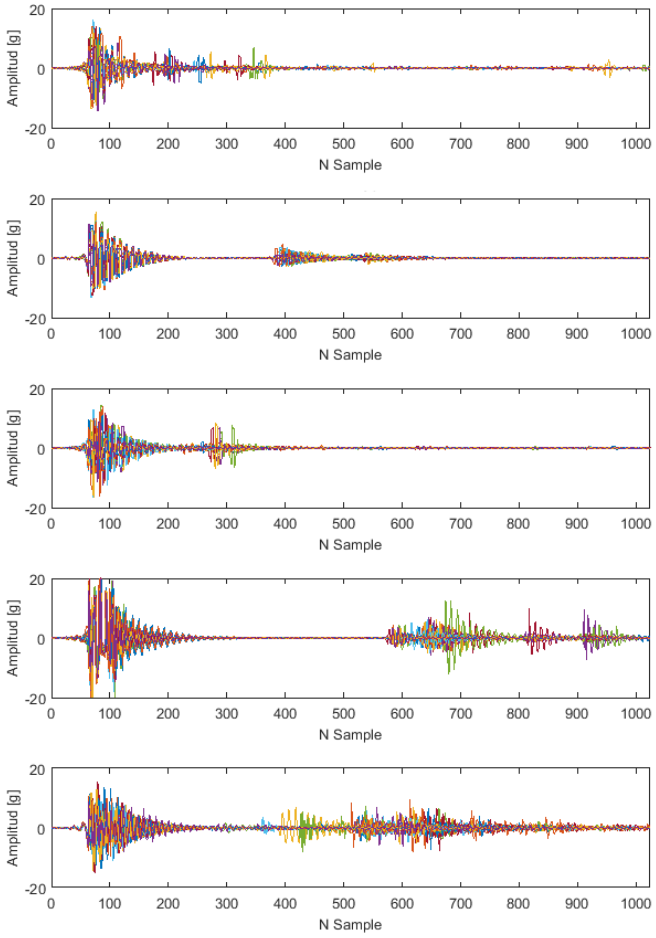
CHALMERS



Analys av sensordata

Examensarbete inom högskoleingenjörsprogrammet Elektroingenjör

MAGNUS EKSTRÖM
PETER LAGERBERG

EXAMENSARBETE 2017

Analys av sensordata

MAGNUS EKSTRÖM
PETER LAGERBERG

Institutionen för Signaler och System
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2017

Analys av sensordata

MAGNUS EKSTRÖM
PETER LAGERBERG

Institutionen för Signaler och System
Chalmers tekniska högskola
SE-412 96 Göteborg
Sverige
Telefon: +46 (0)31-772 1000

Förord

Examensarbetet ingår som ett moment på Chalmers tekniska högskola elektroingenjörsprogrammet 180 HP och omfattar 15 HP.
Arbetet har utförts på Consat Engineering AB´s kontor i Göteborg

Vi vill tacka Consat Engineering AB för att ha gett oss detta uppdrag samt bidragit med en trevlig och inspirerande arbetsmiljö. Vill även rikta ett särskilt stort tack till våra handledare:

- Johan Rubenson, Consat Engineering AB
- Jonas Williamsson, Manager in-house Projects, Consat Engineering AB

Peter Lagerberg och Magnus Ekström, Göteborg, Juni 2017

Sammanfattning

Detta examensarbete är utfört på Consat Engineering AB under våren 2017. Syftet med arbetet är att se om det med hjälp av en algoritm går att skilja på olika specifika testscenarion som en accelerometer utsätts för.

Det finns många olika sensorer och givare ute på marknaden och accelerometern är en av dem. Accelerometrar används i många olika instrument och tekniska system men också i vardagliga apparater som mobiltelefonen, laptops och i bilar. Trots detta har troligtvis inte många hört talas om accelerometern eller än mindre vet hur den fungerar. Accelerometern är enkelt uttryckt en elektromekanisk apparat som mäter accelerationskrafter. I stillastående läge mäts den gravitation som jorden ger upphov till och om accelerometern påverkas av en stöt eller rörelse så mäter den även den accelerationen.

I projektet kommer en accelerometer av typen ADXL345 från tillverkaren Analog Devices och en mikrokontroller av typen PIC16F1719 från tillverkaren Microchip användas för att samla in och behandla testdata. Den inhämtade datan kommer sedan analyseras för att möjliggöra utvecklingen av den algoritm som ska kunna skilja på de olika testscenarion som tagits fram.

Det inledande förarbetet som genomfördes innefattade informationsinhämtning från datablad tillhörande accelerometer och mikrokontroller. Det har även pågått en beslutsprocess gällande utformningen av de testscenarion som ska generera testdata från accelerometern. Diskussionen har varit tidskrävande då dessa testscenarion är sekretessbelagt och på grund av detta kommer de inte att beskrivas i rapporten.

Arbetets huvudsakliga uppgift och där mest tid lagts är att sammanfatta den testdata som accelerometern gett och att hitta skillnader och unika mönster i de scenarion som skall särskiljas. Utveckling och verifiering av algoritmen har utförts i Matlab. Arbetets sista fas innebar implementering av den fungerande algoritmen i mikrokontrollern, vilket visade sig vara en utmaning då minne och beräkningskraft är begränsad.

Arbetet resulterade i en fungerande algoritm som inte kräver stor beräkningskraft eller mycket minne. Algoritmen kunde identifiera 4 av 5 scenarion. Dock kunde en kraftigare processor med högre beräkningskraft samt större minnesutrymme ha underlättat processen och möjliggjort en mer avancerad och precis algoritm.

Abstract

This thesis was performed at Consat Engineering AB in the spring of 2017. The purpose of the thesis is to develop an algorithm to identify various fixed events to which an accelerometer might be exposed.

There are many different sensors on the market, the accelerometer is one of them. The accelerometer is used in many different instruments and technical systems, but also in everyday devices such as mobile phones, laptops and cars. Still, many people might not have heard of the accelerometer and even less know how it works. The accelerometer is, in simple terms, an electromechanical device that measures the proper acceleration, i.e., relative the free-falling reference frame. Hence, in static or rest mode, an accelerometer shows a measure of the gravity acceleration *g* of the earth acting upon it, but directed upwards. In general, if the accelerometer is affected by a shock or movement, it measures the proper acceleration.

The project employed an accelerometer of the type ADXL345 and a microcontroller of type PIC16F1719 to collect and process test data. The data was analyzed to enable the development of the algorithm to distinguish between the different scenarios.

The pre-study consisted of information gathering from data sheets of the accelerometer and the microcontroller. It also included a decision process on the design of the test scenario that will yield test data from the accelerometer. The discussion has been time consuming since the test scenario is confidential and could therefore not be included in the report.

The main tasks of the work are: the summarizing of all the test data provided by the accelerometer and finding differences and unique patterns in the events to be determined. This has been done in Matlab with various toolboxes and features, the code for the algorithm has also been written and tested in Matlab.

The final phase of the project was to implement the algorithm on the microcontroller, which proved to be a challenge due to memory limitations as well as processing power.

The work resulted in a functioning algorithm that does not require much processing power or memory. The algorithm did not solve all test scenarios. However, a more powerful processor with more data memory could have made the process easier and allowed for the algorithm to be more advanced and more accurate.

Ordlista

FFT	Fast Fourier Transform
SPI	Serial Peripheral Interface
USART	Universal Synchronous Asynchronous Receiver Transmitter
PIC	Peripheral integrated circuit
I^2C	Inter integrated circuit
SS	Slave select (chip select)
PCB	Printed circuit board
SCLK	Serial Communications Clock
MOSI	Master out slave in
MISO	Master in slave out
IDE	Integrated Development Environment

Innehållsförteckning

Innehåll

1 Inledning

1.1 Bakgrund

Accelerometerar används i många olika system och tillämpningar. Accelerometern är en relativt billig komponent som i rätt system kan vara användningsbar. Företaget Consat Engineering AB är alltid nyfikna på att se hur man skulle kunna utveckla nya och redan befintliga tekniker.
Consat Engineering AB har tidigare genomfört projekt med liknande tekniker och är nu nyfikna på vad man kan göra med en accelerometer.

1.2 Syfte

Syftet med arbetet är att se om det med hjälp av en algoritm går att skilja på fem bestämda testscenarion som en accelerometer utsätts för. Dessa testscenarion kommer att utvecklats med hjälp av Consats ingenjör Johan Rubenson. De testscenarion som kommer att användas skall vara repeterbara och kommer ej att beskrivas ingående på grund av sekretess. Exempel på hur man kan påverka en accelerometer fysiskt är genom att fästa accelerometern på ett objekt och:

- Släppa objektet och mäta accelerationen vid studs
- Kasta upp objektet i luften och få ut när objektet är högst upp, det vill säga när accelerationen stannar och innan den ökar igen.

Projektmål:

- Analysera och hitta skillnader i den information som accelerometern skickat utifrån det testscenario som används.

- Ta fram en algoritm som kan skilja på de fem testscenarion som framtagits i samråd med ingenjör Johan Rubenson på Consat Engineering AB

- Implementera en algoritm i en mikrokontroller av typen PIC16F1719 samt verifiera dess prestanda.

1.3 Avgränsning

Avgränsningar som gjorts till följd av önskemål från företaget är att en accelerometer skall användas tillsammans med en mikrokontroller. Consat Engineering AB önskade också att en digital accelerometer användes. Testscenariot har utvecklats med hjälp av Consats ingenjör Johan Rubenson. Hur påverkan på sensorn görs kommer inte att beskrivas ingående på grund av sekretess. Algoritmen skall gå att använda i mikrokontrollern PIC16F1719.

1.4 Precisering av uppgiften

- Verifieringen är begränsat till fem stycken testscenarion där accelerometern påverkas fysiskt på ett repeterbart sätt. Dessa testscenarion kommer benämnas scenario A-E. Kap 1.2

- Det färdiga systemet bestående av accelerometern AXDL345 och mikrokontrollern PIC16F1719 skall kunna särskilja de fem specificerade scenarion, A till E, som accelerometern utsätts för.

- Algoritmen skall även implementeras i mikrokontroller.

- Accelerometerns användbarhet i denna tillämpning skall utvärderas med avseende på precision och känslighet.

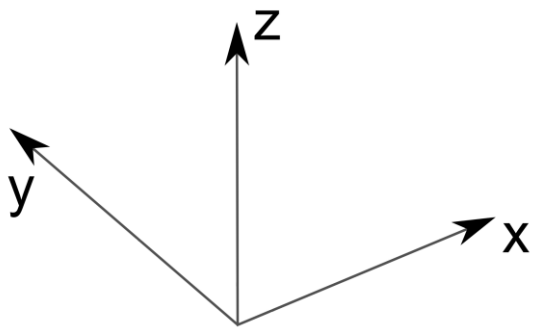
2 Teknisk bakgrund

I detta kapitel beskrivs de olika komponenter och verktyg som har använts.

2.1 Accelerometer

En accelerometer är en elektromekanisk anordning som mäter accelerationskrafter. Acceleration är ett mått på förändringen av hastighet per tidsenhet[1]. Dessa krafter kan vara statiska eller dynamiska samt kan uppstå av direkt eller indirekt påverkan av accelerometern. Ett exempel på statisk påverkan är jordens gravitation som hela tiden påverkar accelerometern, stillaliggande på ett plant underlag ger accelerometern värdet +1g[2] på den axel som ligger vinkelrätt mot jorden.

Accelerometern kan komma med en, två och tre axlars visning. En axels visning ger som det låter bara en dimension av påverkan medan tre axlar kan ge tredimensionell information från påverkan vilket ger ett mer precist och noggrant resultat. Axlarna benämns ofta med X, Y och Z, riktade på det sätt som illustreras i figur 1.

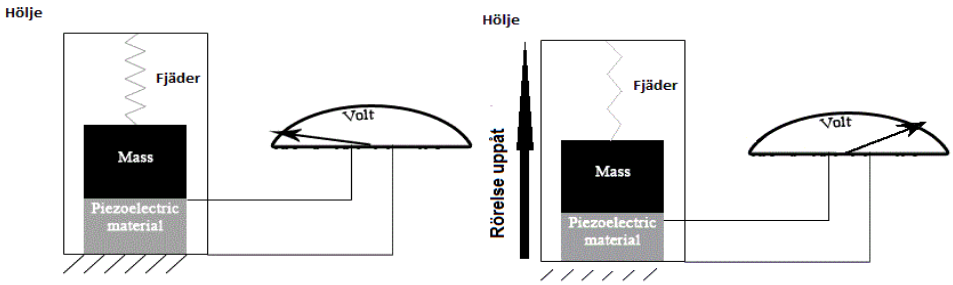


Figur 1 Visar hur axlarna är riktade i en accelerometer.

Det finns olika metoder som accelerometern kan vara uppbyggda efter, en metod som förekommer använder sig av piezo-effekten[3]. Den använder mikroskopiska kristall-strukturer som vid påverkan genererar spänning. En annan metod är att känna av ändringar i kapacitansen, när två mikrostrukturer ligger bredvid varandra har de en viss kapacitans mellan sig. När dessa sedan via acceleration glider isär uppstår en ändring i kapacitansen som sedan konverteras till spänning. Dock är de byggda efter principen ett hölje och en massa som är upphängd i en fjäder. När accelerometern blir påverkad upp eller ner kommer massan att ha en tröghet som drar ut fjädern eller trycker ihop den. Med den rörelsen kan gravitationskraften räknas fram, eller så kan massan i samma fall påverkar ett piezoelektriskt material som i sin tur avger en spänning, se

[1] Acceleration – Hastighet/tid = meter per sekund/sekund = m / s^2 = $m*s^{-2}$
[2] 1g- g står för gravitation och i detta fall jordens tyngdacceleration ca 9.82m/s^2= 1g.
[3] Piezo = Tryck/stress på grekiska. Piezo effekten är när ett material genererar spänning under tryck/stress[12]

figur 2. Den nya teknologin använder samma princip men med mikroskopiska storlekar och andra material, som till exempel kisel.



Figur 2 Visar uppbyggnad och funktion av en accelerometer med piezoelement.

2.1.1 Användningsområden

Ingenjörer har utnyttjat accelerometer inom många olika användningsområden. Till exempel är accelerometern bra att använda vid lutningsberäkningar. Lutningen fås genom beräkningen av tangens mellan den accelerationen som läses från två av axlarna där ena axeln ska vara vid horisontellt läge och den andra i vertikalt läge. Då en axel är helt vertikal är accelerationskraften ±1 g och 0 g om horisontal, om nu accelerometern lutar ändras dessa värden efter hur mycket den lutar.

Ett annat användningsområde är att detektera fritt fall eller om den påverkas av en dynamisk acceleration. Exempelvis på användningsområde på den dynamiska påverkan är att accelerometern används i Indycar race, fäst på förarnas hjälmar. På det sättet kan accelerationskrafterna som föraren utsätts för vid en krock läsas av och säkerheten runt föraren förbättras [1]. Fritt fall funktionen har länge använts av laptop-tillverkare för att stänga av hårddiskarna när datorn hamnar i fritt fall så att läshuvud inte slår i och skadar läsytan när den träffar marken. Andra områden kan vara företags-hälsa vid användandet av vibrerande verktyg och maskiner. Där placeras sensorer på maskiner och verktyg som användaren kommer i kontakt med, sedan avläses värdena för att kunna se om användaren påverkas av för kraftiga vibrationer under för lång tid[2].

Accelerometrar finns med en, två eller tre axlar. Tre axlar används exempelvis ofta i mobiltelefoner för att med hjälp av dessa axlar beräkna hur telefonen är lutad och då rotera displayen efter detta, även för att avgöra hur mobiltelefonen hålls vid fotografering[3]. Känsligheten i givarna kan variera men ju känsligare accelerometer desto lättare blir det att mäta accelerationen och precisionen ökar.

2.1.2 Accelerometern ADXL345

ADXL345 är en liten, energisnål accelerometerkrets med tre axlars mätning. Den digitala utgångsdatan har formatet 16-bitars heltal. Den har en upplösning på 10 till 13 bitar med mätområde upp till ± 16g. Dom resterande bitarna kommer att indikera om

värdet är positivt eller negativt. Tabell 1 visar vilka värden mätdatan kan anta beroende på vald upplösning.

Upplösning [antal bitar]	Motsvarande värde
10	[-1024, 1023]
11	[-2048, 2047]
12	[-4096, 4095]
13	[-8192, 8191]

Tabell 1 Värden som mätdatan kan anta beroende på upplösning

Detta sätt att representera data kallas två-komplement och den generella formeln visas i tabell 2. Det värde som accelerometern ger i g kommer att digitaliseras till den valda upplösningen, 10 bitar till 13 bitar. Ett exempel är om accelerometern ger 1 g i en upplösning på 10 bitar och inställd på ±16g kommer utgående värde vara 2^{10}/16g = 1024/16g = 64.

Normal	2-Komplement
[0, (2^n-1)]	[-2^{n-1}, (+2^{n-1}-1)]
[0, 255]	[-128,+127]
	n= antal bitar

Tabell 2 Visar skillnaden hur normal och 2-komplement räknas ut.

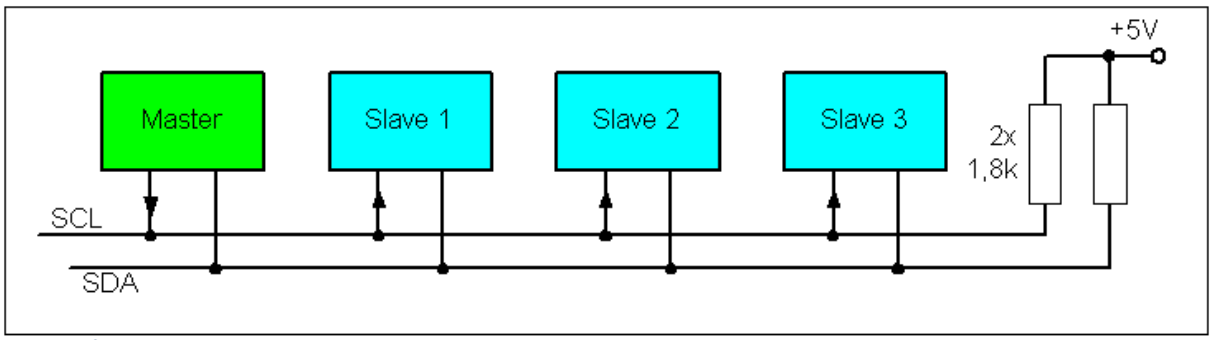
Alla initieringar och inställningar görs via register som ligger på accelerometern. Avläsningar av datavärden från respektive axel görs också via register. Detta måste ske i en specifik ordning och efter ett timingdiagram, se Bilaga A, som finns i dokumentationen för ADXL345 [4]. Dokumentationen visar också att timingen skiljer sig mellan de olika kommunikationsprotokoll, Serial Peripheral Interface och Inter integrated circuit, som AXDL345 stödjer. I fortsättningen kommer SPI och I^2C användas.

Accelerometer ADXL345 har även 2 fysiska utgångar, *int 1* och *int 2*, som används av inbyggda funktioner för att meddela att ett visst scenario har skett som i sin tur exempelvis kan används till att veta när det finns data att hämta från registret. En sådan funktion är *Singel_tap* vilken är en inbyggd algoritm i accelerometern. Denna kan då aktivera en av utgångarna när den dynamiska påverkan var högre än ett förinställt gränsvärde. Det finns även möjlighet att ställa in ett tidsvillkor, d.v.s. en tid som accelerationen skall ha överstigit gränsvärdet under för att utgången skall aktiveras. När väl utgången är definierad måste registret, där värdena lagras, läsas av och i och med avläsningen återställdes utgången [4].

2.1.3 SPI/ I²C

SPI och I²C är båda kommunikationsprotokoll som används i dagens digitala elektroniska system och kommer förmodligen att fortsätta användas i framtiden också då både I²C och SPI passar bra till seriell kommunikation. SPI introducerades med de första mikrokontrollern som härledde från Motorolas ”68000 mikroprocessor” från 1979, den mest kraftfulla mikroprocessorn just då som också användes i Apple Macintosh-datorer 1984 [5].

I²C använder sig av två signaler kallade Serial Data (SDA) och Serial Clock (SCL) som visas i figur 3. Vid användande av I²C behövs inte något Slave Select (SS) oavsett antal masters eller slaves då alla kopplade till dom 2 signalerna har unika adresser. SS är en signal som används för att aktivera rätt slav, en sorts på och av knapp som är direktkopplad till varje enhet kopplad till bussen. Normalt består adressen av 7 bitar, dock finns möjligheten till 10 bitars adresser. Till skillnad från SPI stöder I²C inte *full duplex* [6], vilket innebär att kommunikation i båda riktningarna inte kan ske samtidigt.



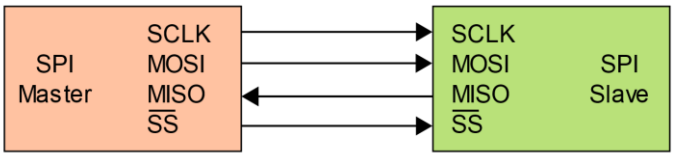
Figur 3 I²C Buss Master och 3 slaves

SPI kopplar ihop master och slave på två olika sätt, och använder sig av tre eller fyra ledningar mellan dem. Vid 4-ledningars koppling används en ledning för varje signal SCLK, MOSI, MISO och SS som visas i figur 4, och vid 3 ledningars koppling använder MOSI och MISO samma ledning.

Kommunikationen styrs av Master-enheten som bestämmer all kommunikation med slaves, när mastern vill skicka data till en slave eller inhämta information så väljer den slav genom att sätta respektive SS signal låg som i sin tur aktiverar klockan som både master och slave synkroniserar efter.

I²C har fördelar i att det bara behöver två kablar till skillnad från SPI som behöver minst 3 och en SS signal till varje slave. När det gäller hastigheten är SPI det klara valet då SPI har full duplex och inte har någon definierad maxfart men går oftast över 10 Mb/s. I²C är begränsad till ca 3,4 Mb/s i high speed mode. I²C erbjuder fler och mer avancerade funktioner men är också mer komplicerat, medan SPI kan ses som enkelt

att förstå och passar bättre till system som handlar om mycket dataströmmar, och I^2C till system med flera masters och slaves.



Figur 4 SPI Master och slave i en 4 lednings kopplad

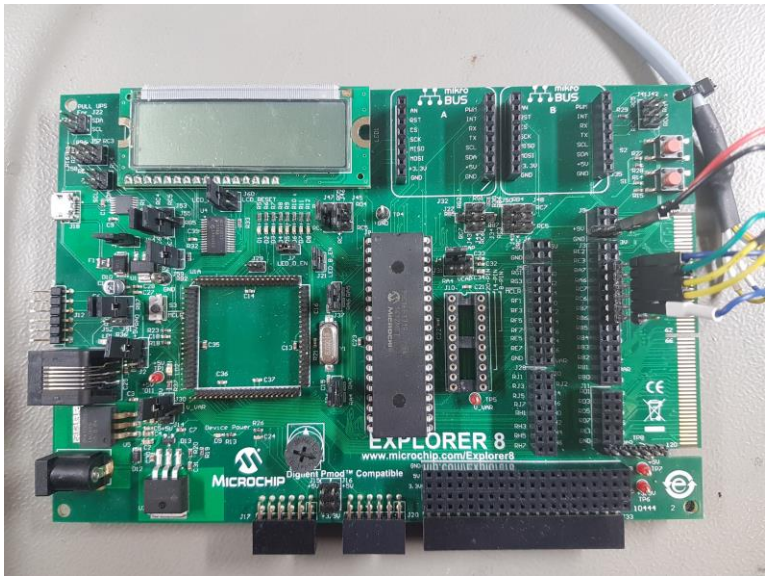
2.2 Explorer 8 Development Board

För att kunna sammankoppla dom olika enheterna på ett lätt sätt utan behov av lödning eller liknande, har ett utvecklingskort använts. Vad detta är och hur det fungerar beskrivs här.

2.2.1 Beskrivning

Explorer 8 Development board är ett fullt utrustat utvecklingskort för 8-bit PIC mikrokontrollers från Microchip. Som utvecklingskort passade Explorer 8 bra då den är anpassad för ett stort antal utbytbara 8-bitars mikrokontroller, bland annat den mikrokontroller som används i denna rapport (PIC16F1719). Fördelen med detta utvecklingskort är att det är samma tillverkare som gjort både mikrokontrollern och utvecklingskortet vilket medför att man kan lita på att dom är helt kompatibla med varandra. Utvecklingskortet är väldokumenterat vilket gör det lätt att komma igång med.

Då vi i detta projekt först och främst vill utveckla en algoritm och mjukvara passar det bra att starta på ett utvecklingskort med färdig strömförsörjning, bra anslutnings-möjligheter och möjlighet att prova olika mikrokontroller utan större problem [7].



Figur 5 Explorer 8 Development board av Microchip Technology Inc.

2.2.2 Parallell / Seriell kommunikation.

Kommunikation mellan enheter kan ske i seriell eller parallell form. I detta projekt kommer seriell överföring användas mellan Explorer 8 kortet och PC.

Parallell kommunikation innebär att databitarna skickas parallellt i grupper och för att detta ska fungera behöver det finnas en ledning för varje bit som ska sändas. Som exempel är det denna sorts kommunikations om används inne i en dator mellan minnen och processor i databussar som har 32 ledningar eller 64 ledningar. 32-lednings buss kan alltså överföra 32 bitar i taget och kallas också 32-bitars buss. Denna kommunikation lämpar sig för kommunikation över korta sträckor med snabb dataöverföring.

Vid seriell kommunikation skickas alla bitar, som namnet antyder, seriellt vilket gör att de inte är lika snabbt som parallell men istället både enklare och mindre störningskänsligt vilket leder till att seriell kommunikation kan användas vid längre transporter av data. Seriell kommunikation behöver till skillnad från parallell bara en ledare i teorin. I praktiken används dock två ledare. Sändare, mottagare och en gemensam jord för att få full duplex tvåvägskommunikation.

För att mottagaren i den seriella kommunikationen skall kunna veta var informationen böjar och slutar används start- och stoppbitar som skickas först och sist i datan, dessa meddelar att data är på väg och när den är slut. Både mottagare och sändaren måste vara överens om hur många bitar som skall sändas samt vilken baud rate[4] som skall användas. Ett annat sätt att säkerställa att mottagna data är korrekt och inga bitar tappats på vägen är att ha en paritetsbit. Innan datan skickas av sändaren, summeras antalet 1 eller 0 i meddelandet och sedan sätts paritetsbiten och anger om det är udda eller jämn antal. Denna metod avslöjar ej vilken bit som är fel utan bara att meddelandet innehåller ett fel.

[4] Baud rate är måttenhet som refererar till antal signal- eller symboländringar per sekund[13]

Det finns två typer av seriell kommunikation: synkron och asynkron. Vid synkron kommunikation synkar sändare och mottagare sig vid start och fortsätter att skicka synkningsbitar under hela förloppet. När ingen data sänds så skickas ett konstant flöde av ”idle” bitar som låter båda enheterna veta var den andra är.
Synkronisk kommunikation tillåter en snabbare dataöverföring tack vare att de extra start- och stoppbitarna kan utelämnas.

Asynkronisk dataöverföring behöver en start och stoppbit då den ej sänder kontinuerligt för att kunna identifiera att data kommer. Detta leder till den lägre överföringshastigheten, dock har det en fördel i att processorn ej behöver behandla dom extra idle-bitarna. Den asynkroniska kommunikationen har i idle-läget istället en konstant 1’a som då indikerar att ingen data sänds. I detta fall sänds startbiten i form av en nolla som gör att den även känner av om kontakten skulle brytas[8].

2.3 Mikrokontroller

En mikrokontroller är en integrerad krets som innehåller en CPU samt diverse kring-utrustning så som minnesenheter, A/D-omvandlare, timers, GPIO-pinnar samt olika kommunikationsinterface. En mikrokontroller används ofta som styrenhet i inbyggda system för att styra de olika delarna av systemet. Ofta programmeras mikrokontrollern i C, eller i Assembler på en dator, där koden sedan kompileras och en maskinkod genereras. Denna laddas sedan in i mikrokontrollern med hjälp av en extern programmerare.

Mikrokontroller finns i många olika pris- och prestandaklasser men generellt är dom billiga då dom tillverkas i stora serier. De är också mycket strömsnåla och passar därför bra i små, batteridrivna system [9].

2.4 Matlab

Matlab är en programvara som använder sig av ett språk som är väl anpassad till forskare och ingenjörer där matematiska och tekniska beräkningar utföras med hjälp av matrisalgebra. Den inbyggda grafiken gör även lätt att visualisera och få bra överblick över datan och grafer. Dokumentationen till Matlab är även den anpassad mot ingenjörer och forskare vilket medför att man inte behöver vara fördjupad i programmering för att förstå programspråk eller instruktioner. Det finns även ett stort bibliotek med professionellt utvecklade toolboxes som är ordentligt testade och med dokumenterad funktionalitet.

Dessa toolboxes tillhandahåller färdiga funktioner för olika tillämpningar som t.ex., signalbehandling, bildbehandling, statistik, optimering, maskininlärning och reglertekniska beräkningar [10].

2.5 Simulink

Simulink är ett simulations och modell-designprogram som jobbar i blockdiagrammiljö. Det stödjer bl.a. simulation, automatisk kodgenerering och realtids simulering av

inbyggda system. Simulink har även ett grafiskt gränssnitt för uppbyggnad och underhåll av blockdiagram samt graffönster och datadisplayer för visning av simuleringsresultat. Simulink är också integrerat med Matlab vilket gör att övergången för vidare analys eller användandet av Matlab-algoritmer blir smidigt [11].

2.6 MPLAB X

MPLAB X IDE är Microchip® egna mjukvaruprogram för utveckling av applikationer för Microchip® mikrokontroller. IDE:n är baserad på NetBeans IDE som är en open source från Oracle. MPLAB X har funktioner som codecompletion, debugging, integrerat versionshantering med mera [9].

3.Metod

Genomgång av de metoder och tankesätt som används.

3.1 Förstudie

Arbetet inleddes med en förstudie som skulle öka kunskapen om hur dom utvalda komponenterna fungerar och hur programvaran behövde utformas för att kommunikationen mellan alla delar skulle fungera. Datablad för utvecklingsplattan Explorer 8 samt för PIC studerades för att få en uppfattning av vilka delar som skulle behövas och hur dessa initieras och ställs in.

3.2 Kommunikation

Dom tillgängliga gränssnitten SPI, I^2C samt USART undersöktes närmare för att i ett senare skede kunna bestämma vilka gränssnitt som skulle användas och hur kommunikationen skulle byggas upp och fungera. Därefter planerades utvecklings-arbetet av programvaran för att inhämta samt analysera data från accelerometern.

För att kunna kommunicera mellan alla delar i systemet var kommunikationen viktig och prioriterades, mellan accelerometern och PIC valdes protokollet SPI eftersom endast en slave och master användes samt att överföringshastigheten var en prioritet då mycket datainformation skulle behandlas.

För att kommunikationen med SPI skulle fungera var initieringen av accelerometern viktig och den gjordes med hjälp av databladet för ADXL345 där all information om timing och register finns. För att kunna ändra i registerna och programmera PIC användes programmet MPLAB samt den tillhörande programmeraren ’PICK IT™ 3’.
När kommunikationen mellan accelerometern och mikrokontrollern var klar var det dags att koppla upp mikrokontrollern mot PC så att testdata kunde överföras och analyseras i Matlab. Detta gjordes via USART Micro-USB och Simulink.

3.3 Testscenarion och Analysen

När all kommunikation var klar konstruerades testerna som skulle användas, dessa gjordes med tanken att fysiskt påverka accelerometern på fem specifika sätt som skulle vara repeterbara och då ge liknande utfall. Testkurvor som skulle analyseras och bearbetas inhämtades och sparades i Matlab via Simulink. Alla analyser och tester gjordes i Matlab. Denna del är den som krävt mest tid eftersom det inte var självklart vilken analysmetod som passade för att skilja på de olika scenarion som accelerometern utsattes för.

Under projektets gång testades en hel del metoder för att hitta det som gjorde varje signal unik, tester med olika filter, frekvensanalys och algoritmer för mönster-igenkänning. Tester samt resultat dokumenterades och utvärderades kontinuerligt för att bestämma vad nästa steg skulle vara för att gå vidare med arbetet.

I takt med att arbetet fortskred och fler tester samt analyser utfördes, sågs indikationer på vilka begränsningar systemet hade och vilka ytterligare avgränsningar som

behövde göras för att säkerställa algoritmens funktion samt att den var inom gränserna för vad PIC klarade av. Detta medförde att algoritmen inte kunde vara för komplex då beräkningskraften samt minnet hos processorn är begränsat.

Efter att ha testat alla scenarion, A-E, med frekvensspektraanalys, amplitudanalys och peakanalys togs beslutet att algoritmen skulle baseras på amplitudanalys för att skilja på de bestämda scenarierna. Den färdiga algoritmen testades genom att köra 125 testsignaler. Algoritmen kunde skilja på scenario D-E i 100 % av fallen, scenario A, B och C överlappade varandras amplitudintervall vilket ledde till att bedömningarna blev mer slumpmässiga. Det uppmärksammades att scenario A var den som var inom både scenario B och C amplitudmedel och gjorde dem svåra att skilja på. Testet gjordes igen med skillnaden att denna gång kördes det utan scenario A och det gav ett resultat på 100 % rätta val av algoritmen. Resultatet visade att amplitudanalys var en bra metod för att kunna hitta olikheter i dom tester som gjorts, detta trots att scenario A var svår att bestämma.

3.4 Algoritmen

Algoritmen fungerade så att den tog ett medel av alla testkurvor som inhämtats på varje scenario och använde detta som en mall. Denna mall användes sedan i algoritmen där den jämfördes med dom enskilda slumpmässigt utvalda testkurvor för att bedöma vilket av de fem scenarion som testats. I två av fallen var scenarierna väldig lika och gick ej att skilja med bara amplitudanalysmetoden, så i ena fallet hittades en olikhet i toppvärdena på signalerna så det användes en extra algoritm som kollade om topparna var över eller under noll det vill säga positiva eller negativa. I det andra fallet hittades en skillnad i avstånd på toppar så en avståndstest mellan topparna infördes i algoritmen

4 Genomförande

I följande kapitel beskrivs genomförandet av projektet mer ingående. Här redogörs för hur vi tänkt med dom olika testerna och analyserna som genomförts samt vilka val vi gjort och motiveringar till detta. Eventuella problem som stötts på kommer också beskrivas och även hur vi löst dessa.

4.1 Mikrokontroller

Mikrokontrollern är systemets styrenhet och det är den som sköter insamling av data från accelerometern och skickar sedan vidare denna till datorn för att kunna analyseras i Matlab. Därför startar arbetet med utveckling av mjukvara som sköter kommunikationen mellan systemets olika delar.

4.1.1 Kommunikation mellan mikrokontroller och accelerometer

Den accelerometer som har valts (ADXL345) i detta projekt stödjer två olika kommunikationsprotokoll. Dessa är I^2C samt SPI. Till följd av att SPI stödjer högre

överföringshastighet samt dess simplicitet jämfört med I^2C (se kapitel 2.1.3 SPI/ I2C) valdes SPI.

Innan accelerometerdata kan börja läsas från sensorn måste den konfigureras med rätt inställningar, dessa redovisas i tabell 3. För att få ut så mycket information som möjligt valdes den högsta samplingshastighet som accelerometern stödjer. Vi har även valt den högsta upplösningen, 13 bitar, samt det största möjliga mätområdet.

Inställning	Vald konfiguration
Samplingshastighet	3200 Hz
Mätområde	+-16 g
Upplösning	13 bitar
SPI typ	4 ledningar

Tabell 3 Inställningar SPI

Accelerometern har även den funktionen att den kan ställas in så att en signal, kallad Data Ready, skickas så fort ett nytt mätvärde finns tillgängligt att läsas. Denna kopplas till en ingång på mikrokontrollern och programmet påbörjar en läsning av accelerometerdata när denna signal aktiveras. Läsningen fungerar så att mikrokontrollern skickar ett 8-bitars meddelande som innehåller instruktioner om att en läsning skall göras samt vilka register som skall läsas. I detta fall skall samtliga 6 register som innehåller accelerometerdatan för x, y och z axeln läsas av.

4.1.2 Kommunikation mellan mikrokontroller och PC

Kommunikationen mellan mikrokontrollern och PC sker med USART. USART enheten i mikrokontrollern har många inställningsmöjligheter för olika överföringshastigheter, avbrott vid olika testscenarier mm. Dom inställningar vi valt visas i tabell 4. Även i detta fall valdes den högsta överföringshastigheten.

Inställning	Vald konfiguration
Baud rate	115200
Antal databitar	8
Antal stoppbitar	1
Paritetsbit	nej
Synkron/asynkron	asynkron

Tabell 4 Inställningar USART

Den seriella kommunikationen är asynkron vilket medför att hastigheten med vilken överföringen skall ske måste vara bestämt i förväg. Datan skickas från mikrokontrollen en byte i taget och kommunikationen avslutas med ett förutbestämt stopptecken som är 1 byte. Totalt skickas det för varje sampel som tas i accelerometern sju byte. Två byte för varje accelerometeraxel samt en byte för stoppteckenet som markerar slutet på överföringen. Tabell 5 visar vilken data som skickas för varje sampel.

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
x-axel, låg byte	x-axel, hög byte	y-axel, låg byte	y-axel, hög byte	z-axel, låg byte	z-axel, hög byte	Stopptecken

Tabell 5 Ordningen som datan skickas i vid en överföring

4.2 Insamling av mätdata

Från accelerometern kan värden för tre axlar x, y och z läsas av. I början fokuserades det på z-axeln för att överföringshastigheten och kommunikationen ej hade optimerats tillräckligt för att kunna ta hand om den extra datamängden som 2 axlar till krävde. Att bara använda z-axelns data gav tillräckligt med information om scenarierna för att kunna hitta skillnader i det flesta fall. I slutskedet av projektet hade kommunikationen optimerats tillräckligt för att möjliggöra tester med tre axlar. Att använda tre axlar ger information om scenarierna från två axlar till vilket kan vara det som behövs för att skilja på två scenarion som är väldigt lika i z-led, men skiljer sig i x- eller y-led. Detta gäller både amplitud och frekvens.

Dom fem insamlade testscenarierna användes inte bara till analys utan också till att skapa referensmallar som sedan användes i algoritmen, detta gjordes genom att ett medelvärde av amplituden togs på alla dom tjugofem testerna tillhörande varje scenario. Medelvärdena på varje tests användes sedan till att hitta inom vilken amplitud varje scenario ligger det vill säga max och min för varje testscenario, detta användes sedan i algoritmen som referensmall för att se inom vilken amplitud scenariot ligger.

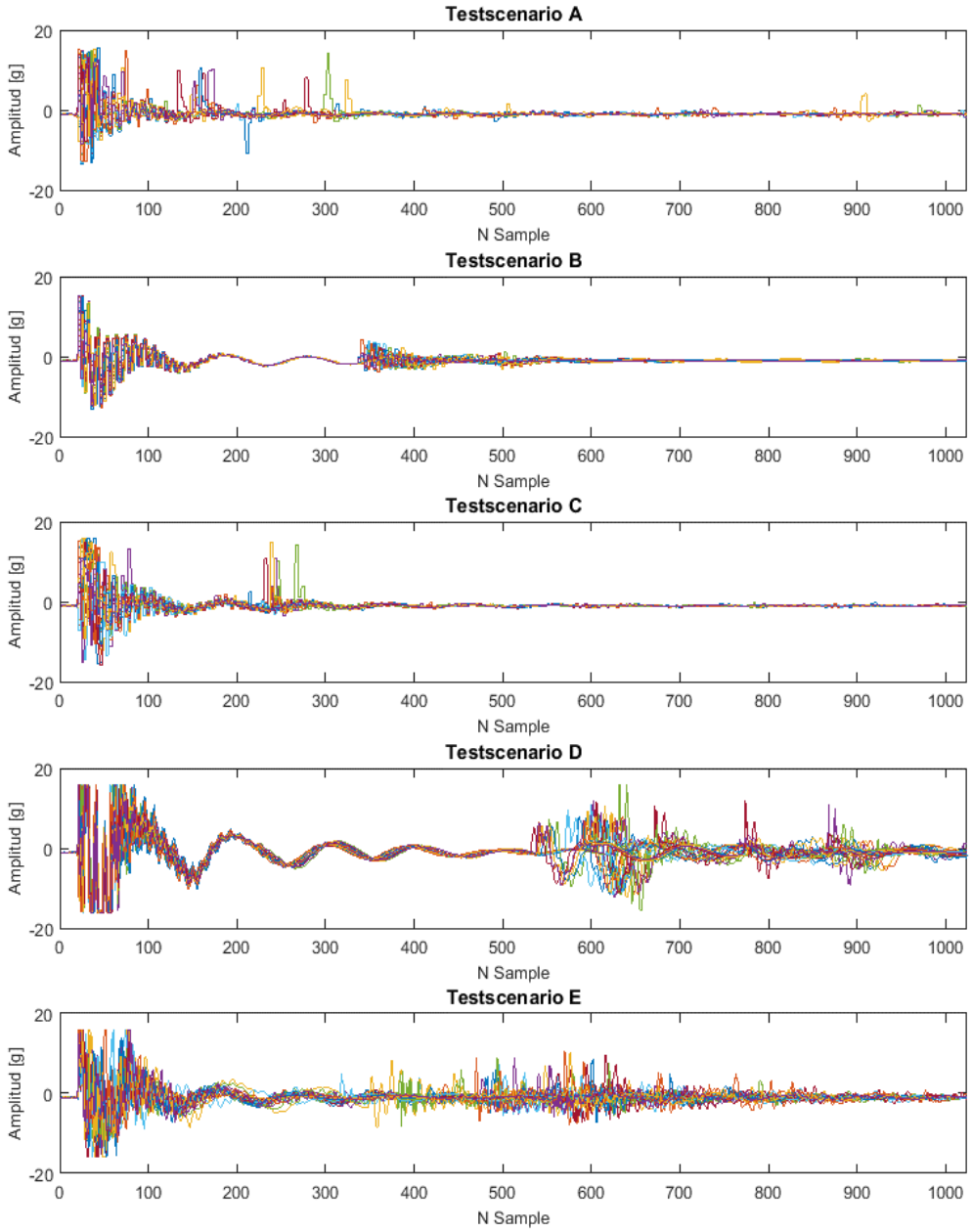
Ett test motsvarar en körning av ett testscenario, och detta gjordes tjugofem gånger per scenario. Värdena som accelerometern gav var i form av en vektor. Dessa värden från de fem olika dynamiska scenarion som accelerometern utsattes för samlades in och sparades. Varje testscenario upprepades mätningen 25 gånger. Den insamlade signal-informationen som nu var i formen av en 2800X25 matris vilket med alla sampels och de 25 testernas resultat, överfördes till Matlab för analys.

4.3 Analys

För att läsa datan som skickas från mikrokontrollern används ett mycket simpelt program i simulink. Detta består endast av två block. Ett block för seriell kommunikation där inställningar görs, exempelvis vilken baud rate som skall användas, vilket format datan har och om det används stopptecken. Det andra blocket överför testdatan till matlab. När väl datan lagrats i matlab, påbörjades analysen av testdatan.

4.3.1 Visuellt

Analysen inleddes med att plotta signalerna för att visuellt jämföra dem och undersöka om det fanns något uppenbart mönster som skulle kunna hjälpa oss vidare. För varje scenario plottades samtliga 25 tester i samma figur för att snabbt kunna se om tydliga mönster förekommer. Som man kan se i Figur 6, finna ett förhållandevis tydligt mönster som skiljer sig mellan scenarierna. Dock förekommer oregelbundna amplitudspikar som bidrar till att signalerna blir svårare att särskilja.

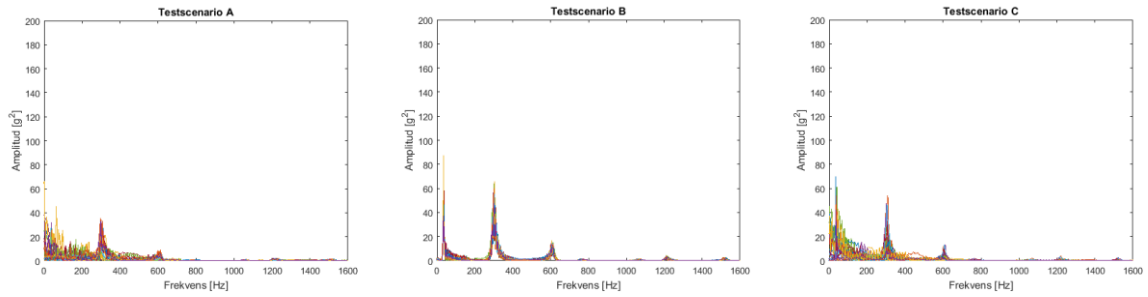


Figur 6 Samtliga 25 tester för varje scenario plottad i respektive graf.

4.3.2 Frekvensanalys

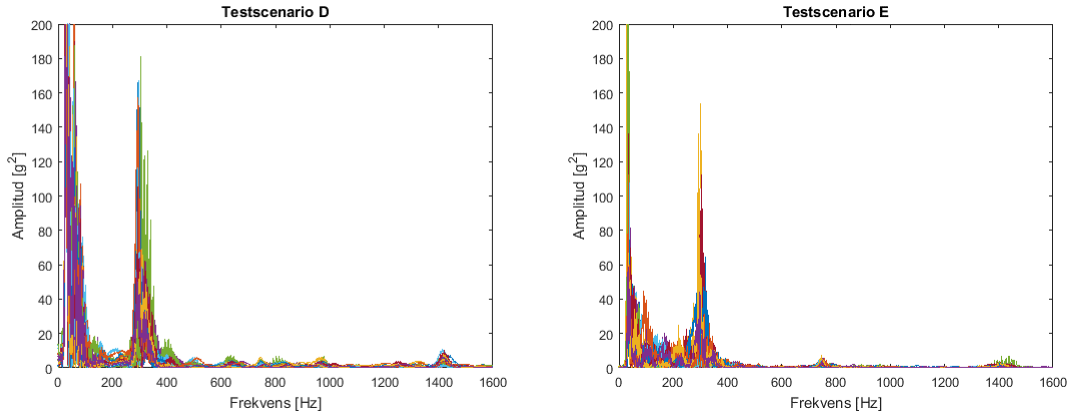
För att utreda vad dom olika signalerna har gemensamt frekvensmässigt och vad som skiljer dem åt analyseras deras frekvensinnehåll. Vid analys av testdatans frekvensinnehåll används FFT (Fast Fourier Transform) för att få fram signalernas frekvenskomponenter. Detta talar om hur mycket av en viss frekvens som en signal innehåller. Tanken är att skilja dom olika situationerna åt med hjälp av vilka frekvenser dessa ger upphov till. Om det visar sig att signalerna har en stor del frekvenser gemensamt och samtidigt har något eller några andra frekvensområden som dominerar beroende på vilken situation som avses, skulle man då i teorin kunna filtrera bort dom gemensamma frekvenskomponenterna och få kvar endast det som skiljer sig mellan signalerna. Tanken är då att dom olika scenarierna skall producera olika frekvensinnehåll i accelerometerdatan.

I Matlab finns en inbyggd funktion för FFT. Denna används för att ta fram ett frekvensspektrum för varje signal. Först plottas spektrumet för alla 25 testkurvor för varje scenario A till E i var sin figur för att snabbt se om tydliga mönster förekommer. Direkt observeras att spektrumet för de 25 olika testerna på testscenario A, B samt C innehåller till stor del samma frekvenser, runt 30, 300 samt 600 Hz, och liknande mängd. Dock syns det att i fallet A samt C finns det en hel del andra frekvenser också och dom har inte lika renodlat mönster bortsett från att topparna ligger vid samma punkter.



Figur 7 Frekvensspektrum för tre av testscenariorna

Om man istället tittar på testscenario D och E i figur 8 ser man att dessa har betydligt högre amplitud på topparna i förhållande till ”bruset” emellan dem. Baserat på detta var en teori att topparna vid 30 samt 300 Hz skulle filtreras ut för att sedan skilja signalerna åt amplitudmässigt. Detta visade sig oanvändbart eftersom att när man tittar på spektrumet för var och en av dom 25 testerna separat så ser man att toppvärdet varierar mycket och endast en handfull påvisar den höga amplituden i figur 8. Det vill säga att amplituden inte var tillräckligt enhetlig för att skilja signalerna med.



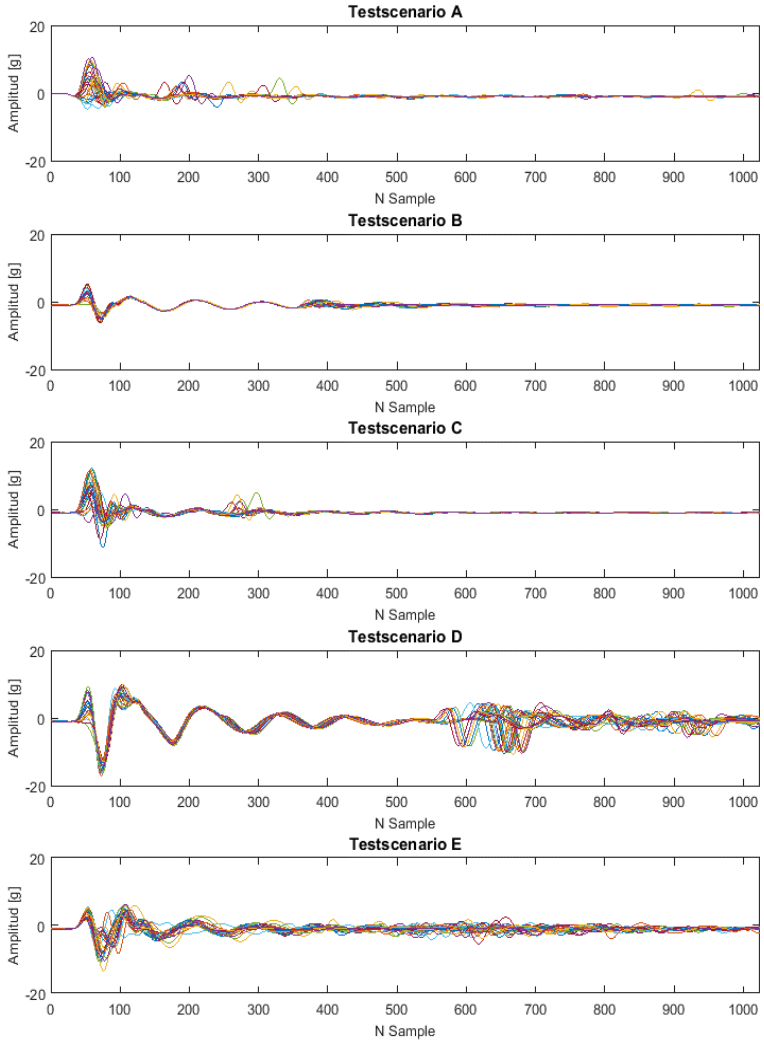
Figur 8 Amplitudspektrum av testscenario D samt E

4.3.3 Test av filter

Då det verkade finnas en del skillnader i frekvensinnehållet görs ett antal försök att filtrera bort frekvenskomponenterna som är lika för att få fram olikheter. Skulle det vara så att man kan skilja på vissa frekvenser borde dessa specifika frekvenser kunna isoleras genom att filtrera bort oönskade frekvenser.

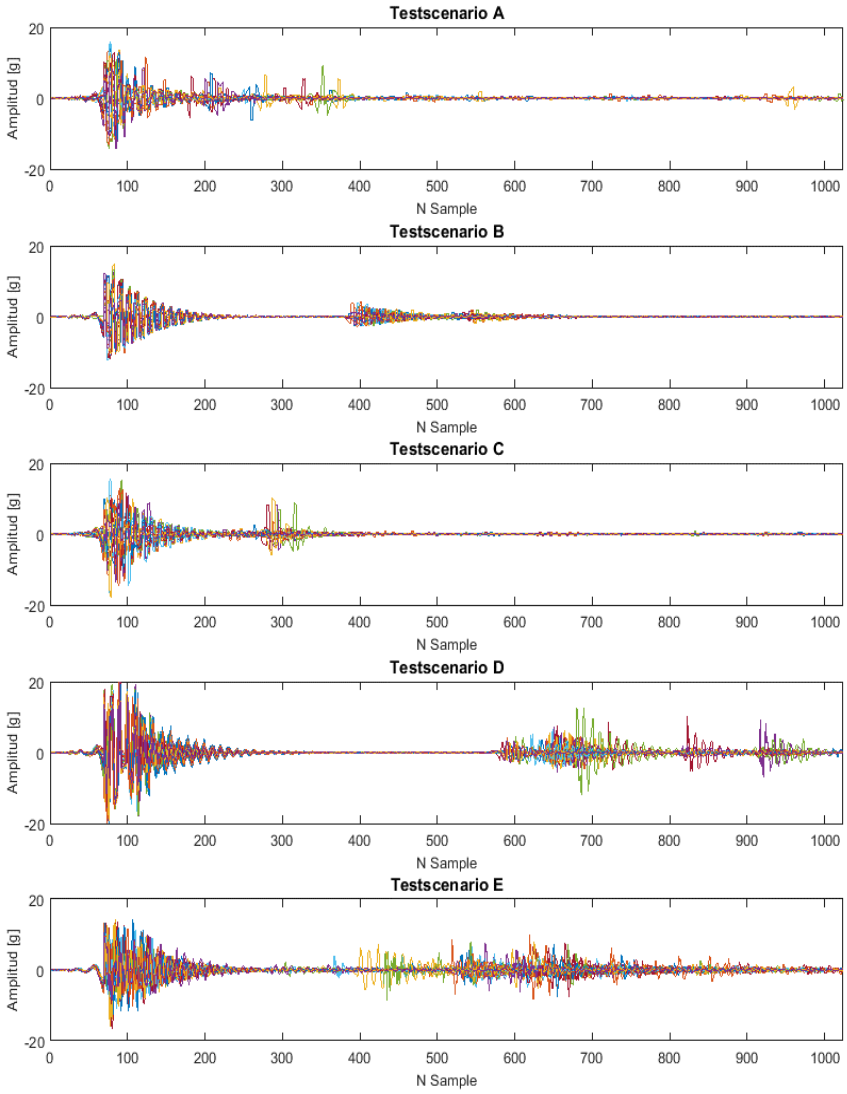
Varje signal verkade innehålla en viss svängning som såg ut att ligga runt en låg frekvens. Detta kan ses i frekvensspektrat för signalerna, se Figur 7 och 8, i form av en tydlig topp vid 30 Hz. Toppens amplitud verkar också variera mellan dom olika testscenariorna och på grund av detta görs ett försök att isolera denna svängning för att se om det går att avläsa amplituden för att särskilja signalerna.

Dessvärre verkar den lågfrekventa svängningen uppstå oberoende av vilken testscenario som inträffat. Visserligen avtar den på något olika lång tid men inte tillräckligt regelbundet för att användas i vår algoritm. Detta kan ses i Figur 9 där dom olika kurvorna inte följer samma mönster.



Figur 9 Plott efter lågpassfiltrering med gränsfrekvens på 100 Hz

Vid högpassfiltrering vid 170 Hz sågs en mer användbar egenskap hos signalerna nämligen att det uppvisas ett tydligt tidsavstånd mellan två områden av aktivitet hos testscenario B-D. Testscenario A och E är dessvärre för oregelbundna för att kunna mäta det avståndet då det har höga amplitudtoppar på mycket olika avstånd. Detta visas i figur 10. Hur denna egenskap skulle kunna användas i detta fall kommer redogöras för i nästkommande kapitel *4.3.4 Peakanalys*.

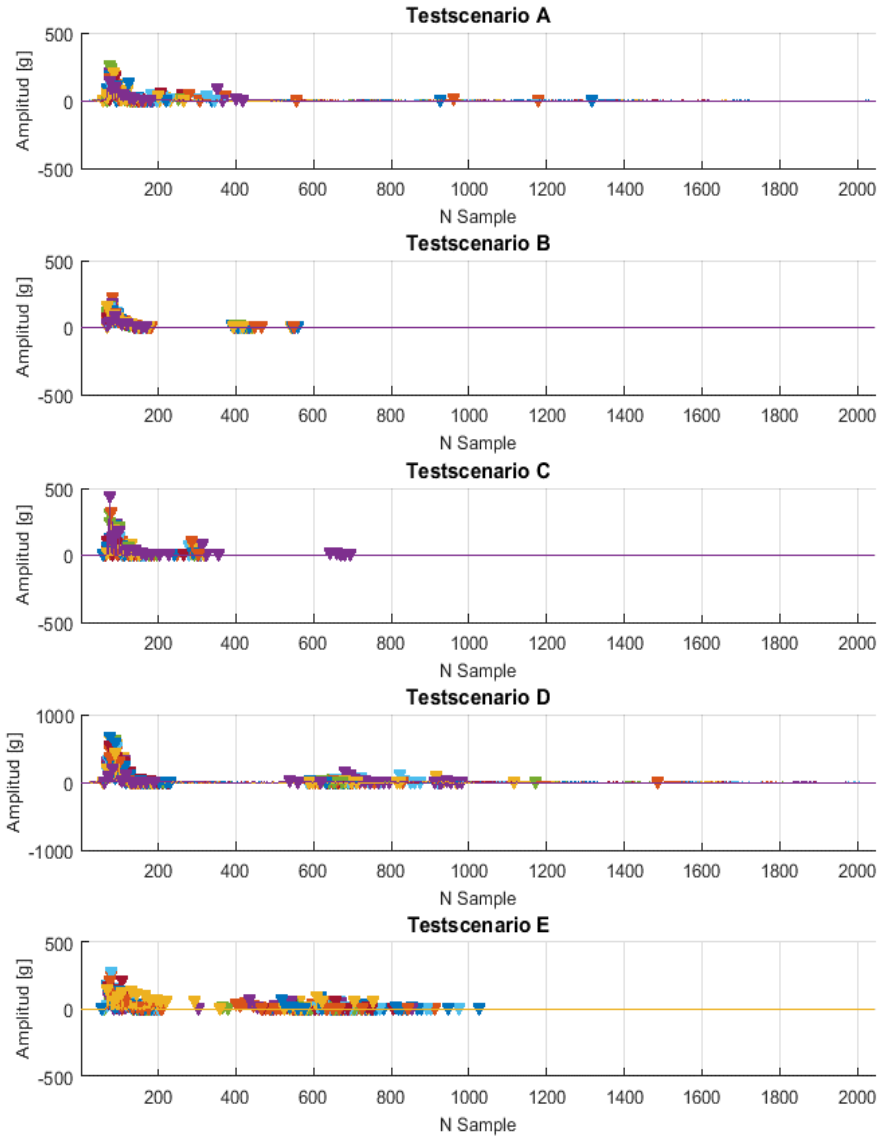


Figur 10 Signalerna efter högpassfiltrering vid 170 Hz

4.3.4 Peakanalys

Vid den visuella analysen observerades det att det fanns ett tydligt avstånd mellan två områden av sensoraktivitet i majoriteten av testfallen. Detta avstånd varierade beroende på vilken testscenario om undersöks, därav var en teori att mäta avståndet mellan områden med aktivitet.

För att undersöka detta utvecklas ett program som först högpassfiltrerar vid 170 Hz. Därefter används Matlabs inbyggda funktion *findpeaks* för att se om konceptet fungerar. Denna funktion hittar toppar i signal och det går att ställa in om det skall vara ett minsta avstånd mellan topparna eller vilken amplitud dom måste ha. Det valdes att topparna skall ha en minsta skillnad från bredvidliggande värden på 1.8. Som man ser i figur 11 så fungerar det i många fall för testscenario B samt C, dock är det några tester som har toppar som hamnar innanför ett annan testscenarios område. Troligtvis hade man kunnat komma fram till ett bättre resultat med fler tester och bättre filtrering.



Figur 11 Plot av toppar

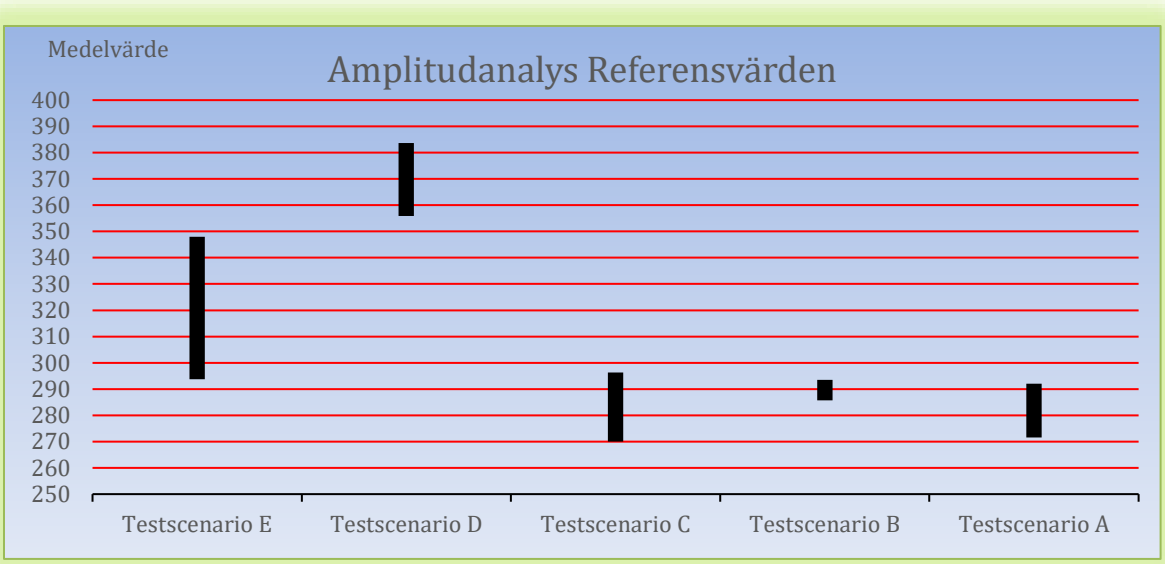
4.3.5 Amplitudanalys

Amplitudanalys är som det låter en analys som använder sig av amplituden för att se om det finns några skillnader. Amplitudanalysen gjordes genom att ta alla 25 tester från testscenario A och ta ett medelvärde på dem, detta ger medel amplituden på testscenario A. Detta gjordes med alla testscenarior A till E, resultatet användes sedan som referensmallar vid jämförande och identifiering i algoritmen. Funktionen som användes var en medelberäkningsfunktion med denna metod som grund i algoritmen kunde separationerna av dem olika testscenariorna börja.

4.4 Algoritm

Algoritmen byggdes upp enligt flödesschemat som finns i bilaga A, och med hjälp av den information som alla analyser som beskrivits och med den begränsningen att rymmas och fungera i PIC. Den skrevs i Matlab och började med att testscenarion som skulle testas, som är i formen av en vektor med ungefär 2800 sampels sparades i 2 variabler. Den första variabeln som heter signal, fick vektorn som den var och den andra variabeln benämnd signalmedel, fick medelvärdet på vektorn. Signalmedel testades mot referensmallarna som delats upp för att bestämma vilken testscenario som testas se kapitel 4.2. I två fall var amplituden inte tillräcklig för att skilja dem åt så i de fallen användes funktioner som i ena fallet utnyttjade att ena testscenarions toppar inte gick över noll utan var negativ den funktionen kallar vi MaxPeak amplitud, i det andra fallet hittades en skillnad i avstånd mellan toppar som beskrevs i kapitel 4.3.3 peakanalys den funktionen kallades Peak distance, se flödesdiagram i bilaga A. Efter att ha testkört algoritmen med alla testfall, klarade algoritmen att placera alla fall med testscenario D och E rätt. Testscenarior A, B och C gick inte att skilja på ett sätt som gav konsekventa resultat, Testscenario A låg hela tiden inom samma amplitud som testscenario B och testscenario C som figur 12 nedan visar, kravet för att det ska gå att skilja dom åt är att dom inte överlappar amplitudområde eller att dom som överlappar går att skilja via någon av dom andra metoder som används, detta klarade inte våran algoritm att göra i fallet testscenario A. Testet gjordes om med alla testscenarior utom A och detta resulterade i att algoritmen kunde skilja på alla dom kvarvarande testscenariorna till 100 %.

Algoritmen lyckades inte lösa alla testscenarior i det utförande som den var i, utan det skulle behövts mer tid, så exempelvis tre axlar kunde användas från accelerometern. Skillnaden att använda tre axlar är att två axlar till ger nya möjligheter till att läsa av testscenariorna och hur dom påverkar accelerometern, som i sin tur kan skilja sig i dom 2 axlarna där det nu inte är möjligt att se någon skillnad.



Figur 12 Diagram över vilka områden medelvärdet sträcker sig över för dom olika testscenariorna

5. Resultat

Verifieringen av algoritmen med hjälp av testscenariot visade att algoritmen klarade av att hitta dom olika scenarion som används med ett undantag. Scenario A gick ej att särskilja tillräckligt bra med någon av dom metoder vi använt. Detta testscenario låg inne i två av dom andra testscenariornas amplitudområde och det kunde inte heller hittas nått unikt för scenario A, med dom andra metoderna som används. Tester med tre axlar på accelerometern han göras men inte fullt ut, dock sågs en stor fördel med användandet av fler axlar, då accelerometern nu har två axlar till med information om vad som hänt, och inte bara två axlar till utan 2 axlar som känner av accelerationen i två riktningar till. Denna information kan innehålla dom skillnader som behövdes och inte hittas vid användandet av bara en axel mellan testscenariorna i fråga.

Uppsatta mål och om vi anser oss lyckats nå dessa:

- *Analysera och hitta skillnader i den information som accelerometern skickat utifrån det testscenario som används:* Skillnader och information har algoritmen lyckats med att hitta och skilja på.

- *Ta fram en algoritm som kan skilja på dom fem fasta testscenarior som accelerometern utsatts för:* Algoritmen som vi utvecklat fungerade inte i alla fallen. Med detta menar vi att den var verkningslös i fall A, men fungerade i fallen B till E om inte A användes. För att helt lyckas måste vidare utveckling göras.

- *Utforma algoritmen så att den går att använda i mikrokontrollern PIC16F1719:* Det går principiellt att utföra men praktiska tester har inte gjorts.

6. Slutsats och diskussion

Syftet med projektet vara att lära oss mer om accelerometerns möjligheter och sedan använda dessa kunskaper och utföra ett testscenario.

Målet med testscenariot var:

- Skapa en algoritm som kan bestämma vilket testscenario som påverkat accelerometern.

- Algoritmen skall skilja på alla fem testscenarion

- Utreda om accelerometerns känslighet och precision är tillräcklig för denna tillämpning.

Algoritmen som skapades använde väldigt enkla metoder för att skilja på de olika testscenariorna. På grund av begränsad beräkningskraft och minneskapacitet hos den valda mikrokontrollern begränsades möjligheten att använda beräkningstunga metoder.

Trots dessa simpla metoder lyckas algoritmen särskilja alla testfallen bortsett från testscenario A. I projektets slutskede uppstod en möjlighet att kort studera mätdata från X- och Y-axel, utöver Z-axeln som vi här använt oss av. Vid ett fåtal extra tester som kunde utföras med inhämtning av data från accelerometerns samtliga axlar, sågs en skillnad mellan dom olika testscenariorna i amplitudnivåerna hos X- och Y-axeln. Dessa skillnader uppstod i flera fall mellan testscenarior som tidigare inte med säkerhet eller inte alls kunnat särskiljas på grund av att deras amplitudområde överlappat. Detta skulle då kunna användas för att med större säkerhet avgöra vilken testscenario som inträffat.

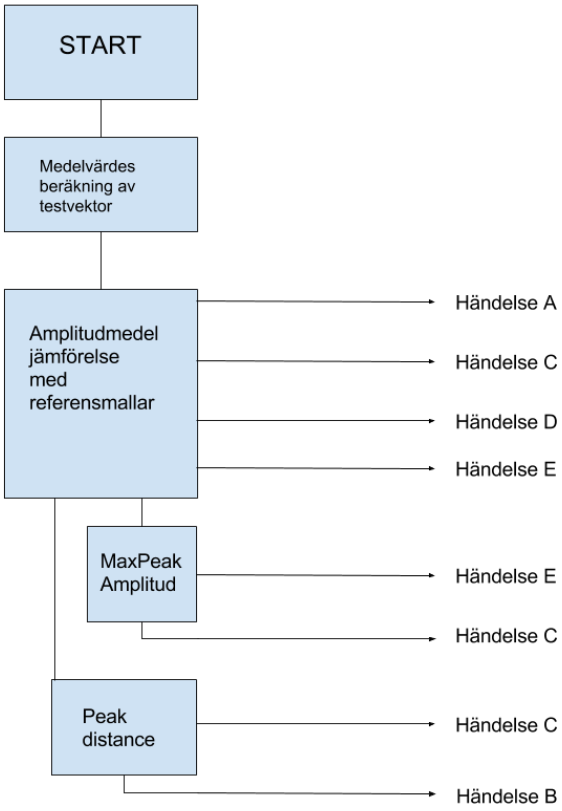
Under projektets gång har en begränsande faktor varit mikrokontrollerns beräknings-kraft och framför allt minnesutrymme. I nuvarande implementation görs för varje nytt sampel, en läsning av data från accelerometern. Datan skickas därefter till PC för att sparas i Matlab, innan nästa läsning av accelerometerdata kan göras. Detta begränsar den mängd data som kan överföras, eftersom kommunikationen mellan mikro-kontrollern och PC:n inte är snabb nog för att överföra mer än två byte, som motsvarar en axels mätdata, vid den valda samplingsfrekvensen, 3200 Hz.

Vid en vidareutveckling av systemet, där accelerometerns samtliga axlar skall användas, samtidigt som en hög samplingsfrekvens önskas, föreslås att en kraftfullare mikrokontroller väljs.

Baserat på de uppnådda resultaten och projektets tidslängd ser vi att systemet har en stark potential för vidareutveckling. Med mer tid till förfogande för utveckling av algoritmen och med en något kraftfullare processor ser vi att det finns goda möjligheter att utveckla en mycket precis metod för att kunna särskilja signaler av denna typ.

Bilaga A Flödesschema

Flödesschema för algoritmen.



Kapitel 4.4 om algoritmen beskriver användandet av flödesschemat och funktionen.

Bilaga B Timingdiagram för SPI

Timingdiagram för SPI kommunikation

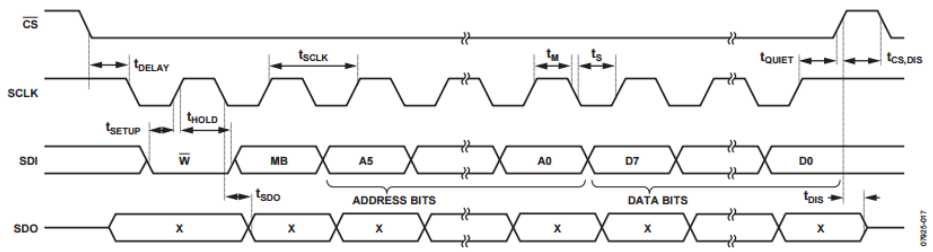


Figure 37. SPI 4-Wire Write

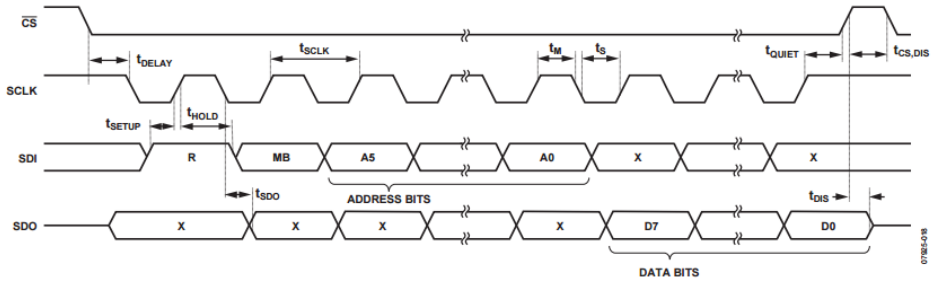


Figure 38. SPI 4-Wire Read

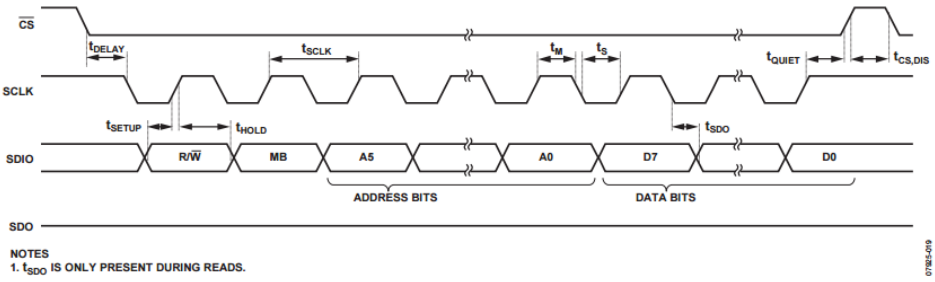


Figure 39. SPI 3-Wire Read/Write

Timing diagrammet används för att kunna läsa och skriva till registerna, detta är viktigt att rätt sak gör vid rätt tillfälle då det inte fungerar annars. mer Kap 2.1.2.

Bilaga C Algoritmen

Algoritmen skriven i Matlab.

```
%få in ett värde.
for k = 5:5     %Väljer vilket test 1 Testscenario A, 2 Testscenario B, 3
Testscenario C,4 testscenario D, 5 Testscenario E.
    for i = 1:25  % Väljer vilken testsignal som körs.

        signal = fixadeMatvarden{k}{i};         % För in test värdet till objekt.
        n = numel(signal);                      % Kollar antal element i matrisen




        signalmedel = mean(abs(signal)) ;            % Tar medelvärdet av
testscenarions amplituder

        if ((signalmedel >= 270) && (signalmedel <= 285.64))||((signalmedel >=
293.5403) && (signalmedel <= 293.8))
            'Testscenario C'
        elseif (signalmedel >= 296.36) && (signalmedel <= 348)

            'Testscenario E'
        elseif (signalmedel >= 355) && (signalmedel <= 383)
            'Testscenario D'
            %     samplespace(obj);
        elseif (signalmedel >= 293.8) && (signalmedel <= 296.36)
            sig1 = signal(545:900,:);
            sigmax = max(sig1);

           if sigmax > 0
               'Testscenario E'
           else
               'Testscenario C'
           end
        elseif (signalmedel >= 285.64) && (signalmedel <= 293.54)
            sig = envelope(bpFilter(double(signal)));
            sig( sig < 400 ) = 0;

            [pks locs] =
findpeaks(sig,'MinPeakProminence',200,'MinPeakDistance',280);
           if numel(locs) > 1
               if(diff(locs)<380)
                   findpeaks(sig,'MinPeakProminence',200,'MinPeakDistance',280)
                   'Testscenario B'
               else
                   'Testscenario C'
               end

           end


        end
```

```
    end
end
```

Referenser

Referenser anges enligt nedan.

[1] “ADXL377 MEMS Accelerometer in INDYCAR Race Car Safety System | Analog Devices.” [Online]. Available: http://www.analog.com/en/education/education-library/videos/1662074678001.html. [Accessed: 07-Jun-2017].

[2] “Teknisk mätning av hand-arm vibrationer.”

[3] Smith Dave, “How Does An Accelerometer Work In A Smartphone? Bill Hammack, The Engineer Guy, Explains [FULL TEXT] | International Business Times,” 2012. [Online]. Available: http://www.ibtimes.com/how-does-accelerometer-work-smartphone-bill-hammack-engineer-guy-explains-full-text-699762. [Accessed: 07-Jun-2017].

[4] “ADXL345* PRODUCT PAGE QUICK LINKS COMPARABLE PARTS SOFTWARE AND SYSTEMS REQUIREMENTS SAMPLE AND BUY.”

[5] F. Leens, “An introduction to I2C and SPI protocols,” *IEEE Instrum. Meas. Mag.*, vol. 12, no. 1, pp. 8–13, Feb. 2009.

[6] “Introduction to I^2C and SPI protocols – Byte Paradigm – Speed up embedded system verification.” [Online]. Available: http://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/. [Accessed: 02-Jun-2017].

[7] “Explorer 8 Development Board | Microchip Technology Inc.” [Online]. Available: http://www.microchip.com/design-centers/8-bit/development-boards/explorer-8-development-board. [Accessed: 06-Jun-2017].

[8] “What is RS232 and Serial Communications? | TALtech.” [Online]. Available: http://www.taltech.com/datacollection/articles/serial_intro. [Accessed: 16-May-2017].

[9] “MPLAB- X IDE | Microchip Technology Inc.” [Online]. Available: http://www.microchip.com/mplab/mplab-x-ide. [Accessed: 24-May-2017].

[10] “Why MATLAB - MathWorks - MATLAB.” [Online]. Available: https://se.mathworks.com/products/matlab/why-matlab.html?s_tid=hp_brand_whymatlab. [Accessed: 24-May-2017].

[11] “Simulink - Simulation and Model-Based Design.” [Online]. Available: https://se.mathworks.com/products/simulink.html. [Accessed: 24-May-2017].

[12] “Advanced Piezoelectric Materials - Science and Technology - 1.1 The History of Piezoelectrics - Knovel.” [Online]. Available: https://app.knovel.com/web/view/swf/show.v/rcid:kpAPMST002/cid:kt0094K1F3/viewerType:pdf/root_slug:advanced-piezoelectric?cid=kt0094K1F3&page=1&b-toc-cid=kpAPMST002&b-toc-root-slug=advanced-piezoelectric&b-toc-url-slug=development-piezoelectric&b-toc-titl. [Accessed: 17-May-2017].

[13] “What’s The Difference Between Bit Rate And Baud Rate?” [Online]. Available: http://www.electronicdesign.com/communications/what-s-difference-between-bit-rate-and-baud-rate. [Accessed: 17-May-2017].