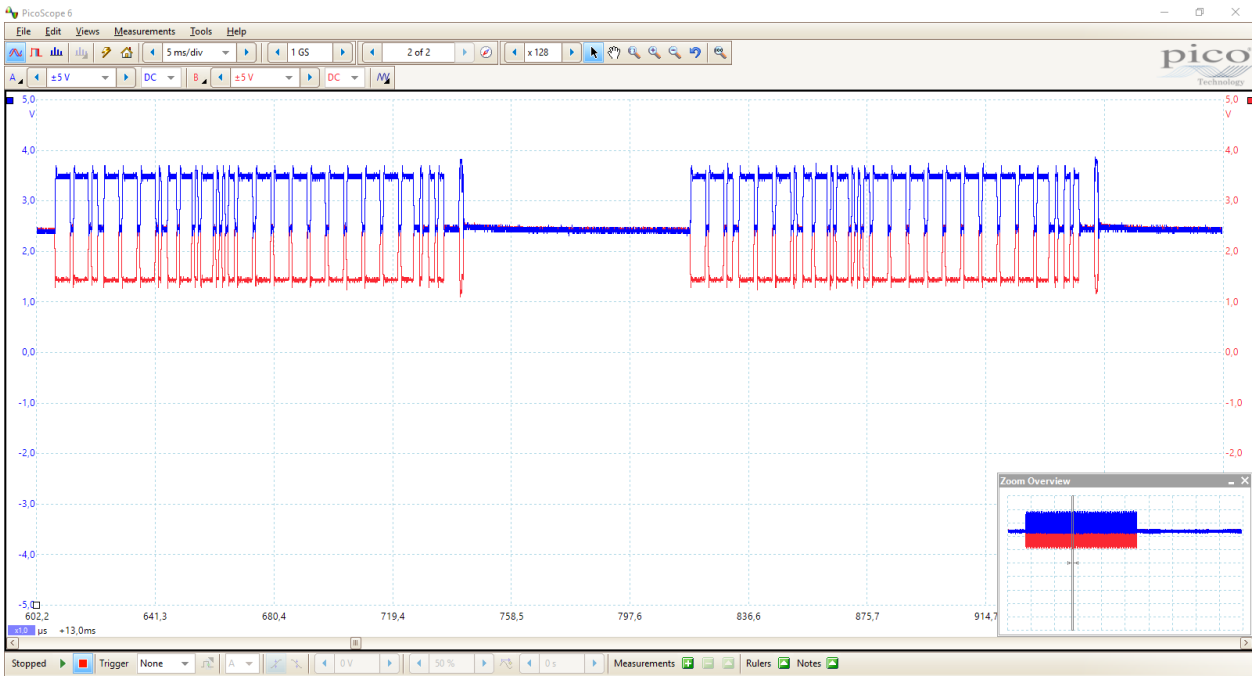
CHALMERS



Analog mätning av CAN-signalkvalité över fordonets livscykel

Analog Measurement of CAN Signal Quality over the vehicles lifespan

Examensarbete inom Mekatronik

Fredrik Andréen

Instutitionen för Signaler och System
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2018

Förord

Detta examensarbete har utförts på Chalmers Tekniska Högskola för programmet mekatronik. Examensarbetets storlek ligger på 15 hp och har genomförts tillsammans med företaget Diadrom i Göteborg.

Jag skulle villja tacka Henrik Fargrell, VD för Diadrom som varit handledare för mig på företaget. Jag vill även tacka Hana Trefna Dobna som varit examinator under examensarbetet samt de kollegor på Diadrom som har hjälpt mig med olika lösningar.

Sammanfattning

Controller area network (CAN) används idag flitigt i fordonsindustrin, navigationssystem och medicinsk utrustning. Det är dess busstopologi som gör att de är så vanligt inom fordonsindustrin, och de är robusta och lätta att diagnotisera digitalt på grund av dess busstopologi. Däremot är det inte lika lätt att diagnotisera CAN-bussen analogt då det behöver ske när fordonet är avstängt. Fordonsindustrin använder sig ännu inte av kontinuerlig analog diagnotisering, vilket är målet med detta arbete. För att diagnotisera CAN analogt krävs att man har tillgång till egenskaperna i de fysiska lagren. Resultatet av arbetet är ett system med tre algoritmer som kan mäta resistans, kapacitans och induktans för att redan innan haveriet inträffar, varna för att systemet håller på att haverera.

Abstract

CAN is very common in today's vehicle industry, navigation systems and medical equipment. It is its bus topology that makes it so common in the vehicle industry, and they are robust and easy to diagnose digitally because of their bus topology. On the other hand, it's not as easy to diagnose the CAN bus analogously because it has to be done when the vehicle is turned off. The vehicle industry does not yet use continuous analog diagnosing, which is the goal of this project. To diagnose CAN analogously you are required to have access to the properties of the physical layers. The result of the project is a system with three algorithms which can measure resistance, capacitance and inductance to be able to already before the failure occurs, warn that the system is about to break.

Innehåll

1

Inledning

1.1 Bakgrund

Självkörande bilar av olika former blir allt vanligare på dagens gator. Denna utveckling kommer kräva en noggrannare diagnostisering av CAN-bussen. Företaget Diadrom i Göteborg är ett konsultbolag inom fordonsindustrin som specialiserat sig på diagnostik. Att kablaget i CAN åldras efter år av användning vet man, men förändras de elektriska egenskaperna tillräckligt mycket för att det ska kunna orsaka fel är frågan arbetet ska besvara. Företaget vill därför utforska möjligheterna med att förutspå error frames samt haveri med olika elektriska egenskaper såsom resistans, kapacitans och induktans. Genom att sätta upp ett artificiellt buss system som ska efterlikna CAN så mycket som möjligt men samtidigt ha en hög hastighet så att fel lättare uppstår.

1.2 Syfte

Projektets syfte är att bygga en grund för att kunna förutspå fel i självkörande fordon i CAN-bussen. Systemet ska kunna förutspå fel baserade på påhittade primitiva elektriska fel.

Först och främst ska en labbmiljö för kommunikation av CAN signaler skapas. Företaget använder sig av något som kallas Diagnostic Communication Equipment (DICE). DICE verktyget har möjlighet att skicka och ta emot signaler med hjälp av ett Python program. Programmet använder ett Application Programming Interface (API) för att skicka och ta emot signaler kontinuerligt. Detta tillsammans med ocilloskop och hårdvara som mäter error frames utgör grunden för replikering av olika elektriska felfall.

1.3 Avgränsningar

Följande ramar har funnits för att hålla projektet inom den angivna tiden.

- Endast framtagning av ett system i labbmiljö ska konstrueras. Varken implementering eller testning i bil kommer utföras.
- Algoritmerna som skapas kommer endast testas för DICE-verktyget via API.

1.4 Precisering av Frågeställning

- Hur kan analog mätning användas för att felsöka CAN-bussar?
- Går det att förutspå fel baserat på elektriska egenskaper i CAN-bussar?
- Kräver alla kablar och mätningar att bilen är stillastående vid analog mätning?

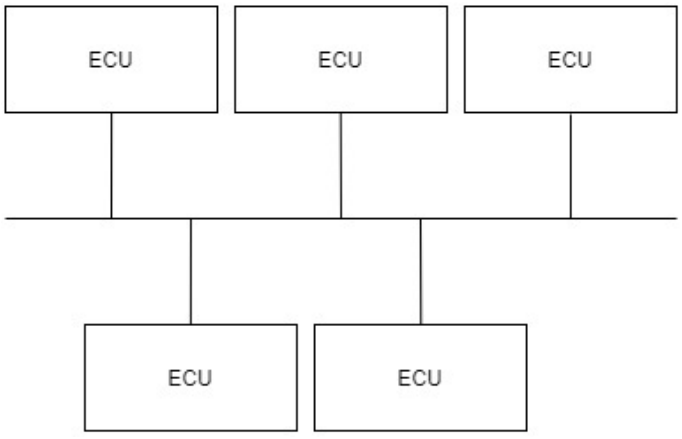
2
Teoretisk bakgrund

I detta avsnitt kommer hårdvara, mjukvara och elektriska komponenter som använts under projektets gång redovisas för att ge läsaren bakgrund till rapportens resultat. För att förstå hur idén uppstod samt hur varje enskild del av projektet fungerar kommer en teoretisk bakgrund ges.

2.1 Controller Area Network

CAN buss används inom fordonsindustrin, medicinsk utrustning och industriell automation. I bilar används CAN-buss som kommunikationsprotokoll för alla Electrical Control Unit (ECU) i bilen. Bilar har idag runt 70 stycken ECU:er. [1]. Antalet ECU enheter i bilen har ökat för varje år då bilarna fått fler och fler elektriska uppgifter för varje år som gått.

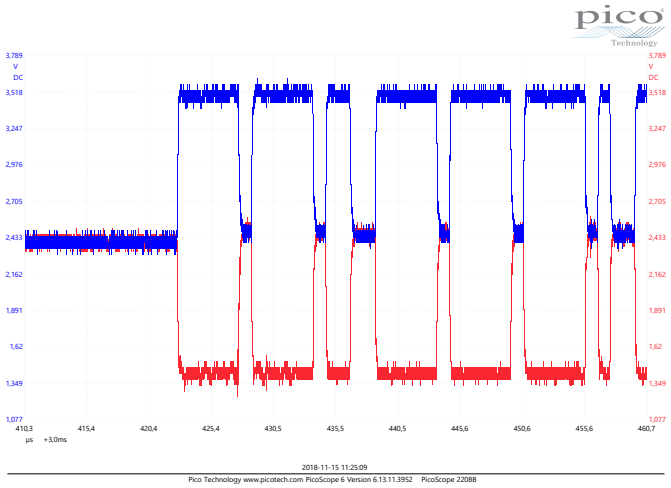
En av CAN:s fördelar är dess buss-topologi. Detta innebär att nätverket är uppyggt på ett Local Area Network där alla ECU:er är sammankopplade med endast en sladd. Detta kan ses nedan i figur 2.1.



Figur 2.1: *Buss topologi.*

En annan fördel med att använda CAN är dess robusthet som den får från dess användning av två förskjutna signaler. CAN använder sig av dominanta och recessiva bitar där den dominanta utgör logiskt 0 och recessiva logiskt 1. När signalen ej är verksam ligger signalen recessivt på logiskt 1. Om en ECU skickar en dominant bit och recessiv bit samtidigt kommer den dominanta biten prioriteras över den recessiva biten. Detta innebär att de meddelanden som har prioritet kommer fram direkt. Samtidigt som den dominanta biten skickas sätts, startar en bit-timer i den ECU

som höll på att skicka sitt meddelande. När klockan har väntat ett visst intervall prövar den återigen att skicka meddelandet. Detta medför att meddelanden som försöker skickas men misslyckas, på grund av högre prioriterade meddelanden alltid kommer fram.[2] Ett exempel på hur en CAN signal kan se ut kan ses nedan i figur 2.2 där den övre, blå signalen är CAN high och den undre, röda är CAN low.



Figur 2.2: *Karakteristik av en CAN signal utan störningar.*

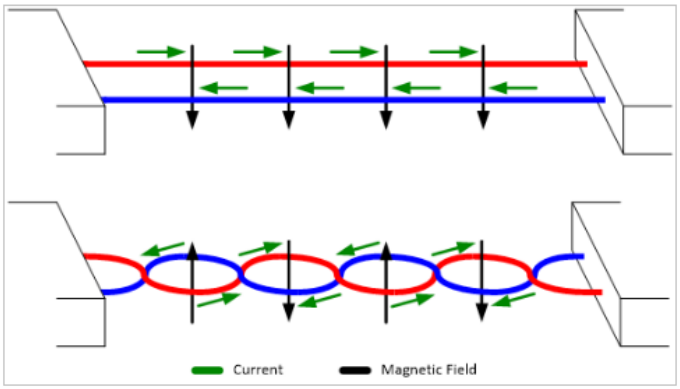
2.1.1 Error frames

En error frame påbörjar stängningen av felaktig data eller remote frames. Detta sker genom att CAN-bussen skickar mer än 5 bitar av samma polaritet i rad. Behöver CAN-bussen skicka ett meddelande som består av 5 bitar eller fler av samma polaritet används bit stuffing.[3]

För att föhindra att CAN-bussen blockeras av error frames skiljer man på error-active och error-passive. När ECU:n blivit defekt är den error-passive och har ECU:n inte reparerats till dess att transmit error counter sätts till 255 kommer ECU:n avlägsna sig själv från CAN-bussen.[4]

2.1.2 Partvinnad kabel

För att motverka störningar använder man sig av två tvinnade ledare som kallas partvinnad kabel. Mellan två parallella kablar förstärks störningarna på grund av att magnetfältet går uppåt mellan båda kablarna. I partvinnade kablar släcks magnetfältet från kablarna ut då magnetfältet från kablarna släcker ut varandra. Samma sak inträffar då kabeln ges en störning. Störningen påverkar ena sladden negativt och den andra positivt vilket gör att den totala störningen blir noll. Detta kan ses i figur 2.3 nedan.

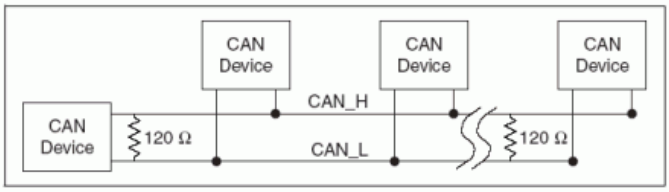


Figur 2.3: *Magnetfält från parallell- samt partvinnad kabel[5].*

2.1.3 Terminerande resistorer

Terminerande resistorer används för att förhindra störningar i form av reflektioner. Reflektioner uppstår på grund av att signalen genomgår ett mediumbyte i form av impedans. Med andra ord kommer signalen från en kabel in till en annan, vid bytet av kabel kommer en reflektion skickas tillbaka på den skickande kabeln. Storleken på den terminerande resistorn utgörs av storleken på ledarens impedans. Är termineringsimpedans större än kabelns impedans kommer kabeln få positiva reflektioner. Är däremot termineringsimpedansen mindre än kabelns impedans kommer kabeln få negativa reflektioner.[6]

Inom hög hastighets CAN används termineringsresistorer längst ut på bussen där de sista ECU:erna sitter. Detta kan ses i figur 2.4 nedan med enheter längst ut samt ett antal enheter mellan de terminerande resistorerna.



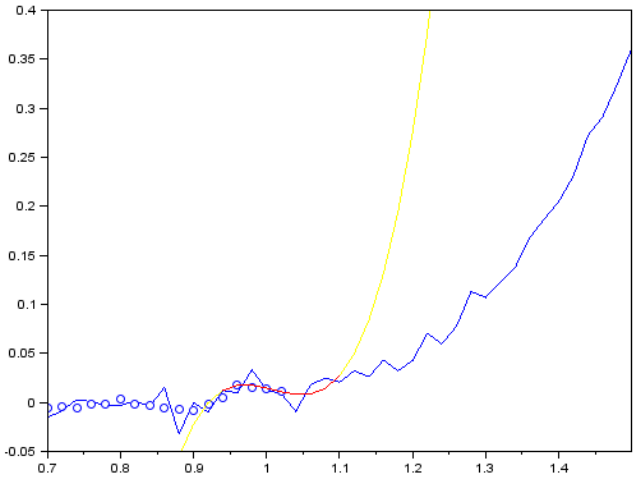
Figur 2.4: *Terminering av höghastighets CAN[7].*

2.2 Signalbehandling

Det är vanligt att jämna ut brusiga signaler med hjälp av lågpassfilter. En typ av lågpassfilter är Savitzky-Golay-filtret som också är ett digitalt filter. Genom att dela upp en funktion i delmängder och sedan beräkna medelvärdet på dessa delmängder får man ut en ny funktion med utjämnade extrempunkter. Alltså kommer avvikelser att förekomma då derivatan är skild från noll. Till exempel blir det glidande medelvärdet för maximipunkter alltid lägre än funktionsvärdet, alltså får man både minskad höjd och ökad bredd i signalen [8].

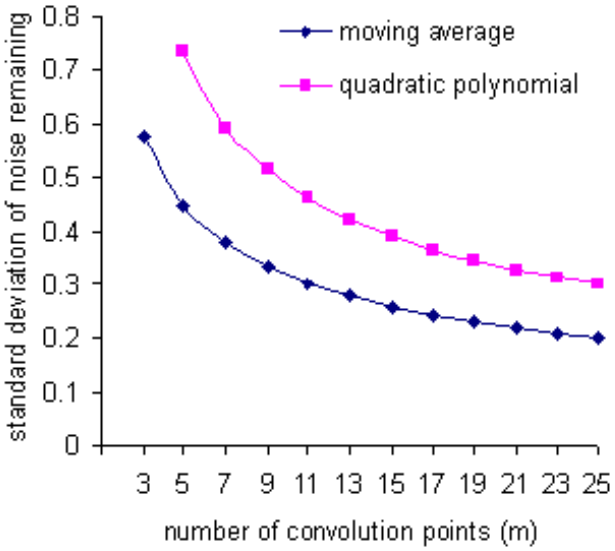
Savitzky-Golay filtret använder sig istället för ett polynom av högre ordning. Precis som glidande medelvärde delar Savitzky-Golay filtret upp funktionen i delmängder,

för att sedan räkna ut ett passande polynom till delmängden av funktionen. I figur 2.5 nedan visas hur funktionen delas upp i delmängder och får ett passande polynom för delmängden. Den blå linjen är den ursprungliga funktionen, de blå punkterna är den utjämnade funktionen. Den röda kurvan representerar polynomets delmängd och den gula kurvan representerar själva polynomet som beräknats från delmängden.



Figur 2.5: *Filtrering av en funktion med hjälp av Savitzky-Golay filter [8].*

För att analysera signaler räcker det oftast inte med ett glidande medelvärde då man får mycket oönskade avvikelser från sin ursprungliga funktion. I figur 2.6 visas korrelationen mellan standardavvikelse och antal faltnings punkter.



Figur 2.6: *Standardavvikelse av det brus som återstår med avseende på ett glidande medelvärde och en andragradsfunktion[9].*

Simpsons regel fungerar likt Savitzky-Golay filtret genom att använda sig av andragradspolynom för att approximera en funktion istället för att använda sig av linjesegment. Simpsons regel kan approximera intergraler exakt upp till polynom av graden kubik. Som exempel aproximerar Simpsons regel ekvationen 2.1 till 1,00228 medan en approximering genom linjesegment, den så kallade trapezoidals regel ger svaret 0,785398. Det rätta svaret för arean under kurvan är 1[10]. Vilket betyder att Simpsons regel approximerar med mycket större noggrannhet än trapezoidals regel.

$$\int_0^{\pi/2} sin(x)dx \tag{2.1}$$

3
Metod

I följande kapitel beskrivs projektets arbetsgång. Arbetsgången är uppdelad i fyra delar. Utformning av laborationsmiljö, mätning av elektriska egenskaper med hjälp av Picoscope, framtagning av algoritmer och till sist, testning av algoritmer.

Först påbörjades undersökningar med den bakomliggande anledningen arbetet med den analoga mätningen. Ett möte med experter på Volvo Personvagnar berättade att man ännu inte mäter analogt för att felsöka bilar och att resistans kan ändras över tiden. Däremot mäter man under produktutvecklingsfasen med oerhört dyra ocilloskop. Ett annat känt problem var att kontakter samlar vatten och salt som sedan oxiderar och ger kontakterna en beläggning.

I början av arbetet lades mycket tid på att hitta rätt hårdvara och verktyg för projektet. Efter att ha diskuterat inköpsförslagen med handledaren på Diadrom köptes dessa in och arbetet med de elektriska lådorna kunde påbörjas.

När elektroniklådorna var ihopkopplade och testade med en multimeter prövades dessa med DICE verktyget för att säkerhetsställa att alla kopplingar stämmer. Efter detta kopplades Picoscopet in och de elektriska komponenter som skulle kopplas in, kopplades in på brödkortet för att smidigt kunna byta komponenter. Efter detta mättes olika egenskaper upp i Picoscope programmet för att få ett riktvärde. Mätningarna i Picoscope programmet har gjorts med hjälp av en zoom funktion samt en inbyggd linjal i programmet. Mätningarna är därför inte perfekta men de är tillräckliga för ändamålet.

När all nödvändig data för algoritmerna hade skapats påbörjade arbetet med framtagning av algoritmer. De olika algoritmerna fungerar på olika sätt och tillvägagångssätt av algoritmerna har kollegor från Diadrom hjälpt till med. Under skapandet har värdena för de mätningar från Picoscopet tagits till åtagande för att se att algoritmen fungerar som den ska.

Testning av algoritmerna har gjorts genom att göra diagram och observera hur signalerna och algoritmerna relaterar till varandra. Under testningen har Picoscope mätningarna också funnits i åtanke.

Slutligen har algoritmerna för var och en av värdena för de olika elektriska egenskaperna körts. Tabeller och korrelationskoefficienter för algoritmerna har räknats ut för att visa resultatet av algoritmernas noggrannhet.

4

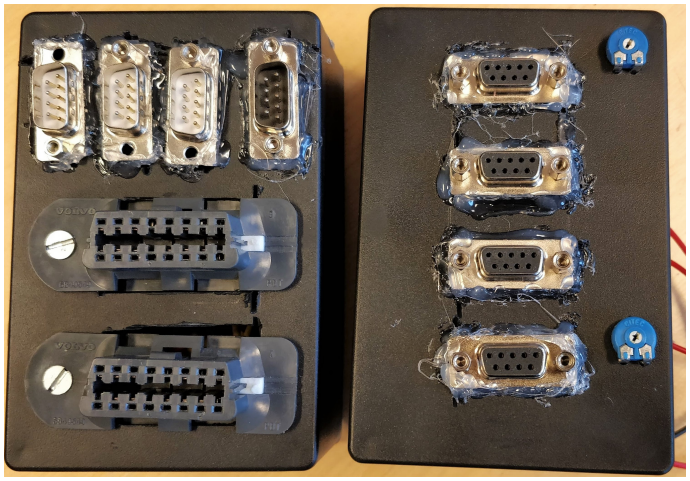
Datainsamling

För att få den önskade datan krävs ett sätt att skicka och mäta CAN hög och CAN låg. Detta har åstadkommits genom att konstruera en artificiell CAN buss som med hjälp av en DICE kan skicka och ta emot data. För att kunna mäta datan som kommuniceras har två elektroniklådor konstruerats.

- En DICE som skickar och tar emot data via ett Pythonprogram.
- Ett Picoscope som fungerar som ett ocilloskop där det går att mäta olika karakteristiker av den elektriska signalen över tid.
- Elektroniklåda 1 med 2 st. ingångar för OBD-II kontakter för DICE verktyget och Kvaser Leaf Light HS V2 samt 4 st. utgångar med DE-9 kontakter.
- Elektroniklåda 2 med 4 st. ingångar med DE-9 kontakter som är ihopkopplade med 2 st. termineringsmotstånd längst ut på bussen. Detta kan ses i figur 2.4.

Elscheman för elektroniklådorna kan ses i bilagan. Komponenter som använts till elektroniklådorna:

- 2 st. OBD-II socket
- 4 st. Dsub socket
- 4 st. Dsub plug
- Kopplingsdäck, brödkort
- Toppklämmor
- Dsub kablage 0,5m



Figur 4.1: *Elektroniklåda 1 till vänster och elektroniklåda 2 till höger.*

4.1 Kommunikation med CAN i praktiken

För att skicka och ta emot signaler på CAN-bussen krävs ett program som kan kommunicera via DICE API. Programmet som använts för detta kommer från Diadrom. Detta program kommunicerar med en baud rate på 1 Mbit/s. Baud är en måttenhet för hur många gånger per sekund signalen ändras. Varje alfabetiskt tecken består av 8-11 databitar. Varje databit består av en puls. Detta innebär att ett program som kommunicerar med 1 MBd kan ta emot 100k tecken per sekund [11].

Då Kommunikationsprogrammet skapades användes Python 2,7. Detta medför att den nuvarande versionen, 3.7 inte fungerar. Istället måste programmet köras med en tolk till Python 2,7.

4.2 Hårdvara

För att analysera signalerna behövs hårdvara som skickar signalerna samt hårdvara för att analysera signalerna.

4.2.1 Picoscope 2208B

Picoscope 2208B är ett oscilloskop som kopplas till datorn. Med hjälp av mjukvaran Picoscope 6 kan man se signalerna i datorn för att sedan analysera dem. Oscilloskopet klarar av att mäta upp till 100 MHz i bandbredd och upp till 128 Mega sample(MS) i buffer minne. Oscilloskopet har funktioner som spektrumanalysator, funktionsgenererare, godtycklig vågformsgenerator samt protokollavkodare.

4.2.2 SAE J2534

J2535 översätter kommunikationsprotokollen som används inom ECU:erna för att kunna läsas av med en dator. Ett J2534 API Dynamic-link library (DLL) kommer från hårdvaruskaparna som gör att deras hårdvara går att använda med en dator. SAE J2534 stöder Universal Asynchronous Receiver/Transmitter (UART), CAN, ISO-TP och liknande RS-232.[12]

4.2.3 Diagnostic Communication Equipment

DICE-verktyget är en så kallad pass through device. DICE-verktygets främsta uppgift är felsökning av bilar och uppdatering av mjukvara i bilen. DICE-verktyget kopplas in via fordonets diagnostikingång, OBDII, och strömsätts via bilens batteri eller vägguttag. Vanligtvis används hårdvaran med mjukvaran VIDA som gör att man kan läsa och återställa felkoder, övervaka bilens parametrar och utföra underhållsuppgifter på bilen. Ett annat sätt att använda hårdvaran på är att använda verktygets API för att göra egna program som tillåter kommunikation via CAN bussen. Vilket kommunikationen i projektet har använt sig av.

4.2.4 Kvaser Leaf Light HS V2

Kvaser Leaf Light HS V2 är en så kallad pass through device. Med denna hårdvara kan man koppla upp sig mot fordonets CAN-buss, precis som med DICE:en. Den har en usb och en 9-pin D-SUB kontakt.I projektet har denna använts med en OBDII till 9-pin D-SUB för att ansluta sig till CAN-bussen. Den har denna använts för att analysera CAN-bussens error frames.

4.3 Python

Fördelen med Python är dess stora mängd öppen källkod inom dataanalysverktyg såsom Numpy, Scipy, Matplotlib och Panda.

4.4 Elektriska egenskaper inom CAN

Kolbaserade resistorers värde sjunker med värmeutveckling och normal användning.[13] Likadant när resistorer utsätts för negativa temperaturförändringar, då ändras resistansen med ekvation 4.1. I Sverige är det vanligt med temperaturförändringar på 40 °C. CAN använder sig av koppar i sladdarna vilket ger temperaturkoefficienten 0,0039 vid 20 °C[14]. Termineringsresistorerna är 120 Ω var. Ändras temperaturen till -20 °C blir den nya resistansen 101,3 Ω.

$$R(T) = R_0[1 + \alpha(T - T_0)] \tag{4.1}$$

Kapacitans mellan CAN hög och CAN låg i en fyrkantsvåg påverkar falltiden negativt. Detta sker på grund av att kapacitansen lagrar en elektrisk laddning som den laddar ur då CAN-bussen får ett signalbyte mellan recessiv till dominant.

Induktans är definierat som förhållandet mellan spänningen och strömmens förändring i tid[15]. En induktor sparar energi i dess magnetiska fält och återger energin när det behövs. Induktans skapas genom att ta tillvara på det magnetfält kabeln har genom att vira kabeln runt en cylindrisk kärna[16].

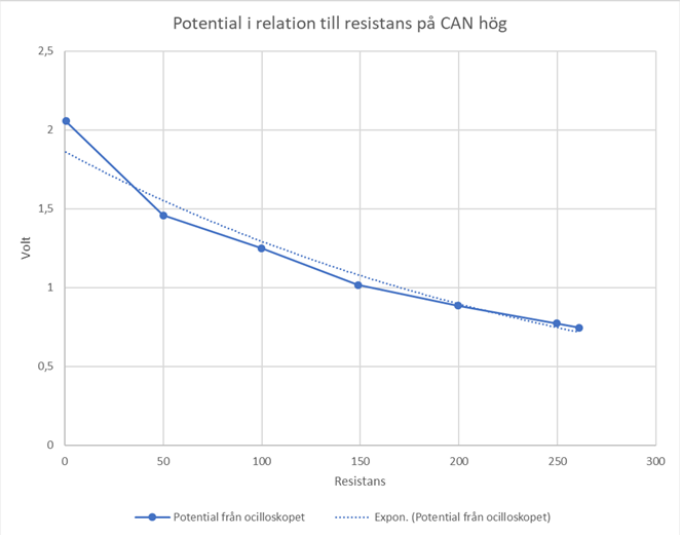
När induktorn matas med DC skapas en elektromagnet. Men eftersom CAN använder sig av en fyrkantsvåg som gör kretsen till en sorts AC kommer en spänningskillnad uppstå i bussen, då uppstår också ett elektromagnetiskt fält i induktorn som skapar en spänningstransient på CAN hög och CAN låg. Spänningskillnaderna sker vid signalbyte mellan recessiv och dominant och tvärtom.

4.5 Mätning av resistans

Intervallet för mätningar av resistans har gjorts i åtanke till det största värdet på resistans då signalerna fortfarande kan tolkas. Detta värde var 261,1 Ω. Intervall för mätningarna av 50 Ω valdes så att en trend ska kunna ses.

För varje ny mätning har potentiometern kopplats ur för att mätningarna inte ska påverkas av de andra komponenterna i kretsen. Den strömsatta kretsen gör också att potentiometern inte längre kan mäta resistansen på ett korrekt sätt. Mätningarna har gjorts med CAN hög på potentiometerns pinne 1 och 2. Mätningarnas värden kan ses i tabellen nedan. Sambandet mellan elektrisk potentialskillnad och resistans kan ses nedan i figur 4.2.

Resistans	Elektrisk potentialskillnad
0,5	2,057
49,9	1,459
99,8	1,251
148,8	1,017
199,7	0,8857
249,8	0,7725
261,1	0,7457



Figur 4.2: *Elektrisk potentialskillnad*

4.6 Mätning av kapacitans

Mätningarna för kapacitans har gjorts med hjälp av kondensatorer med olika storlekar, 4 st med 4,7 nF och 12 st. med 22 nF. Kondensatorerna har kopplats mellan CAN hög och CAN låg så att en elektrisk laddning uppstått i kondensatorerna som sedan laddas ur när signalen går från dominant till recessiv. Ekvationen som använts för att räkna ut kapacitansen är ekvation 4.2.

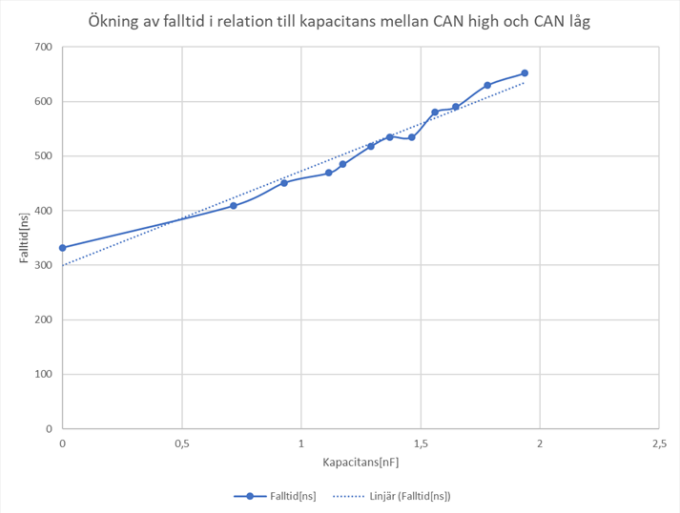
$$\frac{1}{C_{Tot}} = \frac{1}{C_1} + \frac{1}{C_2} + ... \tag{4.2}$$

Det största fungerande värdet var 1,78 nF. Detta räknades ut genom ekvation 4.3.

$$\frac{1}{2 * \frac{1}{4,7} + 3 * \frac{1}{22}} = 1,7797nF \tag{4.3}$$

En tabell för falltiden vid signalbyte från dominant till recessivt beroende på kapacitans kan ses i tabellen nedan. En graf av detta kan ses i figur 4.3.

Kapacitans[nF]	Falltid[ns]
0	332
0,716	409
0,927	451
1,115	469
1,175	485
1,291	518
1,371	535
1,463	555
1,56	580
1,646	590
1,780	630



Figur 4.3: *Resultat av mätningar med olika värden för kapacitans.*

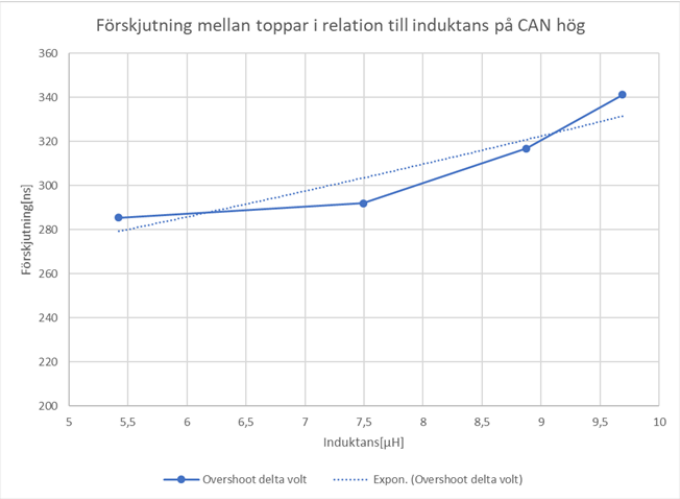
4.7 Mätning av induktans

Induktans mätningen gjordes på CAN hög med hjälp av 6 olika induktanser som satts i parallellkoppling. Induktansen räknades ut med hjälp av ekvation 4.4.

$$\frac{1}{L_{Tot}} = \frac{1}{L_1} + \frac{1}{L_2} + ... \tag{4.4}$$

En tabell för induktans och den förskjutning mellan topp och bott på CAN hög och CAN låg som uppstått kan ses i nedanstående tabell. Sambandet mellan förskjutning av spänningstransienterna kan ses nedan i figur 4.3.

Induktans[µH]	Förskjutning av toppar[ns]
5,42	285
7,492	292
8,875	332
9,69	341



Figur 4.4: *Resultat av mätningar med olika värden för induktans.*

5
Algoritmer

Algoritmerna har tagits fram i avseende på olika elektriska egenskaper. Varje enskild elektrisk egenskap har sin egna algoritm på grund av karakteristiken av vågformen. Alla dessa algoritmer analyserar filer genererade av Picoscopet. Picoscopet har mätt med 500 MS/s för att garantera att man får en bra upplösning på de inspelade vågformerna.

Alla algoritmer har använt sig av Scipy-modulen som är vanlig vid vetenskapliga- och tekniska beräkningar. Numpy från Scipy har använts vid matrisberäkningar. Modulen Matlatplotlib har använts för att kunna felsöka algoritmerna genom att skapa en graf av signalerna.

För att analysera signalerna gjordes filerna om från filformatet psdata till filformatet csv för att kunna analysera vågformerna med python. De algoritmer som skapats använder sig alla av koden nedan. Filen öppnas och sätts i läsläge. Csv-filerna innehåller först beskrivning på första raden och sedan 2 blanka rader innan värdena från vågformerna börjar. Därför hoppar man över de tre första raderna med hjälp av next(csvfile).

Tidens värde, CAN hög värdet och CAN låg värdet är uppdelade med ett kommatecken. Dessa värden läses sedan av med csv.reader och lägger dem i plots. For-loopen lägger första värdet av raden i ett fält x som i detta fall är tiden. Den lägger också andra och tredje värdet CAN hög och CAN låg i varsitt fält.

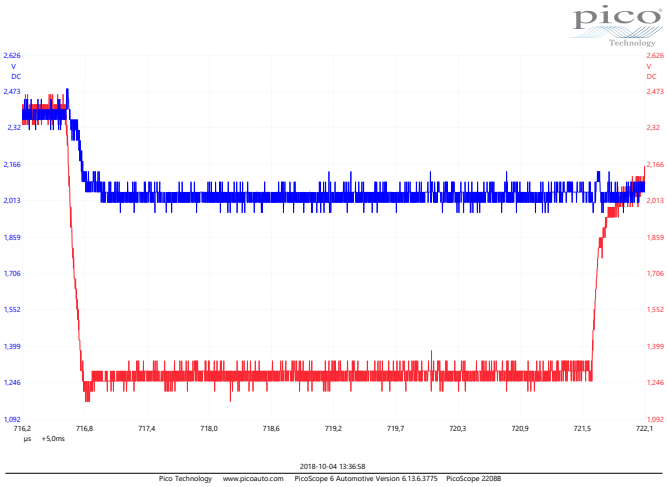
Till slut beräknas medelvärdet av fältet som räknats ut från algoritmen med hjälp av np.mean".

```
with open('Resistans 261,1.csv', 'r') as csvfile:
    next(csvfile)
    next(csvfile)
    next(csvfile)
    plots = csv.reader(csvfile, delimiter=',')
    for col in plots:
        x.append(float(col[0]))
        h.append(float(col[1]))
        l.append(float(col[2]))
```

```
print('The value is:', np.mean(value))
```

5.1 Resistans

Efter att ha jämfört figur 2.2 med figur 5.1 kan man observera att potentialen mellan CAN hög och CAN låg har sjunkit från 2 volt till ungefär 0,8 volt.



Figur 5.1: *Resistansen är 261,1 Ω. Detta motsvarar det största värdet som DICE kan tolka.*

Runt värdet 261,1 Ω börjar error frames att genereras intermittent. Höjs resistansen mer än 261,1 Ω kommer DICE-verktyget inte längre kunna tolka signalerna. När DICE verktyget inte kunnat tolka signalerna under ungefär 10 sekunder kommer lampan firmware status att börja lysa rött. Lyser denna lampa rött innebär det ett hårdvarufel, vilket har framkallats av värdet på resistansen. För att börja skicka signaler igen behöver både DICE-verktyget och pythonprogrammet startas om.

5.1.1 Algoritm för potentialmätning

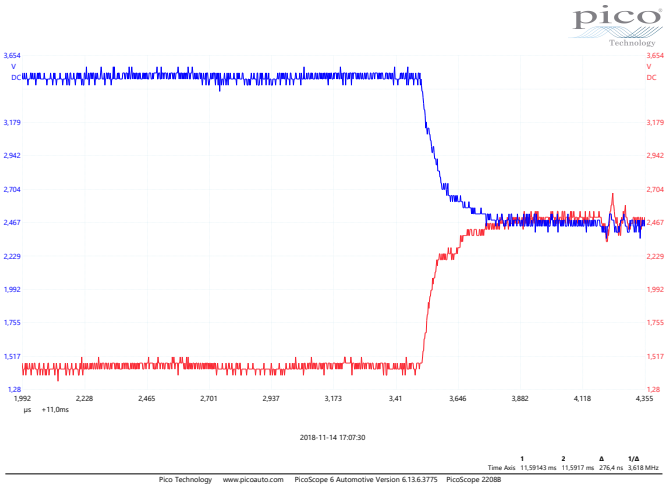
Denna algoritm fungerar på det sätt att för varje gång csv-loopen körs igenom jämför den värdet delta med 0,5 för att se om potentialen mellan CAN hög och CAN låg är tillräckligt stor för att det borde vara en dominant bit som sänds. Det som händer härnäst i algoritmen är att tiden då potentialen är tillräckligt stor sparas undan. När den sparats undan jämför algoritmen det undansparade värdet tid och nästa värde för tiden newtid och lägger det i variabeln intervall.

Det kortaste en dominant bit kan vara är 0.001 ms. Eftersom csv-filens tid är satt till ms letar man efter ett intervall 0.00035 ms till 0.00045 ms som är ett intervall på toppen av den dominanta biten. Detta görs för att säkerhetsställa att potentialen hålls på ett stabilt värde. Sedan sparas alla värden som möter dessa krav i ett fält. För varje dominant bit sparas den dominanta bitens potential mellan 0.00035 ms och 0.00045 ms. Är dock intervallet större än ms sampel så nollställs variablerna och for-loopen börjar om.

```
    delta = float(row[1]) - float(row[2])
          if delta > 0.5:
              if measure == 0:
                  measure = 1
                  tid = float(row[0])
              else:
                  newtid = float(row[0])
                  intervall = newtid - tid
                  if 0.000350 <= intervall <= 0.000450:
                      value.append(float(delta))
                  if intervall > 0.0004500:
                          measure = 0
                          intervall = 0
                          delta = 0
```

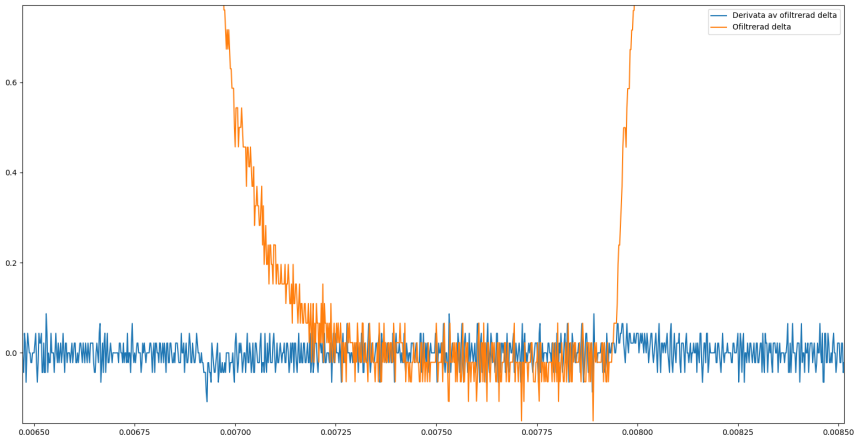
5.2 Kapacitans

Jämför man figur 2.2 som är utan kapacitans med figur 5.2 med kapacitans kan man observera att falltiden av signalen har ökat. Kondensatorn håller en elektrisk laddning som orsakar denna ökning av falltid vilket i sin tur gör att DICE-verktyget får svårt att tolka signalerna. Målet med algoritmen är att få en så hög korrelation som möjligt med mätningarna som gjordes i föregående kapitlet.



Figur 5.2: *Kapacitansen är 1,78 µF. Detta motsvarar det största värdet som DICE kan tolka.*

Problemet med denna algoritmen är att funktionen är så brusig att derivatan av signalen inte blir distinkt nog för att mäta falltiden av potentialen på signalen. Nedan visas en övergång mellan dominant bit till recessiv bit och sedan tillbaka till dominant bit i figur 5.3.

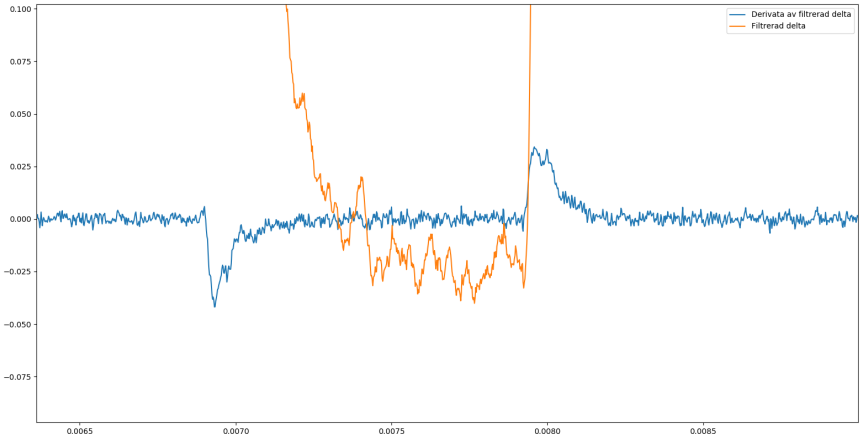


Figur 5.3: *Ofilterad derivata*

För att lösa detta problemet har Savitzky-Golay-filtret applicerats på både CAN hög och CAN låg innan potentialen räknats ut. Första variabeln innehåller den data som ska jämnas ut. Andra variabeln innehåller längden på utjämningsfönstret, alltså antal sampel som ska användas. Tredje variabeln innehåller graden på polynomet som används för att jämna ut funktionen. Värdena har testats fram genom att undersöka grafen av den utjämnade funktionen.

```
filteredhigh = savgol_filter(h, 31, 3)
filteredlow = savgol_filter(l, 31, 3)
filtereddelta = filteredhigh - filteredlow
gradients = np.gradient(filtereddelta)
```

Resultatet av att applicera Savitzky-Golay filtret kan ses nedan i figur 5.4. Resultatet är en distinkt derivata för potentialen mellan CAN hög och CAN låg.



Figur 5.4: *Filterad derivata*

5.2.1 Algoritm för falltid

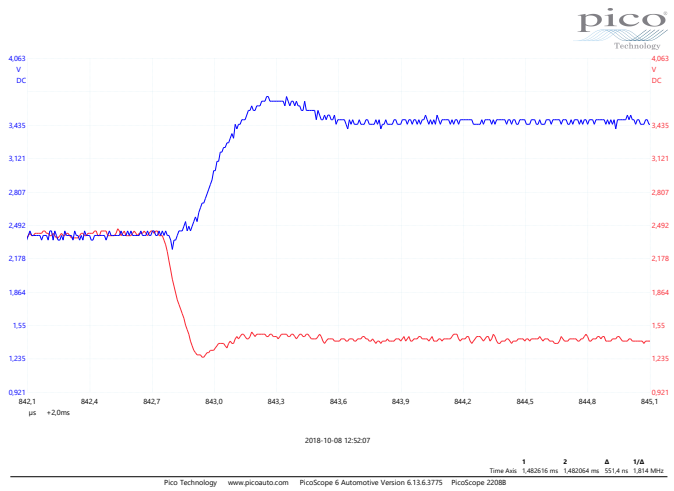
Derivatan av potentialen kan nu användas för att skapa en algoritm. Att endast använda ett värde för derivatan då derivatan är tillräckligt negativ för att kunna tolkas som falltid räcker inte i detta fall då störningarna påverkar för mycket. För att lösa detta använder algoritmen en areafunktion, Simpsons regel för att avgöra om arean av derivatan är tillräckligt stor för att den säkert ska kunna tolkas som en falltid.

En hel falltid mellan dominant och recessiv är beräknad till 0.00008317 ms men eftersom tiden sparas undan med hjälp av en if-sats och ett värde på derivatan gör det att en del av falltiden försvinner. För resultatet gör detta inget då det endast är korrelationen som är viktig. Intervallet för falltiden har därför satts till något mindre, 0,00005 ms och 0,0003 ms. Värdena för att algoritmen ska identifiera att det är en falltid har satts till -0,007 av derivatan. Botten av derivatan är -0,053. När algoritmen identifierat början och slut av intervallet beräknas arean för dessa punkter i avseende till derivatan. När CAN-busen inte har några aktiva störningar inkopplade är medelvärdet av arean 3,16e-06. Arean har satts till ett lägre värde för att få algoritmen inte ska missa några bottnar men ändå filtrera ut de oanvändbara störningar som finns i mätningarna. Efter detta sparas värdet för intervall i ett fält och algoritmen börjar leta efter nästa falltid.

```
if 0.00005 <= intervall <= 0.0003:
    if p > -0.007:
        array_intervall = x[startindex:stopindex]
        area = abs(simps(gradients[startindex:stopindex],
            ↪ array_intervall))
        if area > 1.500e-06:
            value.append(float(intervall))
            counter += 1
            intervall = 0
            measure = 0
        else:
            intervall = 0
            measure = 0
```

5.3 Induktans

Jämför man bilden med hög induktans i figur 5.5 med figur 2.2 ser man att överslängen för CAN hög och CAN låg inte förfaller med varandra i x-led. Denna skillnad har utgjort grunden för algoritmen.



Figur 5.5: *Induktansen är 9,69 µH. Detta motsvarar det största värdet som DICE kan tolka.*

5.3.1 Algoritm för skillnad i tid av spänningstransienter

För att lättare kunna utgöra var mitten på översläng̈en är för CAN hög och CAN låg har återigen en utjämningsfunktion använts.
Algoritmen förlitar sig på att man manuellt ställer in koefficienterna för trösklarna av både höjd(height=) och längd(distance=) mellan toppar. För att veta att rätt koefficienter används behöver man veta antalet toppar i den använda signalen. Längden mellan topparna har mätts upp i matplotlib med tiden i x-led. Då find_peaks använder sig av sampel i x-led beräknas antalet sampel med hjälp av ekvation 5.1

$$\frac{Längd\ mellan\ toppar}{Tidsintervall} = Antal\ sampel \tag{5.1}$$

Fälten subtraheras från varandra för att få ut längden mellan respektive översläng och undersläng.
Nästa del av algoritmen jämför subtraktionen mellan topparna för att sedan spara värdena som stämmer överens med antalet sampel som motsvarar skillnaden i tid av översläng kontra undersläng. Detta hade också kunnat lösas med hjälp av en if-sats men resultatet är detsamma. Tillvägagångssättet med for-satser är en del av en mer avancerad lösning som kommer kunna automatisera en del av algoritmen i framtiden.
Sista delen av algoritmen använder den filtrerade listan för subtraktion av fälten.

```
y_high = savgol_filter(h, 31, 3)
y_low = -1*savgol_filter(l, 31, 3)

peakshigh, _ = find_peaks(y_high, height=3.58, distance=150)
peakslow, _ = find_peaks(y_low, height=-1.355, distance=150)

length_between_peaks = peakshigh-peakslow

for a, b in zip(peakshigh, length_between_peaks):
    if 80 > b > -10:
        new_list_a.append(a)
```

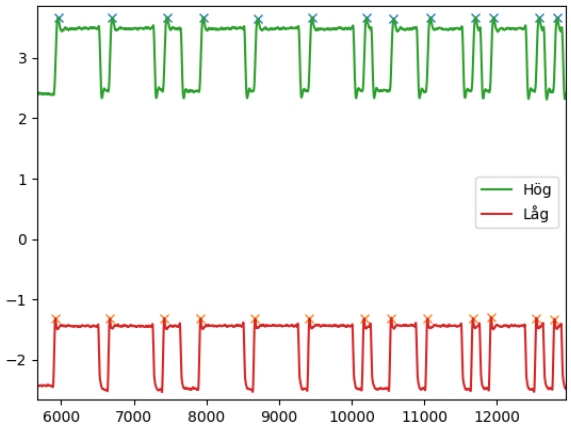
```
for a, b in zip(peakslow, length_between_peaks):
    if 80 > b > -10:
        new_list_b.append(a)

filtered = np.array(new_list_a) - np.array(new_list_b)
```

5.3.2 Testning av skillnad i tid av spänningstransienter

För att säkerhetsställa att find_peaks fungerar som den ska gjordes en graf med de filtrerade signalerna tillsammans med fältet med de uträknade topparna från findpeaks. Detta har gjorts för att kunna se vart algoritmen mäter fel eller helt och hållet missar en topp. Nedan är koden för grafen som gjorts. Den markerar topparna med ett x på det y-värde och x-värde som toppen har. I samma graf läggs de filtrerade signalerna av CAN hög och CAN låg. Detta kan ses i figur 5.6.

```
plt.plot(peakshigh, y_high[peakshigh], "x")
plt.plot(peakslow, y_low[peakslow], "x")
plt.plot(y_high, label='High')
plt.plot(y_low, label='Low')
```



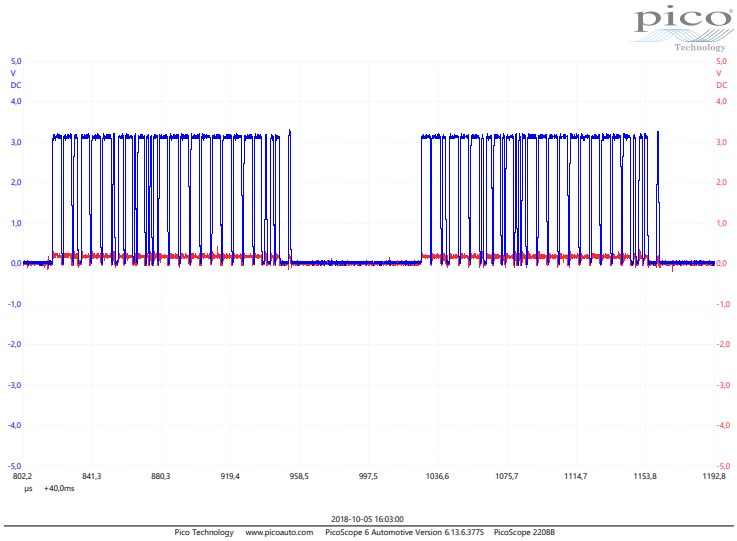
Figur 5.6: *Graf av CAN hög och CAN låg med översläng.*

6
Resultat

Resultatet är tre st. algoritmer som kan förutspå fel i CAN-bussen innan de uppstår. Algoritmerna behöver ett oscilloskop och en dator för att mäta och analysera vågformerna samt en inkoppling på CAN-bussen.

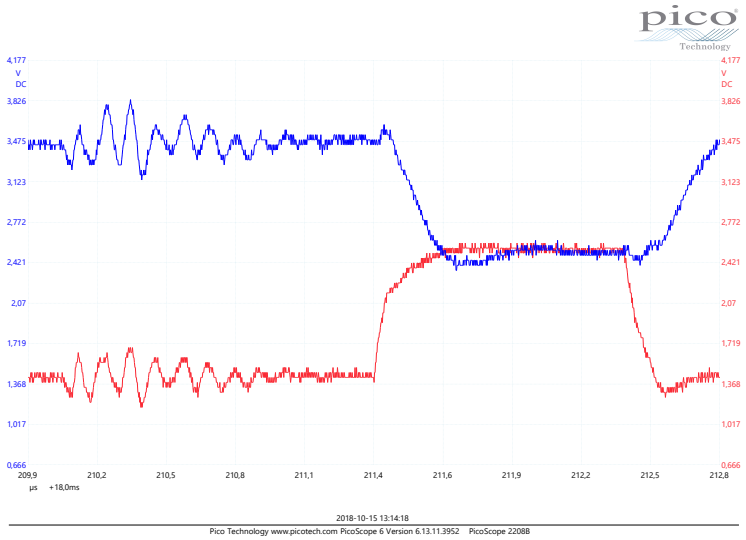
6.1 Mätningar utan felfall

Resultatet av kortslutning mellan jord och CAN hög var att CAN-bussen inte genererade några felramar. Vågformen av kortslutningen kan ses nedan i figur 6.1.



Figur 6.1: *Kortslutning mellan jord och CAN låg.*

Resultatet av att linda 3 m CAN hög runt en mobil som ringer genererade inga felramar. Vågformen av en störning kan ses i figur 6.2. Störningar av detta slag återkommer med jämna mellanrum.



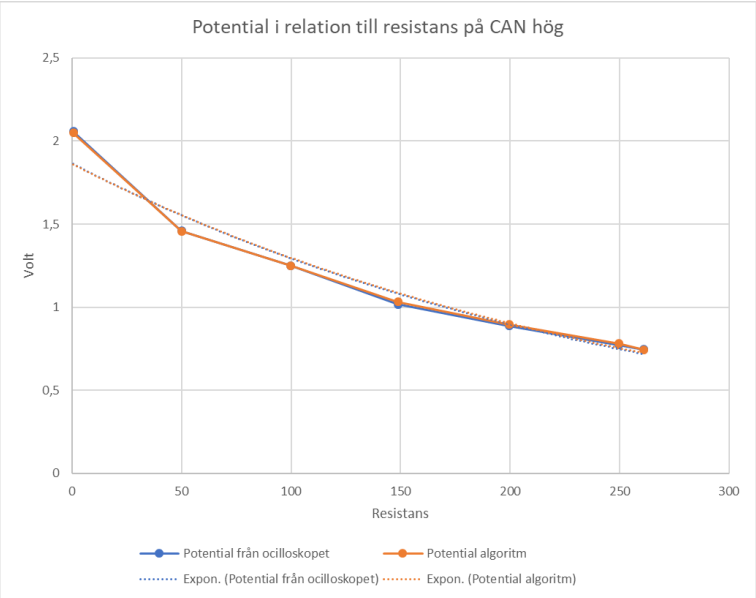
Figur 6.2: *Störning som skett medan telefonen ringde.*

Resultatet av att lägga en DE-9 och 1 m skavd kabel i saltvatten i en veckas tid gav inga märkbara resultat.

Resultatet av att ändra storlek på termineringsresistorerna är att falltiden ökar med ökande storlek på termineringsresistansen och tvärtom. När termineringsresistorerna var 41,4 Ω och 41,4 Ω fungerade fortfarande kommunikationen. Detta betyder att temperaturförändringar på 40 grader inte kommer vara något problem för termineringsresistorerna. Denna temperaturförändring gör att resistansen ändras från 120 Ω till 101,3 Ω och har därför ingen påverkan på CAN-bussen.

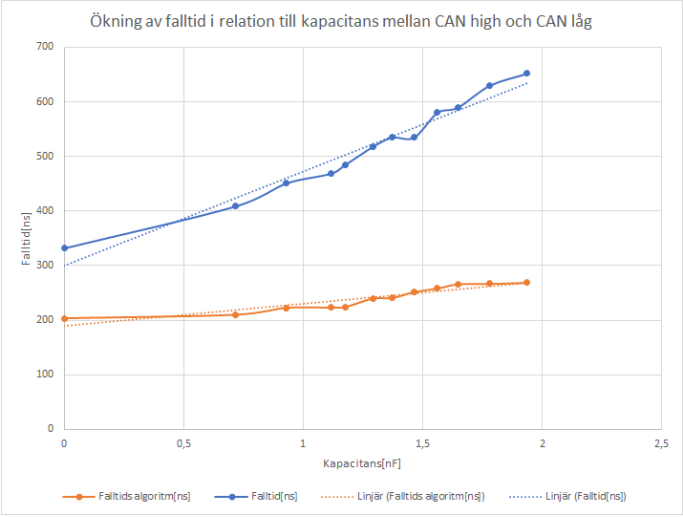
6.2 Algoritmer

Resistansen korrelerade med mätningarna som gjorts med ocilloskopet med en korrelation av 0,999915194.



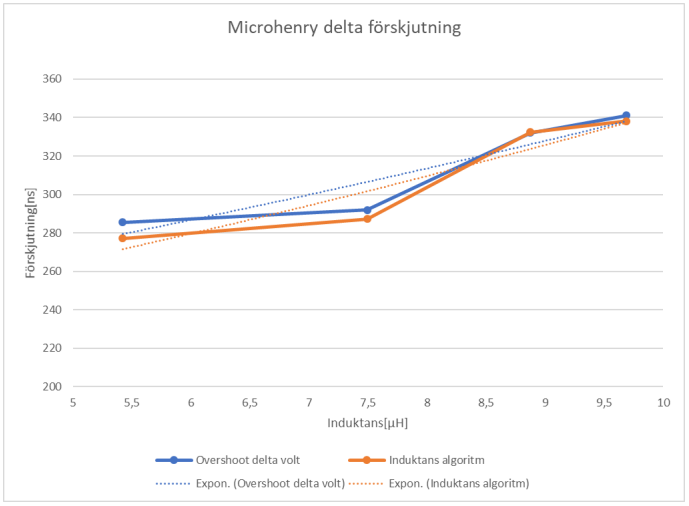
Figur 6.3: *Potential*

Kapacitansen korrelerade med mätningarna som gjorts med ocilloskopet med en korrelation av 0,974815394.



Figur 6.4: *Falltid*

Induktansen korrelerade med mätningarna som gjorts med ocilloskopet med en korrelationen av 0,997545992.



Figur 6.5: *Spole*

6.3 Utvärdering av algoritmer

Eftersom DICE skickar samma signaler varje gång går det snabbt att räkna ut antalet gånger signalen byter från recessiv till dominant läge. Detta gjordes manuellt och jämfördes sedan med antalet funna falltider och spänningstransienter.

6.3.1 Test av kapacitans algoritm

Med hjälp av Matplotlib har grafer av vågformerna gjorts för att räkna ut antalet falltider för csv-filen. Efter detta har counter använts för att se hur många falltider som algoritmen har hittat. Detta testades för 0 nF och 1,9 nF. Vid 0 nF hittades 246 av 246 falltider. Vid 1,9 nF hittades 219 av 224 falltider.

6.3.2 Test av induktans algoritm

Antalet length_between_peaks blev 2800 för 9,69 µH medan algoritmen hittade 1415 godtyckliga värden. För 5,425 µH var length_between_peaks 1644 medan algoritmen hittade 1609 godtyckliga värden.

7
Diskussion

7.1 Svar på frågeställningar

7.1.1 Hur kan analog mätning användas för att felsöka CAN bussar?

Genom att använda sig av ett oscilloskop på CAN hög och CAN låg kan man felsöka med hjälp av återskapade felfall från tre elektriska komponenter, resistans, kapacitans och induktans.

7.1.2 Går det att förutspå fel baserat på elektriska egenskaper i CAN-bussar?

Med de algoritmer som skapats går det att förutspå fel med en korrelation upp till 0,999915194.

7.1.3 Kräver alla kablar och mätningar att bilen är stillastående vid analog mätning?

Av säkerhetsskäl kommer bilen behöva vara avstängd för att dessa mätningar ska gå att utföras. Den data som samlas in efter bilen stängts av borde vara tillräcklig för att kunna se trender.

7.2 Förslag på vidareutveckling

Algoritmerna kan alltid bli bättre och fler. En vidareutveckling på falltiden är att placera om de spänningstransienter med falsk positivt utfall från find_peaks med ett omplaceringsprogram innan man börjar räkna ut medellängden mellan spänningstransienterna. Omplaceringsprogrammet påbörjades men det har fortfarande vissa bitar kvar innan det fungerar. Tanken bakom programmet är att det ska kunna hitta fler falltider under det uppmätta intervallet. I framtiden är tanken att en styrkrets med ett inbyggt oscilloskop ska kunna mäta kontinuerligt med begränsad processorkraft. Då skulle detta programmet kunna användas för att placera om spänningstransienterna som styrkretsen hittat istället för att behöva mäta på CAN-bussen under en längre tid och därav behöva behandla ett större dataintervall.

Har man data från riktiga CAN-bussar med mätningar gjorda över en längre tid med till exempel 10-bitars upplösning och 4 GS/s och samtidigt olika väderförhållanden kommer man kunna se ändringar i vågformerna som detta projekt inte haft möjlighet till. Då skulle man kunna se skillnader över fordonets livscykel och har det sedan blivit fel så ska det förhoppningsvis gå att förutspå dessa med hjälp av den data

man har samlat in.

En vidareutveckling hade kunnat vara att installera en Arduino som kan mäta med 227,3 KS/s.[17] Med denna hade man kunnat mäta resistansen med resistans algoritmen som gjorts. Däremot hade falltiden inte kunnat räknas ut på grund av att sampling hastigheten är för låg. Är hastigheten under 62,5 MS/s börjar sampeltiderna bli för långa för att kunna användas för mätningar. Eftersom induktansmätningarna behöver ännu högre sampling hastighet på grund av att ett ännu mindre område mäts kommer de algoritmerna inte heller att fungera. Finns det däremot andra billiga sätt man hade kunnat använda som kommer upp i 62,5 MS/s så hade man kunnat använda de andra algoritmerna med.

En annan vidareutveckling som skulle kunna tillämpas när en krets väl mäter på CAN-bussen är ett kortslutningslokaliseringssystem.[18] Detta system hade varit användbart om det hade kunnat samarbeta med de algoritmerna som skapats under detta projekt. Då hade algoritmerna som skapats i detta projekt kunnat köras varje gång bilen stängs av, för att inte påverka CAN-bussen, för att förutspå fel som sker i samband med fordonets åldrande och kortslutningslokaliseringssystemet felsöka hur långt bort kortslutningen är.

Referenser

[1] "CAN Bus Explained - A Simple Intro (2018)." [Online]. Available: https://www.csselectronics.com/screen/page/simple-intro-to-can-bus/language/en

[2] "CAN Bus Explained - A Simple Intro (2018)." [Online]. Available: https://www.csselectronics.com/screen/page/simple-intro-to-can-bus/language/en

[3] Wilfried Voss, "CAN Bus Guide – Error Frame – Copperhill Technologies – SAE J1939 & CAN Bus Prototyping." [Online]. Available: http://www.copperhilltechnologies.com/can-bus-guide-error-frame/

[4] "CAN Bus Error Handling." [Online]. Available: https://www.kvaser.com/about-can/the-can-protocol/can-error-handling/

[5] T. C. Leads, "How Cable Twisting Improves EMI – EMI Analyst." [Online]. Available: https://emianalyst.wordpress.com/2016/08/31/how-cable-twisting-improves-emi/

[6] Trung Le, "Forced perfect termination," 1990. [Online]. Available: http://www.t10.org/ftp/x3t9.2/document.91/91-037r0.pdf

[7] "CAN Physical Layer and Termination Guide - National Instruments." [Online]. Available: http://www.ni.com/white-paper/9759/en/

[8] "4.8 Savitzky-Golay Smoothing Filters," Tech. Rep., 1986. [Online]. Available: http://www.nr.com

[9] "File:Lissage sg3 anim.gif - Wikimedia Commons." [Online]. Available: https://commons.wikimedia.org/wiki/File:Lissage{_}sg3{_}anim.gif

[10] E. W. Weisstein, "Simpson's Rule." [Online]. Available: http://mathworld.wolfram.com/SimpsonsRule.html

[11] W. Stallings, *Data and computer communications*, 5th ed. Upper Saddle River, N.J.: Prentice Hall, 1997. [Online]. Available: http://libris.kb.se/bib/4563539

[12] "Introduction to SAE J2534." [Online]. Available: https://www.kvaser.com/about-can/can-standards/j2534/

[13] "Resistors dropping value with age??" [Online]. Available: https://www.eham.net/ehamforum/smf/index.php?topic=68663.0;wap2

[14] D. Kraus, E. Leitgeb, T. Plank, and M. Loschnigg, "Replacement of the Controller Area Network (CAN) protocol for future automotive bus system solutions by substitution via optical networks," in *2016 18th International Conference on Transparent Optical Networks (ICTON)*. IEEE, jul 2016, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/document/7550335/

[15] G. Elert, "Inductance - Summary – The Physics Hypertextbook." [Online]. Available: https://physics.info/inductance/summary.shtml

[16] "Inductor in a DC Circuit." [Online]. Available: http://macao.communications.museum/eng/exhibition/secondfloor/MoreInfo/2_3_6_ResistanceInductance.html

[17] "Arduino High Speed Oscilloscope With PC Interface: 8 Steps." [Online]. Available: https://www.instructables.com/id/Arduino-High-speed-Oscilloscope-with-PC-interface/

[18] X. Du, S. Jiang, A. Nagose, Y. Zhang, and N. Wienckowski, "Locating Wire Short Fault for In-Vehicle Controller Area Network with Resistance Estimation Approach," *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*, vol. 9, no. 1, pp. 2016–01–0065, apr 2016. [Online]. Available: http://papers.sae.org/2016-01-0065/

A
Bilaga

A.1 Algoritmer

Resistans algoritm

```
import csv
import numpy as np
x = []
h = []
l = []
measure = 0
i = 0
intervall = 0
value = []

with open('Resistans 266,9.csv', 'r') as csvfile:
    next(csvfile)
    next(csvfile)
    next(csvfile)
    plots = csv.reader(csvfile, delimiter=',')
    for row in plots:
        x.append(float(row[0]))
        h.append(float(row[1]))
        l.append(float(row[2]))
        delta = float(row[1]) - float(row[2])

        if delta > 0.5: #Letar efter en etta mellan can high och low
            if measure == 0:#Kollar s att man inte brjat spara undan
                ↪ vrden
                measure = 1
                tid = float(row[0]) #Sparar tiden d skillnaden mellan
                    ↪ can high o low r tillrckligt signifikant fr att
                    ↪ man ska kunna frustp en etta
            else:
                newtid = float(row[0]) #Sparar en senare tid som till
                    ↪ slut ska representera att biten r 1
                intervall = newtid - tid
                if 0.000350 <= intervall <= 0.000450: #Det intervall
                    ↪ som frhoppningsvis r toppen p biten som snds
```

```
                    if delta > 0.5:
                        value.append(float(delta)) #Lgger de aktuella
                            ↪ delta vrdena i en array
                if intervall > 0.0004500: #D mtningen r klar nollstlls
                    ↪ mtningssekvensen, intervall vrdet samt delta.
                        measure = 0
                        intervall = 0
                        delta = 0
print(value)
print('value', np.mean(value)) #Medelvrdet av en array
```

Algoritm för kapacitans

```
from __future__ import print_function
from scipy.signal import savgol_filter
import matplotlib.pyplot as plt
import csv
from pylab import *
import numpy as np
from scipy.integrate import simps

x = []
h = []
l = []
measure = 0
tid = 0
i = 0
value = []
intervall = 0
counter = 0
array_intervall = []

with open('Induktans 5,425.csv', 'r') as csvfile:
    next(csvfile)
    next(csvfile)
    next(csvfile)
    plots = csv.reader(csvfile, delimiter=',')
    for row in plots:
        x.append(float(row[0]))
        h.append(float(row[1]))
        l.append(float(row[2]))

filteredhigh = savgol_filter(h, 31, 3)
filteredlow = savgol_filter(l, 31, 3)
filtereddelta = filteredhigh - filteredlow

gradients = np.gradient(filtereddelta)
```

```
while i < len(filtereddelta):
    p = gradients[i]
    #q = filtereddelta[i]
    row = x[i] #row r tiden
    i += 1

    if p < -0.009:
        if measure == 0: # Kollar s att man inte brjat spara undan
            ↪ ett vrde
            measure = 1
            tid = row #Sparar starttiden
            startindex = i
            intervall = 0
        else:
            newtid = row # Sparar en senare tid som ska representera
                ↪ att filtereddelta brjar nrma sig 2.2 V igen
            intervall = newtid - tid
            stopindex = i
            if intervall > 0.00013:
                measure = 0
                tid = 0
    if 0.00006 <= intervall <= 0.00013: # Storleken p intervall som
        ↪ gr att det frmodligen r en falltime (0.00008317 en hel
        ↪ falltime isch)
        if p > -0.009:
            array_intervall = x[startindex:stopindex]
            area = -1*simps(gradients[startindex:stopindex],
                ↪ array_intervall) #scipy.integrate.simps(y, x=None,
                ↪ dx=1, axis=-1, even='avg')
            print("intervall =", intervall)
            print("tid =", tid)

            if area > 1.00e-06:
                print("area =", area)
                value.append(float(intervall)) # Sparar undan
                    ↪ falltiden i en array
                counter += 1
                print(counter) # count ska bli exakt 119
                intervall = 0
                measure = 0
            else:
                intervall = 0
                measure = 0
print('Falltiden i ms r:', np.mean(value)) # Tidsaxeln i datafilen r
    ↪ i ms
```

```
#dx r 0.00005600 (ms)
```

Algoritm för Induktans

```
from __future__ import print_function
from pylab import *
from scipy.signal import *
import csv
import numpy as np
import matplotlib.pyplot as plt


x = []
h = []
l = []
filtered = []
i = 0
peakslow = []
peakshigh = []

new_list_a = []
new_list_b = []

with open('Induktans 5,425.csv', 'r') as csvfile:
    next(csvfile)
    next(csvfile)
    next(csvfile)
    plots = csv.reader(csvfile, delimiter=',')
    for row in plots:
        x.append(float(row[0]))
        h.append(float(row[1]))
        l.append(float(row[2]))

y_high = savgol_filter(h, 31, 3)
y_low = -1*savgol_filter(l, 31, 3)

peakshigh, _ = find_peaks(y_high, height=3.58, distance=150) #Lngd
    ↪ mellan tv toppar under sndning=0,00455
peakslow, _ = find_peaks(y_low, height=-1.355, distance=150) #Lngd i
    ↪ sekunder p sampel 0,000008. 0,00455/0,000008=568
                                                        #sampel/2=284
np.set_printoptions(threshold=np.nan) #Gr s att man printar alla
    ↪ vrden


length_between_peaks = peakshigh-peakslow #Tar absolubeloppet av
    ↪ skillnaden i y axel(vrdelst). Tar bort 1 frn slutet av arrayen
```

```
print('Number of length_between_peaks', len(length_between_peaks))


for a, b in zip(peakshigh, length_between_peaks):
    if 80 > b > -10:
        new_list_a.append(a)

for a, b in zip(peakslow, length_between_peaks):
    if 80 > b > -10:
        new_list_b.append(a)

#filtrerar ut de dliga vrdena p ett ondigt svrt stt fr att jag
    ↪ egentligen
#ville aligna om vrdena som blev felaktiga s att jag fick ett strre
    ↪ dataset av vrden men
#det kan vi ta en annan gng :)))))

filtered = np.array(new_list_a) - np.array(new_list_b)

print(filtered)
print('Len of filtered', len(filtered))

print(np.mean(filtered))

plt.plot(peakshigh, y_high[peakshigh], "x")
plt.plot(peakslow, y_low[peakslow], "x")
plt.plot(y_high, label='Hg')
plt.plot(y_low, label='Lg')

plt.legend()
plt.show()
```

Ej färdigställd induktans algoritm.

```
from __future__ import print_function
from pylab import *
from scipy.signal import *
import csv
import numpy as np


x = []
h = []
l = []
filtered = []
i = 0
peakslow = []
```

```
peakshigh = []

listlow = []
listhigh = []
ihigh = 0
ilow = 0
length_between_peaks = []
imin = 0
shape = 0
count_less = 0
peaks_length_temp = []

with open('5,944 litemindre.csv', 'r') as csvfile:
    next(csvfile)
    next(csvfile)
    next(csvfile)
    plots = csv.reader(csvfile, delimiter=',')
    for row in plots:
        x.append(float(row[0]))
        h.append(float(row[1]))
        l.append(float(row[2]))

y_high = savgol_filter(h, 31, 3)
y_low = -1*savgol_filter(l, 31, 3)

peakshigh = find_peaks(y_high, height=3.58, distance=150) #Lngd
    ↪ mellan tv toppar under sndning=0,00455
peakslow = find_peaks(y_low, height=-1.334, distance=150) #Lngd i
    ↪ sekunder p sampel 0,000008. 0,00455/0,000008=568
                                                        #sampel/2=284
np.set_printoptions(threshold=np.nan) #Gr s att man printar alla
    ↪ vrden
"""
print(peakslow[0][:20])
print(peakshigh[0][:20])
print(peakshigh[0][:20]-peakslow[0][:20])
"""
shape = len(peakshigh[0])-len(peakslow[0])
print(shape)

if shape > 0:
    peakshigh = peakshigh[:-shape]
if shape < 0:
    peakslow = peakslow[:-shape]
print(peakshigh)
print('-----')
```

```
while ilow < len(peakslow) or ihigh < len(peakshigh):
    imin = min(ilow, ihigh) #Lgger minsta vrdet av ilow eller ihigh
        ↪ fr att veta vilket vrde som r lngst bak i fltet
    count_less += 0 # nollstller antalet steg bort frn det evaluerade
        ↪  vrdet
    length_between_peaks = np.array(peakshigh[ihigh]) - np.array(
        ↪ peakslow[ilow]) # findpeaks skickar ut en matris istllet fr
        ↪  en array behver man specifiera frsta raden


    if length_between_peaks[imin] < 80 and length_between_peaks[imin]
        ↪  > 5: #Ligger vrdet fr lngden inom rtt intervall
        listhigh.append(peakshigh[ihigh])
        listlow.append(peakslow[ilow])
        ilow += 1
        ihigh += 1

    elif length_between_peaks[imin] < 5: #Letar reda p nsta hga vrde
        ↪ i peakshigh

        print('hello')
        peaks_length_temp[imin] = peakshigh[ihigh + count_less] -
            ↪ peakslow[ilow] #peakshigh skickar ut en matris istllet
            ↪ fr en array behver man specifiera frsta raden
        print(peaks_length_temp[imin])

        while count_less < 2 and peaks_length_temp < [5]: # gr en
            ↪ rknare som sparar undan ihighs vrden och sedan kollar p
            ↪  fljande vrden om det r ngra av dem som skulle kunna
            ↪ vara rtt vrde. Kollar endast 2 gnger
            peaks_length_temp[imin] = peakshigh[ihigh+count_less] -
                ↪ peakslow[ilow]
            print('Print caounter', count_less)
            if 5 > peaks_length_temp[imin] > 80:
                try:
                    listhigh.append(peakshigh[ihigh+count_less])
                    listlow.append(peakslow[ilow])
                    ihigh = 1 + ihigh + count_less
                    ilow += 1
                    print('-----')

                except IndexError:
                    print('indexerrror')
                    break
```

```
            elif peaks_length_temp < 5:
                print('countless')
        try:
            peaks_length_temp[imin] = peakshigh[ihigh+count_less] -
                ↪ peakslow[ilow] # findpeaks skickar ut en matris
                ↪ istllet fr en array behver man specifiera frsta
                ↪ raden
        except IndexError:
            print('fel')
            break

    elif length_between_peaks[imin] > 80:##Letar reda p nsta lga vrde
        ↪  i peakslow
        peaks_length_temp = peakshigh[0][ihigh] - peakslow[0][ilow]

        while count_less < 2 and peaks_length_temp > 80: # gr en
            ↪ rknare som sparar undan ihighs vrden och sedan kollar p
            ↪  fljande vrden om det r ngra av dem som skulle kunna
            ↪ vara rtt vrde
            peaks_length_temp = peakshigh[0][ihigh] - peakslow[0][ilow
                ↪  + count_less]

            if 5 > peaks_length_temp > 80:
                try:
                    listhigh.append(peakshigh[0][ihigh + count_less])
                    listlow.append(peakslow[0][ilow])
                    ihigh = 1 + ihigh
                    ilow = 1 + ilow + count_less

                except IndexError:
                    break

            elif peaks_length_temp > 80:
                count_less += 1

"slnger de sista vrdena i den array som r hgst av high och low"
shape = len(listhigh)-len(listlow)
if shape > 0:
    listhigh = listhigh[:-shape]
if shape < 0:
    listlow = listlow[:-shape]



print(shape)
print(len(listhigh))
```

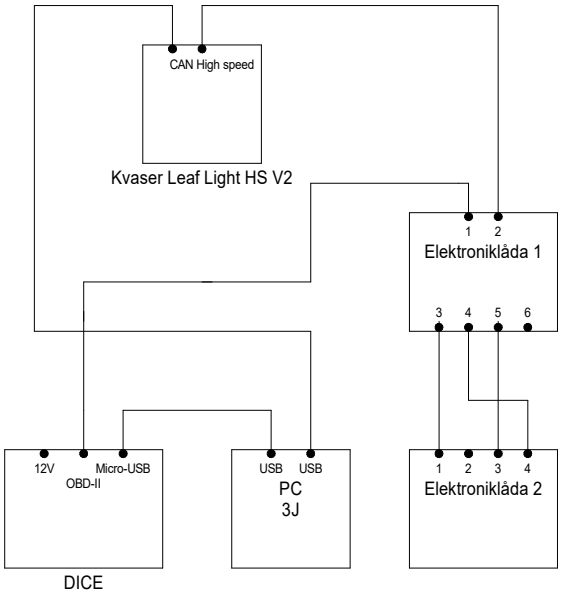
```
print(listlow)
filtered = np.array(listhigh) - np.array(listlow)

print('Filtered: ', filtered)
print(np.mean(filtered*0.00000800*1000))
```

A.2 Kommunikation med DICE

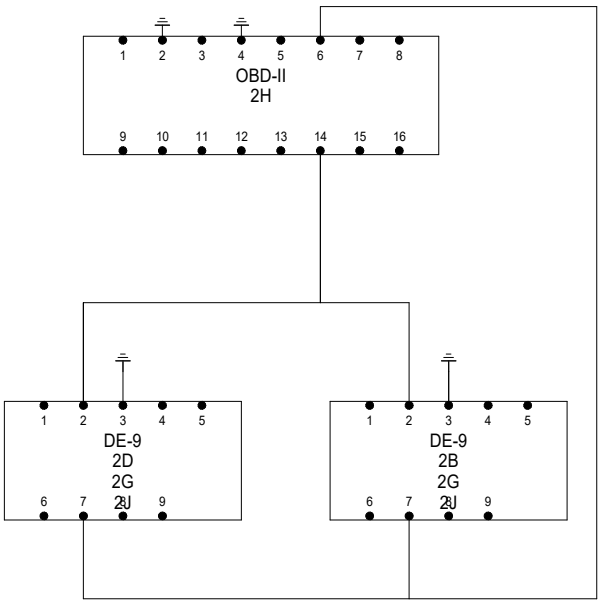
A.3 Elscheman

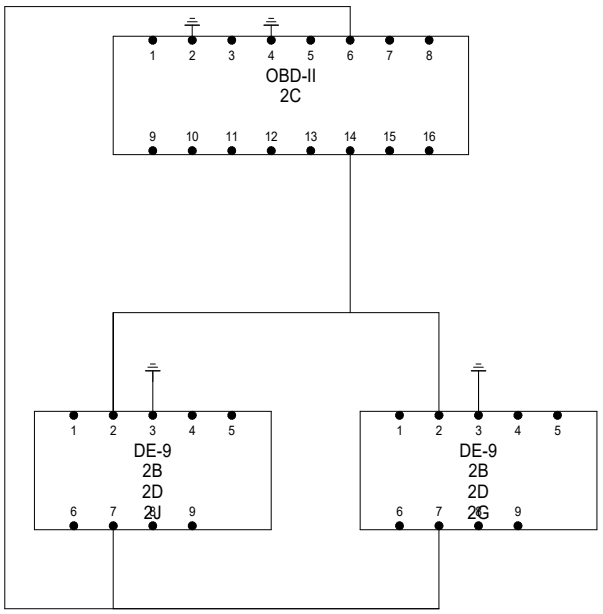
A	B	C	D	E	F	G	H	J	K



Chalmers Tekniska Högskola

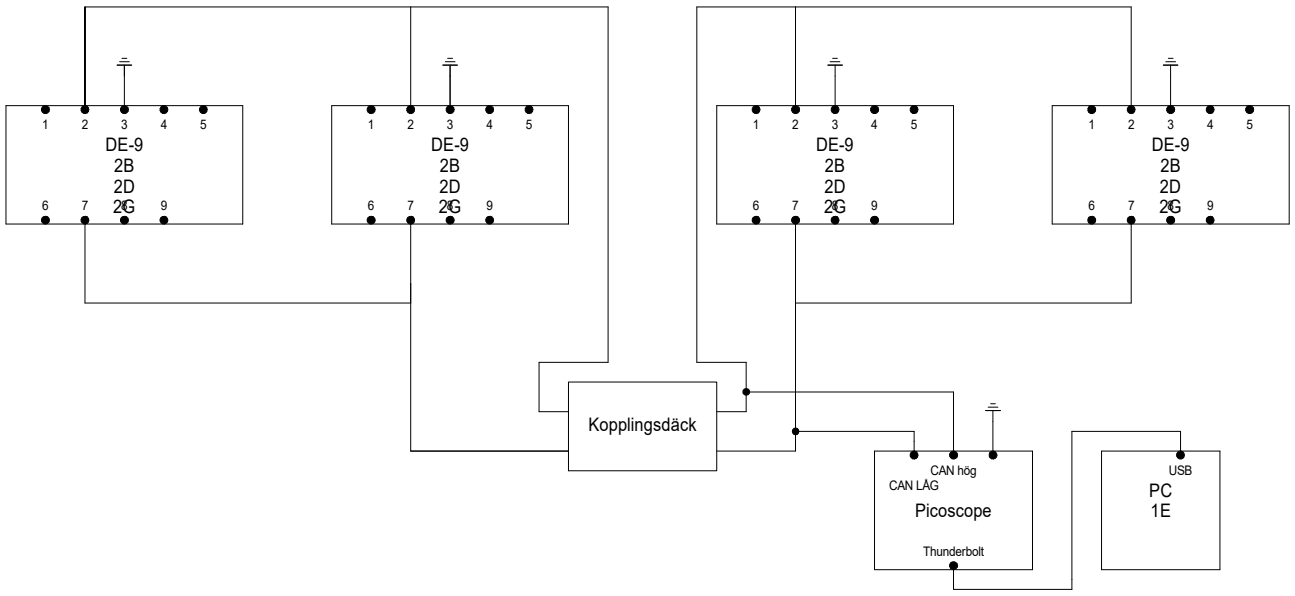
		Översikt	FastEL / EasyEL www.pkdata.se	KONST.	DATUM	NÄSTA BLAD 2	BLAD 1
				GRANSKAD	UTGÅVA A*	RITN.NR	





Chalmers Tekniska Högskola

		Elektroniklåda 1	FastEL / EasyEL www.pkdata.se	KONST.	DATUM	NÄSTA BLAD 3	BLAD 2
				GRANSKAD	UTGÅVA A*	RITN.NR	



Chalmers Tekniska Högskola

		Elektroniklåda 2	FastEL / EasyEL www.pkdata.se	KONST.	DATUM	NÄSTA BLAD	BLAD 3
				GRANSKAD	UTGÅVA A*	RITN.NR	