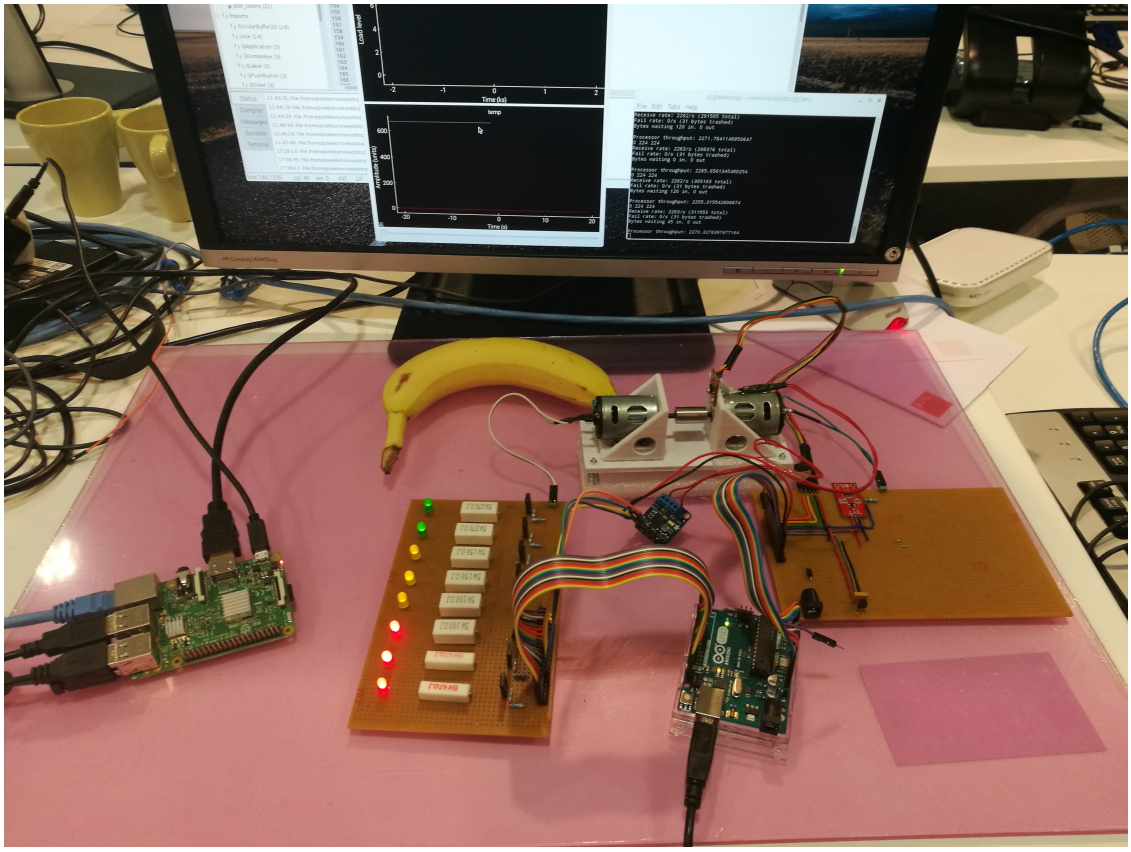


Machine Learning on the Edge

Examensarbete hos Institutionen för Data- och Informationsteknik

JOHAN LÖVGREN & ANTON OLSSON

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA / GÖTEBORGS UNIVERSITET
Göteborg 2017

Machine Learning on the Edge

ANTON OLSSON JOHAN LÖVGREN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg 2017

Machine Learning on the Edge
ANTON OLSSON
JOHAN LÖVGREN

Handledare: Thomas Hallgren, Institutionen för Data- och Informationteknik
Examinator: Peter Lundin, Institutionen för Data- och Informationsteknik

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola / Göteborgs Universitet
412 96 Göteborg
Telefon: 031-772 1000

Cover: En överblick över det system som konstruerades under projektets gång, inklusive hårdvara och visualisering, med en banan för att ge en känsla av storleken.

Machine Learning on the Edge
JOHAN LÖVGREN & ANTON OLSSON
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola / Göteborgs Universitet

Sammanfattning

Maskininlärning är ett dataanalytiskt verktyg som utan djupare kunskap om ett system kan konstruera en modell som kan användas för att analysera detta system. Denna sorts analys är av stort intresse för så kallat prediktivt underhåll, att kunna i förväg upptäcka ett fel i exempelvis ett mekaniskt system. I detta arbete användes en maskininlärningsplattform från företaget Ekkono att för att undersöka möjligheten att förutsäga tillståndet hos en liten DC-motor och hur detta kommer att utvecklas utifrån dess nuvarande tillstånd. Feldetektering anses ligga utanför projektets omfattning. För att genomföra detta konstruerades ett system för datainsamling. Detta bestod av en Arduino Uno som var kopplad till ett annat sensorer som samlade in data om motorn. Denna data analyseras sedan på en Raspberry Pi och då utfördes även prediktionen. Träffsäkerheten hos denna prediktion analyserades sedan.

Machine Learning on the Edge
JOHAN LÖVGREN & ANTON OLSSON
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola / Göteborgs Universitet

Abstract

Machine learning is a tool for data analysis which can construct a model of a system without necessarily requiring deeper insight into the system. This model can then be used for analysis of the system. This type of analysis is of great interest for use with so called predictive maintenance; to be able to discover an abnormality in the behaviour in the system which can lead to the system breaking, before it actually happens. A platform for machine learning was provided by Ekkono and used to predict the state of a small DC motor. The prediction used data from various sensors to gain information about the current state of the system. This data was used to predict how the state would change a short time in the future. Fault detection was not within the scope of this project, we only concern ourselves with predicting how the values read from the sensors would change. In order to achieve this, a system which could be used for data collection was constructed. The system consisted of an Arduino Uno which collected data about the DC motor using various sensors. This data was analysed using a Raspberry Pi. The prediction was also done on the Raspberry. The accuracy of the predictions was then analysed.

Författarnas tack

Vi vill tacka våra handledare Thomas Hallgren, Mazdak Sanati och Mina Alibeigi för åtskilligt stöd och hjälp under arbetets gång.

Vi vill även tacka Cybercom, framförallt Gabriel Ibanez, men även alla andra medarbetare i Innovation Zone för att de gav oss möjlighet att genomföra detta samarbete.

Vi vill också tacka Sakib Sistek och Lennart Widen från Chalmers Tekniska Högskola för deras stöd och råd, framförallt vad gällde konstruktionen av det elektriska systemet. Detta hjälpte oss väldigt mycket eftersom våra kunskaper inom detta område knappast kan beskrivas som expertkunskaper.

Vi vill tacka Ekkono och våra kontaktpersoner hos dem, Rikard König och Peter Alm, för att vi fick tillgång till deras maskininlärningsplattform och deras stöd och hjälp med att använda denna. Vi har under projektets gång mottagit flera uppdateringar och förbättringar av deras plattform och de har kontinuerligt arbetat för att vi skall kunna genomföra detta projekt.

Sist men inte minst vill vi tacka Adrian på GTC för hjälpen med att bestämma vilken sorts motor vi ville ha och att få tag på denna.

Innehåll

Nomenklatur **xv**

Figurer **xvii**

Tabeller **xix**

1 Inledning **1**
1.1 Bakgrund . . . 1
1.2 Syfte . . . 1
1.3 Mål . . . 2
1.4 Avgränsningar . . . 2

2 Teori **3**
2.1 Likströmsmotor . . . 3
2.2 Motor-generator . . . 4
2.3 Reostatisk broms . . . 4
2.4 Enkortsdatorer . . . 5
2.4.1 Arduino . . . 5
2.4.2 Raspberry Pi . . . 6
2.4.3 Komunikation mellan Raspberry PI och Arduino . . . 6
2.5 Ekkonos plattform för maskininlärning . . . 6
2.5.1 Träning av modell . . . 7
2.5.2 Användning av modell . . . 7

3 Systemkonstruktion **9**
3.1 Sammanfattning av hårdvarusystemet . . . 9
3.2 Kravspecifikation . . . 10
3.3 Motor-generator . . . 11
3.3.1 Test av motor-generatorns överföringskaraktäristik . . . 12
3.4 Lastkrets . . . 12
3.4.1 Val av resistorer i lastbanken . . . 14
3.4.2 Val av transistorer som styr lastbankens last . . . 15
3.5 Val av sensorer . . . 16
3.5.1 Strömsensor . . . 16
3.5.2 Accelerometer . . . 17
3.5.3 Rumstemperatursensor . . . 18

3.5.4 Motortemperatursensor 19
3.6 Arduino 20
3.6.1 Arduinons in- och utportar 20
3.6.2 Arduinons mjukvara 21
3.6.2.1 Avläsning av sensordata 21
3.6.2.2 Protokoll för kommunikation mellan Raspberry Pi och Arduino 22
3.6.2.3 Test av sändhastighet hos Arduinon 22
3.7 Raspberry Pi 23
3.7.1 Mottagarprocessen 23
3.7.1.1 Test av mottagarhastighet hos Raspberry Pi 23
3.7.2 Processering- och prediktionsprocessen 24
3.7.2.1 Test av dataprocesseringshatighet 24
3.7.3 Visualiseringsprocessen 25
3.7.4 Kommunikation mellan Raspberry Pi:ns tre processer 26
3.8 Visualisering 27
3.9 Prestandatester av systemet 27
3.9.1 Test av samplingsfrekvens för vibration 28
3.9.2 Test av prestanda för alla ingående system 28
3.9.3 Test av prestanda för prediktion 29

4 Resultat **31**
4.1 En överblick av systemets beteende under datainsamling 31
4.2 Resultat av prestandatester 32
4.3 Resultat av test av prediktion 32
4.3.1 Prediktion av vibration 32
4.3.1.1 5 ms prediktion av vibration i x-led 33
4.3.1.2 10 ms prediktion av vibration i x-led 35
4.3.2 Prediktion av motorström. 37
4.3.2.1 1000 ms prediktion av motorströmmen. 37
4.3.2.2 5 ms prediktion av motorströmmen. 40
4.3.3 Prediktion av motortemperaturen, 5 s. 42

5 Slutsats **45**
5.1 Hur väl har projektet uppnått sitt syfte? 45
5.2 Möjlig vidareutveckling och förbättringar 46
5.2.1 Träning av modeller 46
5.2.2 Koppling mellan motorer, dämpning m.m. 46
5.2.3 Vidareutveckling 47

Bibliography **49**

A Appendix **I**
A.1 Lista av komponenter I
A.2 Testmodeller för prestandatest II
A.3 Resultat av tester av systemet IV
A.3.1 Systemets beteende under datainsamling IV

A.3.2 Uppmätt acceleration V
A.3.3 Uppmätta frekvensspektran VII
A.3.4 Vibrationskurvor IX
A.4 Prediktion av motortemperaturen, 10 ms. X

Nomenklatur

ADC Analog-till-digital-omvandlare, sida 5

BOB breakout board, sida 17

Checksum En sekvens av siffror som beräknas genom att interera över en mängd data. Om datan ändras påverkas också med största sannolikhet checksumman, sida 22

csv comma-separated-values, sida 24

DSI Display interface, sida 6

FFT fast fourier transform, sida 28

GPIO General-purpose input/output, sida 5

I/O input/output, sida 5

mape Mean-absolute-percentage-error, sida 29

ML Maskininlärning, sida 1

Moving average Sv. glidande medelvärde; medelvärdet över x antal datapunkter bakåt., sida 7

PdM Prediktivt underhåll, sida 1

SBC Single-board computer, sida 5

SPI Serial Peripheral Interface, sida 19

Figurer

2.1 En bild på en likströmsmotor. 3
2.2 Motor-generatorkoppling. 4

3.1 Den plattform som motorn monterades på, samt motorerna. 12
3.2 Kopplingsschema för generator och lastbanksanordning. 13
3.3 Lastbanksanordningen som konstruerades. 13
3.4 Kopplingsschema för motor, dess strömförsörjning och shunt-resistor för att mäta motorns strömtillförsel. 17
3.5 Kopplingschema Hall-effekt-baserad strömsensor som sedan byttes ut. 18
3.6 Kopplingsschema för accelerometer. 18
3.7 Kopplingsschema för rumstemperatursensor. 19
3.8 Kopplingsschema för motortemperatursensor. 19
3.9 Kopplingsschema för Arduino. 21
3.10 En bild som visar det grafiska interface som konstruerades. 27

4.1 Överblick av amplitudskillnaden vid 5 ms prediktion av vibration i x-led. 33
4.2 MAPE 1000 vid 5 ms prediktion av vibration i x-led. 34
4.3 Vågformen av 5 ms prediktion av vibration i x-led, vid låg MAPE. . . 34
4.4 Vågformen av 5 ms prediktion av vibration i x-led, vid hög MAPE. . 35
4.5 Överblick av amplitudskillnaden vid 10 ms prediktion av vibration i x-led. 35
4.6 MAPE 1000 vid 10 ms prediktion av vibration i x-led. 36
4.7 Vågformen av 10 ms prediktion av vibration i x-led, vid låg MAPE. . 36
4.8 Vågformen av 10 ms prediktion av vibration i x-led, vid hög MAPE. . 37
4.9 Överblick av amplitudskillnaden vid 1000 ms prediktion av motorströmmen. 38
4.10 MAPE 1000 vid 1000 ms prediktion av motorström. 39
4.11 Lastskifte under ca 1 sekund från 1000 ms prediktion av motorström. 39
4.12 Vågformen av 1000 ms prediktion av motorström. 40
4.13 Överblick av amplitudskillnaden vid 5 ms prediktion av motorströmmen. 41
4.14 MAPE 1000 vid 5 ms prediktion av motorström. 41
4.15 Lastskifte under ca 1 sekund från 5 ms prediktion av motorström. . . 42
4.16 Vågformen av 5 ms prediktion av motorström. 42
4.17 5 s prediktion av motortemperaturen. 43

4.18 MAPE 1000 för 5 s prediktion av motortemperaturen. 44

A.1 Motorströmmen vid de första 2 miljoner insamlade datapunkterna. . IV
A.2 Lasten vid de första 2 miljoner insamlade datapunkterna. IV
A.3 Motortemperaturen vid de första 2 miljoner insamlade datapunkterna. V
A.4 Acclererationen i x-led vid de första 2 miljoner insamlade datapunkterna. V
A.5 Acclererationen i y-led vid de första 2 miljoner insamlade datapunkterna. VI
A.6 Acclererationen z-led vid de första 2 miljoner insamlade datapunkterna. VI
A.7 Frekvensspektrat som uppmättes då samplingsfrekvensen var 500Hz. . VII
A.8 Frekvensspektrat som uppmättes då samplingsfrekvensen var 1000Hz. VII
A.9 Frekvensspektrat som uppmättes då samplingsfrekvensen var 1500Hz. VIII
A.10 Frekvensspektrat som uppmättes då samplingsfrekvensen var 2000Hz. VIII
A.11 En kurva som visar de vibrationer som uppmättes gentemot tiden då samplingsfrekvensen var 500Hz. IX
A.12 En kurva som visar de vibrationer som uppmättes gentemot tiden då samplingsfrekvensen var 1000Hz. IX
A.13 En kurva som visar de vibrationer som uppmättes gentemot tiden då samplingsfrekvensen var 1500Hz. X
A.14 En kurva som visar de vibrationer som uppmättes gentemot tiden då samplingsfrekvensen var 2000Hz. X
A.15 10 ms prediktion av motortemperaturen. XI
A.16 MAPE 1000 för 10 ms prediktion av motortemperaturen. XI

Tabeller

3.1 Data från det första motor-generator-testet (12V motorspänning) . . 13
3.2 Testdata för accelerometer. 18
3.3 Antal GPIO-portar som enskilda sensorer använder samt deras strömförbrukning. 21
3.4 Data från test av Arduinons sändhastighet 23
3.5 Data från test av Raspberryns mottagarhastighet 24
3.6 Data från test av dataprocesseringshastighet 25
3.7 Data från test av det totala systemet. 29

1

Inledning

1.1 Bakgrund

Maskininlärning (ML) är ett dataanalytiskt verktyg som skapar en generaliserad modell av tidigare insamlad data (modellträning), för att hitta mönster i den och för att beräkna förväntad framtida data (prediktion). Speciellt i stora datamängder med många variabler kan mönstren vara svåra att identifiera via traditionell analys.

Prediktivt underhåll (PdM) innebär att analysera driftinformation från utrustning (t.ex. motorer, eller andra mekaniska system) för att kunna förutsäga var och när eventuella problem kommer att inträffa. PdM kan användas i många områden, men kanske främst inom industri, för att öka säkerhet, minska reparationskostnader och minimera driftstopp i produktion.

Även om det finns stor potential att använda ML inom PdM (och andra områden) så är användningen begränsad på grund av ofta svåranvända system. Det finns ett antal tjänster som söker att adressera detta, varav många som är "molnbaserade", som endast använder tidigare uppmätt data för att sedan bygga modellen centralt och därmed ökar användarvänligheten. I många sådana ML-tjänster utförs också prediktioner centralt.

En nackdel med helt molnbaserade produkter är att en internetanslutning krävs, men för många tillämpningar finns ingen eller endast tillfällig internetanslutning. En annan nackdel är att fördröjningar introduceras när modellen finns på annat håll än där det faktiskt behövs, vilket gör modellen oanvändbar i många realtidsapplikationer.

Arbetet ska genomföras hos Cybercom i samarbete med Ekkono, som tillhandahåller produkter inom maskininlärning. Ekkono utvecklar en maskininlärningstjänst som kan träna en modell i molnet, som sedan kan användas, på ett plattformsagnostiskt vis, där systemet behövs, s.k. "on the edge".

1.2 Syfte

Syftet med detta arbete är att tillämpa ML med hjälp av Ekkonos plattform för att analysera hur ett system kommer att bete sig (i syftet att i framtiden möjliggöra

PdM). Detta system skall sedan utvärderas utifrån hur bra dess prediktioner sammanfaller med uppmätta värden. Det existerar produkter på marknaden som fyller denna funktion idag, men som antingen är väldigt specialiserade, komplicerade att använda, eller som inte kan utnyttjas "on the edge". Syftet med detta arbete är också att detta skall fungera som en uppvisningsprodukt för Ekkono och deras tjänst.

1.3 Mål

Målet med projektet är att konstruera ett litet elektriskt system, baserat kring en liten elmotor och en mikrodator (Raspberry PI), som kan utföra prediktion av en elmotors tillstånd (vibration, strömförbrukning, samt temperatur) genom att modellera systemet med hjälp av Ekkonos ML-tjänst och applicera den på kontinuerligt uppmätt sensordata.

Data uppmäts genom sensorer ansluta till elmotorn. Prediktionen samt systemets tillstånd skall i realtid (med en viss fördröjning) åskådliggöras och visuellt jämföras med det verkliga utfallet för att demonstrera prediktionens tillförlitlighet. Visualiseringen samt systemets elektro-mekaniska dynamik ska vidare vara intuitivt att förstå för icke insatta personer (exempelvis potentiella kunder till Ekkono). För att få mer intressanta värden ska vi till motorn koppla en varierbar last.

1.4 Avgränsningar

Arbetet kommer endast att utnyttja Ekkonos plattform för maskininlärning. För ett praktiskt fungerande uppvisningsprodukt för Ekkono krävs att systemet inkapslas på ett adekvat sätt, både för transport och av estetiska anledningar. Detta är inte inkluderat i projektet, annat än som en extrauppgift om tid finnes.

Arbetet kommer endast att undersöka hur väl Ekkonos plattform kan förutsäga systemets beteende i termer av hur dess tillståndsvariabler som till exempel temperatur och vibration kommer att utvecklas i framtiden. Någon feldetektering eller liknande kommer ej ske.

2

Teori

I detta kapitel presenteras teknisk bakgrund som ligger till grund för genomförandet av projektet. Denna information gäller funktionen av de hårdvarukomponenter som har använts inom detta projekt, som till exempel den likströmssmotor som användes och tillhörande lastbank som konstruerades. Information rörande viktiga koncept inom projektet, exempelvis maskininlärning och prediktivt underhåll redogörs också inom detta kapitel.

2.1 Likströmsmotor

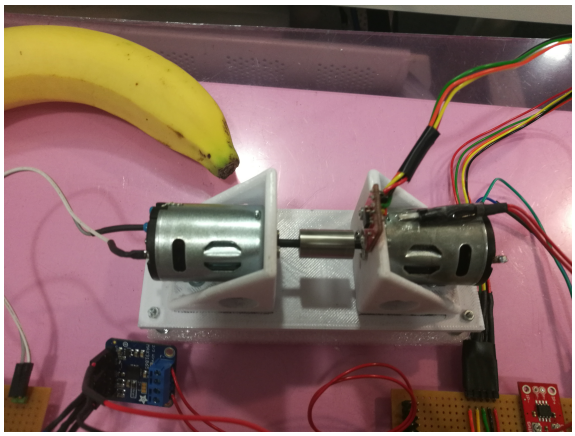
En likströmsmotor är en konstruktion som drivs av likström och kan omvandla elektrisk energi till mekanisk energi eller mekanisk energi till elektrisk energi om denna används som en generator. En bild av den motor som användes i detta projekt kan ses i figur 2.1.

Figur 2.1: En bild på en likströmsmotor.

Det finns flera varianter av likströmsmotorer med olika för- och nackdelar. I denna rapport behandlas enbart borstade likströmsmotorer med permanentmagneter eftersom detta är den typ av motor som är enklast att använda som en generator.

2.2 Motor-generator

Denna typ av koppling innebär att en elektrisk motor används för att driva en generator. Ett exempel på detta kan ses nedan i figur 2.2.



Figur 2.2: Motor-generatorkoppling.

Det huvudsakliga syftet med motor-generatorkoppling är att på ett precist och kontrollerat vis kunna reglera den last som motorn jobbar mot. Ett sätt att uppnå detta beskrivs i nedan i sektion 2.3.

2.3 Reostatisk broms

Reostatisk inbromsning syftar på den metod där energin från en likströmsmotor dissiperas i en tillhörande resistorbank.

Principen bakom denna form av inbromsning är följande: då motorn roterar utan att energi tillförs agerar motorn istället som en generator. Detta innebär att motorns mekaniska energi konverteras till elektrisk energi i form av en ström som rör sig från motorns två poler. Denna elektriska energi konverteras till värme enligt formen $P = \frac{U^2}{R}$ där R är den total resistansen för kretsen och U är spänningsfallet över denna.

Detta uttryck är maximalt då U är så stort som möjligt och R är så litet som möjligt. Resistorbanken som används i detta arbete består av ett antal resistorer som är parallellkopplade. Den totala resistansen ges då av den välkända formeln:

$$\frac{1}{R_{tot}} = \frac{1}{R_1} + \frac{1}{R_2} + ... + \frac{1}{R_n} \tag{2.1}$$

Den totala resistansen hos resistorbanken är alltså minimal då alla resistorer leder ström. Detta innebär i sin tur maximal effektutveckling i resistorbanken eftersom $P = \frac{U^2}{R}$ ger att effektutvecklingen är omvänt proportionell mot resistansen i kretsen. Det är då som så stor andel som möjligt av den mekaniska energi som

tillförs till generatorn omvandlas till värme i resistorerna vilket leder till maximal inbromsning av generatorn.

Denna effektutveckling kan direkt relateras till den last som motorn upplever genom kvoten mellan tillförd effekt och genererad effekt enligt:

$$\eta = \frac{P_{ut}}{P_{in}} \tag{2.2}$$

Ju större värde på η, desto större förlust i lastbanken. Det teoretiskt största värdet på η är 1 vilket motsvarar att all energi som tillförs till systemet går förlorad genom värmeutveckling i lastbanken.

2.4 Enkortsdatorer

Med enkortsdatorer avses datorer som består av enbart ett kretskort men samtidigt innehar de funktioner som krävs för att den skall kunna benämnas som dator. Dessa funktioner kan inkludera exempelvis en processor, portar för input och output, med mera [9]. På engelska kallas enkortsdatorer Single-board computer (SBC).

2.4.1 Arduino

En typ av SBC som avses användas i detta projekt är Arduino. Den modell som kommer användas i detta projekt är Arduino Uno. Arduino Uno är baserad på en 16MHz ATmega328P mikrokontroller. Denna använder en arbetsspänning på 5V. Hög logiknivå motsvarar 5V och låg logiknivå motsvarar 0V. Arduinon har också en inbyggd spänningsregulator vilket innebär att den rekommenderade spänningen på strömförsörjningen är 6-12V vilket sedan regleras ner till 5V av Arduinon.

Det finns också ett antal portar som kan användas för kommunikation med andra enheter. Dessa portar samt deras antal anges i listan nedan:

- En port för seriell kommunikation (USB typ B)
- 14 digitala input/output (I/O) pinnar
- 6 analoga ingångar

I/O pinnarna kallas kollektivt för General-purpose I/O (GPIO). Då seriell kommunikation används begränsas antalet tillgängliga digitala I/O pinnar till 12 stycken. GPIO pinnarna som mest ge och ta emot 20 mA vid kontinuerlig användning.

Arduinon är väl lämpad för avläsning av sensorer med analog output på grund av Arduinons inbyggda Analog-till-digital-omvandlare (ADC) [10].

2.4.2 Raspberry Pi

Raspberry Pi är en SBC som tillverkas av Raspberry Pi Foundation. Den har en 1.2GHz 64-bitars fyrkärnig ARMv8 CPU. Den har stöd för 802.11n Wireless LAN vilket innebär att den kan agera som en trådlös accesspunkt [11].

Det finns också ett antal portar som kan användas för kommunikation med andra enheter. Dessa portar samt deras antal anges i listan nedan:

- 4 USB portar
- 40 GPIO pinnar
- Full HDMI port
- En ethernet port
- En Display interface (DSI)
- Plats för ett MicroSD kort

Inom detta projekt används operativsystemet *Raspbian Jessie with Pixel version April 2017*, vilket är baserat på Debian, vilket i sin tur är ett Linuxsystem.

2.4.3 Komunikation mellan Raspberry PI och Arduino

Både Arduino och Raspberry PI har inbyggt möjlighet för seriell digital kommunikation genom USB-gränssnitt. Båda enheter har dessutom GPIO pinnar för seriell kommunikation.

2.5 Ekkonos plattform för maskininlärning

Företaget Ekkono tillhandahåller en maskininlärningsplatform samt Python-bindningar för denna. Plattformen använder sig av en Random Forest-modell för att utföra prediktioner. En Random Forest-modell består av ett antal så kallade Decision Trees. Ett Decision Tree är träliknande grafstruktur där varje nod representerar ett test på en given attribut och grenarna representerar den väg som skall tas beroende på resultatet av testet hos en nod.

Random Forest-modellen arbetar med så kallade instanser av ett antal attribut som tillförs som input till modellen som sedan ger ett utvärde baserat på den givna instansen. Utputen från en Random Forest-modell kommer från ett antal Decision Trees, som alla är tränade på modellens träningsdata men med olika inställningar så att de enskilda träden ger olika utput för samma input. Ett medelvärde bildas av trädens utput som sedan blir Random Forest-modellens utput.

Varje instans indikerar för modellen att de attributvärden som instansen innehar är kopplade till samma tidpunkt. Varje attribut har ett värde som motsvarar ett inläst sensorvärde eller ett värde producerat på annat sätt (till exempel ett tidigare

sensorvärde från ett antal punkter bakåt eller ett medelvärde för ett antal punkter bakåt).

2.5.1 Träning av modell

Träningen av Ekkonos Random Forest-modeller går till så att ett set träningsdata extraheras från ett set av tidigare insamlad data. Till varje datapunkt i träningssetet läggs sedan ett prediktionsattribut, som består av värdet för ett utvalt sensordataattribut som läses från ett visst antal datapunkter framåt i datasetet (värdet är känt vid träningtillfället som är baserad på tidigare insamlad data). Prediktionsattributet markeras som målattibut för modellen, för att indikera att det är det attributet som modellen ska tränas att förutsäga.

Även fler attribut kan läggas till för att förbättra prediktionsförmågan hos den tränade modellen, till exempel: “lags”, värdet av ett attribut från ett antal punkter bakåt; moving average , medelvärdet för ett attribut övre ett antal punkter bakåt. Ekkonos API tillåter att deklarativt lägga till ett eller flera av dessa sorters attribut, som då läggs till instanser automatiskt vid träning och användning av modellen, vilket förenklar användandet av modellen.

2.5.2 Användning av modell

Då Ekkonos plattform för maskininlärning skall användas krävs ej någon förståelse av de inre mekanismerna av denna plattform. En datapunkt skickas in och prediktionen exekveras sedan. Prediktionen fås ut i form av en siffra som indikerar det predicerade värdet på den relevanta attributen. All denna kommunikation sker med hjälp av Python-bindningar.

3

Systemkonstruktion

Konstruktionsprocessen för hela systemet, som består av både hårdvara och mjukvara, beskrivs i detta kapitel. Först kommer en överblick av systemets komponenter och dess syften, därefter kommer en kravspecifikation som val och utformning av komponenter utgår från. Därefter kommer ingående beskrivningar av systemets alla komponenter, samt motivering av val och utformning av dessa komponenter. Under konstruktionens gång utfördes ett antal tester där vissa kritiska komponenters prestanda och karaktäristik utforskades. Dessa test beskrivs och resultat redovisas i slutet av detta kapitel.

3.1 Sammanfattning av hårdvarusystemet

Systemet kan delas in i följande logiska moduler:

- Motor-generator
- Lastkrets, samt kontrol av last
- Sensorer:
  - Accelerometer
  - Termometer (motor)
  - Termometer (omgivning)
  - Amperemeter
- Arduino Uno
- Raspberry Pi
- Bildskärm
- Strömförsörjning

En mer detaljerad lista av dessa komponenter, för den intresserade läsaren, finns i A.1.

Motorn-generatorn är den centrala delen av systemet och består av en motor som driver en tillkopplad generator. Generatorn är i sin tur kopplad till en lastkrets

bestående av ett antal resistorer som kan kopplas till och från genom en styrsignal och därmed stegvis variera motståndet i generatorkretsen. Lastkretsen agerar därför som en ställbar reostatisk broms till motor-generatorn (se sektion 2.3).

En accelerometer och en temperaturgivare är fastsatta på motorn, vars temperatur och vibrationsmönster varierar med last och hur länge motorn har körts. Ytterligare en termometer finns kopplad till systemet som mäter rumstemperaturen för att ge en referenstemperatur till den uppmätta motortemperaturen. Till motorns strömförsörjningskrets finns en amperemeter kopplad för att läsa av hur mycket ström som motorn drar.

Alla sensorer är kopplade till en mikrodator av typen Arduino som agerar som datainsamlare. Arduinon samplar sensorvärdena med en viss frekvens och skickar vidare dessa datavärden till Raspberry Pi över en seriell databuss. Arduinon är även inkopplad till lastbankens styrkrets och ansvarar för att ställa in dess motstånd till önskat värde.

Raspberry Pi agerar som den huvudsakliga processorenheten (med 4 kärnor på 1.2Ghz, jämfört med Arduinons enkla CPU på 16MHz) och hanterar all sensordata vidare, utför prediktion och förbereder data för visualisering.

Visualisering sker på en till Raspberry Pin kopplad bildskärm. Visualiseringen består av en graf som i realtid uppdateras och beskriver hur sensorvärdena ser ut just nu samt för en tid tillbaka. Även prediktionen för en tid framåt samt tidigare prediktioner och uppmätta värden ritas ut så att skillnader och likheter visuellt åskådliggörs. På bildskärmen visas också statistiska värden för den data som presenteras, till exempel ett värde på hur väl prediktionen stämmer med det motsvarande uppmätta värdet.

Systemets strömförsörjning består utav en 12V/2.25A DC strömadapter som styr både Arduino (som regulerar spänningen till 5V internt) och motor, samt en 5V/2.4A DC strömadapter till Raspberry Pin.

3.2 Kravspecifikation

Syftet med hårdvarusystemet är att leverera sensordata som PdM-modellen kan baseras på och sedan i realtid appliceras på det körande systemet. För att detta ska fungera måste systemet leverera konsekventa sensorvärden över lång tid. För att prediktionen ska vara intressant måste dessutom tillräckligt stor dynamik uppvisas i systemet. De uppmätta sensordata är: vibration, strömförbrukning, och temperatur (motortemperatur, men även rumstemperatur som referenstemperatur). Kraven på hårdvarussytemet har utformats så att insamlingen av denna data möjliggjorts.

Vidare begränsas valet av komponenter så att systemet blir översiktligt och förståeligt (inte för litet och plottrigt), samtidigt som att det lätt ska kunna transporteras (inte för stort och tungt). Systemets fysiska storlek bör vara i runt 30-50 cm i bredd och inte mer än 20-30 cm i höjd.

För att uppnå tillräcklig dynamik i systemet är ett krav att lasten på motorn ska kunna varieras i ett tillräckligt stort omfång; motorns last påverkar strömförbrukning, varvtal, vibration samt värmeutvecklig i motorn.

Eftersom den centrala komponenten i systemet är motorn, som alla sensorvärden utgår från, har tillgängligheten av motorer och sensorer varit den största faktorn i valet av komponenter för det elektriska systemet. Motorn valdes vidare så att dess vartal vid operation av systemet inte är för stort och att variationer i varvtalet ska vara tydligt observerbara av icke insatt publik.

Ett annat krav är att läsningar av sensorer skall ske med tillräckligt hög frekvens för att fånga de delar av motorns vibration som anses vara intressanta. Detta bör innefatta grundfrekvensen hos vibrationen samt en del av de övertoner som uppstår. På grund av Nyqvist-kriteriet bör samplingsfrekvensen vara dubbelt så stor som det område som önskas mätas.

3.3 Motor-generator

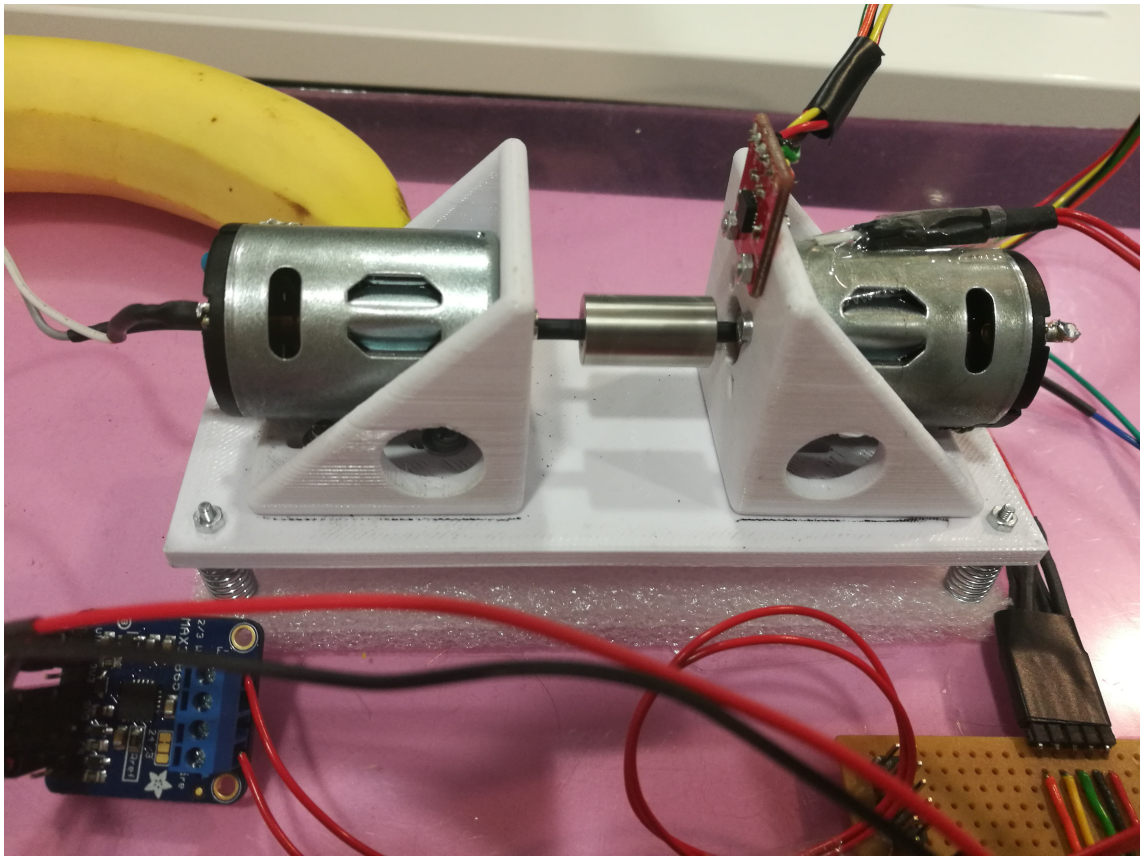
Motor-generatorn består av två borstade permanentmagnet-DC-motorer av identisk modell (N2738-125 tillverkad av Igarashi Motoren GMBH), specificerade att kunna drivas av 6 till 24 V likspänning, med 12V som nominell spänning. Icke-lastad drivström är 0.08A (vid 12V), medan maximal drivström är specificerad till 4.3A, enligt datablad [5].

Motor och generator kopplas ihop med en skruvkoppling. Ett problem som uppstod här var att skruvkopplingens diameter (3mm) var större än motoraxlarnas (2.75mm). Detta medförde att kopplingen blev lös efter en kort tids körning, vilket innebar en omfattande underhåll av kopplingen. För att åtgärda detta täcktes axlarna vid kopplingssytan med två bitar av 3mm krympslang. En ytterligare lösning som togs i åtanke var att använda lim som kopplingsmaterial, eventuellt i kombination med krympslang och skruvkoppling. Detta alternativ beaktades inte vidare då enbart krympslang och skruvkoppling visade sig fungera tillräckligt bra.

För att minimera för stora oönskade vibrationer, som ger upphov till oljud och minskad livslängd hos motorerna, behöver motorn och generatorns axlar sättas upp i en så rak linje som möjligt. Att minska oljudet är viktigt då systemet ska fungera som uppvisningsprodukt och då vara tillräckligt estetiskt attraktiv.

Motorerna monterades på en 3D-skriven monteringsplatform. Plattformen består av tre delar: två identiska kubiska bitar som motorn respektive generatorn skruvas fast i (med hjälp av skruvhålen i motorernas hölje), samt en platta med 8 stycken hål för montering av de två nämnda bitarna plus fyra hål för att montera plattformen via vibrationsdämpare på den akryliska bottenplattan.

De två kubiska bitarna är utformade med en färdig 3D modell som grund, där dimensioner anpassats för våra motorer. Plattformen skrevs ut med 3D-skrivaren Flashforge Creator Pro.



Figur 3.1: Den plattform som motorn monterades på, samt motorerna.

3.3.1 Test av motor-generatorns överföringskaraktäristik

För att testa karaktäristiken hos motor-generatorn utfördes ett experiment där motorns strömförbrukning och generatorns utspänning noterades. Resultat användes främst för att anpassa resistansen hos resistorerna i lastbanken på ett sådant vis att systemet får önskad karakteristik, se sektion 3.4.1.

Genom ett justerbart nätaggregat tillförs 12V över motorns terminaler, först utan last (dvs. öppen generatorkrets), sedan lastad med olika resistanser mellan 204-13.5 Ohm. Motorns matningsström mäts upp genom nätaggregatets inbygga amperemeter. Generatorns utspänning mäts med voltmeter. Generatorkretsens ström samt effektutveckling över lasten beräknas enligt Ohms lag. Resultatet syns i tabell 3.1.

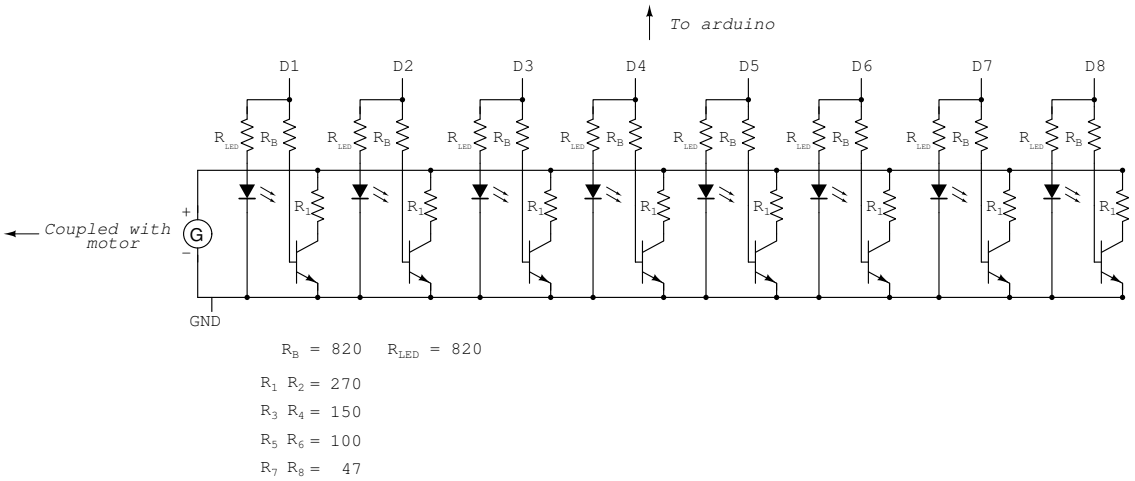
3.4 Lastkrets

Lastkretsen består av åtta parallellkopplade resistorer som kan kopplas in och ur med hjälp av åtta transistorer (BD135G), som agerar som strömbrytare för varje resistor. Resistorerna är parallellkopllade vilket innebär att den totala resistansen i kretsen minskar då fler av dessa leder ström (se sektion 2.3). Den kretskoppling

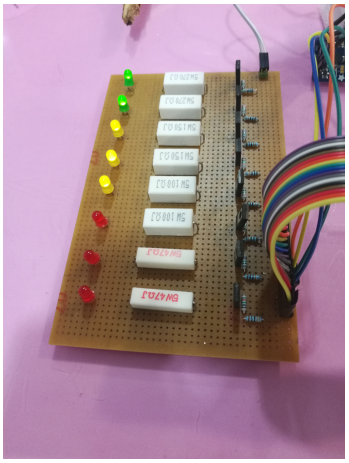
Resistans last (Ohm)	Motorström (mA)	Genererad spänning (V)	Genererad ström (mA)	Effekt i lasten (W)
-	200	9,6	-	-
204	280	9,3	46	0,342
136	300	9,08	67	0,61
68	360	8,18	120	0,98
47	400	7,51	160	1,2
27	480	6,3	233	1,47
13,5	590	4,56	338	1,54

Tabell 3.1: Data från det första motor-generator-testet (12V motorspänning)

som användes i detta projekt kan ses i figur 3.2. Den fysiska konstruktionen kan ses i figur 3.3.



Figur 3.2: Kopplingsschema för generator och lastbanksanordning.



Figur 3.3: Lastbanksanordningen som konstruerades.

3.4.1 Val av resistorer i lastbanken

Lastresistorerna är valda så att strömmen som går genom generatorkretsen kan varieras med så stort intervall att det ger upphov till märkbar skillnad i motorns hastighet, men inte så mycket att strömmen blir så stor att den kan skada motorn eller ge upphov till för stor effektutveckling i lastbanken, vilket skulle innebära att resistorerna blir orimligt varma och skulle riskera att brännas.

För att utröna lämpliga resistorvärden beaktades testresultaten från sektion 3.3.1 utifrån ett antal olika faktorer:

- Strömmen i resistorbanken får ej överstiga rekommenderade värden för resistorerna.
- Det skall tydligt märkas att det går tyngre för motorn att driva generatorn.
- Skillnaden i effektutveckling då resistansen ändras skall ej vara för stor eller för liten.
- Resistorerna skall väljas från E12-serien så att varje steg i lastbanken enbart består av en resistor. Detta sparar utrymme och undviker att introducera fler komponenter, vilket skulle leda till fler "points of failure".

Effektutvecklingen vid 13.5 Ohm är 4.5 gånger större än vid 204 Ohm. Denna effektutveckling kan direkt relateras till lasten som motorn upplever då den driver på det vis som beskrivs i ekvation 2.2. Vid 204 Ohm ger denna ekvation att ca. 13% av den effekt som tillförs till motorn omvandlas till värme i resistorbanken. Motsvarande värde vid 13.5 Ohm är istället ca. 57%. Detta bedöms som tillräckligt stor variation i last då motorn är märkbart ansträngd vid 13.5 Ohm.

Följande resistorer valdes:

- 2 resistorer á 270 Ohm
- 2 resistorer á 150 Ohm
- 2 resistorer á 100 Ohm
- 2 resistorer á 47 Ohm

Vidare antas också att dessa alltid kommer att aktiveras i viss ordning och i begränsade kombinationer. Resistorerna aktiveras alltid i ordning från högst till lägst och vice versa då de inaktiveras. Detta innebär att följande resistansvärden kan uppnås för lastbanken:

- 270 Ohm
- 135 Ohm
- 71 Ohm
- 48 Ohm
- 33 Ohm

- 25 Ohm
- 16 Ohm
- 12 Ohm

Detta ger oss den önskade karakteristiken för lastbanken och ger dessutom bra noggrannhet vid lägre resistans vilket är viktigt eftersom det gäller att effektutvecklingen förändras snabbare vid låga resistanser. På grund av denna egenskap är det önskvärt att kunna variera den lastbankens totala resistans mer exakt vid låga resistanser, men ej nödvändigt med samma precision vid höga resistanser.

Den maximala effektutvecklingen hos en enskild resistor kan beräknas med hjälp av värden från tabellen 3.1 och blir då $\frac{4.56^2}{47} = 0.442$W. Detta är nära den maximala avsedda effektutvecklingen hos den vanligaste typen av resistorer på 0.5W [8]. Då dessa resistorer förväntas leda ström under en längre tid, samt att den totala effektutvecklingen som mest ligger på ca 1.54 W valdes därför att använda resistorer avsedda för en högre effektutveckling än 0.5 W. På grund av tillgänglighet, samt att dessa resistorer är enklare att se på grund av deras storlek valdes resistorer som var avsedda att klara upp till 5 Watt.

3.4.2 Val av transistorer som styr lastbankens last

Valet av transistorer begränsas främst av hur mycket ström och spänning de tål, samt att de ska kunna styras som strömbrytare av Arduinons digitala pinnar. Strömmen som går genom transistorerna är dels strömmen från generatorn (kollektorströmmen, I_C), och dels styrströmmen från Arduinon (basströmmen, I_B). Respektive maxvärden är enligt datablad 1.5 A (I_C) och 0.5 A I_B. Maxvärdet för kollektor-emitterspänningen, V_{CE}, är enligt datablad 80 V, vilket är långt under de spänningsnivåer som är aktuella (< 12V). Maxvärdet för backspänningen över emitter-bas, V_{EB}, är noterat till 5V, men det är ej ett problem då Arduinons digitala utput-pin kopplad till basen endast kan variera mellan 0 till +5V relativt GND vilken är ansluten till emittern.

För att transistorn ska agera som strömbrytare utan att begränsa strömmen onödigt när den är aktiv, måste basströmmen vara tillräckligt stor för att den ska in i mättat läge (saturation). Detta läge karaktäriseras av att kollektorströmmen inte påverkas av högre ström genom basen.

Resistorerna (R_B) som kopplas till basen på transistorerna valdes till 820 Ohm, efter experimentell jämförelse av kollektorströmmen med olika värden på R_B. Härefter följer ett teoretisk resonemang om transistorerna faktisk bör gå in i mättat läge:

Arduinons digitala pinnar ger vid på-läge ger en spänning (V_{bb}) på ca 5V, och den maximala ström som rekommenderas dras ur dem är 20 mA [10]. Transistorns bas-emitter-spänning vid mättad operation $V_{BE(sat)}$ är enligt datablad 0.6V till 1V. Detta ger en basström på $\frac{V_{bb}-V_{BE(sat)}}{R_B\Omega} = 4.8$ mA till 5.4mA.

Eftersom den totala generatorströmmen ej överstiger 338 mA (se tabell 3.1) vid någon lastnivå, kan I_C nödvändigtvis ej överstiga den storleken. Notera att strömmen från generatorn delas på de aktiva transistorerna, så I_C över de individuella transistorerna kommer vara mindre än den totala generatorströmmen. En övre gräns för förstärkningen en transistor kan uppnå blir då $h_{FE\_krets\_max} = \frac{I_C}{I_B} = \frac{338mA}{4.8mA} = 70.4$ gånger, vilket är mindre än det typiska värdet för h_{FE} som är på över 100 gånger vid de strömmar som är aktuella (under 1A). [12]

3.5 Val av sensorer

En viktig del av projektet var att bestämma vilken typ av data som skulle samlas in. Då detta skulle göras togs hänsyn till hur enkel själva datainsamling är: sensorer som är anpassade för Arduino och är lättillgängliga måste finnas. Parametern som skall mätas måste också variera nämnvärt då en datainsamling sker. Storheten som mäts skall vara enkel att förstå och ha en tydlig fysisk manifestation.

Dessa kriterier ledde till beslutet att följande storheter skulle mätas:

- Motorns strömförbrukning
- Motorns vibration
- Motorns temperatur
- Det mekaniska motstånd som motorn upplever

Under systemets konstruktion uppmättes dessa parametrar och den data som erhölls undersöktes sedan för att bedöma huruvida dessa parametrar uppvisade beteende som var värt att analysera. För temperaturen innebar detta exempelvis att den varierade nämnvärt under en körning. Resultatet av denna typ av datainsamling kan ses i sektion 4.1.

3.5.1 Strömsensor

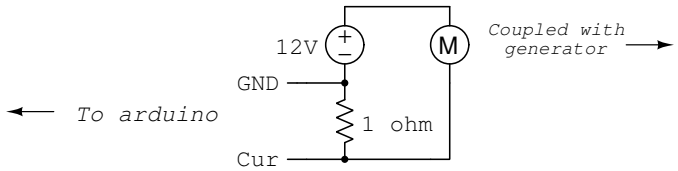
Det finns olika sätt att mäta ström på, men speciellt för användning med Arduino övervägdes och testades två metoder: mätning över shunt-resistor, samt mätning med hall-effektsensor.

Den första och enklaste metoden är direkt mätning av spänningen över en shunt-resistor seriekopplad med motorn. Arduinons nollpunkt (jord) måste då anslutas på ena sidan av shunt-resistorn och en av de analoga ingångarna till andra sidan av resistorn. Strömmen beräknas sedan enligt Ohms lag. En shunt-resistor är en resistor med litet motstånd och hög noggrannhet på resistansen, och lämpar sig därför till att med exakthet mäta strömen i en krets. Shunt-resistorns Ohm-värde måste väljas så att en tillräckligt stor potentialskillnad uppstår (beroende på ADC:ns känslighet), men så lågt som möjligt för att inte onödigt bromsa strömmen i kretsen

och därmed påverka kretsens funktion. Nedan följer ett resonemang om hur shunt-resistorn valdes för detta projekt.

Enligt data från 3.3.1 drar motorn en ström på mellan cirka 200 till 600 mA, beroende på generatorns last. Dessa värden motsvarar ett motstånd i motorn på mellan 12V/0.2A = 60 Ohm och 12V/0.6A = 20 Ohm. Med en shunt-resistor på 1 Ohm får vi ett ungefärligt spänningsfall på mellan 0.2V och 0.6V. I praktiken blir detta spänningsfall mindre eftersom detta spänningsfall får motorn att rotera långsammare vilket innebär en lägre ström eftersom motorns ersättningsresistans varierar med dess rotationshastighet (se ovan).

Arduinons ADC har en upplösning på 5mV per datapunkt, vilket skulle innebära att det på intervallet 0.2V-0.6V finns (0.6V-0.2V)/(0.005V/datapunkt) = 80 unika datapunkter. Med en 1 Ohms resistor och en ADC-upplösning på 5mV kan vi mäta strömmen vid 5 mV / 1 Ohm = 5mA. Med en 0.5 Ohms resistor får vi motsvarande upplösning på 10mA. Dessa beräkningar verifierades sedan experimentellt.



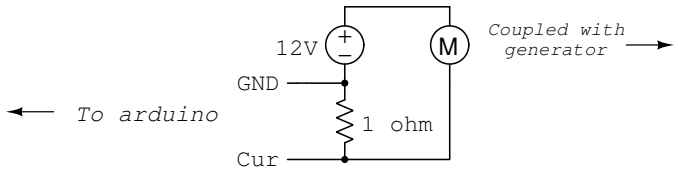
Figur 3.4: Kopplingsschema för motor, dess strömförsörjning och shunt-resistor för att mäta motorns strömtillförsel.

Mätning av strömmen genom motorkretsen kan också göras med en Halleffekt-sensor. En Halleffekt-sensor ger en varierande spänning beroende på hur magnetfältet ser ut runt sensorns givare. En strömsensor som är Halleffekt-sensorbaserad låter den uppmätta strömmen gå genom en liten krets som skapar ett magnetfält vars styrka växer med strömmen, och som sedan kan mätas med Halleffekt-sensorn. Fördelen med denna strömmätningsmetod är att den uppmätta strömmens krets inte påverkas nämnvärt. Den enda påverkan är en mycket liten extra impedans som sensorn medför [2].

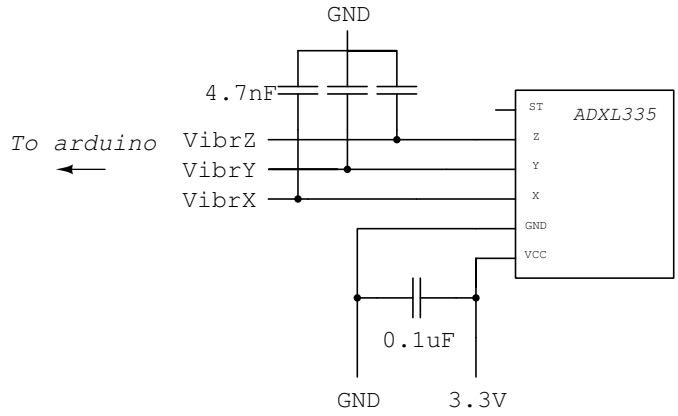
Det finns en hallsensor-baserad strömmätare för Arduino i form av en utbrytningsmodul, kallad breakout board (BOB) . Enligt specifikationer ska sensoren kunna mäta AC eller DC ström upp till 5A. Strömsensorn är ratiometrisk, dvs att utputspänningen är proportionell mot dess matningsspänning [1]. Utifrån specifikationerna och att sensorn är gjord för att fungera med en Arduino, samt att sensorn inte påverkar mätningskretsen valde först vi att använda denna sensor. Halleffektsensorn visade sig dock efter tester vara otillförlitlig och väldigt känslig för elektromagnetiska störningar från omgivningen.

3.5.2 Accelerometer

Accelereometern som används i detta projekt är ADXL335. Denna används för att mäta de vibrationer som uppstår då motorn arbetar [2].



Figur 3.5: Kopplingschema Hall-effekt-baserad strömsensor som sedan byttes ut.



Figur 3.6: Kopplingsschema för accelerometer.

Vi matar accelerometern med $V_s = 3.3$V. Accelerationen 0g bör enligt datablad ge en output spänning på $\frac{V_s}{2} = 1.65$. Utvärdet från accelerometern är ratiometriskt, vilket innebär att matningsspänningen är proportionell mot känsligheten för ändringar i acceleration. Med matningsspänningen 3.6V är känsligheten ca. 360 $\frac{mV}{g}$, enligt datablad. Med $V_s = 3.3$V fås då en känslighet $\frac{3.3}{3.6} \cdot 360 = 330\frac{mV}{g}$.

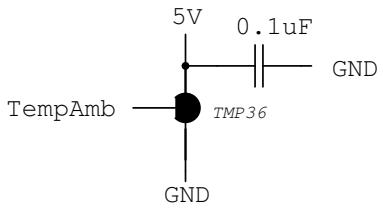
Vi har monterat accelerometern på motorns kortsida med x-axeln i sidled, y i höjdled och z i riktningen parallell med motoraxeln. Då motorn ej arbetar bör de avlästa värdena se ut som i tabell 3.2.

Axel	Acceleration (g)	Utspänning (V)
x	0g	1,65
y	1g	1,98
z	0g	1,65

Tabell 3.2: Testdata för accelerometer.

3.5.3 Rumstemperatursensor

Temperatursensorn som användes för att mäta rumstemperaturen är av modell TMP36. Temperatursensorn ger en utspänning proportionell mot temperaturen vid en matningspänning på 2.7V till 5.5V [3]. Den har en mätosäkerhet på ± 0.5°C. Detta anses tillräckligt då rumstemperaturen ej förväntas variera signifikant och ej heller påverka systemet till den grad att mindre ändringar behöver mätas.

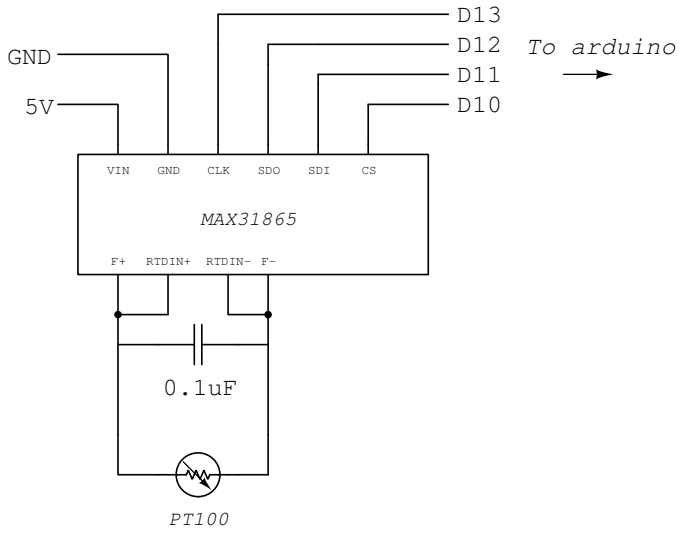


TempAmb = 0.1V...2V ~= -40º..150º (10mV/ºC)

Figur 3.7: Kopplingsschema för rumstemperatursensor.

3.5.4 Motortemperatursensor

Denna temperatursensor används för att mäta hur temperaturen hos motorn varierar då lasten och andra parametrar varieras. Temperatursensorn består av två delar: en PT100-givare och en särskild BOB från Sparkfun. PT100-givaren består av en platinumfilm vars resistans varierar med temperaturen på ett sätt som kan betraktas som linjärt inom temperaturområdet $0 - 100^{o}$C. Platinumfilmen sitter på ett keramiskt substrat och har två anslutningsben som går genom ett avlastningsmaterial. Vid 0^{o}C är resistansen 100 Ohm, därav namnet. Felmarginalen på denna resistans ger upphov till en felmarginal på temperaturen på $\pm 0.25^{o}C$



Figur 3.8: Kopplingsschema för motortemperatursensor.

För att läsa av PT100-givaren på ett noggrant sätt, används en extra BOB som kopplas mellan Arduinon och givaren. BOB-en har bland annat en inbyggd förstärkare, 15 bitars ADC, och automatisk kompensering för extra motstånd i kopplingen till PT100-givaren. BOB-en kommunicerar till skillnad från övriga sensor med Arduinon digitalt över Serial Peripheral Interface (SPI) [7]. SPI är ett synkront dataprotokoll som används som en standard för kommunikation mellan mikrodatorer.

PT100 har en känslighet på ungefär $0.385\Omega/^{o}C$.

Anledningen till att just denna temperatursensor valdes var i hög grad på grund

av dess precision samt på grund av att sensorn är liten och okomplicerad att fästa på motorn som man avser mäta temperaturen på.

Kopplingen mellan BOBen och givaren gjordes genom att löda två lika långa kablar till givarens respektive ben, för att sedan täckas över med 3mm krympslang för att förstärka kopplingen. I ett första försök täcktes respektive bens lödkoppling med ett lager krympslang, men under den vidare konstruktionsprocessen uppstod ett allvarligt problem med kopplingen. På grund av givarens sköra ben och att kopplingen var otillräckligt förankrad gick givarens ben av vid dess fästpunkt på givarens substrat, vilket resulterade i att givaren blev oanvändbar och en ny PT100 fick monteras. I det andra försöket att förankra givarens ben användes ett lager krympslang för respektive ben, precis som tidigare, men sedan täckes båda benen tillsammans samt en bit av PT100 substratet med ytterligare ett lager krympslang och sedan några lager eltejp virat runt. Detta för att försäkra att kopplingen ej skulle kunna brytas av av misstag.

3.6 Arduino

Arduinon är ansvarig för att samla in sensordata med en bestämd frekvens och sedan skicka dessa data till Raspberry Pi:n för vidare behandling. Arduinon ansvarar dessutom för att kontinuerligt vänta på lastnivåinställningar från Raspberry Pi:n och applicera dessa värde på lastbanken genom digitala output pinnar.

3.6.1 Arduinons in- och utportar

Den Arduino som valdes är en Arduino Uno (se sektion 2.4.1), vilken är den vanligaste typen av Arduino. Den största anledningen till valet av Arduinon är att alla kringkomponenter skall kunna kopplas in samtidigt: antalet portar samt strömförbrukning behöver tas i åtanke. Nedan kommer en sammanställning av dessa krav för respektive komponent.

Komponenter som ska kopplas till Arduinon är följande: Sensorer (3 analoga, 4 digitala inputs), Lastbanksstyrning (8 digitala outputs), Raspberry Pi (1 USB port).

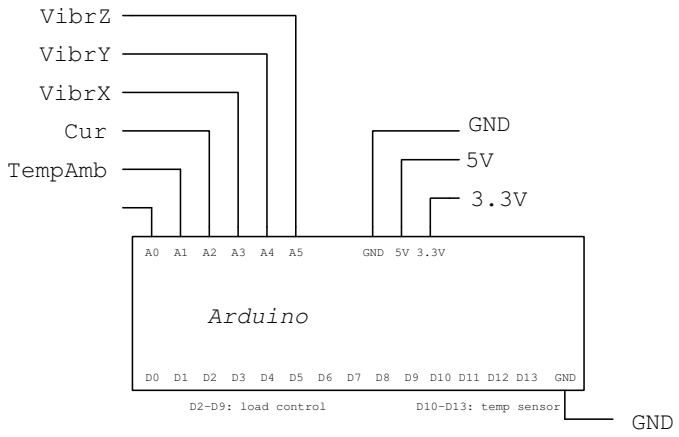
Sensorerna och lastbankstyssignalerna ska kopplas till Arduinons GPIO-portar. Följande GPIO-portar behöver då vara tillgängliga:

Arduino Uno har 12 tillgängliga digitala I/O portar; 1 USB port, och 6 analoga inputs. Detta är tillräckligt för ovan beskrivna konfiguration.

[1]Medelvärde av angivna värden för 3.0V och 3.6V, motsvarande 3.3V, från datablad [1].

Komponent	GPIO-typ	Antal	Max strömförbrukning
Vibrationsensor	Analog	1 (en axel)	363 μA [1]
Rumstemperatursensor	Analog	1	50 μA
Strömsensor	Analog	1	-
Motortemperatursensor	Digital	4	2 till 3.5 mA
Lastbank	Digital	8	

Tabell 3.3: Antal GPIO-portar som enskilda sensorer använder samt deras strömförbrukning.



Figur 3.9: Kopplingsschema för Arduino.

3.6.2 Arduinons mjukvara

På Arduinon exekveras kod som läser från sensorer och skickar denna data vidare till Raspberryn som sedan behandlar denna data. Mjukvaran till Arduino programmerades i Arduino IDE och ett språk som liknar C och C++ användes för detta syfte.

3.6.2.1 Avläsning av sensordata

Avläsningen av sensorer skedde med två huvudsakliga metoder beroende på vilken typ av sensor som skulle avläsas. Den inbyggda funktionen *analogRead* användes för att avläsa de sensorer vars output är analog och funktionen *max.readRTD_auto*, vilket är en modifikationen av funktionen *max.readRTD* från biblioteket Adafruit_MAX31865 [7], användes för att läsa av PT100 sensorn, eftersom denna använder en egen ADC och därmed är kopplad till en digital inport.

Eftersom Arduinon önskades utföra dessa läsningar med en viss frekvens ställde detta ett krav på hur ofta de inkopplade sensorerna behöver avläsas. Som kan ses i tabellen 3.4 kräver detta att ADC:n skall omvandla de analoga insignalerna med en viss hastighet. Detta innebar att den snabbare omvandlingen behövde användas för att uppnå målet med 1500 sändningar per sekund. Detta uppnåddes genom att välja en klockdivisor på 16, se sektion 3.6.2.3.

3.6.2.2 Protokoll för kommunikation mellan Raspberry Pi och Arduino

De faktorer som påverkade hur datan skulle skickas från Arduinon var framförallt hur stora paket som kunde skickas, samt hur snabbt dessa kunde skickas samtidigt som en acceptabel mängd av dessa datapaket nådde fram till Raspberryn och samtidigt bibehöll sin integritet, det vill säga nådde fram oförändrade.

För att bedöma vilken datamängd och hastighet som var bäst anpassad efter projektets behov. Då detta beslutet skulle tas beaktades resultatet av sändar- och mottagarexperimenten från sektionerna 3.7.1.1, 3.6.2.3, 3.9.2, samt vilken samplinghastighet som krävs för att fånga motorvibrationerna enligt sektion 3.9.1. En annan faktor som beaktades var hur den data som behövde skickas kunde förpackas.

För att ta detta beslut krävdes först att de storheter som skulle mätas var tydligt definierade. Dessa storheter kan ses i 3.5. Utöver dessa ansågs det också nödvändigt att bifoga den tidpunkt vid vilken datapunkten uppmättes. För att försäkra datans integritet bifogades även en så kallad checksumma samt ett sekvensnummer för datapaketet.

Denna data komprimeras till 15 bytes och kommunikationen sker sedan med hastigheten 1500 datapunkter per sekund vilket, enligt utförda tester är en bra kompromiss mellan prestanda och tillräcklig datamängd.

3.6.2.3 Test av sändhastighet hos Arduinon

Syftet med följande test är att avgöra med vilken hastighet som Arduinon kan utföra de läsningar av sensorer som krävs samt hur ofta dessa kan skickas med hjälp av seriell kommunikation. Detta test ger då ett värde på maximal sändhastigheten utan att ta hänsyn till hastigheten som mottagaren kan ta emot och behandla denna data.

För att uppnå detta syfte används inte Raspberryn på mottagarsidan, för att undvika att mottagaren begränsar Arduinons sändhastighet. Istället används en dator med en Intel I7-27200M 2.2 GHz processor som mottagare.

I detta test undersöks hur sändhastigheten påverkas av konfigurationen av Arduinons ADC. Denna konfigureras genom att sätta en divisor på så vis att ADC:ns klockfrekvens blir Arduinons processorhastighet delat med denna divisor [6]. Arduinons standarddivisor på 128 samt en divisor på 16 testades. Avläsningar av sensorer sker så ofta som möjligt, då data enbart skickas från Arduinon i det fall då en läsning av en sensor har skett.

Det skickas 100 000 paket och sedan beräknas hur många av dessa som överförts korrekt. Resultatet kan ses i tabellen 3.4.

Ur denna tabell kan utläsas att Arduinon utan problem kan skicka åtminstone än 2000 datapunkter av storleken 15 bytes per sekund om den låga ADC divisorn används. Om den större ADC divisorn används begränsas hastigheten till 1381 paket. Vilken divisor som skall användas beror i viss utsträckning på resultaten från de andra testerna.

Antal bytes per datapunkt	ADC divisor	Datapunkter per sekund	Antalet misslyckade sändningar
15	16	2263	0
15	128	1381	0

Tabell 3.4: Data från test av Arduinons sändhastighet

3.7 Raspberry Pi

Mjukvaran hos Raspberry Pi delas in i tre processer: mottagarprocess (MP) av data från Arduinon, dataprocessering och prediktionsprocess (DPPP), och visualiseringsprocess (VP). Dessa tre processer körs med hjälp av Pythons multiprocessing-bibliotek i separata processer (i datorteknisk mening) och kommunicerar genom delade datastrukturer. En flerprocessorkitektur tillämpas för att utnyttja Raspberry Pi:ns fyra CPU-kärnor, vilket gör att dessa processer kan köras parallellt. En flerprocess-arkitektur innebär dock också vissa komplikationer, som främst har att göra med kommunikation dem emellan.

I följande undersektioner förklaras hur dessa tre processer fungerar, och därefter hur kommunikationen mellan processerna går till mer detaljerat.

3.7.1 Mottagarprocessen

Mottagarprocesser ansvarar för att kommunicera med Arduinon, att sätta ihop mottagna databytes till enhetliga paket med sensordata värden, samt att kontrollera att datan är oskadd (se sektion 3.6.2.2 för kommunikationsprotokollet mellan Arduinon och mottagarprocessern). Mottagerprocessen skickar sedan vidare datapunkterna till DPPP.

För att läsa och skriva data via den seriella porten från pythonprogrammet användes biblioteket pyserial. Med detta bibliotek kan bland annat baudrate och andra överföringsprotokollsparametrar ställas in så att de matchar de parametrar som Arduinon använder. När mottagerprocessen startar upp behöver Arduinons program och mottagarprogrammet synkroniseras innan data kan samlas in. Detta görs genom att seriella portens signal Data Terminal Ready (DTR) sätts till hög nivå av Raspberry Pi:n, vilket aktiverar Arduinons interna reset-signal.

3.7.1.1 Test av mottagarhastighet hos Raspberry Pi

För att hitta en övre begränsning på mottagarhastigheten hos Raspberryn utformades ett test varvid ett antal datapaket med varierande storlek skickas över USB från Arduinon till Raspberryn. Dessa paket innehåller en så kallad checksumma som används för att kontrollera huruvida paketet oskatt fullbordat sin resa mellan dessa enkortsdatorer. Om datan som paketet innehåller har förändrats på något vis, till

exempel på grund av elektriska störningar kan checksumman med stor sannolikhet påvisa detta.

Testet utformas på så vis att 100 000 paket med en given storlek och med en viss hastighet skickas och antalet paket vars checksumma var inkorrekt räknas som misslyckade sändningar. Datapaketen behandlas inte på något vis i detta test. Resultatet kan ses i tabellen 3.5.

Antal bytes per datapunkt	Datapunkter per sekund	Antalet misslyckade mottagningar
15	2263	0
16	2205	0

Tabell 3.5: Data från test av Raspberryns mottagarhastighet

Ur detta kan utläsas att både 15 och 16 bytes är acceptabla storlekar på de datapaket som skickas eftersom inga paket går förlorade. Dock är hastigheten marginellt lägre då 16 bytes skall skickas i stället för 15.

3.7.2 Processering- och prediktionsprocessen

När processeringsprocessen mottar data från mottagarprocessen är datan fortfarande i "rå" form, det vill säga de värden som Arduinon läste från sensorerna genom ADC:n. Processeringsprocessen ansvarar för att transformera dessa värden till konventionella enheter för de storheter som uppmätts. Processen skickar även in dessa värden till prediktionsmodellen, samt tar det aktuella prediktionsvärdet från modellen och skickar detta tillsammans med uppmätt data till visualiseringen.

Efter att ett paket med obehandlad ADC-data mottagits från Arduinon behöver det packas upp inför prediktion och visualisering. Processeringen av en datapunkt måste gå åtminstånde lika snabbt som sampelhastigheten för att inte missa datapunkter (se mjukvarusystemets konstruktion sektion 3.7.4) För att säkerställa detta utfördes ett antal tester av systemets prestanda. (Se sektion 3.7.2.1)

3.7.2.1 Test av dataprocesseringshatighet

Syftet med detta test är att bedöma med vilken hastighet inläst data kan behandlas av Raspberryn. Med detta menas hur lång tid det tar för Raspberryn att konvertera rådata till enheter som kan utläsas av en människa, till exempel att omvandla strömsensorns data från ett tal mellan 0 och 1024 till enheten ampere.

För att göra detta används en så kallad comma-separated-values (csv) fil med tidigare uppmätt data och ett känt antal rader. Varje rad i denna fil motsvarar ett paket med data från Arduinon.

Tiden det tar att bearbeta denna fil uppmäts på två olika vis:
I ena fallet bearbetas datan rad för rad och divideras sedan med antalet rader för

att få ett medelvärde på hastigheten hos dataprocesseringen.
I det andra fallet används satsvis bearbetning. Datan delas upp i satser innehållande en viss mängd punkter och dessa bearbetas samtidigt.

Varje test utförs tre gånger och sedan beräknas ett medelvärde på resultatet. De satsstorlekar som användes var: 10 datapunkter, 20 datapunkter, 50 datapunkter och 100 datapunkter. Resultatet ses i tabellen 3.6:

Antal datapunkter per sats	Satser per sekund	Datapunkter per sekund
1	13480	13480
10	6609	66090
20	6409	128180
50	5885	294250
100	5157	515700

Tabell 3.6: Data från test av dataprocesseringshastighet

Ur detta test kan utläsas att antalet datapunkter som bearbetas per sekund ökar avsevärt då storleken på satsen ökar. Det betyder dock nödvändigtvis att den bästa strategin är att bearbeta så stora satser som möjligt. Detta beror på att även om antalet datapunkter per sekund ökar, så minskar samtidigt antalet satser som behandlas per sekund. Det tar alltså större tid att behandla en enskild sats.

Detta måste tas hänsyn till då tiden det tar att bearbeta en sats ej får bli så stor att denna beräkning kräver så mycket tid att den blockerar någon annan process som använder sig utav dess utput. Vilken storlek på satsen som är optimal kan endast beräknas då denna process arbetar tillsammans med de andra processer som exekveras på Raspberryn. Se sektion 3.9.2, där systemet som helhet testas tillsammans.

3.7.3 Visualiseringsprocessen

Visualiseringsprocessen består i att kontinuerligt ta emot inkommande data och regelbundet rita upp utvalda kurvor (se sektionen om visualiseringsdesignen 3.8). Kurvorna ritas upp med hjälp av biblioteket PyQtPlot vilket använder sig av PyQT, som är Python-bindningar till QT 5-ramverket för användarsgänssnitt.

Den största problemet att ha i åtanke när det kommer till visualieringsprocessen är att den inte ska vara för långsam så att visualieringen blir lidande (vilket skulle motverka projektets syfte som uppvisnings-produkt). Ett första försök till visualisering var genom en webbstacks-baserad lösning, det vill säga en webbserver och en HTML-sida som med javascript ritar ut visualieringen i en webbläsare. Detta visade sig snabbt vara för tungt att köra på Raspberry Pi:n, och alternativet PyQtGraph hittades, som är mycket snabbare, men inte utan att dess resursanvändning måste tas i åtanke.

Generellt gäller att högre utritningsfrekvens ger mjukare rörelser av kurvorna, men tar samtidigt mer processorkraft, vilket kan ge motsatt effekt. Updateringsfrekvensen av utritningen av kurvorna är något arbiträrt valt till ca 30Hz för detta projekt, men Beroende på vilken prediktionsmodell som är aktiv vid ett tillfälle, och således vilka kurvor som ritas ut, kan viss prestanda erövras genom viss nedsampling av utritningskurvorna, till exempel att endast rita ut var annan datapunkt. Detta fungerar bäst om kurvan som ritas ut visar många datapunkter samtidigt så att nedsamplingen inte påverkar den visuella representationen av kurvan negativt. PyQTGraph har en inbyggd nedsamplings funktion, men denna visade sig vara något svårinställd och gav ibland upphov till konstiga utritningsartefakter.

3.7.4 Kommunikation mellan Raspberry Pi:ns tre processer

Kommunikationen mellan de tre processerna sker genom en kedja av cirkulära databuffer[2] som är parvis delade mellan processerna, där varje processpar kommunicerar enligt ett producent-konsument-mönster[3].

Arduino-mottagarprocessen tar emot data över den seriella porten och hanterar som beskrivet ovan, och för varje paket som har kommit fram oskadat läggs de mottagna datapunkterna i ett cirkulärt buffer för att hanteras vidare av DPPP, som på samma sätt kontinuerligt väntar på nya datapunkter att processa, för att skicka vidare till visualiseringsprocessen.

En utmaning med en multiprocessingarkitektur är att hantera om processerna hanterar data med olika hastighet, vilket kan orsaka en flaskhals i datakommunikationen. Detta beror på att en buffer har en bestämd storlek för hur många sampelpunkter som kan sparas samtidigt då tillväxt av buffern utan begränsning skulle denna använda allt tillgängligt RAM-minne. Förutom att se till att efterföljande konsument alltid tar ut data snabbare än föregående producent, vilket är det optimala, men kanske ej alltid möjligt att garantera, kan detta lösas på två sätt: ett är att om buffret är fullt kommer producenten att behöva vänta på att konsumenten frigör platser i buffret. Denna metod är standardmetoden för exempelvis Pythons multiprocessingkö, *multiprocessing.Queue*, vilken användes under ett första försök med flerprocesseringsarkitektkur (men som valdes bort då *multiprocessing.Queue* visade sig vara för långsamt). Problemet med denna metod är att om en flaskhals uppstår kommer alla nymottagna paket att kastas bort i den seriella kommunikationen (detta manifesteras i att datapaket kommer verka vara korrupta i mottagarprocessen).

Den andra metoden är att inte blockera producenten och istället slänga bort de äldsta datapunkterna ur buffret som konsumenten inte har plockat ut för att få

[2] Ett cirkulärt databuffer är ett buffer av bestämd storlek, allokerat som ett fixt datablock i minnet, var i dataelement kan läggas till och tas ut enligt FIFO (“first in, first out”) utan att det behövs flytta runt data i minnet när element läggs till eller tas ut.

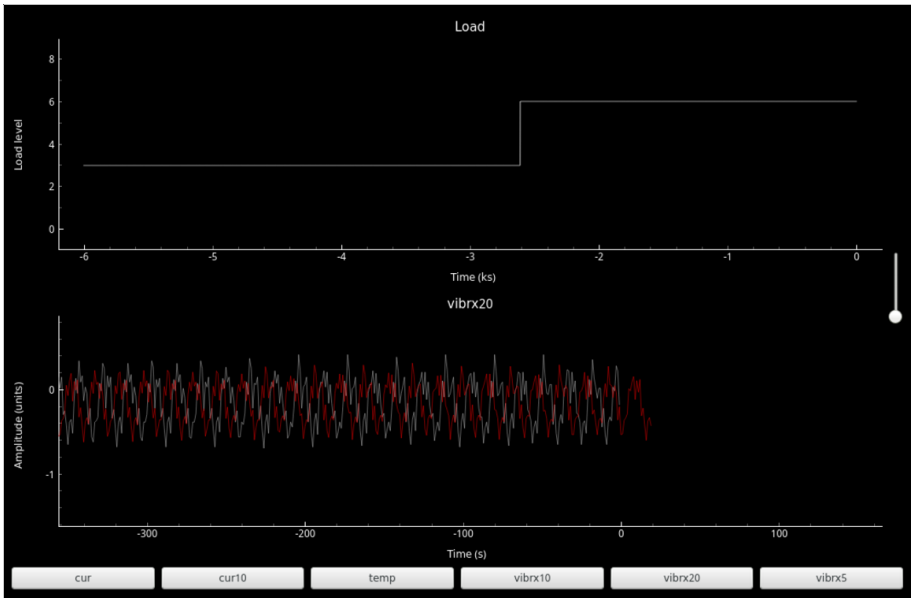
[3] Producent-konsument-mönstret är ett välkänt designmönster involverande separata processer där en producentprocess ansvarar för att producera data, exempelvis till ett databuffer, och en konsumentprocess som ansvarar för att ta emot denna data.

plats med de nya datan. Fördelen med denna metod är att alla paket i den seriella kommunikationen kan tas emot och hanteras som de ska i MP. Denna metod är den som används i den slutgiltiga arkitekturen.

3.8 Visualisering

Då ett av syftena med detta projekt var att konstruera en produkt som kan fungera som en uppvisningsprodukt krävdes en intuitiv form av visualisering av den data som tas och behandlingen av denna data. Detta gjordes genom att använda en bildskärm kopplad till Raspberryn.

De kurvor som ritas upp är de intressanta för den aktuella prediktionsmodellen: till exempel lastnivå, predicerat värde med en förskjutning framåt som motsvarar prediktionshorisonten, samt det uppmätta värdet, så förhållanden mellan dessa synliggörs. I figur 3.10 syns en skärmdump från visualiseringsprogrammet.



Figur 3.10: En bild som visar det grafiska interface som konstruerades.

3.9 Prestandatester av systemet

I denna sektion beskrivs hur ett antal tester utförs. Syftet med dessa tester är att kunna analysera hur snabbt och väl systemet kan utföra ett antal operationer och hur systemets prestanda påverkas av detta. Denna data används sedan då beslut rörande systemets design behöver tas, till exempel hur ofta Arduinon skall avläsa dess sensorer och skicka denna information till Raspberryn.

Tidtagning i all python-kod är gjord med *time.process_time()* [14], vilken är en metod för att noggrant mäta hur lång processortid som passerat i exekveringen av

processen (utan att räkna in tid som passerat då processen väntat på exekveringstid).

3.9.1 Test av samplingsfrekvens för vibration

Detta test undersöker vilken samplingsfrekvens av vibrationssensorn som krävs för att det frekvensområde som anses vara intressant skall kunna undersökas. Detta område bör innehålla grundfrekvensen för motorvibrationerna samt en del av de övertoner som uppstår.

Detta test genomförs genom att variera samplingsfrekvensen hos Arduinon samtidigt som lasten hos motorn hålls konstant. Lasten hålls konstant eftersom amplituden samt frekvensen av vibrationerna från motorn ej varierar nämnvärt då lasten hålls konstant. Den enda faktorn som påverkar hur väl vibrationerna kan uppmätas blir då samplingsfrekvensen.

Med hjälp av fast fourier transform (FFT) undersöks sedan vibrationernas frekvensspektra visuellt. En bedömning av huruvida frekvensspektrat innehåller de frekvenser som anses intressanta görs sedan.

Resultatet av detta test kan ses i figurerna A.7, A.8, A.9, A.10. Dessa bilder visar de frekvensspektra som uppstod vid samplingsfrekvenserna 500Hz, 1000Hz, 1500Hz, respektive 2000Hz.

Ur detta kan utläsas att de toppar som uppstår med nämnvärd amplitud kan återskapas för alla samplingsfrekvenser mellan 500-2000Hz. Intressant data kan alltså erhållas så länge samplingsfrekvensen ligger i detta intervall.

För ytterligare undersökning av detta fenomen togs därför beslutet att närmare undersöka utseendet på de vibrationskurvor som användes för att skapa dessa frekvensspektra. Detta syns i figurerna A.11, A.12, A.9, och A.14.

Då figurerna A.12, A.9, och A.14 jämförs noteras ej stor skillnad mellan dessa. Då dessa i sin tur jämförs med figur A.11 noteras en skillnad i form på denna kurvan. Detta beror troligtvis på att de mindre toppar som noteras för större frekvenser ej existerar motsvarande frekvensspektrum för denna kurva.

Eftersom det framförallt är formen på kurvan som avgör hur väl denna kan utnyttjas för prediktion dras då slutsatsen att om systemets prestanda tillåter bör samplingsfrekvensen ej understiga 1000Hz.

3.9.2 Test av prestanda för alla ingående system

Avsikten med detta test är att avgöra hur de olika processer som tidigare har testats individuellt påverkas då de alla exekveras parallellt. Detta innefattar datainsamling i realtid, prediktion med hjälp av Ekkonos maskininlärningsmotor, behandling av data, samt visualisering av data. Detta motsvarar den avsedda användingen av systemet.

Det som skall mätas är hur många datapunkter per sekund som kan behandlas av systemet utan att mottagarhastigheten faller under sändhastigheten. I fall detta sker kommer datapunkter att behöva förkastas för att databehandlaren skall hinna med.

Resultatet av detta test ses i tabellen 3.7. Ur denna tabell kan utläsas att det ibland uppstår misslyckade sändningar då sändhastigheten överstiger 2000Hz. Systemet bör alltså ej överstiga denna gräns för att förhindra att det uppstår ett möjliga fel i insamlingen och behandlingen av datan. Notera att inga flaskhalsar (resulterande i buffer overflow och därmed bortkastade datapunkter) uppstod vid detta test.

Sändhastighet (Hz)	Mottagningsfel	Buffer overflow-fel
500	Nej	Nej
1000	Nej	Nej
1500	Nej	Nej
2000	Ibland	Nej
2300	Ibland	Nej

Tabell 3.7: Data från test av det totala systemet.

De vibrationstester som genomförts indikerar att en samplingshastigheten bör ligga mellan 1000-2000Hz. De tester av sänd- och mottagarhastighet samt data-processeringshastigheten indikerar att dessa hastigheter är möjliga att uppnå. Då det sammanlagda testet körs visar dock detta att viss försiktighet bör iakttas då sändhastigheten närmar sig 2000Hz. För att sammanfatta bör alltså sändhastigheten ligga någonstans mellan 1500-2000Hz. För säkerhetsskull används 1500Hz för att minimera antalet mottagningsfel men ändå fånga tillräckligt med högfrekventa motorvibrationer.

3.9.3 Test av prestanda för prediktion

Den centrala delen av mjukvarusystemet, förutom datainsamling, databehandling och visualisering, är prediktionen.

Testen utförs genom att med en fix sekvens av 100 000 tidigare insamlade datapunkter, för varje datapunkt, lägga till datapunkten till modellens cache samt utföra prediktion enligt modellens inställning. Detta åstadkoms för ett antal olika modeller, se A.2. Tidsåtgång för varje prediktion samt tillägg till modellens cache sparas.

Värden för prediktionens korrekthet noteras också. Detta görs genom att beräkna den så kallade “mean-absolute-percentage-error” eller mape (se [13] för mer information). Då mape skall beräknas för en given datapunkt görs detta genom att ta hänsyn till de 1000 föregående datapunkterna.

4

Resultat

De slutgiltiga resultat som har uppnåtts under projektets gång redovisas här. Detta innefattar en överblick av den fysiska konstruktionen, systemets beteende och bedömningar av det systemets prestanda. De viktiga resultaten presenteras här i form av tabeller och figurer. Övriga figurer som refereras till återfinns i appendix A.3.

4.1 En överblick av systemets beteende under datainsamling

Här följer plotter av lasten, motortemperaturen, motorströmmen, och vibrationen i x-led, y-led och z-led för de 2 000 000 första insamlade datapunkterna. Datan har samlats in med ca 2000 datapunkter/sekund, så hela tidintervallet som visas motsvarar 1000 sekunder. Under insamlingen varierade lasten enligt ett slumpmässigt genererat program. I figur A.2 syns lasten under intervallet, som här varierar mellan nivåerna 1, 2, 7 och 8. 8 innebär i detta fall så stor last som möjligt och 0 innebär den lägsta möjliga lasten.

Motortemperaturen visas i figur A.3. Som synes stiger motorns temperatur långsamt från ca 25 grader (rumstemperaturen) till runt 46 grader under denna tidsperiod. Ökningen är som väntat kraftigast då lasten är som högst, vilket syns om man jämför tidpunkterna för laständringarna och motortemperaturkurvans lutning.

Motorns strömförbrukning visas i figur A.1. Ur denna figur kan utläsas att motorströmmen framför allt beror på den lastnivå som används.

I figurerna A.4, A.6, A.5 visas motorvibrationerna i axlarna x, y, respektive z. Notera att graferna för Y och Z är mycket lika, trots att Z-axeln är parallell med den axeln kring vilken motorn roterar och Y är vinkelrät mot den. Notera också att amplituden i accelerationen i y-led är förskjuten uppåt på grund av att gravitationskraften (motsvarande accelerationen 1g) som syns i mätdatat. Grafen för X är mycket mer uniform och med lägre amplitud, vilket troligen beror på obalanser i monteringen under datainsamlingen. Detta är inte något som undersöktes närmare.

4.2 Resultat av prestandatester

I denna sektion beskrivs resultat av utförda tester vars syfte är att bedöma prestanda för komponenter i systemet eller systemet i helhet.

4.3 Resultat av test av prediktion

I denna sektion presenteras en överblick av prediktionsresultaten för ett antal olika modeller, samt kort analys av dessa resultat. Samtliga modeller är tränade med samma dataset (samma som visas sektion 4.1) med en samplingsfrekvens på 2000Hz. Storleken av träningsdatasetet var 1 000 000 punkter, dvs ett tidsintervall av 500 sekunder. Datorn som användes vid träningen har en Intel I7-27200M 2.2 GHz processor.

I följande diagram visas prediktioner med modeller tränade enligt ovan, applicerade på data från samma datainsamlingssession, men med utökat tidsintervall till 2 000 000 punkter, dvs 1 000 sekunder. De riktiga uppmätta värdet visas förskjutet tillsammans med prediktionen så att de kan jämföras visuellt.

För varje modells prediktion visas också en MAPE-1000 graf, d.v.s. Mean Absolute Percentage Error för de 1000 föregående punkterna (500 ms). Anledningen till att fönstret för medelvärdesbildningen valdes till 1000 punkter är att åskådliggöra MAPE på den tidsskala (1000 sekunder) som visas i bilderna. Ett för högt eller lågt värde skulle gömma toppar eller dalar av MAPE värdet.

Som synes i resutatbilderna är prediktionsfelet som väntat generellt större i den andra halvan av graferna, då de första 500 sekunderna av datamängd är exakt samma som träningsmängd.

Noterbart är att prediktionen av motorströmmen (figur 4.13) inte alls verkar få sämre resultat i den andra halvan, medan prediktionen av motortemperaturen (figur A.15) blir helt fel när modellen matas med data som inte var med i träningssettet. Anledningen att temperaturförutsägelsen blir så mycket sämre beror troligen på att temperaturen ändras så långsamt att detta tidsintervall ej uppvisar dynamiken på ett sätt som ej är optimalt för träning av Random Forest-modeller, då det predikterade värdet blir mycket starkt kopplat till träningsdatat.

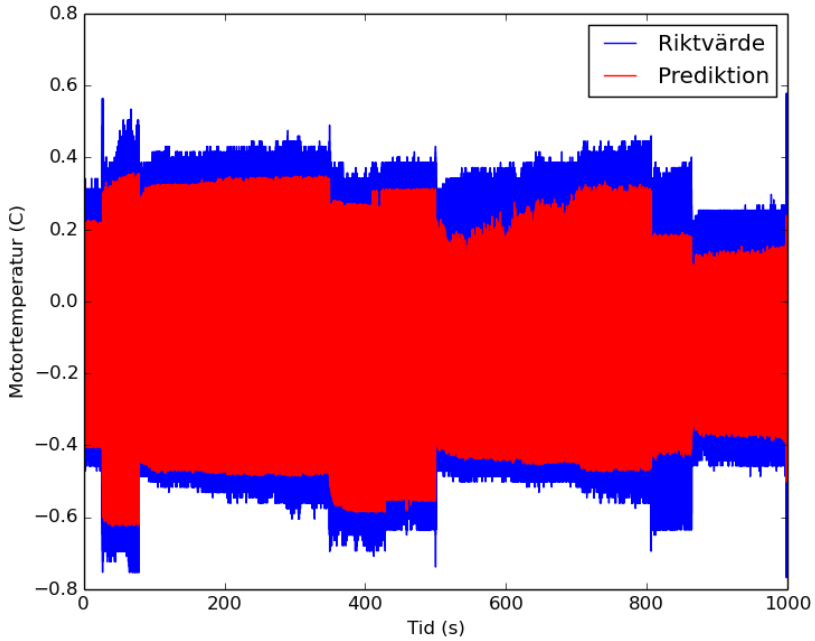
4.3.1 Prediktion av vibration

I denna sektion presenteras resultatet av två olika modeller som predicerar vibrationen i x-led. Modellerna förutsäger värdet för 5 respektive 10 millisekunder framåt i tiden. De attribut som användes som input till modellerna var nuvarande last för motorn (LOAD) och prediktionsmål (vibration i x-led). Lag-attribut på upp till 50 ms användes på vibration i x-led (VIBRX). Även ett moving-average-attribut på 10 ms respektive 20 ms var applicerat.

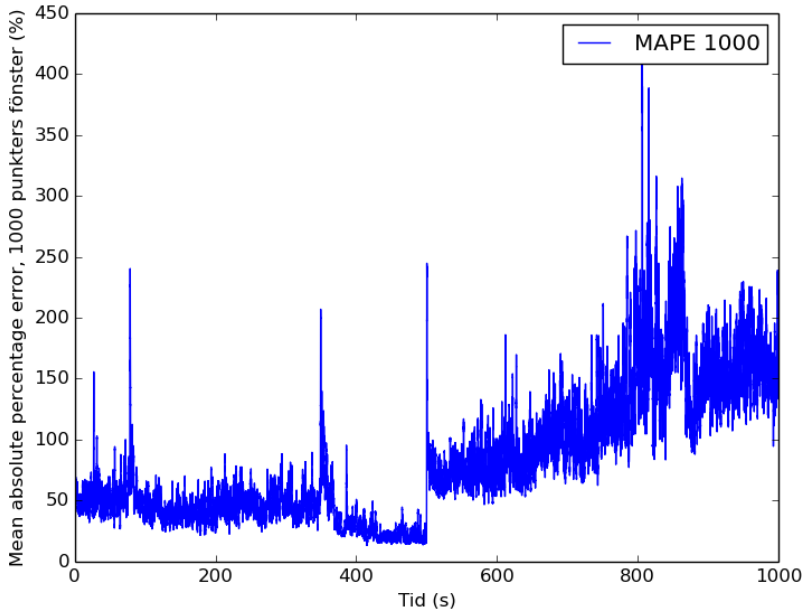
Modellerna tog 1100 sekunder (ca 18 min) respektive ca 1230 sekunder (ca 20 minuter) att träna.

4.3.1.1 5 ms prediktion av vibration i x-led

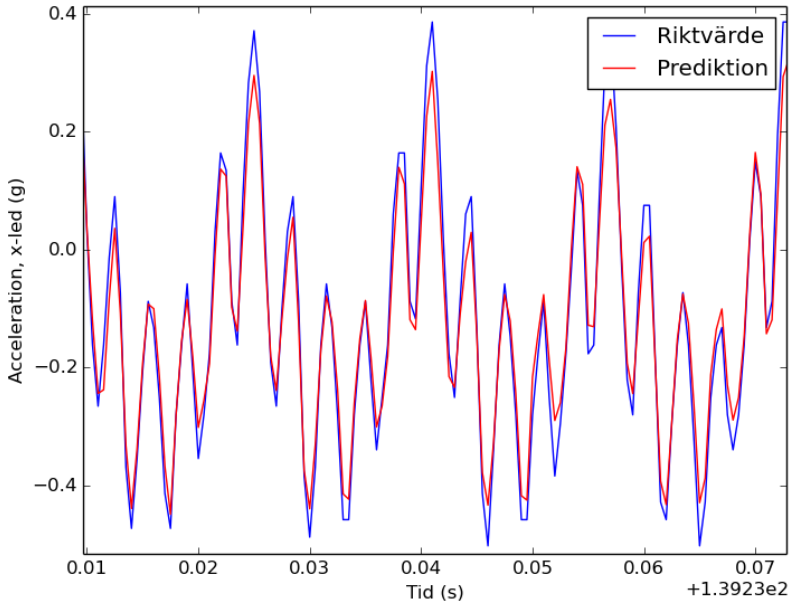
Denna modell förutsäger vibrationen i x-led för 10 punkter frammåt, d.v.s. 5 millisekunder och resultatet av detta kan ses i figurerna 4.1, 4.2, 4.3 och 4.4.



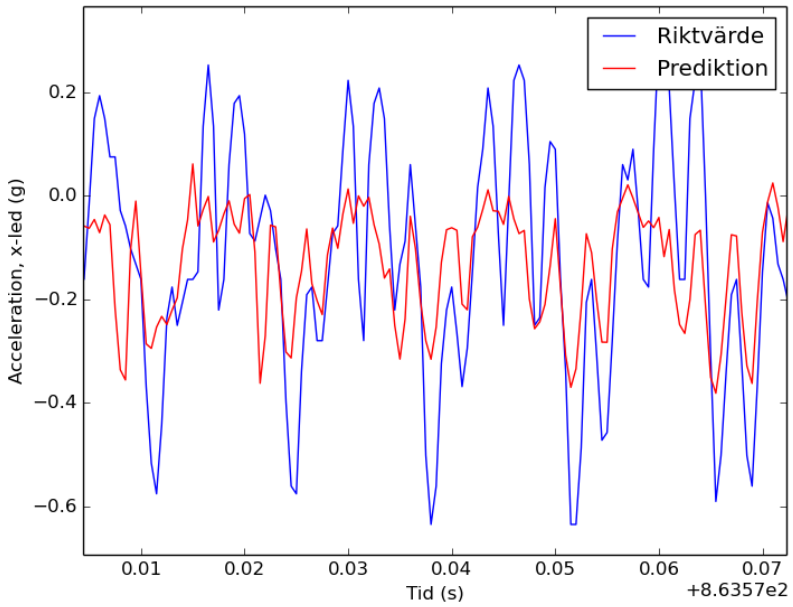
Figur 4.1: Överblick av amplitudskillnaden vid 5 ms prediktion av vibration i x-led.



Figur 4.2: MAPE 1000 vid 5 ms prediktion av vibration i x-led.



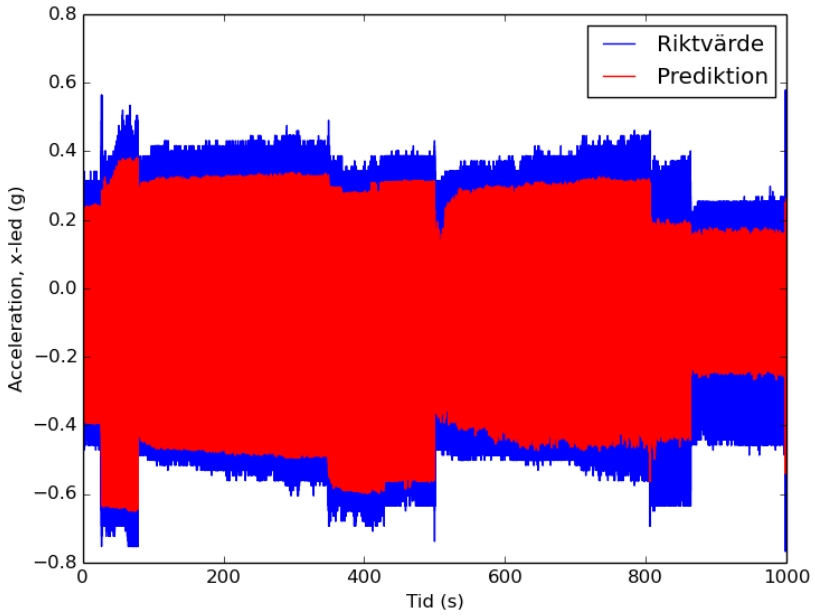
Figur 4.3: Vågformen av 5 ms prediktion av vibration i x-led, vid låg MAPE.



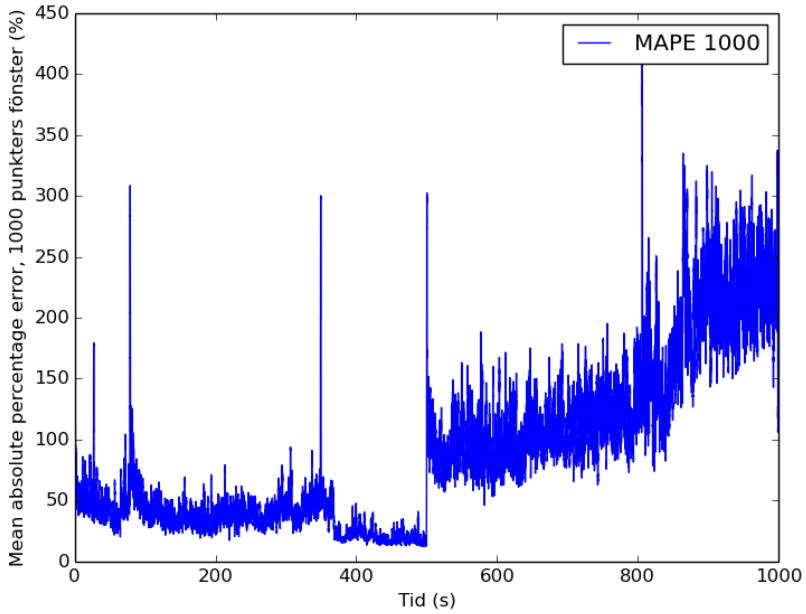
Figur 4.4: Vågformen av 5 ms prediktion av vibration i x-led, vid hög MAPE.

4.3.1.2 10 ms prediktion av vibration i x-led

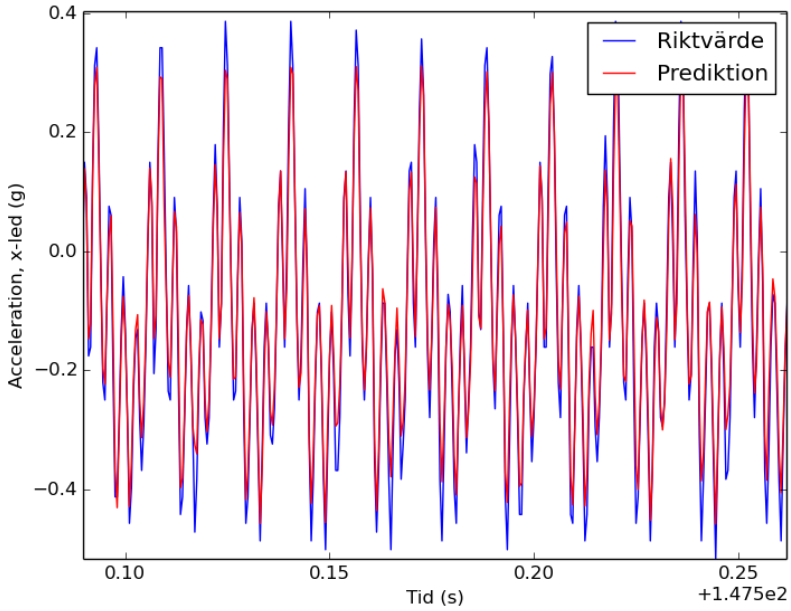
Denna modell förutsäger vibrationen i x-led för 20 punkter framåt, d.v.s. 10 millisekunder och resultatet av detta kan ses i figurerna 4.5, 4.6, 4.7 och 4.8.



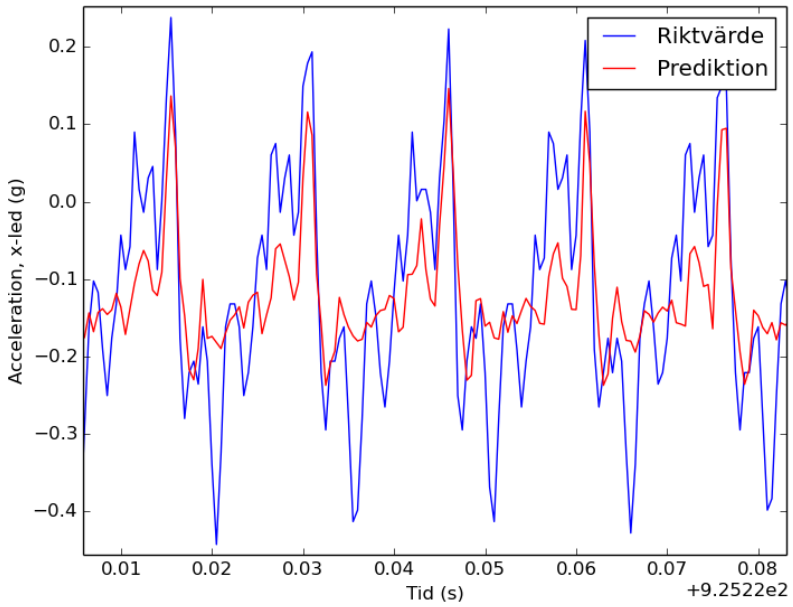
Figur 4.5: Överblick av amplitudskillnaden vid 10 ms prediktion av vibration i x-led.



Figur 4.6: MAPE 1000 vid 10 ms prediktion av vibration i x-led.



Figur 4.7: Vågformen av 10 ms prediktion av vibration i x-led, vid låg MAPE.



Figur 4.8: Vågformen av 10 ms prediktion av vibration i x-led, vid hög MAPE.

4.3.2 Prediktion av motorström.

Två modeller för prediktion av motorströmmen har testats. En med längre prediktionshorisont, 1 sekund, och en med kortare, 5 millisekunder.

Motorströmmens medelvärde är mycket stark kopplad till lastnivån och är näst intill statisk för en viss lastnivå (se figur 4.9). Strömmens amplitud varierar dock snabbt och regelbundet (som kan ses i figur 4.12); frekvensen bör följa motorns rotationshastighet. Ett sågtandsmönster med en period på ca 10 ms (100Hz) kan ses i figur 4.12 och 4.16.

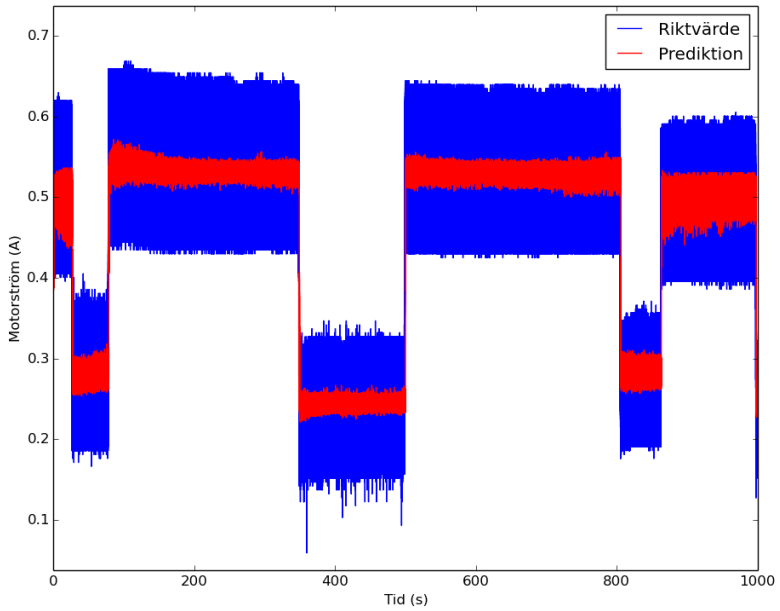
4.3.2.1 1000 ms prediktion av motorströmmen.

Denna modell förutsäger motorströmmen för 2000 punkter framåt, d.v.s. 1 sekund framåt.

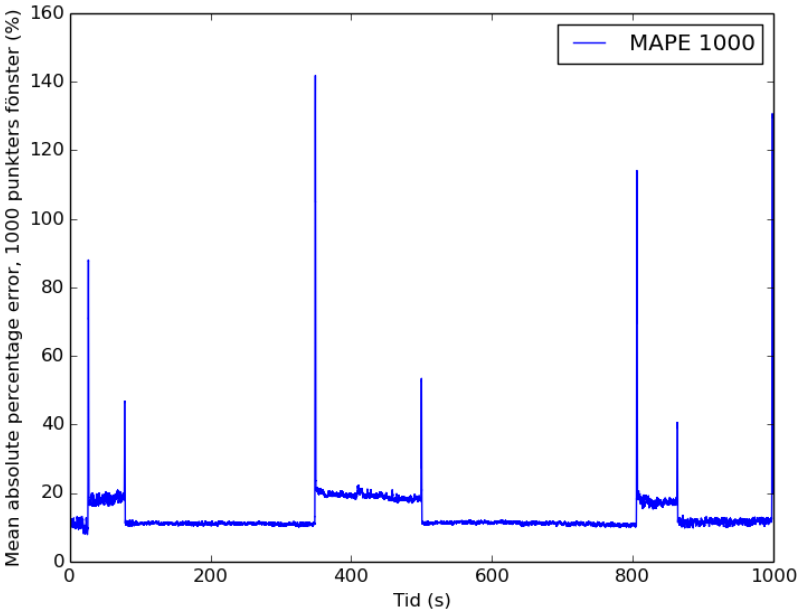
Attribut som användes var motorns last och strömförbrukning. Moving average på motorströmmen var på 10 ms. Lags upp till 100 ms bakåt på strömförbrukningen var applicerade.

Denna modell har som synes mycket konstant MAPE, utom precis vid lastskiften. I figur 4.11 som visar bilden vid lastskiftet precis innan 500 sekunder har passerat, ser vi att den topp som uppstår beror på att prediktionen helt enkelt fortsätter att förutsäga samma värde ända fram till att lasten ändras, en ändring som modellen omöjligen kan förutse baserat på de attribut som finns tillgängliga.

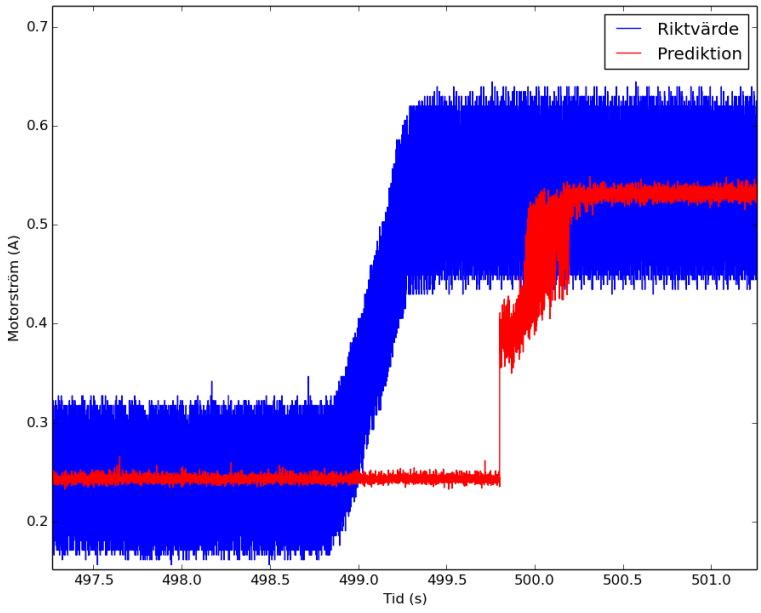
Vid inspektion av vågformen i figur 4.12 ser vi att modellen har svårt att förutsäga en så snabbt skiftande amplitud med en så lång prediktionshorisont. Medelvärdet är dock mycket nära. Detta visar den medelvärdesbildande funktionen hos Random Forest-modeller, som när prediktionen är osäker tenderar att ge ett medelvärde av de många Decision Tree-modeller som Random Forest-modellen består av.



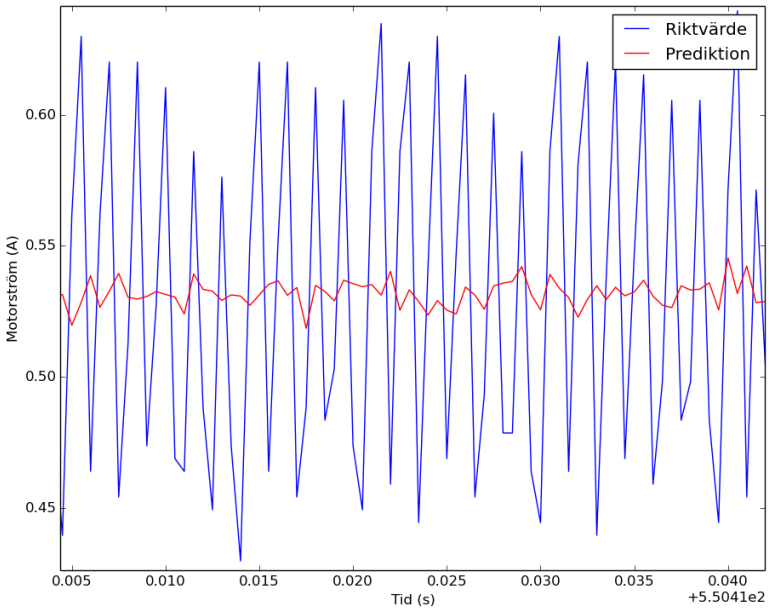
Figur 4.9: Överblick av amplitudskillnaden vid 1000 ms prediktion av motorströmmen.



Figur 4.10: MAPE 1000 vid 1000 ms prediktion av motorström.



Figur 4.11: Lastskifte under ca 1 sekund från 1000 ms prediktion av motorström.



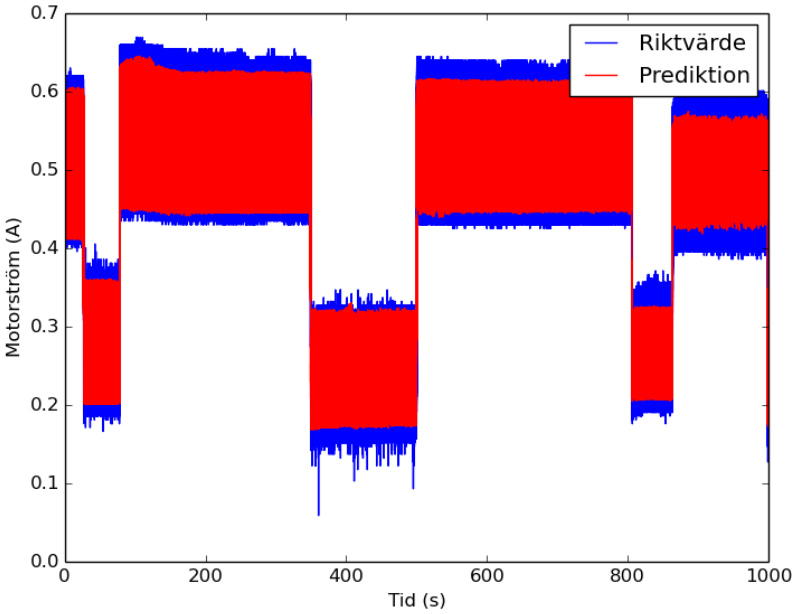
Figur 4.12: Vågformen av 1000 ms prediktion av motorström.

4.3.2.2 5 ms prediktion av motorströmmen.

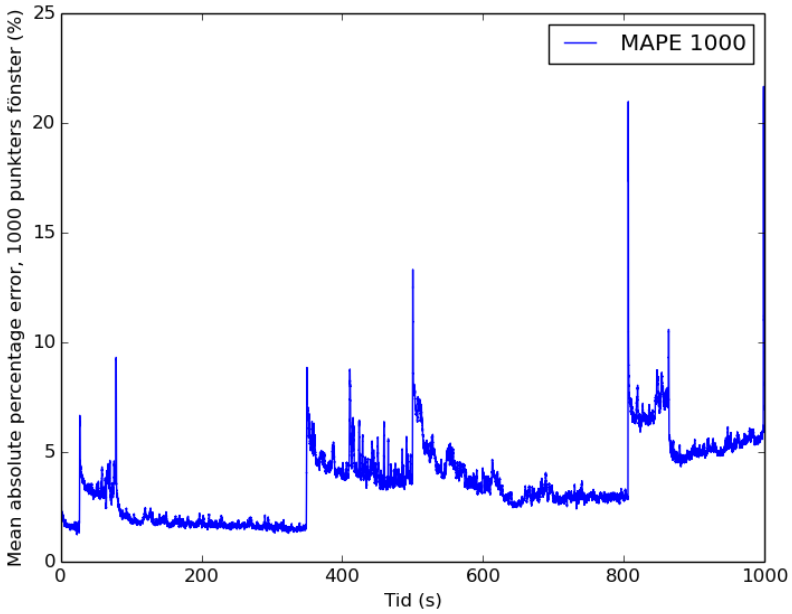
Denna modell förutsäger motorströmmen för 10 punkter framåt, d.v.s. 5 millisekunder.

Attribut som användes var motorns last, motorns temperatur och motorströmmen. Moving average på motorströmmen var på 50 ms. Lags upp till 50 ms bakåt på motorströmmen var applicerade.

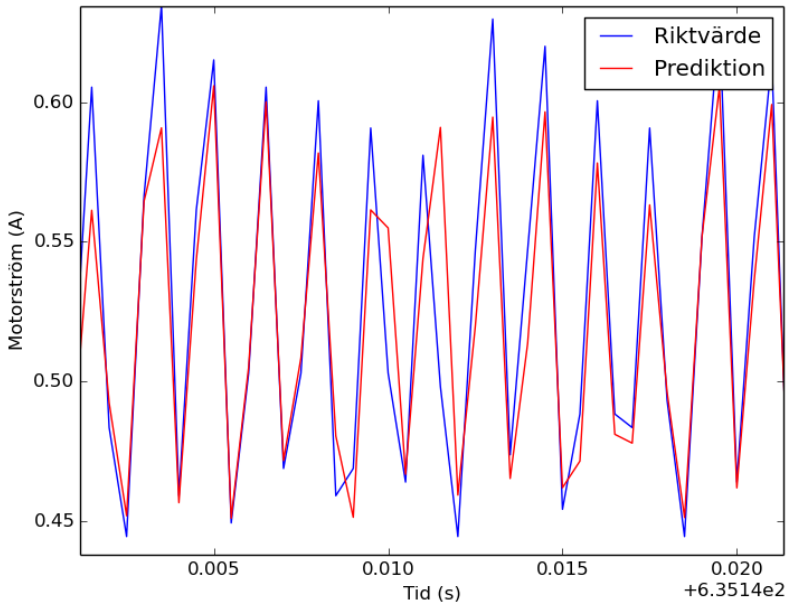
Med bara 5 ms prediktionshorisont blir både prediktionerna av både amplitud och vågform mycket mer korrekta jämfört med modellen med 1000 ms prediktion.



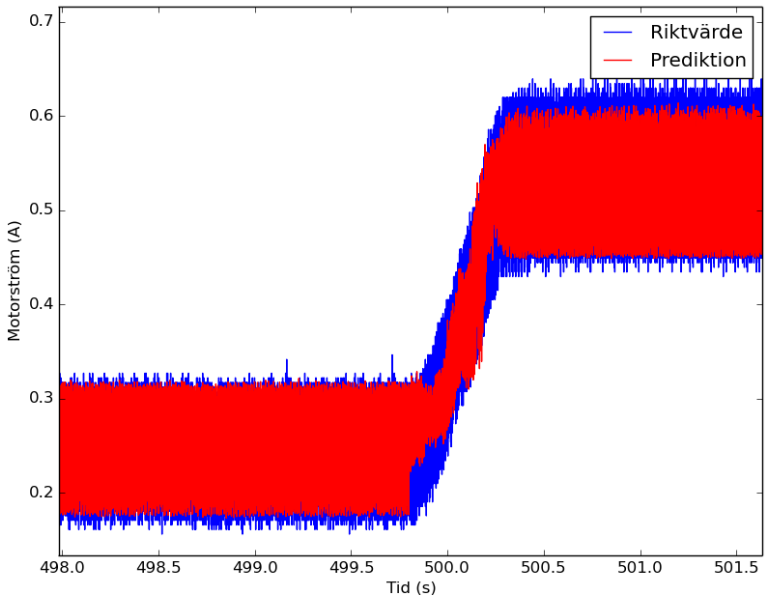
Figur 4.13: Överblick av amplitudskillnaden vid 5 ms prediktion av motorströmmen.



Figur 4.14: MAPE 1000 vid 5 ms prediktion av motorström.



Figur 4.15: Lastskifte under ca 1 sekund från 5 ms prediktion av motorström.



Figur 4.16: Vågformen av 5 ms prediktion av motorström.

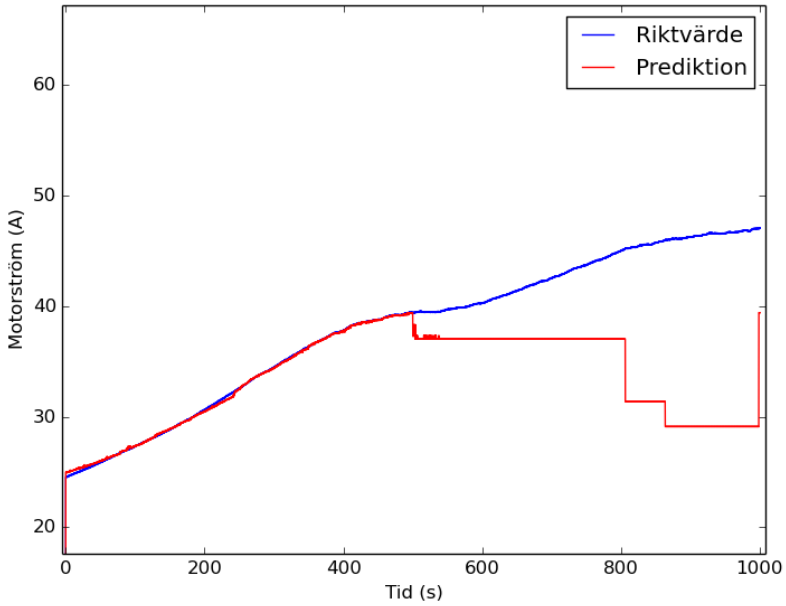
4.3.3 Prediktion av motortemperaturen, 5 s.

Denna modell förutsäger motortemperaturen för 10 000 punkter framåt, d.v.s. 5 sekunder. Som vi ser blir prediktionen efter 500 sekunder snabbt dålig, även om

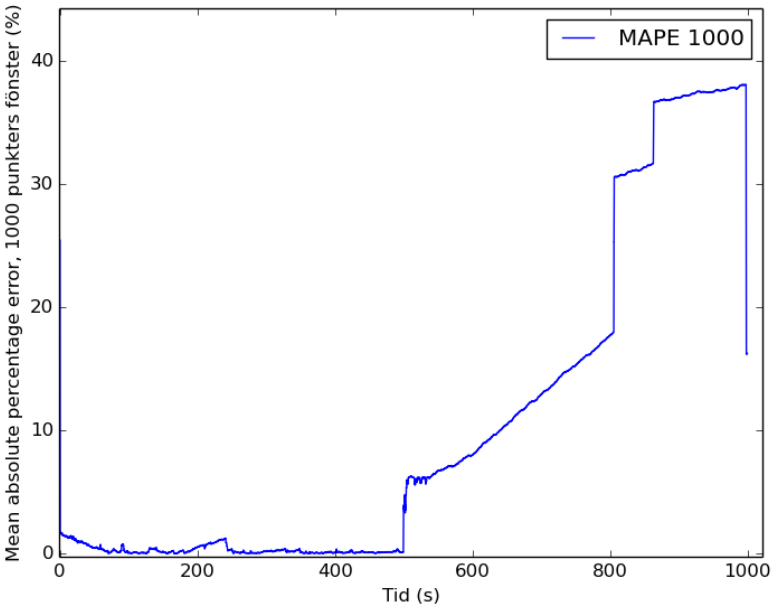
temperaturen följer sitt tidigare värde väldigt nära, med en liten ökning (som främst bör bero på lasten för tillfället). Detta tyder på en svaghet hos denna modelltyp.

Attribut som användes var motorns last och dess temperatur. Moving average på motorns temperatur var på 500 ms. Inga lags var applicerade. Modellen var inställd att göra ny prediktion var 500 ms (ny prediktion var 1000 sampel). Träningen tog 6 sekunder för denna modell.

En liknande modell men med 10 ms prediktionshorisont fick nästan exakt samma resultat (se appendix A.4).



Figur 4.17: 5 s prediktion av motortemperaturen.



Figur 4.18: MAPE 1000 för 5 s prediktion av motortemperaturen.

5

Slutsats

I detta arbete har ett elektriskt system konstruerats. Detta system bestod av en motor-generator koppling samt en lastbank för att reglera det mekaniska motstånd som motorn upplever då denna skall driva generatorn. Lastbanken bestod av ett antal parallellkopplade resistorer samt transistorer som används för att reglera huruvida en enskild resistor kan leda ström eller ej. De motorer som användes var 12V borstade permanentmagnets DC-motorer.

Lastbanken styrdes med hjälp av en Arduino Uno. Sensorer som mäter temperatur, vibration, samt strömförbrukning var kopplade till Arduinon. Denna information skickades sedan vidare med hjälp av seriell kommunikation till en Raspberry Pi model 3B. Med hjälp av Ekkonos plattform för maskininlärning utfördes sedan prediktion på denna data i syfte att predicera systemets beteende ett kort tid framåt i tiden.

Prestandan av detta system utvärderades sedan utifrån ett antal kriterier, såsom tillförlitlighet hos prediktionen, hur väl denna information förmedlas till användaren av systemet och hur snabbt systemet klarar av att behandla data. En närmare diskussion av detta kan ses i detta kapitel.

5.1 Hur väl har projektet uppnått sitt syfte?

En del av syftet med detta projekt var att utvärdera hur väl Ekkonos plattform kan användas för att predicera ett systems beteende utifrån dess nuvarande tillstånd. Utvärderingen av detta gjordes med hänsyn till mape, samt en visuell inspektion av den data som insamlades. Den visuella utvärderingen är något subjektiv, eftersom detta beror på vad observatören anser acceptabelt. Författarna anser att det resultat som synes i sektion 4.3 tyder på en bra noggrannhet hos prediktionen. Dock krävs för närvarande viss konfiguration av systemet i form av lags och moving average. För detta krävs enbart begränsad kunskap om hur Ekkonos plattform fungerar för att uppnå bra resultat. Författarna besitter inga djupare kunskaper om maskininlärning men har trots detta uppnått ett resultat som anses tillräckligt bra för att uppnå projektets syfte (se sektion 1.2).

Då det system som konstruerades skall agera som uppvisningsprodukt var det viktigt att de fysikaliska storheter som uppmäts skall uppvisa tydliga fysikaliska

tecken. Detta gäller framförallt vibration och temperatur. Temperatur varierar markant då systemet används och har varierat med ca. 20^o C (se figur A.3). Detta är en så pass stor variation att vem som helst kan märka detta och relatera denna storhet till informationen som visas i användargränssnittet.

Denna information skulle förmedlas på ett tydligt vis till en användare på så vis att även oinsatta personer skall kunna förstå systemets syfte. Genom samtal med utomstående personer avgjordes att detta behöver förbättras något. De kurvor som visas är fina att se på men den information de innehåller framförs ej tillräckligt tydligt. Det krävdes viss förklaring för att användaren skulle förstå innebörden av de kurvor som ritats. Efter muntligt förtydligande ansåg dock användarna att systemet var lättförståeligt i övrigt.

Trots detta lämpar sig inte den slutgiltiga produkten så väl som uppvisningsprodukt. Detta beror framförallt på att motor-generator kopplingen vibrerar betydligt mer än förväntat. Detta är bra på det sättet att vibrationsdatan blir mer distinkt, men inte så bra för en uppvisningsprodukt eftersom producerar ett obehagligt ljud. Detta är systemets största tillkortakommande i detta avseende.

5.2 Möjlig vidareutveckling och förbättringar

I denna sektion beskrivs möjligheter till vidareutveckling eller förbättring av det genomförda projektet.

5.2.1 Träning av modeller

Som kan ses i sektion 4.3 finns det möjlighet till förbättring av den utförda prediktionen. Detta skulle kunna jämföras genom att bättre undersöka hur parametrarna för träningen skulle kunna konfigureras optimalt för detta användningsområdet, till exempel lags, backwards moving average, etc. Träningen skulle också kunna förbättras genom att utföras på ett större och mer varierat dataset för att få med mer variation i träningen. Anledningen till att dessa inte gjorts är huvudsakligen på grund av tidsbegränsningar.

5.2.2 Koppling mellan motorer, dämpning m.m.

Det mekaniska systemet som innefattar plattformen de båda motorerna är monterade på, kopplingen mellan dessa, samt dämpningen som skall förhindra att vibrationer från underlaget sprider sig till denna är i huvudsak egentillverkade. Detta har lett till viss instabilitet i systemet, särskilt på grund av kopplingen mellan motorerna. Denna kräver kontinuerligt underhåll för att hålla systemets vibrationer på en stabil nivå, vilket är viktigt för att prediktionerna skall ge bra resultat. En mer permanent lösning hade varit fördelaktig.

5.2.3 Vidareutveckling

Detta projekt har endast påvisat möjligheten till att använda Ekkonos plattform för maskininlärning för att förutse ett enkelt systems beteende en kort tid framöver. För att ge detta en mer praktisk användning skulle detta kunna används för så kallad feldetektering, där avvikelser från ett systems normala beteende används för att påvisa någon typ av fel. Förslags skulle ett onormalt stort värde på mape kunna användas för att indikera att något fel har inträffat.

Litteraturförteckning

[1] Analog Devices, "ADXL335," ADXL335 datablad, https://www.sparkfun.com/datasheets/Components/SMD/adxl335.pdf, 2009 [Rev 0]

[2] Allegro Microsystems Inc, "Fully Integrated, Hall Effect-Based Linear Current Sensor with 2.1 kVRMS Voltage Isolation and a Low-Resistance Current Conductor," ACS712 datablad, https://www.sparkfun.com/datasheets/BreakoutBoards/0712.pdf, 2006 [Rev 7]

[3] Analog Devices, "Low Voltage Temperature Sensors TMP35/TMP36/TMP37," TMP36 datablad, http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Temp/TMP35_36_37.pdf, 2010 [Rev F]

[4] Jumo, "Platinum-chip temperature sensors with connecting wires to EN 60 751," PT100 datablad, https://www.electrokit.com/productFile/download/3819, [Data Sheet 90.6121]

[5] Igarashi Motoren GMBH, "N2738-125," N2738-125 datablad, http://www.produktinfo.conrad.com/datenblaetter/225000-249999/244475-da-01-de-IGARASHI_N2738_125_12V_E_MOTOR.pdf

[6] Atmel, "AVR120: Characterization and Calibration of the ADC on an AVR," http://www.atmel.com/Images/Atmel-2559-Characterization-and-Calibration-of-the-ADC-on-an-AVR_ApplicationNote_AVR120.pdf, 2016 [Rev 2559E]

[7] Limor Fried, "Adafruit_MAX31865", https://github.com/adafruit/Adafruit_MAX31865, 2016 [Commit d59fe8c]

[8] Resistor Guide, "Resistor Sizes and Packages", sektion: "Axial resistor size", [Online], besökt: 6 juni, 2017, tillgänglig: http://www.resistorguide.com/resistor-sizes-and-packages/

[9] Wikipedia, "Single-board computer", [Online], besökt: 6 juni, 2017, senast ändrad: 24 maj, 2017, tillgänglig: https://en.wikipedia.org/wiki/Single-board_computer

[10] Arduino, "Technical specs", [Online], besökt: Jun 6, 2017, tillgänglig: https://www.arduino.cc/en/main/arduinoBoardUno#techspecs

[11] Raspberry Pi Foundation, “Raspberry Pi 3 Model B”, [Online], besökt: 6 juni, 2017, tillgänglig: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[12] On Semiconductor, “BD135G, BD137G, BD139G, Plastic Medium-Power Silicon NPN Transistors”, BD139G datablad, http://www.farnell.com/datasheets/1911665.pdf, 2013 [Rev 17]

[13] Wikipedia, “Mean absolute percentage error”, [Online], besökt: 8 juni, 2017, senast ändrad: 9 maj, 2017, tillgänglig: https://en.wikipedia.org/wiki/Mean_absolute_percentage_error

[14] Python Software Foundation, “The Python Standard Library v.3.4”, sektion: “16.3. time — Time access and conversions”, [Online], besökt: 8 juni, 2017, senast ändrad: 25 Jun, 2016, tillgänglig: https://docs.python.org/3.4/library/time.html#time.process_time

A

Appendix

A.1 Lista av komponenter

En fullständig lista innehållande de komponenter som använts under projektets gång:

- En Raspberry Pi 3 model B
- En kylfläns för Raspberry Pi
- En Arduino Uno
- En bildskärm för visualisering
- Lastkretsen:
  - Två resistorer á 270 Ohm 5 Watt
  - Två resistorer á 150 Ohm 5 Watt
  - Två resistorer á 100 Ohm 5 Watt
  - Två resistorer á 47 Ohm 5 Watt
  - Åtta transistor av typ BD139G
  - Åtta lysdioder
  - Åtta resistorer á 330 Ohm
  - Åtta resistorer á 820 Ohm
- Två borstade permanentmagnet DC-motorer Igarashi N2738-125 12 V
- Sensorer:
  - PT100 givare
  - PT100RTD MAX31865
  - Accelerometer av typ ADXL335
  - Temperatursensor av typ TMP36
  - En resistor á 1 Ohm som används för strömmätning
- Strömförsörjning:

- 12 V 2.5 A DC spänningskälla från Deltaco
- 5 V 2.4 A DC spänningskälla från Clas Ohlson

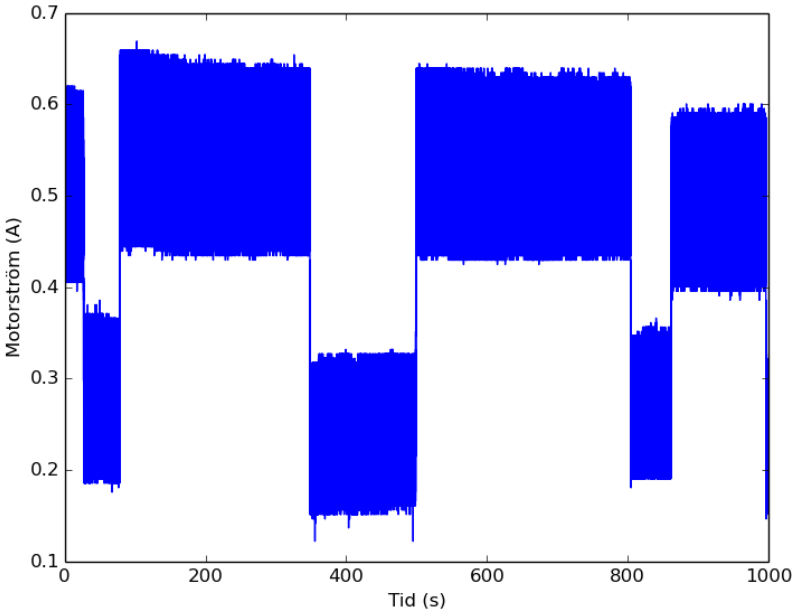
A.2 Testmodeller för prestandatest

```
models_meta = {
'temp': dict(
    freq = 10,
    attrs = [LOAD, TEMP],
    target = TEMP,
    future_prediction = 20,
    lags = {TEMP:[1,2]},
    moving_average = {TEMP:[100]},
),
'cur': dict(
    freq = 1,
    attrs = [LOAD, CUR],
    target = CUR,
    future_prediction = 2000,
    lags = {CUR:[10, 100]},
    moving_average = {CUR:[10]},
    plotting = dict(
        y_range = [0.1, 0.8],
    ),
),
'cur10': dict(
    freq = 1,
    attrs = [LOAD, TEMP, CUR],
    target = CUR,
    future_prediction = 10,
    lags = {CUR:[1,2,3,4,5,6,7,8,9,10,15,20,25,100]},
    moving_average = {CUR:[100]},
    plotting = dict(
        y_range = [0.1, 0.8],
    ),
),
'vibrx20': dict(
    freq = 1,
    future_prediction = 20,
    attrs = [LOAD, VIBRX],
    target = VIBRX,
    lags = {VIBRX:[1,2,3,4,5,6,7,8,9,10,20,30,40,50,60,70,80,90,100]},
    moving_average = {VIBRX:[40]},
    plotting = dict(
```

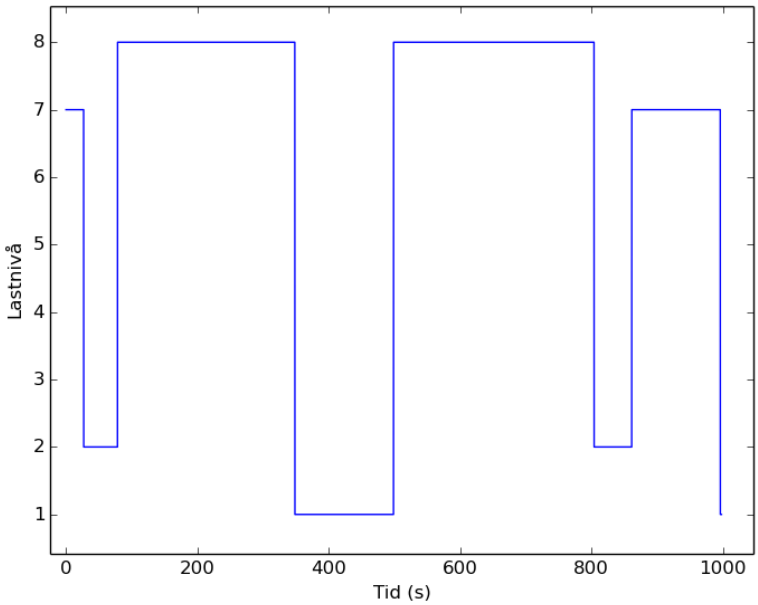
```
        x_range = [-60,60],
        y_range = [-4, 4],
    ),
),
'vibrx10': dict(
    freq = 1,
    future_prediction = 10,
    attrs = [LOAD, VIBRX],
    target = VIBRX,
    lags = {VIBRX:[1,2,3,4,5,6,7,8,9,10,20,30,40,50,60,70,80,90,100]},
    moving_average = {VIBRX:[20]},
    plotting = dict(
        x_range = [-60,60],
        y_range = [-4, 4],
    ),
),
'vibrx5': dict(
    freq = 1,
    future_prediction = 5,
    attrs = [LOAD, VIBRX],
    target = VIBRX,
    lags = {VIBRX:[1,2,3,4,5,6,7,8,9,10,20,30,40,50,60,70,80,90,100]},
    moving_average = {VIBRX:[10]},
    plotting = dict(
        x_range = [-60,60],
        y_range = [-4, 4],
    ),
),
}
```

A.3 Resultat av tester av systemet

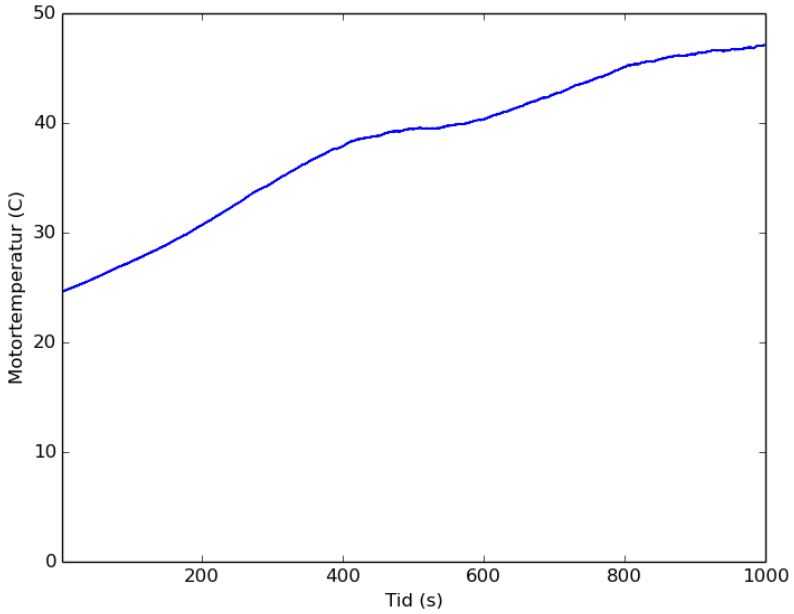
A.3.1 Systemets beteende under datainsamling



Figur A.1: Motorströmmen vid de första 2 miljoner insamlade datapunkterna.

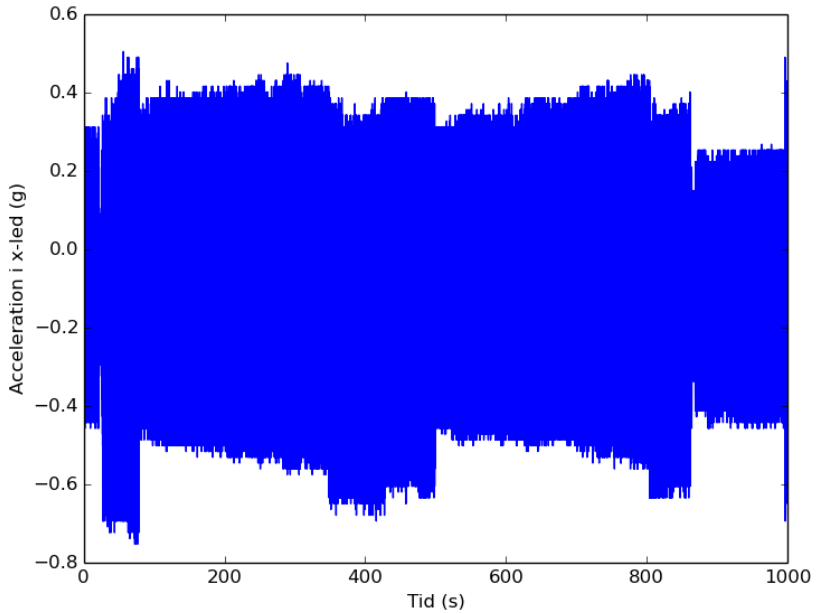


Figur A.2: Lasten vid de första 2 miljoner insamlade datapunkterna.

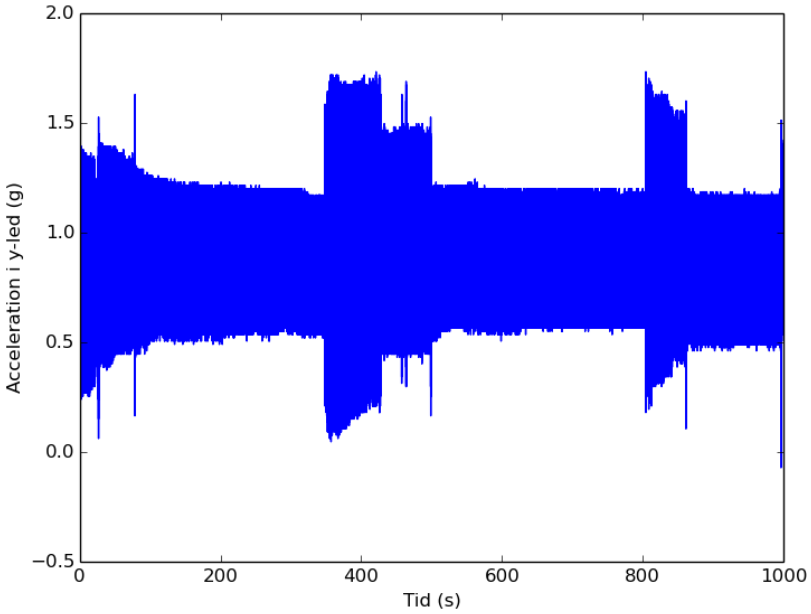


Figur A.3: Motortemperaturen vid de första 2 miljoner insamlade datapunkterna.

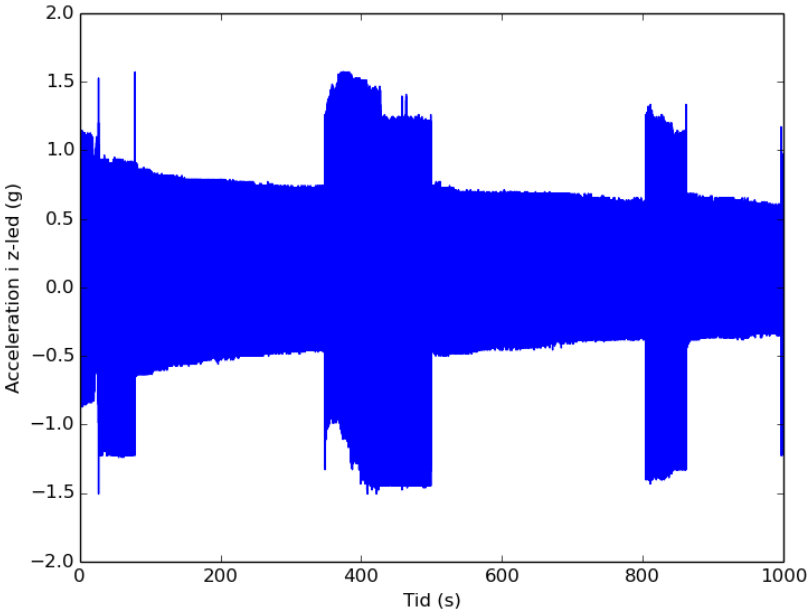
A.3.2 Uppmätt acceleration



Figur A.4: Acclererationen i x-led vid de första 2 miljoner insamlade datapunkterna.

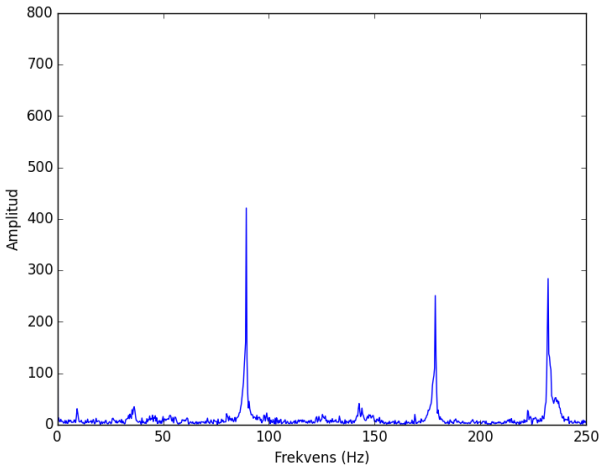


Figur A.5: Acclererationen i y-led vid de första 2 miljoner insamlade datapunkterna.

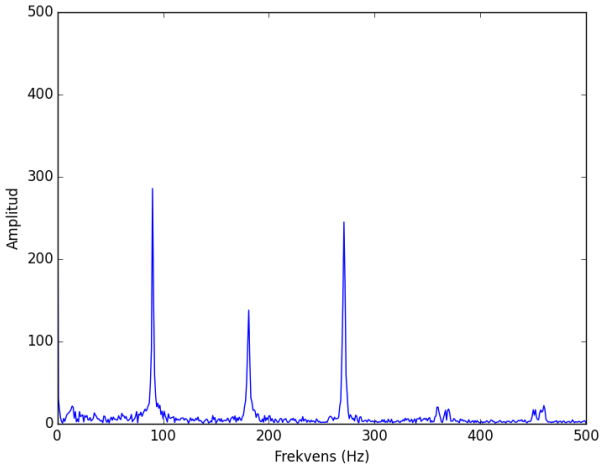


Figur A.6: Acclererationen z-led vid de första 2 miljoner insamlade datapunkterna.

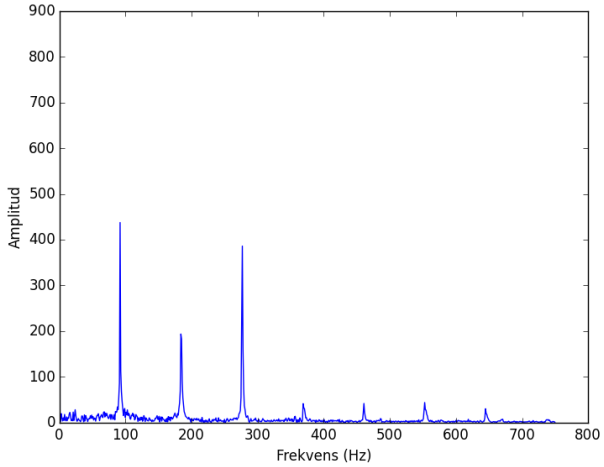
A.3.3 Uppmätta frekvensspektran



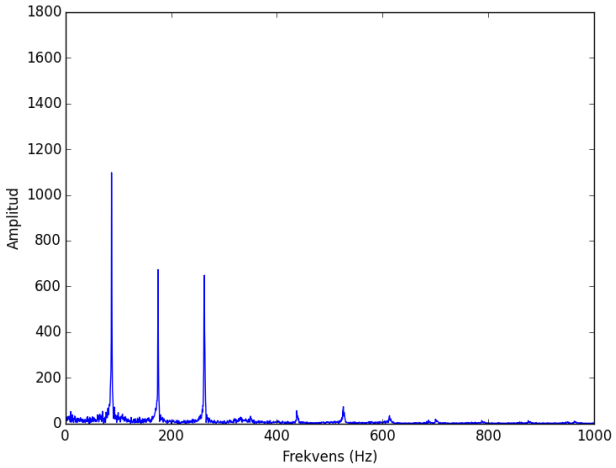
Figur A.7: Frekvensspektrat som uppmättes då samplingsfrekvensen var 500Hz.



Figur A.8: Frekvensspektrat som uppmättes då samplingsfrekvensen var 1000Hz.

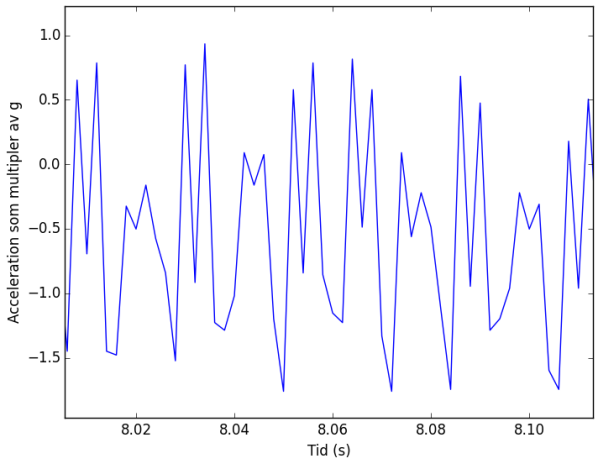


Figur A.9: Frekvensspektrat som uppmättes då samplingsfrekvensen var 1500Hz.

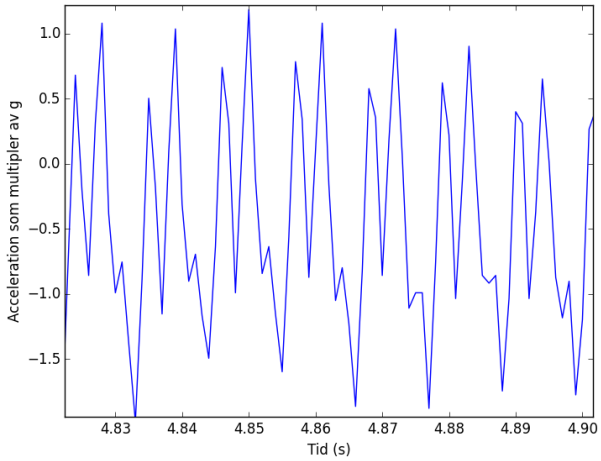


Figur A.10: Frekvensspektrat som uppmättes då samplingsfrekvensen var 2000Hz.

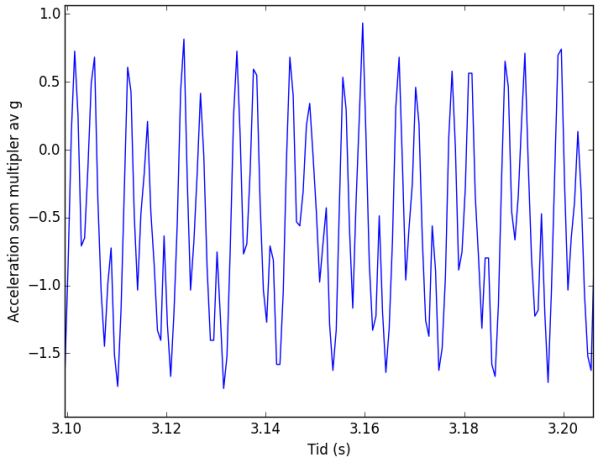
A.3.4 Vibrationskurvor



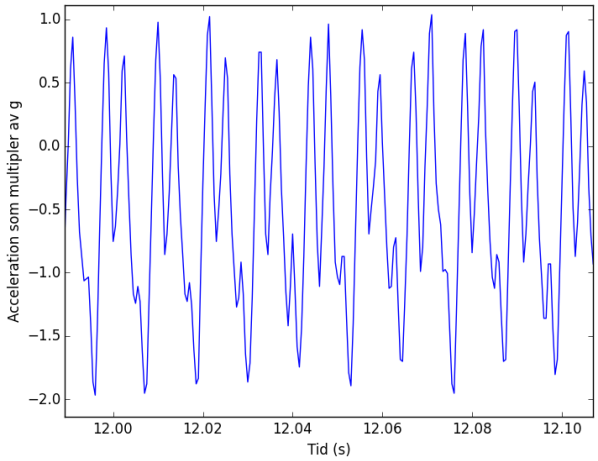
Figur A.11: En kurva som visar de vibrationer som uppmättes gentemot tiden då samplingsfrekvensen var 500Hz.



Figur A.12: En kurva som visar de vibrationer som uppmättes gentemot tiden då samplingsfrekvensen var 1000Hz.



Figur A.13: En kurva som visar de vibrationer som uppmättes gentemot tiden då samplingsfrekvensen var 1500Hz.

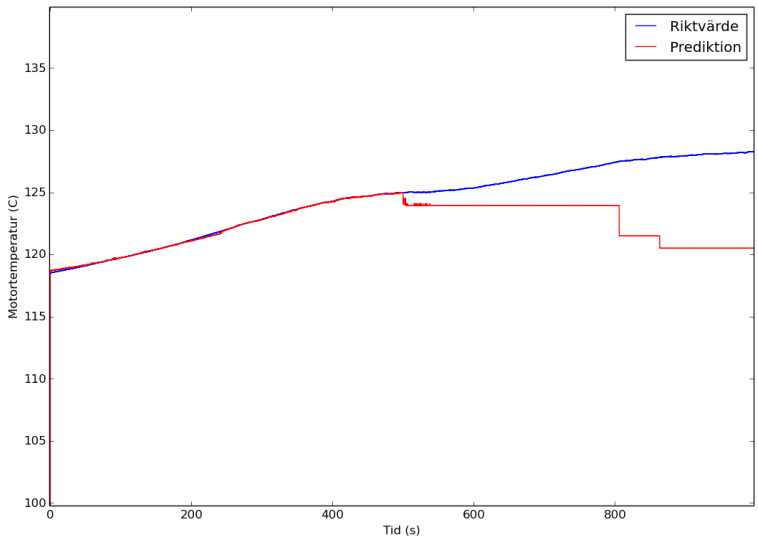


Figur A.14: En kurva som visar de vibrationer som uppmättes gentemot tiden då samplingsfrekvensen var 2000Hz.

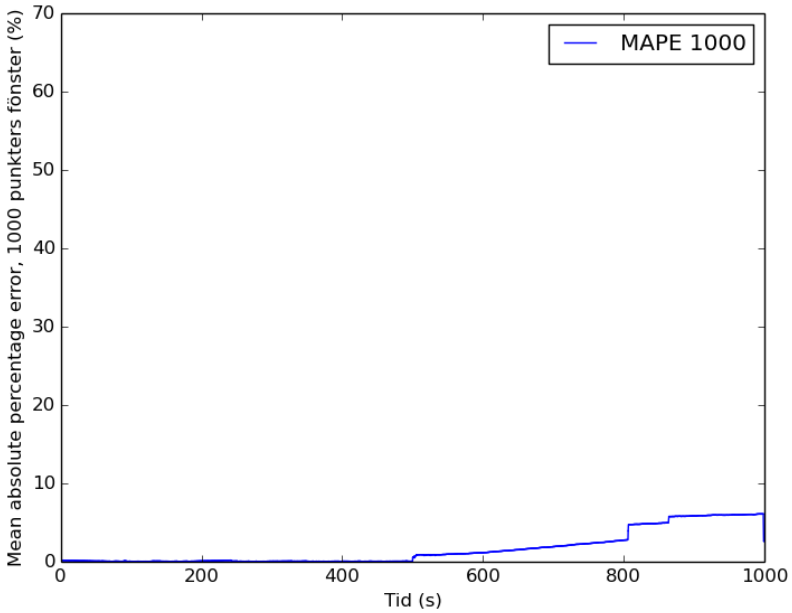
A.4 Prediktion av motortemperaturen, 10 ms.

Denna modell förutsäger motortemperaturen för 20 punkter framåt, d.v.s. 10 ms framåt. Vi ser nästan exakt samma resultat som prediktionen med 5 sekunders horisont.

Attribut som användes var LOAD och TEMP. Moving average på TEMP var på 10 ms. Lags på 0.5 och 1 ms bakåt på TEMP var applicerade. Modellen var inställd att göra ny prediktion var 5 ms (ny prediktion var 10:nde sampel). Träningen tog 51 sekuder för denna modell.



Figur A.15: 10 ms prediktion av motortemperaturen.



Figur A.16: MAPE 1000 för 10 ms prediktion av motortemperaturen.