.3 Ryckanalys

Ryck beräknades med hjälp av den diskreta derivatan av accelerationen med ett mätintervall på 10 millisekunder. Ett lågpassfilter implementerades där 80% av första värdet adderas med 20% av nästa ryckvärde, för att minska brus. Metoden beskrivs mer i detalj i tidigare metodberäkningsstycke som handlar om ryck och acceleration.

Test 1:
Värdet på ryck vid jämn körning varierade mellan -1.192 och 1.397 m/s³. Det högsta värdet uppkom precis vid bromsaktivering. Direkt efter bromsningen registrerades värdet -2.072 m/s³ vid tidpunkten 9.095 sekunder, vilket är väldigt rimligt eftersom bromsen introducerade en kraftig förändring i retardation, se Tabell 3.

Test 2:
Vid tidpunkten 12.587 sekunder visade det ett extremt högt ryckvärde, på grund av att bromssystemet aktiverades vilket resulterade ett ryckvärde på 4.893 m/s³. Det extrema värdet är på grund av retardation vid bromsningen. Vid accelerationen på tidpunkten 3.814 sekunder när fordonet var stillastående, ett ryckvärde på 3.100 m/s³ betraktades när fordonet accelererade från 0.164 till 0.185 m/s. Data hänvisas i Bilaga F.

Testerna visade hur kraftiga hastighetsskiften är för ett fordon med låg hastighet vid acceleration från stillastående och vid bromsning. Ryckvärdet var högst vid bromsningen i de två testerna som utfördes, vilket är fysikaliskt motiverat eftersom ett bromssystem utför en mer abrupt kraftförändring än en gradvis motoracceleration.

Tid (s)	Hastighet (m/s)	Ryck (m/s³)
0.060	0.210	0.254
0.321	0.392	-0.203
0.473	0.206	-1.552

0.661	0.204	1.397
0.886	0.198	-0.004
———		
8.870	0.509	1.093
9.095	0.293	-2.072
9.283	0.272	1.156
9.470	0.197	-0.184
9.695	0.160	-0.991

Tabell 3: Ryck mätdata

4.4 Bromssystemets funktion

Bromssystemet implementeras med hjälp av en cykelbroms där servomotorn drar i en vajer som är kopplad till bromsen som i sin tur klämmer åt skivan som sitter fast på axeln. Funktionen fungerade vid 80° i servomotorns vinkel som startpunkt. Detta upptäcktes då klämstyrkan uppnåddes. Därefter går det att styra med PMW för justering av bromsstyrka. Servomotorn går tillbaka till sitt ursprungliga läge direkt efter att bromsen har avstannat, i väntan på nästa signal. Signalen fås via applikationens tryckknapp eller nödstoppsknappen. Vid aktiveringen stoppades motorn omedelbart.

4.5 Human-machine interface (HMI)

En webbapplikation utvecklades för styrning av fordonet där kommunikationen skedde via en 4.0 Bluetooth-modul. Applikationen erbjöd även realtidsvisning av hastighet, acceleration och ryck samt funktionell styrning av fordonets motor och bromssystem. Värdet loggas med en samplingsfrekvens på 5 Hz för varje Bluetooth-paket. De loggade värdena kan sedan laddas ner i CSV-format för vidare analys där värdena är uppdelade i sex 34olika kolumner som visas i Bilaga F.
Inställningspanelen installerades i applikationen för att möjliggöra snabb anpassning där PWM-värdet på motorn och servomotorns bromsvinkel kan justeras utan att ändra på mjukvaran.

4.6 Felkällor

Under testkörningen identifierades några felkällor som påverkade mätresultatets noggrannhet.

Hall-sensorer felkällor:
Hastighetsvärden från Hallsensor introducerade ojämna hastighetsvärden beroende på hastighetsskivans rotationshastighet. Med båglängden 25.5 mm/s per puls där magneterna hade olika dimensioner på grund av magnetklippning. Storleken på magneterna har påverkat pulssignalen. En drifthastighet på 0.200 m/s genererade en ny puls var 127 ms. Det innebär att hastighetsuppdateringen hände med ojämna intervall som resulterade i variation i det beräknade accelerationsvärdet på kring ± 0.100 m/s² vid konstant hastighet, vilket observerats i tidigare mätdata där accelerationen visade varierande värden som ligger mellan -0.099 och 0.234 m/s² under oföränderlig körning.

MPU-6050 brus och störningar:
Servomotorns placering nära IMU-sensorn introducerade en ny felkälla. Där PWM-signaler via I2C-kommunikationen till servomotorn genererar elektromagnetiska störningar, vilket påverkar accelerometerns värde. För att minska bruset kopplas servomotorns PWM-signaler av när servomotorn inte är aktiverad. Kvarstående brus reducerades med hjälp av sensorfusionsmetoden, men eliminerades inte helt. Ett approximationstal på -0.099 till 0.234 m/s² kvarstod, vilket kan orsakas av olika omständigheter som friktion mellan däcken och marken eller helt enkelt inte tillräckligt slät mark.

Arduino UNO Q Kommunikation:
Arduino har en inbyggd hårdvara som tillåter kommunikation via Bluetooth i bakgrunden utan att blockera resten av programmet. För extern kommunikation fungerar inte hårdvaran på Arduino UNO Q-modellen. Det löstes genom att använda bitbanging-metoden istället. Där Arduino läser in bit via bitbanging där den blockerar pulser som kommer från hall-sensorerna. Chansen att den missar en puls är ganska liten, men inte noll, eftersom bitbanging tar 1 millisekund per bit som den läser in.

Faktabaserad filter parametrar:
Sensor fusionen som räknar ut accelerationen har en viktfördelning på 70% och 30% och beräkningen av ryck har ett lågpassfilter på 80% och 20% dessa parametrar valdes empiriskt genom observation av mätdata i stället för användning av analytisk beräkning.

Det innebär att dessa parametrar är optimerade för det läge som efterfrågas i projektet och kan påverka resultatet. Ändringen av parametrarna är ibland nödvändig vid ett ändrat underlag eller vid högre hastighet.

5. Diskussion

Systemet innehåller många begränsningar där känsligheten hos MPU-6050, hall-sensorernas upplösning och noggrannhetsmätning via bitbanging-kommunikation utgör störningar. Påverkan av dessa begränsningar har inte hindrat systemet från att uppnå sitt syfte, att vid låg hastighet kunna räkna ut hastighet, acceleration och ryck. Det visade att ryckvärdet som systemet gav var fysiskt motiverat. Filterkonstanten för fusionssensorn visade ett bra värde, men under speciella testförhållanden, då konstanten valdes ut genom praktiska tester, skulle den ha förbättrats genom att utföra en analytisk kalibrering. Vilket hade möjliggjort ett kalibrerat system för olika testmiljöer snarare än ett specifikt.

Testkörningen gav observationer av intressant värde för ryck där ryckvärdet var mer känsligt för rörelseskiften än för accelerationen. Loggen visade tydliga värde på hur ryck skiftade sig under acceleration från stillastående. Det visade att ryck värde var som högst under broms aktiveringen enligt Tabell 3 och Bilaga F, vilket är en bra mätdata att användas inom fordon dynamiska studier. Det hör ihop med ryck inom lastbils perspektiv är en kritisk faktor där den utsätter framför allt förarens säkerhet samt lasten för fara. Små förändringar i hastigheten kan orsaka märkbara ryck effekter vilket påverkar komforten hos föraren.

Att använda en webbapplikation som kommunikationssätt med systemet för styrning och datainsamling är en praktisk och flexibel lösning. Webbapplikationen underlättade testkörningen och sparade mycket tid med hjälp av inställningspanelen, där ändringen av motorns hastighet och servomotorers vridvinkel underlättades för att utföra flera tester utan att behöva justera eller ladda upp en ny programkod varje gång. Men användningen av Bluetooth innebär en ny begränsning, där programmet kör via bitbanging, vilket försämrar upplösningen på resultaten. För att eliminera risken för störningar i framtida tester bör Arduino UNO Q ersättas med en mikrokontroller där UART-funktionen är tillgänglig för extern överföring.

En fordonsprototyp som är billig i kostnad och har tillgänglig hårdvara har kunnat användas för att framställa ett system där värdena kan användas på ett meningsfullt sätt för att spegla en verklig simulation av en lastbil som kör i låg hastighet. Betydelsefulla mätvärden kan plockas upp och analyseras utan avancerad utrustning, vilket öppnar en värld av fordonsdynamik och analys av fordon som kör i låg hastighet, där forskning kan ta sin gång.

När det gäller vidare forskning finns det flera intressanta riktningar som kan utvecklas. En intressant faktor kan vara fortsättningen av studier av hur ryckvärdet förändras i olika situationer, som bland annat vid olika bromskrafter och lastnivåer. Till exempel genom att

lägga till en större massa och studera hur bromsbeteendet och ryckvärdet förändras. En intressant studie kan även vara att undersöka hur olika underlag påverkar ryck, tillexempel testkörning på asfalt, betong och gummigolv, vilket möjliggör jämförelse av bromsdynamiktester. Ett annat utvecklingsprojekt kan vara att installera en IMU-sensor separat på ett av hjulen för att mäta hastighet eller ryck. Men detta kan behöva ett mindre Arduinosystem med en strömkälla och en kommunikationskrets som kan skicka löpande information. Detta möjliggör ett mer trovärdigt informationsflöde som hjälper till att jämföra olika ryckvärden med varandra och även en noggrannare hastighetsmätning. För ett noggrant och trovärdigt mätresultat rekommenderas ett jämförelsetest med ett kalibrerat referensinstrument. Att jämföra med ett simuleringsverktyg skulle också vara intressant.

För att skydda fordonet från damm eller skador rekommenderas att utveckla ett hölje som passar elektronikkomponenterna eller hela fordonet. Att 3D-Printa ett hölje som håller fast kablarna i samma position för just breadboardet är nödvändigt. Detta kan vara ett säkrare sätt eftersom en liten ändring i kablarna kan orsaka en kortslutning som kan leda till oväntade och allvarliga konsekvenser.

Fordonet har potential att användas i olika utbildningssammanhang. Det fungerar som ett pedagogiskt verktyg för att enkelt beskriva grundläggande begrepp inom fordonsdynamik. Den billiga kostnaden och den enkla konstruktionen kan möjliggöra reproduktion i ett utbildningssyfte, Det ökar förutsättningarna för ett par praktiska experiment som annars kan vara dyra om de utförs med andra instrument.

6. Slutsats:

Fordonsprototypen konstruerades för att detektera mätdata, bland annat hastighet, acceleration och ryck. Mätdatan verifierades med två testkörningar och visade sig uppfylla de angivna målen. Konstruktionen visade sig vara en kostnadseffektiv lösning som kombinerar sensorfusion och kan detektera tillförlitliga mätresultat för rörlesevariationer vid låga hastigheter.

Detta resultat visade tydliga siffror som tyder på att ryckvärdet är som högst vid bromsningen och i början av körningen, då fordonet accelererar, vilket är fysiskt motiverat. Bromsningen bidrog med en mer oförväntad kraftförändring än motoracceleration, eftersom kraften som drar fordonets tyngd under kort tid är högre, vilket resulterar i en mer abrupt reaktion. Mätresultatet har även visat skillnad vid körning på olika underlag där ryck visade mindre variationer när den körde på ett jämnt underlag än på asfalt.

Projektet utgör en bra grund för vidare forskning och utveckling inom fordonsdynamik vid låg hastighet. Föreslagna förbättringar kan vara en IMU-sensor som sitter på hjulen, analys av filterkonstanter och test av fordonet på olika underlag för mer heltäckande mätresultat. Tilläggsfunktioner är även möjliga men kräver ändringar i elkretschema och förnyad programkod. Tillägg av ett 3D-printat skyddshölje kan också vara intressant.

7. Referenser

Hayati, H., Eager, D., Pendrill, A.-M., & Alberg, H. (2020). *Jerk within the context of science and engineering—A systematic review*. Vibration, 3(4), 371–409. https://doi.org/10.3390/vibration3040025

EngThink. (2023, januari 1). *HALL EFFECT SENSORS, Magnetic sensors #halleffect*. Youtube. Hämtad 20 mars 2026, från https://www.youtube.com/watch?v=bTubDmesSTI

Shawn. (2020, januari 17). *What is an IMU sensor? Overview with Arduino usage guide*. Seeed Studio. Hämtad 20 mars 2026, från https://www.seeedstudio.com/blog/2020/01/17/what-is-imu-sensor-overview-with-arduino-usage-guide/

DwyerOmega. (u.å.). *What is an accelerometer?* Hämtad 20 mars 2026, från https://www.dwyeromega.com/en-us/resources/accelerometers

LaboTest AB. (u.å.). *Accelerometer*. Hämtad 20 mars 2026, från https://www.labotest.se/17/81/accelerometer/

Dingman, J. (u.å.). *What is a gyroscope?* Honeywell. Hämtad 20 mars 2026, från https://aerospace.honeywell.com/us/en/about-us/blogs/what-is-a-gyroscope

Daisch Sensor. (u.å.). *Understanding magnetometers for an IMU system*. Hämtad 22 mars 2026, från https://daischsensor.com/understanding-magnetometers-for-an-imu-system/

Monolithic Power Systems. (u.å.). *Brushless vs. brushed DC motors: When and why to choose one over the other*. Hämtad 22 mars 2026, från https://www.monolithicpower.com/en/learning/resources/brushless-vs-brushed-dc-motors

TYHE Motors. (u.å.). *How does a DC motor work?* Hämtad 22 mars 2026, från https://www.tyhemotors.com/sv/blog/how-does-a-dc-motor-work366

SEW-EURODRIVE. (u.å.). *Servomotorer – synkrona och asynkrona*. Hämtad 22 mars 2026, från https://www.sew-eurodrive.se/produkter/motorer/servomotorer/servomotoren_3.html

Handson Technology. (u.å.). *BTS7960 high current 43A H-bridge motor driver user guide* [PDF]. Hämtad 22 mars 2026, från https://www.handsontec.com/dataspecs/module/BTS7960%20Motor%20Driver.pdf

Arduino. (u.å.). *UNO Q WiFi*. Hämtad 22 mars 2026, från https://store.arduino.cc/products/uno-q

Arduino. (u.å.). *Analog output*. Arduino Documentation. Hämtad 22 mars 2026, från https://docs.arduino.cc/learn/microcontrollers/analog-output/

Joy-IT. (u.å.). SEN-MPU6050 motion sensor (gyroscope/accelerometer).Hämtad 5 maj 2026, från https://www.joy-it.net/en/products/SEN-MPU6050

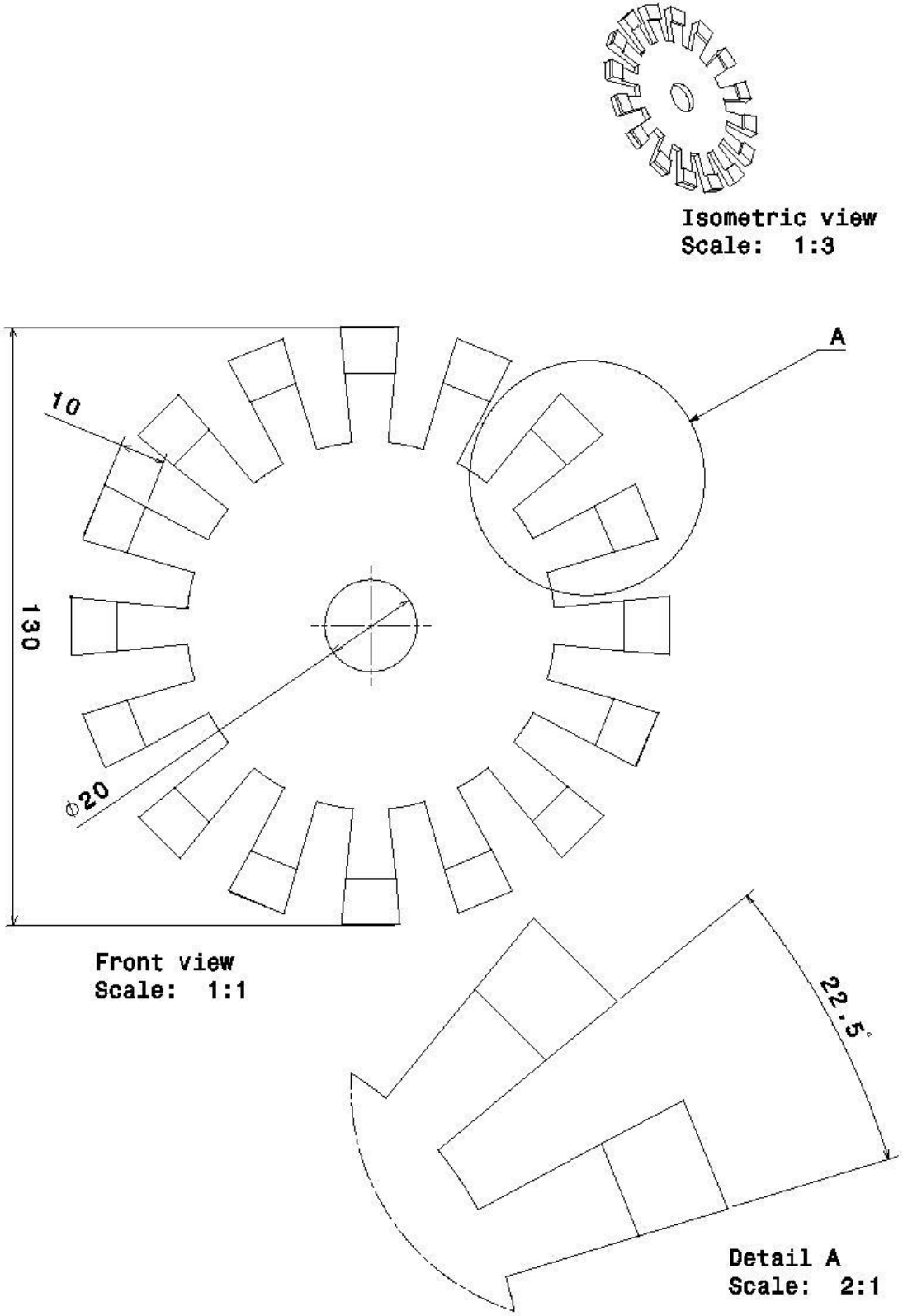
Eager, D., Pendrill, A.-M., & Reistad, N. (2016). *Beyond velocity and acceleration: Jerk, snap and higher derivatives*. *European Journal of Physics, 37*(6), 065008. https://doi.org/10.1088/0143-0807/37/6/065008

8. Bilagor

Bilaga A: Kravspecifikation

Kriterier	Krav/Önskemål	Målvärde	Enhet	Verifiering	Referens
Hastighet	Krav	0-1	m/s	Sensormätning	Examinator
Längd	Krav	ca 0,5	m	Mätning	Examinator
Vikt	Önskemål	Max 15	kg	Vägning	Projektteamet
Kostnad	Önskemål	Max 3500	kr		Projektteamet
Ryck mätning	Krav		m/s^3	IMU-sensor	Examinator
Friktionsbroms	Krav	upp till 13	kg	Beräkning	Examinator
Inbyggd styrenhet	Krav			Arduino	Examinator
Fjärrstyrning	Önskemål			Bluetooth	Examinator
Inbyggd Minne	Önskemål	Max 16	GB		Examinator
Snabb ryck dokumentering	Krav	Max 2	GB	Projekt	Examinator

Bilaga B: Hall-effekt skiva



Bilaga C: C++ programkod

```cpp
#include <Servo.h>
#include <Wire.h>
#include <Arduino_RouterBridge.h>

#define MPU_ADDR 0x68

int R_IS = 1;
int R_EN = 2;
int R_PWM = 3;
int L_IS = 4;
int L_EN = 5;
int L_PWM = 6;

Servo broms;
const int SERVO_PIN = 9;
int CFG_BROMS_PA = 80;
int CFG_BROMS_AV = 0;
int CFG_PWM = 30;

const int BT_RX = 7;
const int BT_TX = 8;
const int BAUD_DELAY = 104;
String btBuffer = "";

const int HALL_FRAM = 10;
const int HALL_BAK  = 11;
const float METER_PER_PULS = (0.130f * 3.14159f) / 16.0f;

volatile unsigned long hallF_prevMicros    = 0;
volatile unsigned long hallF_period        = 0;
volatile unsigned long hallF_senasteMicros = 0;
volatile unsigned long hallB_prevMicros    = 0;
volatile unsigned long hallB_period        = 0;
volatile unsigned long hallB_senasteMicros = 0;
unsigned long HALL_TIMEOUT_US = 400000UL;
volatile unsigned long countFram = 0;
volatile unsigned long countBak  = 0;

float acc_imu  = 0.0f;
float hastighet_ms = 0.0f;
float forraHastighet = 0.0f;
float forraAcc = 0.0f;
float jerk     = 0.0f;
unsigned long tidImu = 0;

bool motorPa = false;
bool korFram = false;
bool korBak  = false;
unsigned long tidSkick = 0;
```

```
50
51  void btSendChar(char c) {
52    digitalWrite(BT_TX, LOW);
53    delayMicroseconds(BAUD_DELAY);
54    for (int i = 0; i < 8; i++) {
55      digitalWrite(BT_TX, (c >> i) & 1);
56      delayMicroseconds(BAUD_DELAY);
57    }
58    digitalWrite(BT_TX, HIGH);
59    delayMicroseconds(BAUD_DELAY);
60  }
61
62  void btPrint(String s) {
63    for (int i = 0; i < s.length(); i++) btSendChar(s[i]);
64  }
65
66  void btPrintln(String s) {
67    btPrint(s);
68    btSendChar('\r');
69    btSendChar('\n');
70  }
71
72  int readBit() {
73    delayMicroseconds(BAUD_DELAY);
74    return digitalRead(BT_RX);
75  }
76
77  char btReadChar() {
78    unsigned long start = micros();
79    while (digitalRead(BT_RX) == HIGH) {
80      if (micros() - start > 10000) return 0;
81    }
82    delayMicroseconds(BAUD_DELAY / 2);
83    char c = 0;
84    for (int i = 0; i < 8; i++) {
85      c |= (readBit() << i);
86    }
87    readBit();
88    return c;
89  }
90
91  void isrFram() {
92    unsigned long nu = micros();
93    if (hallF_prevMicros > 0)
94      hallF_period = nu - hallF_prevMicros;
95    hallF_prevMicros    = nu;
96    hallF_senasteMicros = nu;
97     countFram++;
98  }
```

```
99
100 void isrBak() {
101   unsigned long nu = micros();
102   if (hallB_prevMicros > 0)
103     hallB_period = nu - hallB_prevMicros;
104   hallB_prevMicros    = nu;
105   hallB_senasteMicros = nu;
106   countBak++;
107 }
108
109 String floatToStr(float val, int decimals) {
110   if (isnan(val) || isinf(val)) return "0.000";
111   long mult = 1;
112   for (int i = 0; i < decimals; i++) mult *= 10;
113   long whole = (long)val;
114   long frac = abs((long)((val - (float)whole) * mult));
115   String result = "";
116   if (val < 0 && whole == 0) result = "-0";
117   else result = String(whole);
118   String fracStr = String(frac);
119   while ((int)fracStr.length() < decimals) fracStr = "0" + fracStr;
120   return result + "." + fracStr;
121 }
122
123 void setup() {
124   pinMode(R_IS, OUTPUT); pinMode(R_EN, OUTPUT);
125   pinMode(R_PWM, OUTPUT); pinMode(L_IS, OUTPUT);
126   pinMode(L_EN, OUTPUT); pinMode(L_PWM, OUTPUT);
127   digitalWrite(R_IS, LOW);
128   digitalWrite(L_IS, LOW);
129   digitalWrite(R_EN, HIGH);
130   digitalWrite(L_EN, HIGH);
131   analogWrite(R_PWM, 0);
132   analogWrite(L_PWM, 0);
133
134   broms.attach(SERVO_PIN);
135   broms.write(CFG_BROMS_AV);
136   delay(500);
137   broms.detach();
138
139   pinMode(BT_RX, INPUT_PULLUP);
140   pinMode(BT_TX, OUTPUT);
141   digitalWrite(BT_TX, HIGH);
142
143   pinMode(HALL_FRAM, INPUT_PULLUP);
144   pinMode(HALL_BAK,  INPUT_PULLUP);
145   attachInterrupt(digitalPinToInterrupt(HALL_FRAM), isrFram, RISING);
146   attachInterrupt(digitalPinToInterrupt(HALL_BAK),  isrBak,  RISING);
```

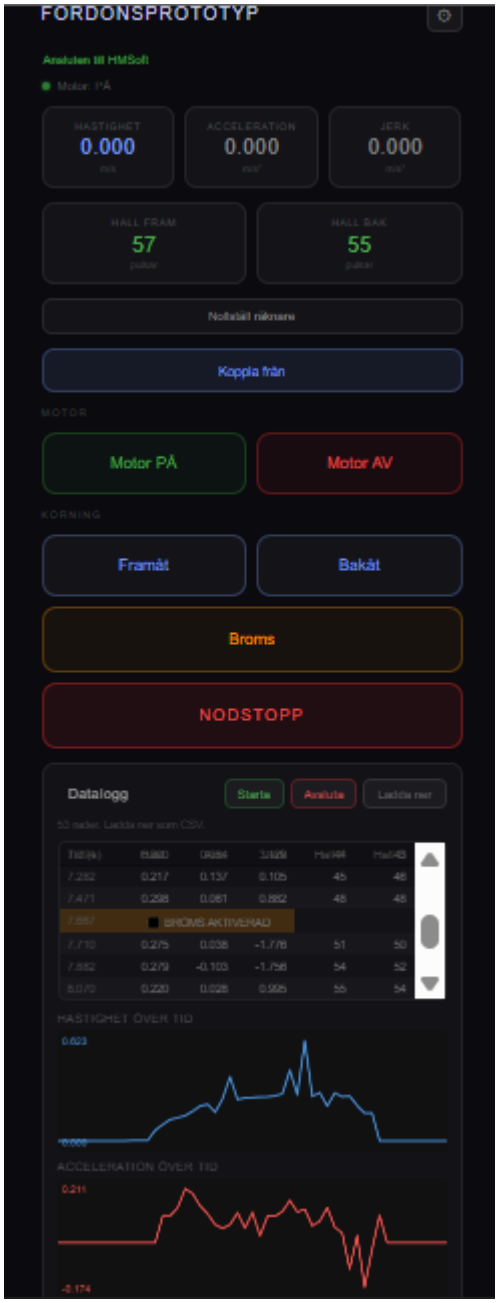
```
147
148     Wire.begin();
149     Wire.beginTransmission(MPU_ADDR);
150     Wire.write(0x6B);
151     Wire.write(0x00);
152     Wire.endTransmission(true);
153
154     tidImu   = millis();
155     tidSkick = millis();
156   }
157
158   void loop() {
159     if (digitalRead(BT_RX) == LOW) {
160       char c = btReadChar();
161
162       if (c == '\n' || c == '\r') {
163         if (btBuffer.length() > 0) {
164           hanteraStraeng(btBuffer);
165           btBuffer = "";
166         }
167       } else if (btBuffer.length() > 0 || c == 'C') {
168         btBuffer += c;
169       } else {
170         hanteraKommando(c);
171       }
172     }
173
174     unsigned long nu = millis();
175     if (nu - tidImu >= 10) {
176       float dt = (nu - tidImu) / 1000.0f;
177
178       forraAcc = acc_imu;
179
180       Wire.beginTransmission(MPU_ADDR);
181       Wire.write(0x3B);
182       Wire.endTransmission(false);
183       Wire.requestFrom(MPU_ADDR, 2, true);
184       int16_t ax = (Wire.read() << 8) | Wire.read();
185
186       if (hastighet_ms > 0.01f) {
187         float acc_hall = 0.0f;
188         if (dt > 0) acc_hall = (hastighet_ms - forraHastighet) / dt;
189         float acc_mpu = -(ax / 16384.0f) * 9.81f;
190         acc_imu = 0.7f * acc_hall + 0.3f * acc_mpu;
```

```
191
192       if (dt > 0) jerk = (acc_imu - forraAcc) / dt;
193       if (isnan(jerk) || isinf(jerk)) jerk = 0.0f;
194       static float jerkFilter = 0.0f;
195       jerkFilter = 0.8f * jerkFilter + 0.2f * jerk;
196       jerk = jerkFilter;
197     } else {
198       acc_imu      = 0.0f;
199       jerk         = 0.0f;
200       forraHastighet = 0.0f;
201     }
202     forraHastighet = hastighet_ms;
203
204     tidImu = nu;
205   }
206
207   if (nu - tidSkick >= 200) {
208     tidSkick = nu;
209
210     unsigned long nuMicros = micros();
211     noInterrupts();
212     unsigned long perF = hallF_period;
213     unsigned long perB = hallB_period;
214     unsigned long senF = hallF_senasteMicros;
215     unsigned long senB = hallB_senasteMicros;
216     interrupts();
217
218     float vFram = 0.0f, vBak = 0.0f;
219     if (perF > 0 && (nuMicros - senF) < HALL_TIMEOUT_US) {
220       float vTemp = METER_PER_PULS / (perF / 1000000.0f);
221       if (vTemp < 2.0f) vFram = vTemp;
222     }
223     if (perB > 0 && (nuMicros - senB) < HALL_TIMEOUT_US) {
224       float vTemp = METER_PER_PULS / (perB / 1000000.0f);
225       if (vTemp < 2.0f) vBak = vTemp;
226     }
227
228     float hastighet = 0.0f;
229     if      (vFram > 0 && vBak > 0) hastighet = (vFram + vBak) / 2.0f;
230     else if (vFram > 0)             hastighet = vFram;
231     else if (vBak  > 0)             hastighet = vBak;
232     hastighet_ms = hastighet;
233
```

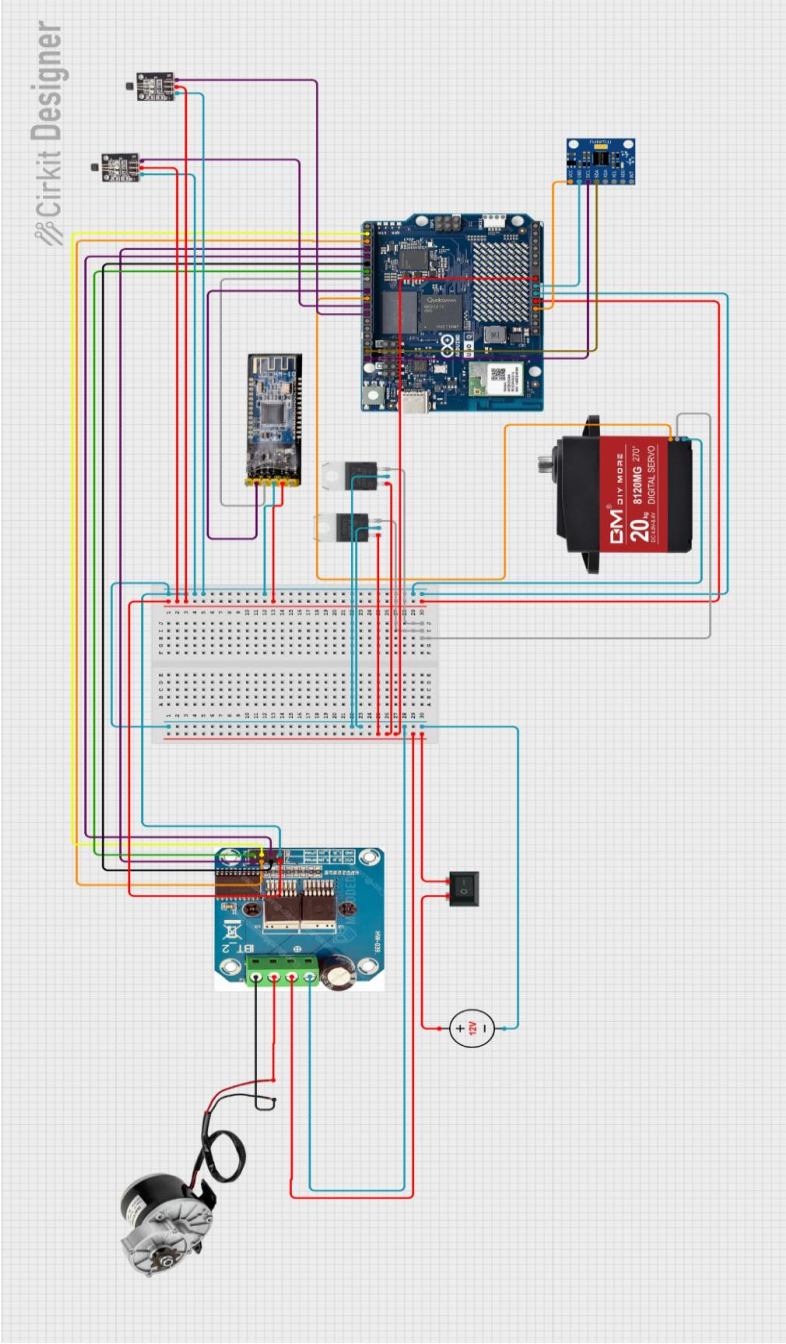
```
    noInterrupts();
    unsigned long cf = countFram;
    unsigned long cb = countBak;
    interrupts();
    btPrint("H:"); btPrint(floatToStr(hastighet, 3));
    btPrint(" A:"); btPrint(floatToStr(acc_imu, 3));
    btPrint(" J:"); btPrint(floatToStr(jerk, 3));
    btPrint(" CF:"); btPrint(String(cf));
    btPrint(" CB:"); btPrintln(String(cb));
  }
}

void hanteraKommando(char c) {
  switch(c) {
    case 'P':
      motorPa = true;
      btPrintln("STATUS:PA");
      break;

    case 'O':
      motorPa = false; korFram = false; korBak = false;
      analogWrite(R_PWM, 0); analogWrite(L_PWM, 0);
      btPrintln("STATUS:AV");
      break;

    case 'F':
      if (motorPa) {
        korFram = true; korBak = false;
        broms.write(CFG_BROMS_AV);
        analogWrite(R_PWM, 0);
        analogWrite(L_PWM, CFG_PWM);
      }
      break;

    case 'f':
      korFram = false;
      analogWrite(R_PWM, 0); analogWrite(L_PWM, 0);
      break;

    case 'B':
      if (motorPa) {
        korBak = true; korFram = false;
        broms.write(CFG_BROMS_AV);
        analogWrite(R_PWM, CFG_PWM);
        analogWrite(L_PWM, 0);
      }
      break;

```

```
    case 'b':
      korBak = false;
      analogWrite(R_PWM, 0); analogWrite(L_PWM, 0);
      break;

    case 'K':
      korFram = false; korBak = false;
      analogWrite(R_PWM, 0); analogWrite(L_PWM, 0);
      broms.attach(SERVO_PIN);
      broms.write(CFG_BROMS_PA);
      break;

    case 'k':
      broms.write(CFG_BROMS_AV);
      delay(500);
      broms.detach();
      break;

    case 'E':
      motorPa = false; korFram = false; korBak = false;
      analogWrite(R_PWM, 0); analogWrite(L_PWM, 0);
      broms.attach(SERVO_PIN);
      broms.write(CFG_BROMS_PA);
      btPrintln("STATUS:AV");
      break;
  }
}

void hanteraStraeng(String s) {
  if (s.startsWith("CFG:")) {
    s = s.substring(4);
    int i0 = s.indexOf(',');
    int i1 = s.indexOf(',', i0+1);
    if (i0 > 0 && i1 > i0) {
      CFG_PWM      = constrain(s.substring(0,i0).toInt(), 5, 255);
      CFG_BROMS_PA = constrain(s.substring(i0+1,i1).toInt(), 0, 180);
      CFG_BROMS_AV = constrain(s.substring(i1+1).toInt(), 0, 180);
      btPrintln("CFG_OK");
    }
  }
}
```

Bilaga D: HMI layout



Bilaga E: Kopplingsschema



För komponenternas namn samt kopplings portar se den bifogade länken:
[Circuit.html]

Bilaga F: Datalogg CSV-file

Tid (s)	Hastighet (	Acceleratic	Jerk (m/s3)	Hall Fram	Hall Bak
0.174	0.000	0.000	0.000	0	0
0.397	0.000	0.000	0.000	0	0
0.587	0.000	0.000	0.000	0	0
0.774	0.000	0.000	0.000	0	0
1.005	0.000	0.000	0.000	0	0
1.187	0.000	0.000	0.000	0	0
1.372	0.000	0.000	0.000	0	0
1.599	0.000	0.000	0.000	0	0
1.787	0.000	0.000	0.000	0	0
1.974	0.000	0.000	0.000	0	0
2.197	0.000	0.000	0.000	0	0
2.383	0.029	0.000	0.000	1	1
2.608	0.029	0.030	-1.764	1	1
2.796	0.054	0.130	-0.541	2	2
2.988	0.122	0.198	0.832	2	2
3.213	0.259	0.074	-0.098	3	3
3.434	0.228	0.085	-1.161	4	4
3.584	0.164	0.086	-0.079	5	5
3.814	0.185	0.203	3.100	6	6
4.000	0.199	0.009	-1.299	7	7
4.187	0.292	0.140	1.019	9	9
4.412	0.184	0.096	-0.662	10	10
4.597	0.213	0.129	1.619	11	11
4.787	0.212	0.086	0.569	13	13
5.012	0.157	0.137	1.062	15	15
5.201	0.237	0.036	-0.608	17	17
5.387	0.229	-0.007	-2.316	19	19
5.612	0.251	0.090	0.460	21	21
5.804	0.261	0.055	0.153	22	22
5.985	0.312	-0.007	-1.362	24	24
6.212	0.229	0.169	0.979	26	26
6.399	0.255	0.092	1.484	28	28
6.587	0.248	0.115	0.352	30	30
6.846	0.261	0.051	-1.262	32	32
6.999	0.287	0.048	-0.855	34	34
7.186	0.265	0.170	1.342	36	36
7.413	0.264	0.066	0.613	38	38
7.637	0.225	0.057	0.168	40	40
7.787	0.210	0.111	0.794	42	42
8.012	0.310	0.069	-0.734	44	44
8.199	0.217	0.024	-2.727	46	46
8.387	0.151	0.114	0.710	49	49
8.612	0.311	0.164	1.184	51	51
8.799	0.231	0.122	-1.697	54	54
8.984	0.204	0.075	-0.519	56	56
9.213	0.213	0.071	-0.255	58	58
9.399	0.216	0.085	-0.509	61	61
9.587	0.197	0.084	0.448	63	63
9.847	0.178	0.227	1.704	66	66
9.999	0.192	-0.016	-2.811	68	68
10.187	0.187	0.234	2.579	69	69
10.412	0.180	0.174	1.081	71	71
10.596	0.183	0.128	-0.646	73	73
10.787	0.313	0.098	0.537	75	75
11.012	0.185	0.074	-2.050	78	78
11.199	0.203	0.206	3.464	80	80
11.387	0.212	0.076	-0.164	82	82
11.612	0.195	-0.032	-1.692	83	83
Broms				84	84
11.796	0.204	0.194	1.501	85	85
12.001	0.188	0.074	-1.202	87	87
12.209	0.201	-0.061	-0.579	89	89
12.399	0.163	-0.099	-1.687	90	90
12.587	0.129	0.158	4.893	92	92
12.812	0.129	0.095	0.460	92	92
12.999	0.129	0.094	0.165	92	92
13.187	0.129	0.089	0.183	92	92

Bilaga G: Fordonguide uppstart

Uppstart av fordonet:

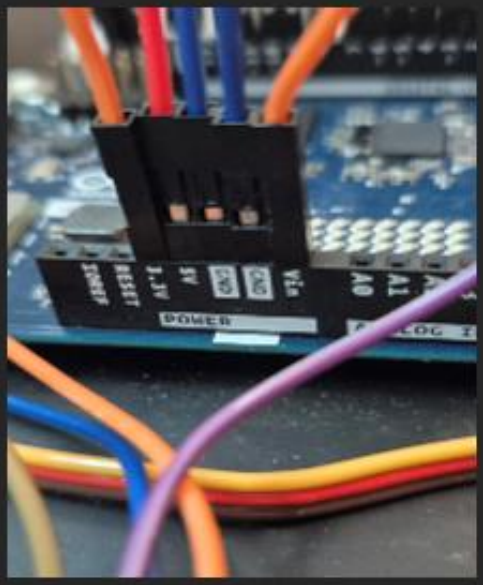
1. Se till att alla kablar är anslutna och att ingenting är löst.
2. Slå på fordonet genom att trycka på på-knappen.

3. Vänta tills LED-displayen på Arduino har laddats klart och ett hjärta visas.
4. Öppna "fordonsprototyp_app.html" på webben.
5. Tryck “Connect Bluetooth” och leta runt I listan efter “HMsoft”.
6. En grön text visas högst upp på webbplatsen med texten “HMsoft connected”.
7. Tryck “Motor On” för att aktivera motorn.
8. Nu borde den vara redo att testas.

Bilaga H: Fordonguide koduppladdning

När du laddar upp en ny kod:

1. Se till att stänga av fordonet helt genom att vrida strömbrytaren tillbaka till avstängt läge.
2. Koppla även bort "vin"-kabeln som är ansluten till Arduino för att undvika att Arduino strömförsörjs från två olika spänningskällor.



3. Nu kan du ansluta USB-C-kabeln för att ladda ner en ny kod.
4. Glöm inte att dra ur USB-C-kabeln innan du ansluter "vin"-kabeln igen och startar fordonet igen.

CHALMERS