CHALMERS
UNIVERSITY OF TECHNOLOGY

Portabel och trådlös skåpssensor

Examensarbete inom högskoleingenjörsprogrammet Elektroteknik

NASIM SALEHINEJAD

Institutionen för Elektroteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2020

EXAMENSARBETE 2020

Portabel och trådlös skåpssensor

Nasim Salehinejad

CHALMERS
UNIVERSITY OF TECHNOLOGY

Institutionen för Elektroteknik
Chalmers Tekniska Högskola
Göteborg, Sverige 2020

Portabel och trådlös skåpssensor
Nasim Salehinejad

Handledare: Björn Bergholm, Broccoli Engineering AB
Examinator: Bertil Thomas, Institutionen för Elektroteknik

Institutionen för Elektroteknik
Chalmers tekniska högskola
SE-412 96 Göteborg, Sverige
Telefon: +46-(0)31 772 10 00

Omslagsbild: Bild tagen på prototyp

Typeset in LaTeX
Printed by Chalmers Reproservice
Göteborg, Sverige 2020

Förord

Examensarbetet har genomförts under våren 2020 på Chalmers tekniska högskola och omfattar 15 högskolepoäng. Arbetet är ett avslutande moment på högskoleingenjörsprogrammet Elektroteknik omfattande 180 högskolepoäng och utfördes i samarbete med Broccoli Engineering AB.

Jag vill tacka min sambo Carl-Henrik som har stöttat och hjälpt mig under detta examensarbete och hela utbildningen, likaså min familj som alltid stöttat och motiverat mig i allt jag gör.

Jag vill tacka min handledare Björn Bergholm på Broccoli för möjligheten att få utföra detta examensarbete. Jag vill dessutom tacka min handledare och examinator Bertil Thomas på Chalmers som har gett stöd och vägledning under detta examensarbete.

Nasim Salehinejad, Göteborg, Juni 2020

Portabel och trådlös skåpssensor
Nasim Salehinejad
Institutionen för Elektroteknik
Chalmers Tekniska Högskola

Sammanfattning

Det finns många produkter ute i samhället idag som kan detektera huruvida en dörr är öppen eller stängd, dock är de oftast beroende av att ha flera fästpunkter för upptäckt av eventuell rörelse. Som det ser ut idag, finns det inte en generisk och portabel produkt som endast behöver en fästpunkt.

Syftet med detta arbete var att ta fram en portabel och trådlös skåpssensor. Den behöver endast en fästpunkt och ska kunna detektera när en yta (dörr/lock) har varit i rörelse samt räkna upp antalet gånger ytan använts. Projektet genomfördes i samarbete med företaget Broccoli Engineering AB.

I arbetet har en mikrokontroller (Arduino Nano 33 BLE sense) med inbyggd IMU (Inertial Measurement Unit), en kopplingsplatta, en powerbank och ett 9 V batteri använts. De olika sätt en lucka/dörr kan öppnas och stängas på (horisontal- eller vertikalled) utfördes först på en liten låda där Arduinokortet var kopplat till dator via USB-kabel. Sedan övergick testningen till att fästa skåpssensorn på dörrar, skåp och luckor med powerbank eller 9 V batteri.

En prototyp av skåpssensorn har tagits fram och fungerar i många fall som avsett. Den är trådlös och relativt enkel och liten. Skåpssensorn kan detektera rörelser på ytor såsom dörrar/lock/skåp som är fästa med gångjärn. Arduinokortet kan räkna upp, registrera och behålla värden.

Nyckelord: Arduino, IMU, accelerometer, gyroskop, magnetometer.

Portable och wireless cabinet sensor
Nasim Salehinejad
Department of Electrical Engineering
Chalmers University of Technology

Abstract

There are a lot of products on the market today that can detect if a door is open or closed, but they usually depend on having several points of attachment for detection of possible movement. As it seems today, there is no generic and portable product which needs only one point of attachment.

The purpose of this project is to develop a portable and wireless cabinet sensor. It should only need one point of attachment and should be able to detect when a surface (door/lid) has been in movement and also be able to count the number of times the area/surface has been open. The project was performed in collaboration with the company Broccoli Engineering AB.

In this work, a microcontroller (Arduino Nano 33 BLE Sense) with a built in IMU (Inertial Measurement Unit) and also a breadboard, a power bank and a 9 V battery has been used. There are different ways doors and lids can be opened and shut (horisontal or vertical). The detection of opening and shutting a door was first tested on a little box where the Arduinocard was attached to a computer via USB-cable. The tests with the cabinet sensor were later carried out on doors and lids powered by either a power bank or a 9 V battery.

A prototype of the cabinet sensor is developed and works in most cases. It is wireless and rather simple and small. The cabinet sensor is able to detect movements on surfaces like doors/lids/cabinets with hinge. The Arduinocard can count, register and keep previous values.

Keywords: Arduino, IMU, accelerometer, gyroscope, magnetometer.

Innehåll

1 Introduktion **1**
- 1.1 Bakgrund . . . 1
- 1.2 Syfte . . . 1
- 1.3 Avgränsningar . . . 1
- 1.4 Precisering av frågeställningen . . . 2

2 Teoretisk bakgrund **3**
- 2.1 Arduino . . . 3
- 2.2 Arduino Software IDE . . . 3
- 2.3 Arduino Nano 33 BLE Sense . . . 4
  - 2.3.1 Arduino LSM9DS1 . . . 5
  - 2.3.2 Varför en IMU? . . . 6
- 2.4 Accelerometer . . . 6
  - 2.4.1 Funktion . . . 7
- 2.5 Gyroskop . . . 8
- 2.6 Magnetometer . . . 9
- 2.7 Madgwickfilter . . . 9

3 Metod **11**

4 Implementation **13**
- 4.1 Datainsamling . . . 13
  - 4.1.1 Val av mikrokontroller . . . 13
- 4.2 Utveckling . . . 13
  - 4.2.1 Första testningarna . . . 13
  - 4.2.2 Avläsning av värden . . . 14
  - 4.2.3 Tolkning av värden . . . 15
  - 4.2.4 Olika sätt Arduinon ska upptäcka position och rörelse . . . 17
  - 4.2.5 Återkommande problem . . . 18
  - 4.2.6 Feedback . . . 19

5 Resultat **21**
- 5.1 Konstruktionen . . . 21
- 5.2 Bluetooth och feedback . . . 22
- 5.3 Resultat från tester . . . 23

6 Diskussion och Slutsats **25**

6.1 Återkommande problem 25
6.2 Tänkbara orsaker 26
6.3 Förslag på förbättringar 27
6.4 Miljömässig aspekt 27

Referenser **29**

A Bilagor **I**
A.1 Grafer I
A.2 Arduino kod XII

Förkortningar

BLE - Bluetooth Low Energy

CPU - Central Processing Unit

DAC - Digital-to-Analog Converter

I^2C - Inter-Integrated Circuit

IDE - Integrated Development Environment

IMU - Inertial Measurement Unit

LED - Light Emitting Diode

PWM - Pulse Width Modulation

SPI - Serial Peripheral Interface

SRAM - Static Random-Access Memory

UART - Universal Asynchronous Receiver/Transmitter

1

Introduktion

1.1 Bakgrund

I dagsläget finns det produkter som rörelsedetektorer och garderobsbelysning som reagerar på ljusändringar och även larm som använder sig av magneter där ena änden fästs i fönsterkarmen och den andra i fönstret. Dessa har oftast endast ett syfte och kräver dessutom montering med flera fästpunkter. Däremot finns inte idag en avtagbar, portabel produkt, som kan fästas på olika ytor, som endast har en fästpunkt och som samtidigt kan ha flera funktioner. Funktioner såsom att ha koll på statistiken över hur många gånger något har öppnats eller stängts, eller som signalerar, kanske via ett applikationsprogram, när till exempel skåpsdörren är öppen och bör stängas. Därför kan man istället för att köpa in nya produkter, så som ett nytt kylskåp eller ett nytt skåp med dessa funktioner inbyggt, installera en liten portabel apparat som enkelt kan fästas på vad som helst oavsett om det är en dörr, byrålåda eller lock. Detta är också ett mer miljövänligt val, då man tillför en redan inköpt och befintlig produkt, en ny funktion.

Projektet framfördes till ingenjörsbolaget Broccoli och skall utföras i samarbete med dem. Broccoli kommer att ta del av en fördjupande undersökning om hur man kan konstruera denna typ av mekatronisk sensor, eftersom de arbetar mycket med olika typer av sensorer implementerade i små och eller stora reglersystem.

1.2 Syfte

Uppdraget är att tillverka en generisk och portabel sensor som kan fästas på olika ytor. Den ska med hjälp av olika typer av sensorer kunna lokalisera var det ursprungliga läget är och registrera rörelse. På så sätt ska den kunna signalera antingen med LED, ljud eller till ett applikationsprogram att ytan har rört på sig eller öppnats/stängts.

1.3 Avgränsningar

Mikrokontrollern byggs inte från grunden utan köps in färdig. Ett applikationsprogram för insamling av statistik kommer inte att skapas. Prototypen görs endast för att testa funktionaliteten och kommer inte att uppfylla några estetiska krav.

1.4 Precisering av frågeställningen

- Kan man tillverka en portabel apparat som känner av om en dörr har öppnats eller stängts med endast en fästpunkt?
- Hur liten och enkel kan apparaten bli samt hur väl kommer den fungera?
- Vilka typer av sensorer är lämpliga för detektion av rörelse, orientering och position?
- Hur strömeffektiv kan lösningen bli?

2
Teoretisk bakgrund

I detta kapitel presenteras de komponenter och den teknik som har använts under projektet.

2.1 Arduino

Arduino är en elektronisk plattform som använder sig av open-source för utformning av olika sorters projekt och tillämpningar [1]. Då den är enkel att använda passar den bra för både mjuk- och hårdvaruprojekt. På Arduinokort har man möjligheten att implementera och skapa diverse projekt på grund av dess enkla språk och kompatibilitet med både olika sensorer för insamling av data och dess förmåga att kommunicera detta till användaren via Bluetooth, Wifi eller kompatibla skärmar.

Arduinokortet har en mikrokontroller som tar emot instruktioner så att den kan utföra det man vill att den ska göra. För implementering av kod kopplas kortet via USB-kabel till datorn. På datorn används programmeringsmiljön Arduino Software IDE som också är av typen open-source.

Det var på Ivrea Interaction Design Institute som Arduino först skapades för studenter som inte hade någon bakgrund inom varken programmering eller elektronik. Men idag kan vem som helst använda den, från nybörjare till erfarna programmerare. En stor fördel med Arduinokort är att de är publicerade under Creative Commons license (CC) så att alla har tillgång till att bygga sin egen version [1].

2.2 Arduino Software IDE

Den programmeringsmjukvara som tillämpas för Arduino är Arduino Software IDE, där IDE står för "Integrated Development Environment" [2]. Den består av bland annat en texteditor för möjligheten att skriva kod, meddelanderuta och en verktygsrad med knappar för vanliga funktioner samt en rad med olika menyval. Tillgängligt i menyn finns även färdigskrivna exempelkoder för att komma igång för användning av specifika funktioner på Arduinon.

I Arduino Software IDE kan man välja det Arduniokort man ska använda då det finns flera olika versioner, funktioner och storlekar på Arduinos. När man gjort valet sätter den automatiskt parametrarna CPU-hastighet och "baud rate" = hur många gånger en signal ändras per sekund. Dessa är till för när man kompilerar och laddar

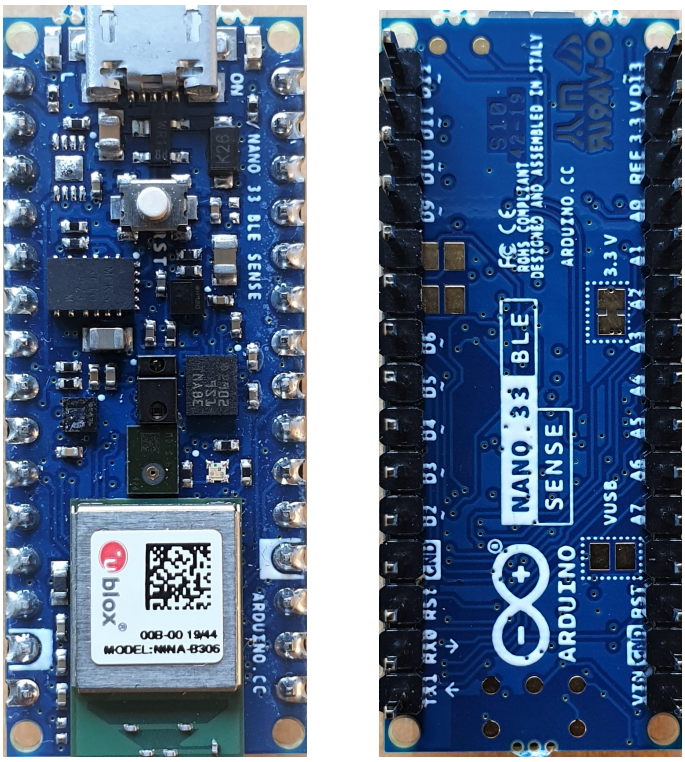
upp koden [2]. Det finns även många olika bibliotek med kod man kan ladda ner och använda sig av.

2.3 Arduino Nano 33 BLE Sense

Arduino Nano 33 BLE Sense är det minsta kortet tillgängligt ute på marknaden och den drivs på 3,3 V istället för 5 V som många andra Arduinokort gör [3]. Den har ett antal inbyggda sensorer såsom en 9-axlig tröghetssensor (IMU-sensor), fukt- och temperatursensor, barometer, mikrofon, rörelsesensor, närhetssensor samt ljus- och färgsensor. Denna version av Arduino har en mer kraftfull processor kallad nRF52840. Det är en 32-bit ARM Cortex-M4 CPU med en klockhastighet på 64 MHz. Den har 1 MB programminne och flertal variabler. I processorn finns även Bluetooth parning via NFC samt ultralåg strömförbrukningsläge. Med en micro-USB kabel kopplas kortet direkt till datorn. Bilder på Arduinokortet ses i Figur 2.1 och 2.2.

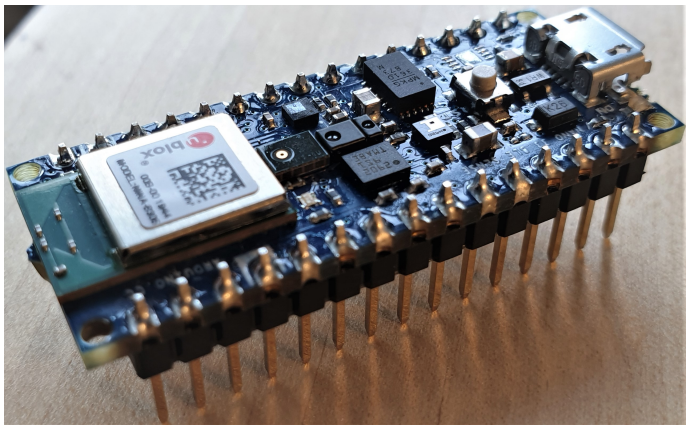
Specifikationer [3]

Mikrokontroller	nRF52840 (datablad)
Driftspänning	3,3 V
Inspänning (gräns)	21 V
DC ström per I/O pinnar	15 mA
Klockhastighet	64 MHz
CPU flash minne	1 MB (nRF52840)
SRAM	256 KB (nRF52840)
Digital ingång/utgång	14
PWM pinnar	Alla digitala pinnar
UART	1
SPI	1
I^2C	1
Analoga ingångspinnar	8 (ADC 12 bit 200 ksampel)
Analoga utgångspinnar	Bara genom PWM (ingen DAC)
`LED_BUILTIN`	13
USB	Native in the nRF52840 Processor
IMU	LSM9DS1 (datablad)
Mikrofon	MP34DT05 (datablad)
Gest, ljus, närhet	APDS9960 (datablad)
Barometertryck	LPS22HB (datablad)
Temperatur, fukt	HTS221 (datablad)
Längd	45 mm
Bredd	18 mm
Vikt	5 g (med lister)



(a) Framsida *(b) Baksida*

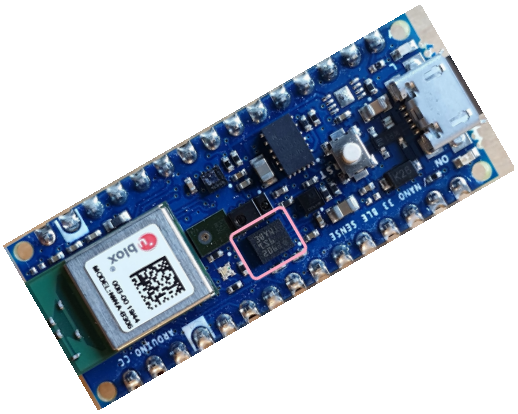
Figur 2.1: *Arduinokortet.*



Figur 2.2: *Arduinokort sett från sidan.*

2.3.1 Arduino LSM9DS1

Arduino LSM9DS1 är ett bibliotek som kan användas då man vill få tillgång till Arduino Nano 33 BLE IMU sensorn [4]. IMU står för "Inertial Measurement Unit" (tröghetsmätningsenhet) [5] och består av en 3D accelerometer, ett 3D gyroskop och en 3D magnetometer [6], se Figur 2.3. IMU kan användas för att till exempel bestämma ett flygplans kurs och den kan hittas i andra applikationer såsom mobiltelefoner eller bilar som använder sig av position, riktning och orientering.



Figur 2.3: *Den rosa färgmarkeringen visar IMU-sensorn (LSM9DS1) på Arduino-kortet.*

Bra att ha i åtanke innan man använder Arduinon är att följande initialisering sätts automatiskt för involverade sensorer:

- Räckvidd/omfång för accelerometern är [-4,+4]g -/+0.122 mg
- Omfång för gyroskop sätts till [-2000, +2000] dps +/-70 mdps
- Omfång för magnetmeter sätts till [-400, +400] µT +/-0.014 µT
- Utgångsdata för accelerometer är fixerad på 104 Hz
- Utgångsdata för gyroskop är fixerad på 104 Hz
- Utgångsdata för magnetometer är fixerad till 20 Hz

2.3.2 Varför en IMU?

För en väl fungerande IMU som kan bestämma dess orientering och läge, behövs alla tre sensorer som har nämnts, det vill säga accelerometer, gyroskop och magnetometer. Tillsammans, genom sensorfusion där mätvärdena slagits samman, kan de rätta till varandras driftningar och störningar för att få ett bättre resultat samt ett mer stabilt värde [7].

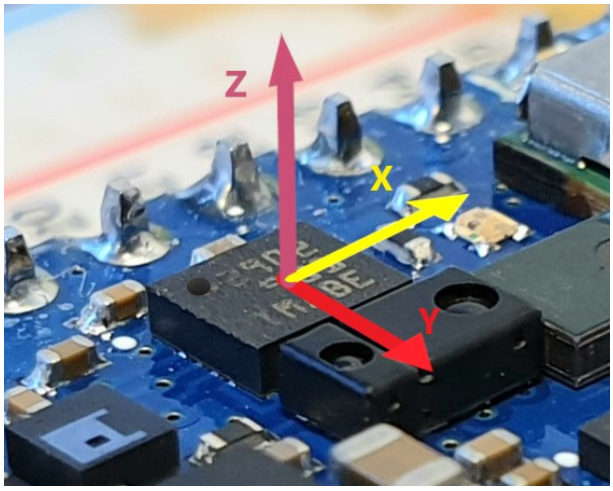
2.4 Accelerometer

En accelerometer har förmågan att mäta hastighetsförändringar (acceleration) av ett objekt, framförallt i tre olika riktningar [8]. Riktningarna är i x-,y- och z-axeln. Den påverkas dock av gravitation och dess enhet är meter per kvadratsekund (m/s^2) eller i G-krafter (g) [9]. Eftersom accelerometern är en tröghetssensor (intertialsystem) betyder det att när apparaten ligger helt stilla, exempelvis om den ligger platt på ett bord, finns en acceleration uppåt på grund av att den mäter en kraft från bordet, som trycker apparaten uppåt [8]. Detta gör att den blir lika med jordens gravitation, g = 9.8 m/s^2. Accelerometern reagerar eller känner av när apparaten är i fritt fall, det vill säga att accelerationen är i den riktning den faller och blir lika med 0 m/s^2. Accelerometrar brukar oftast samverka med andra sensorer då den inte är lika användbar för sig själv.

2.4.1 Funktion

En accelerometer kan känna av två olika krafter av acceleration: statisk kraft eller dynamisk kraft [9]. Gravitation är ett exempel på statisk kraft, medan vibration och rörelse är exempel på dynamisk. Eftersom accelerometern är en elektromekanisk apparat kan den känna av dessa krafter.

Den accelerometer som börjar bli mer och mer vanlig är den 3-axliga (se Figur 2.4). Ett sätt för accelerometrar att känna av dessa krafter är att de innehåller kapacitiva plattor. Plattorna kan antingen vara fixerade eller fästa på väldigt små fjädrar inuti, som förflyttas och får rörelse när det uppstår en kraft [9]. Acceleration kan då bestämmas när kapacitansen mellan plattorna ändras då de får rörelse i förhållande till varandra. Det andra sättet är användning av piezoelektriska material. Accelerometern placeras då i mitten av dessa, och när de piezoelektriska små kristallerna utsätts för mekanisk stress/acceleration, bildas en elektrisk laddning.



Figur 2.4: *De 3-axlar/riktningar (z,x,y) som accelerometern känner av. Bilden är tagen på ovansidan av projektets IMU-sensor och därefter skapat enligt sensorns datablad [6].*

Accelerometrar drar generellt lite ström och brukar behöva ström inom ramen mikro (µ)- eller milliampere [9]. Spänningen den behöver ligger på 5 Volt eller mindre. Dessa egenskaper gör att accelerometrar oftast passar bra vid liten strömförsörjning som kan fås av batterier och dylikt.

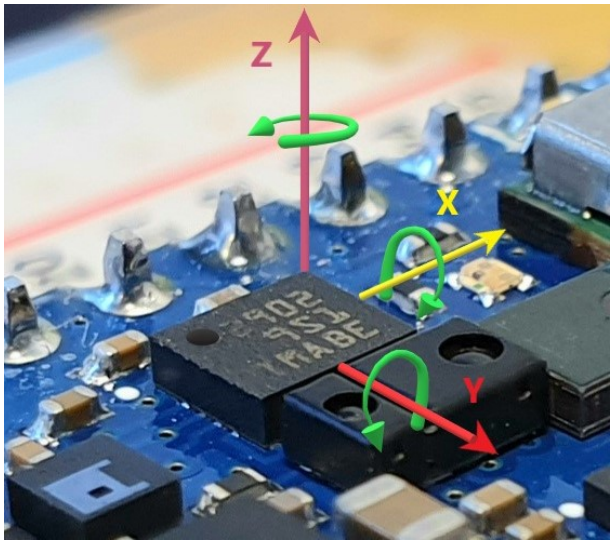
2.5 Gyroskop

Ett gyroskop känner av vinkel/rotationshastigheter i förhållande till sig själv [8]. Den använder sig av tröghetskraften Corioliseffekten[1] och ger alltså information om vinkeln då den känner av rotation, se Figur 2.5. I gyroskopet finns det en massa som oscillerar konstant och när gyroskopet utsätts för en rotation förskjuts massan åt något håll enligt Corioliseffekten [11]. Man kan alltså säga att med hjälp av Corioliskraften kan gyroskopet bestämma rotationshastigheten.

Enheten för gyroskop är grader per sekund (dps = degrees per second) [8]. För uträkning av vinkeln från vinkel/rotationshastigheten behöver följande integration göras (där f ≡ frekvens):

$$\int \cos(2\pi \times ft)) = (1/(2\pi \times f)) \times \sin(2\pi \times ft) \tag{2.1}$$

Vid denna integrering kan dock störningar ge upphov till en driftning i vinkel och gör då resultatet opålitligt. Den drar även mycket energi vid mätning av rotation på grund av att den oscillerar på relativ hög frekvens.

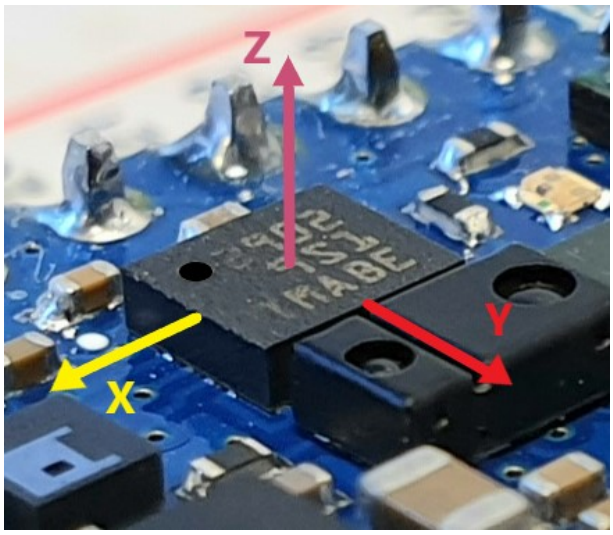


Figur 2.5: *De 3-axlar/vinklar (z,x,y) som gyroskopet känner av. Bilden är tagen på ovansidan av projektets IMU-sensor och därefter skapat enligt sensorns datablad [6].*

[1]Corioliseffekten har en påverkan på föremål som är ovanför jord och som ska färdas långt längs jordens axel. Det corioliseffekten gör är att den påvisar en avvikelse i den färdriktning föremålet färdas [10].

2.6 Magnetometer

För att känna av magnetiska fält så som exempelvis jordens magnetfält kan man använda en magnetometer. Magnetometern känner av och kan upptäcka var det starkaste magnetfältet kommer från [12]. Magnetometern känner alltså av magnetiska fält i de tre fysiska axlarna x, y och z, se Figur 2.6. Magnetiskt fält kan uppstå av elström, magnetiska material eller jordens magnetiska fält [13]. Denna sensor används gärna med två andra typer av sensorer: accelerometrar och gyroskop. Tillsammans kan dessa sensorer upptäcka dess egna orientering då de bildar en 9-axlad IMU [12].



Figur 2.6: *De 3-axlar/riktningar (z,x,y) som magnetometern känner av från magnetiska fält. Bilden är tagen på ovansidan av projektets IMU-sensor och därefter skapat enligt sensorns datablad [6].*

2.7 Madgwickfilter

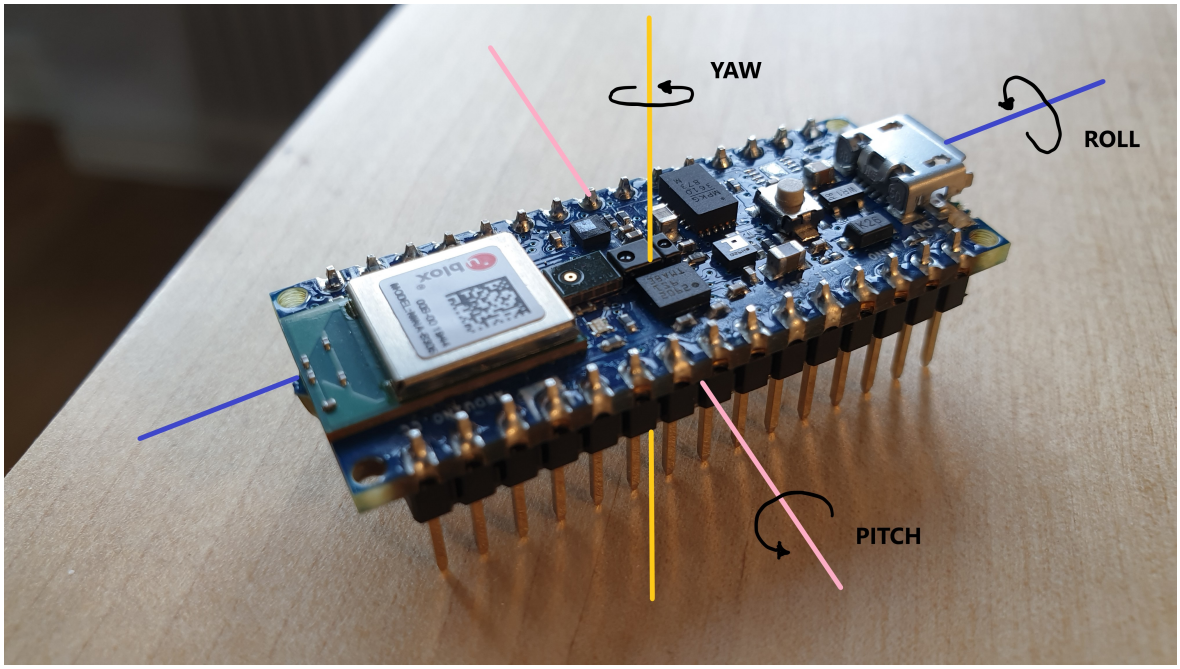
Ett Madgwickfilter är en algoritm som skapades av Sebastian Madgwick år 2010 [14]. Det var under hans forskarutbildning som han utvecklade denna algoritm så att den blev billigare och även mer effektiv på låga samplingsfrekvenser.

För att kunna bestämma orientering i tre dimensioner måste man ha vissa referenspunkter [15]. Dessa får man av magnetometerns egenskap att känna jordens gravitationsfält och accelerometerns förmåga att känna av jordens gravitation. Ett sådant system kallas AHRS (Attitude and Heading Reference Systems). Dock räcker det inte att rätta sig efter dessa två, då de är utsatta för störningar. Störningar kan innebära att något ändras i magnetometerns omgivning, till exempel att en elektronisk apparat är i närheten och ändrar de närmaste magnetfälten. Accelerometern och gyroskopet kan störas av accelerationsändringar i olika led. Dessa störningar läggs på varandra och resulterar i ett driftande resultat, det vill säga ett resultat som man inte kan orientera sig efter. Det är dessa störningar som man motverkar

med filtrering, med till exempel hjälp av Madgwickfilter.

Madgwickfiltret tar sensorvärdena från magnetometer, accelerometer och gyroskop, och sammanslår dessa för en enda uträkning av nuvarande orientering. För en mer djupgående förklaring, finns det ett fyrdimensionellt komplext nummer, ett så kallat kvaternion, som representerar x, y och z värdena, samt ω värdet [14].
x, y och z värdena motsvarar den axel runt den rotation som uppstår, medan ω värdet representerar det värdet som uppstår då rotation sker runt samma axel. Dessa värden tillsammans kan användas för att få fram orienteringen. Madgwick algoritmen tar därför dessa råa värden från accelerometern, gyroskopet och magnetometern för att returnera de fyra kvaternion. "Yaw, pitch and roll" är de tre vinklar som används i Madgwickfiltret och bestäms med det fyrdimensionella talsystemet, se Figur 2.7.

Madgwickfiltret utvecklades för att kunna utföra denna typ av orienteringsfiltrering på mindre system, något som inte gick med det annars accepterade Kalmanfiltret [16] som också använts till denna typ av filtrering. Dessa filter kräver en högre samplingsfrekvens för sina uträkningar.



Figur 2.7: *Förklarande bild på hur yaw, pitch och roll verkar.*

3
Metod

Arbetet började med att hitta vilken mikrokontroller som skulle passa för projektet samt information om vilka sensorer som var lämpliga för att kunna genomföra uppgiften. Senare togs det reda på vilken strömförsörjning prototypen behövde.

För datainsamlingen användes olika databaser. Designen av apparaten var simpel och berodde på hur mikrokontrollern och sensorerna var uppbyggda samt var beroende av batteriernas storlek och vikt. När tillräckligt med information om alla delar hade tagits fram och beslut om vilka verktyg som behövdes för att bygga prototypen hade gjorts, gjordes en beställning av dessa och byggandet påbörjades.

Under utvecklingsprocessen utfördes enkla tester där alla delar sattes ihop och testning av dess registrering av rörelse i x-led gjordes. Sedan skulle den kunna detektera sin ursprungsposition. Därefter testades det om liknande funktioner kunde tillsättas för detektion av y-led.

Prototypen testades genom att fästas på olika ytor som öppnas på olika sätt (lock, dörr) samt utvärderades utifrån hur den betedde sig i de yttersta fallen (om man öppnade väldigt snabbt, om man öppnade långsamt). Tid och utrymme gavs också för ytterligare modifiering och testning av konstruktionen innan den utvärderades.

En utvärdering utfördes på den färdiga prototypen för att se om den uppfyllde målen och för att se om eller hur den skulle kunna förbättras.

4
Implementation

I detta kapitel kommer utförandet och arbetsgången för projektet att beskrivas och förklaras.

4.1 Datainsamling

Projektet inleddes med att utvärdera olika sensorer som kan tänkas passa för detektion av rörelse och läge. Resultat på sökningen gav att accelerometer, gyroskop och eller att en IMU sensor ska fungera för detta avseende. Därefter utvärderades fler sensorer för att säkertställa att en IMU är ett av de bättre alternativen. Magnetometer sensorer kom också upp som ett bra alternativ för att utföra experimentet.

4.1.1 Val av mikrokontroller

Sedan tidigare var det bestämt att ett mikrokontrollerkort skulle köpas och inte byggas från grunden, därför gjordes en sökning på olika mikrokontroller som skulle passa för ändamålet. Arduino är väldigt populärt, lättanvänt och fungerar väldigt bra för sådana här projekt. Nästa mål var således att hitta vilket av flera Arduinokort som passar detta projekt. Eftersom det inte var helt säkert om vilka sensorer som skulle behövas och ett av målen var också att försöka få apparaten att bli liten och enkel, beställdes därför ett Arduinokort som har flera funktionaliteter, sensorer och är en av Arduinos minsta modeller. Arduino Nano 33 BLE Sense var det Arduinokort som beställdes hem samt en kopplingsplatta. När tillräcklig information hade hämtats om IMU-sensorn, Arduinon och hur de fungerar, påbörjades konstruktionen av prototypen.

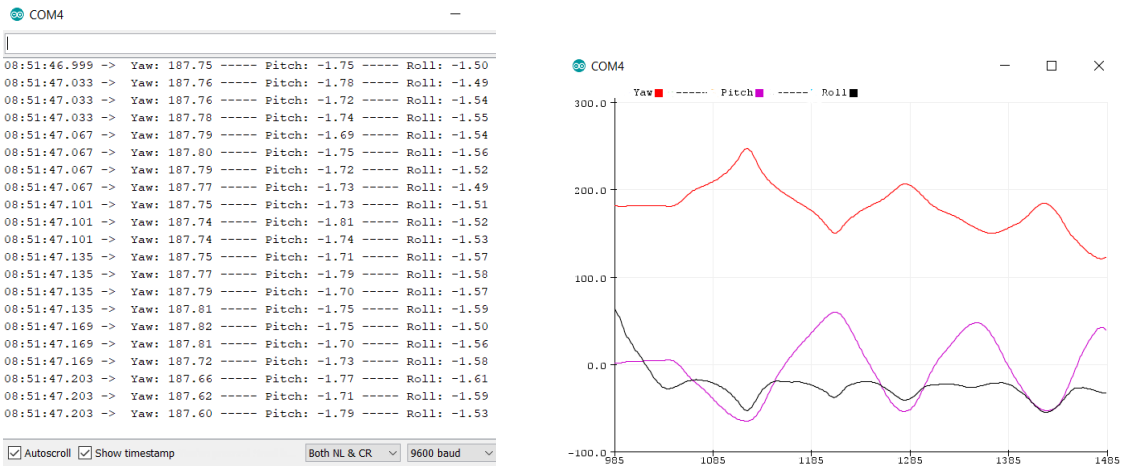
4.2 Utveckling

4.2.1 Första testningarna

Installation av programmeringsmiljön Arduino Software IDE och biblioteket för IMU sensorn (LSM9DS1) gjordes. Därefter söktes information på hur IMU-biblioteket ska användas och hur sensorerna fungerar. Många olika exempel och lösningar testades för att hitta rätt sätt att kommunicera med Arduinons sensorer. Exempel hittades på olika forum samt Arduinos egna hemsida och testades för att få en känsla för hur och om Arduino sensorerna fungerar. Med hjälp av dessa olika exempel och pusselbitar kunde en första version av skåpsensorn implementeras på Arduinokortet.

4.2.2 Avläsning av värden

Testerna gjordes till en början för hand, det vill säga, kortet var fäst på kopplingsplattan och datorn via USB-kabel, där sedan vertikala och horisontella rörelser utfördes för hand i luften. Sensorerna verkade fungera bra vid testning var för sig, men när de användes tillsammans uppstod förskjutningar. Vid informationssökning hittades en algoritm kallad Madgwickfilter och vid implementering av detta bibliotek löstes några av förskjutningarna. Resultatet avlästes och utvärderades på Arduinos programmeringsmiljö, Arduino Software IDE. Där skrevs all kod då den även har inbyggd uppladdningskompatibilitet med utvecklingskortet. Denna miljö har även två olika sätt att läsa av data som skickas från kortet under körtid: "Serial Monitor" (som visar utdatavärdena i text och siffror) och "Serial Plotter" (som visar värden i en graf). Bilder på hur dessa kunde se ut ses i Figur 4.1.



(a) *Serial Monitor* ***(b)*** *Serial Plotter*

Figur 4.1: *De två olika sätt Arduino Software IDE kan visa resultat.*

4.2.3 Tolkning av värden

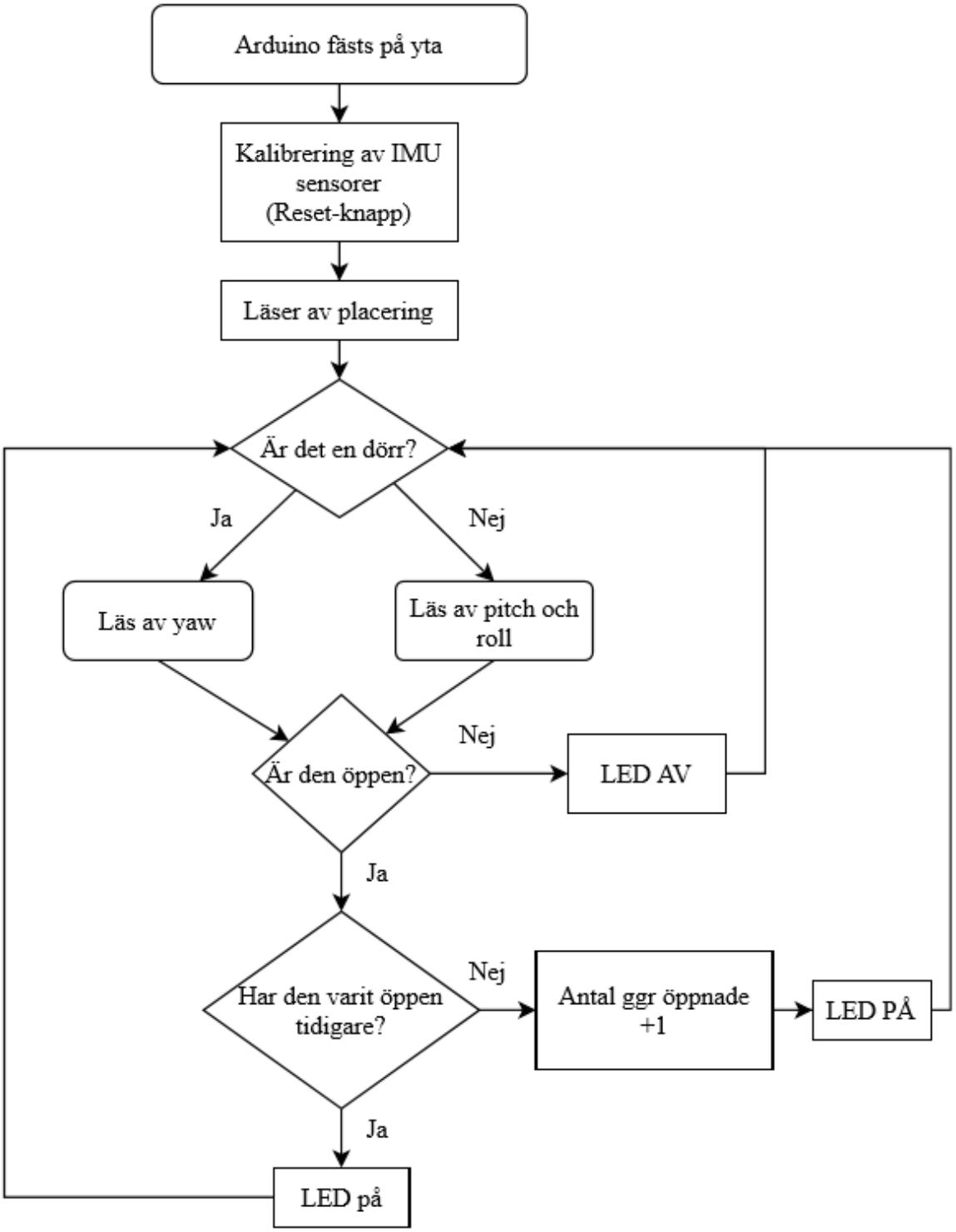
När sensorerna kunde fungera tillsammans utfördes nya tester där prototypen istället fästes på en liten låda med hjälp av tejp, se Figur 4.2. Sedan testades alla de olika sätt som en dörr eller ett lock kan öppnas och stängas på.

Figur 4.2: *Tester som utfördes på liten låda, Arduinokortet kopplad till dator via USB-kabel.*

När prototypen började fungera någorlunda bra, testades en annan typ av strömförsörjning än från datorn. En powerbank med liten formfaktor och som kan ge ut max 2.4 A användes. Med hjälp av tejp, tejpades kortet och laddaren ihop för att kunna fästas (också med tejp) på olika ytor för testning.

För att kunna detektera ursprungsläget, det vill säga den position då dörren eller lådan är i ett stängt läge, tänktes det från början att en switch-knapp skulle implementeras och tillföras till kopplingsplattan och Arduinon. Dock snabbt därefter började Reset-knappen komma till användning och blev därefter också knappen som används för detektering av ursprungsläget, istället för en extra knapp. Detta genom att avläsningen av "ursprungsläge" istället sker under "setup fasen" för kortet som då bara körs en gång.

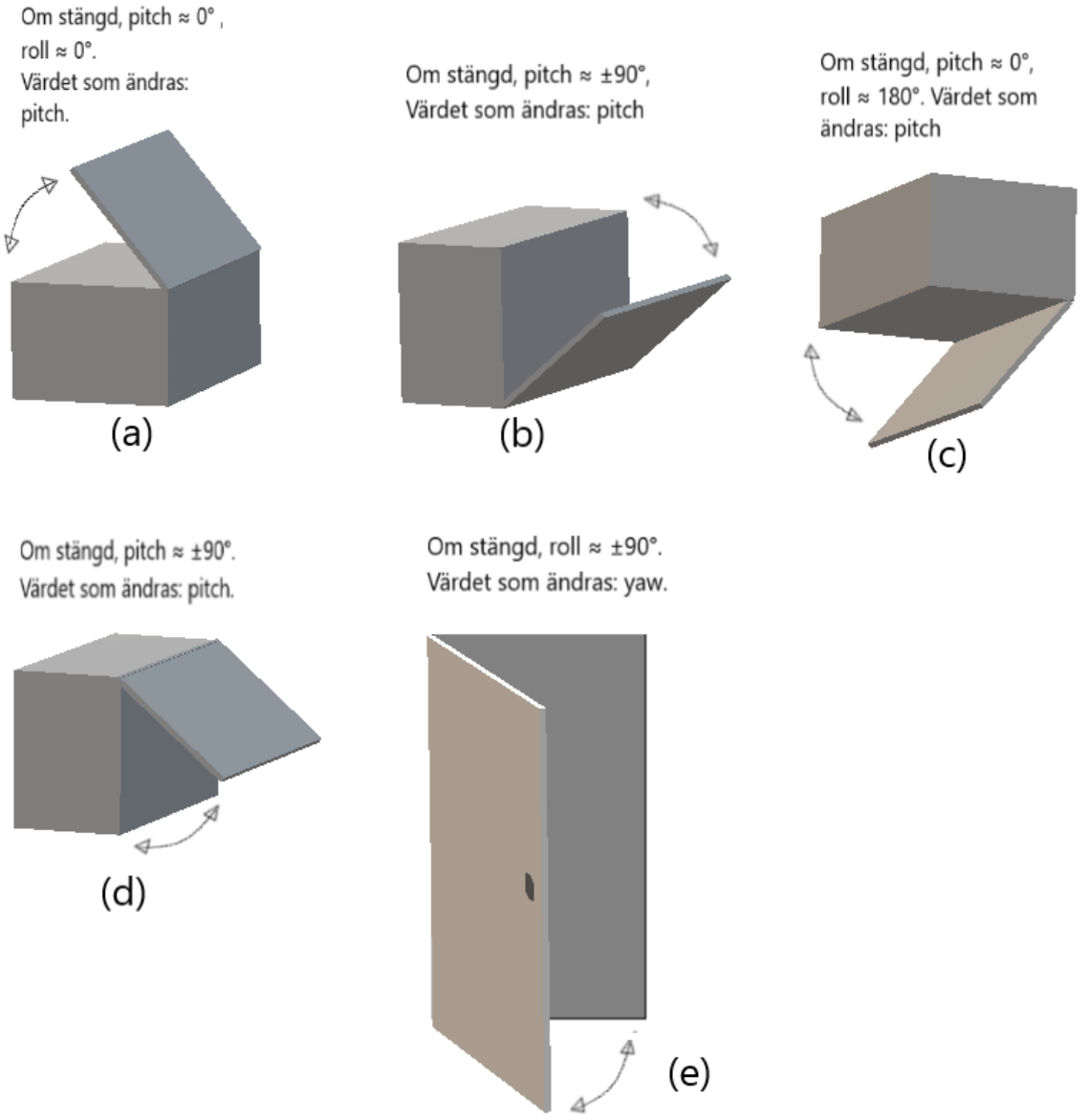
I ett försök att förstå hela processen lättare och för att kontrollera att alla aspekter finns med i programmeringskoden, skissades det fram ett flödesschema, se Figur 4.3. Det blev då klart att det fanns några olika saker som behövdes tas reda på innan avläsning kunde göras av de olika värdena. "Är det en dörr?" behövde bestämmas för att kunna avgöra vilket värde som är viktigt att läsa av samt att detta var första gången "antalet gånger öppnade" också behövdes tas hänsyn till. Då krävdes också någon typ av information om huruvida dörren redan står öppen, och därmed inte har öppnats (alltså ändrat läge).



Figur 4.3: *Flödesschema som visar hela processen.*

4.2.4 Olika sätt Arduinon ska upptäcka position och rörelse

Ytterligare en skiss gjordes, för lättare förståelse och kodning, av olika sätt Arduinokortet kan sättas på en låda för att kunna detektera rörelse och position, se Figur 4.4. Testningarna utfördes på en liten låda samtidigt som Arduinon var kopplad till datorn med USB-kabel. Det första som testades och som till en början fungerade relativt bra var upptäckten av "pitch"-rörelsen, det vill säga då lådan kunde öppnas i y-led (vertikalled). Samtidigt skrevs också en kod att denna rörelse ska upptäckas och registreras då "roll" har ett visst värde i grader. Hur avläsning av värden gick till kan lättare förstås av att titta på figur 4.4 (a-d) där alla de olika sätten listas. All testning kontrollerades och avstämdes med hjälp av Serial monitor och Serial plotter på Arduinos programmeringsmiljö. Sedan skulle Arduinon också kunna upptäcka rörelse i horisontalled (x-led) då den är fäst på exempelvis en dörr med gångjärn, se Figur 4.4 (e). Detta kunde upptäckas med hjälp av "yaw".



Figur 4.4: *Olika sätt som en låda/skåp och dörr kan öppnas och stängas på.*

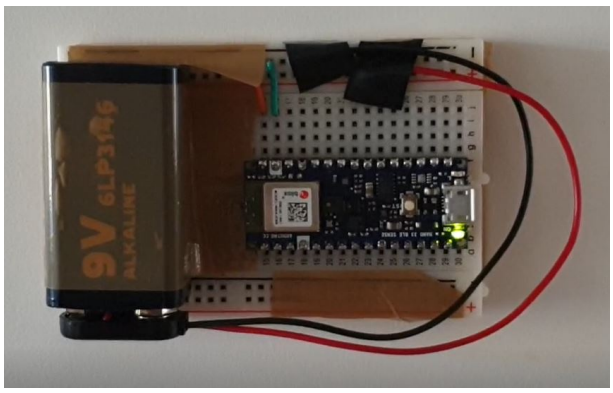
Alla ändringar som gjordes i koden och implementerades på kortet, testades alltså på denna lilla låda varje gång. Om prototypen fungerade, testades detta med powerbank på riktiga ytor (skåp, dörr och kista). När de olika sätten kunde registreras och indikera med hjälp av LED att dörren/lådan är öppen eller stängd, implementerades också en kod som räknar upp antalet gånger ytan hade använts.

4.2.5 Återkommande problem

Många gånger fungerade prototypen bra, andra gånger (även om ingen ändring hade gjorts) kunde sensorernas värden bli förskjutna. Vid sådana förskjutningar fick mycket felsökning och ändringar göras tills det rättade till sig. Dessa ändringar kunde till exempel vara att testa olika uppdateringsfrekvenser samt vidare analysera sättet att mata in värden till filtret.

Olika problem som uppstått under hela arbetsgången har varit förskjutningar och fördröjningar. Förskjutningar som lett till att yaw, pitch eller roll ändrats rejält. Till exempel när Arduinon har varit i ett "öppet läge", återgick (förskjöts) värdena sakta tillbaka till sitt ursprungsläge utan någon fysisk interaktion med prototypen, vilket var stora förskjutningar. Fördröjningarna kunde vara att värdena (yaw, pitch och roll) rörde sig väldigt långsamt, det vill säga att den registrerade eventuell rörelse för långsamt. Detta medförde att skåpssensorn tog för lång tid på sig att registrera öppning och stängning innan den visade rätt. Fördröjningar och förskjutningar är något som har arbetats på under hela projektet, då det alltid dök upp något nytt.

Efter ett flertal lyckade tester med powerbank där prototypen fungerade någorlunda bra, gjordes testerna under en lite längre tidsperiod. Då började det märkas att Arduinon startar om på något vis. Vid sökning på Google om detta var det många som hade skrivit att Arduinon drar för lite ström, vilket medför att powerbanken inte försörjer Arduinon med tillräcklig ström och "stänger av sig" och resulterar i att Arduinon startar om. Efter läsning om detta testades olika resistorvärden så att hela prototypen skulle dra mer ström, men resistorvärdena som fanns tillgängligt var inte tillräckligt låga och testning med ett 9 V batteri var istället nästa steg och lösning. Det fungerade bra med 9 V batteriet, se Figur 4.5.



Figur 4.5: *Kortet med 9V batteri ansluten och fäst på dörr.*

4.2.6 Feedback

Arduinokortet har sitt egna inbyggda LED-lampa och den används som feedback till användaren i detta projekt för att kunna se om sensorerna uppfattat att ändring av position har skett. Lampan indikerar om exempelvis dörren är öppen (LED på) eller om den är stängd (LED av). Under testningar kunde man inte heller avläsa någon data för att se om prototypen verkligen räknar upp och vilka slags fördröjningar eller förskjutningar den kan ha. Detta försvårade testningen. Ett test gjordes med en LED-skärm först, men det blev snabbt väldigt otympligt. Då denna Arduinomodell har Bluetooth, därav namnet "BLE" som står för Blutetooth-Low-Energy, laddades biblioteket för BLE ner och implementerades i koden. Efter flertal försök att hitta en existerande applikation tillgänglig för mobilen, upptäcktes appen "LightBlue" som kunde läsa av datan. Detta underlättade testningen avsevärt och gav en bekräftelse på att Arduinokortet använder sig av sitt minne, behåller det och uppdaterar värdet den har även om anslutning via Bluetooth sker när som helst.

Vid tryckning av Reset-knappen var det till en början svårt att avgöra om Arduinokortet förstod att den skulle kalibrera om, och det var svårt att förstå om den var redo att användas. Tack vare detta problem ändrades prototypen så att LED-lampan blinkar tre gånger efter att Reset-knappen har tryckts in och gav även användaren feedback på att det går bra att använda skåpssensorn.

För korrekt registrering av läge ska skåpssensorn som den fungerar just nu alltid placeras med USB-ingången riktad mot gångjärnen.

5
Resultat

En prototyp av den portabla och trådlösa skåpssensorn har tagits fram och fungerar i många fall som avsett. Under arbetets gång stöttes det naturligtvis på motgångar, men dessa åtgärdades och arbetet följde ändå tidigare planerad tidplan. Skåpssensorn är portabel då den får sin strömförsörjning via powerbank eller batteri, vilket var ett av målen. Strömförsörjningen blev inte ett lika stort fokus på under arbetets gång då mycket mer tid lades på att prototypen skulle fungera. Arduinokortet kan med hjälp av sitt eget minne registrera, räkna upp och behålla värdet på antal gånger ytan använts.

5.1 Konstruktionen

Ett annat mål var att skåpssensorn skulle bli liten och enkel, vilket delvis har åstadkommits genom att ha valt Arduino Nano 33 BLE Sense, då den är det minsta mikrokontroller kortet inom Arduino familjen. När kortet däremot kopplas till kopplingsplattan och powerbanken blir helheten större, likaså vid användning av 9 V batteriet. Detta kan dock enkelt åtgärdas med exempelvis en mindre powerbank (som ändå ger tillräckligt med ström) och om man gör kopplingsplattan mindre. I figur 5.1 kan man få en bättre bild på hur skåpssensorn kunde se ut.

(a) *Skåpssensorn fäst på en dörr med powerbank.*

(b) *Skåpssensorn fäst på en dörr med 9 V batteri.*

Figur 5.1: *De två olika typer av strömförsörjning som testats i projektet kopplat till Arduinokortet.*

Det tar en minut för prototypen att kalibrera och registrera ursprungsläge. Denna specifika tid togs fram genom flertalet tester där tiden det tog för värdena att stabiliseras mättes. En av de inbyggda LED-lamporna på Arduinokortet blinkar tre gånger med orange färg efter att man har tryckt på Reset-knappen (den andra LED-lampan lyser grönt och betyder att kortet är på). Denna visar att kalibreringen är klar och att skåpssensorn har registrerat ett ursprungsläge. Detta görs för att ge en bättre användarvänlighet och indikation på att skåpssensorn är redo att börja användas. Detta var dock inte en del av målen för projektet utan lades till för att lättare utvärdera resultatet då skåpssensorn är fäst på en yta.

5.2 Bluetooth och feedback

Ett tillägg till projektet (som heller inte var ett av målen med projektet) är att med hjälp av kortets inbyggda Bluetooth, visa antalet gånger skåpssensorn har använts samt om man är intresserad, se värdena för yaw, pitch och roll. Datan kunde enkelt läsas av med valfri BLE läsningsapplikation. Det var dock till fördel om applikationen lät dig översätta de inlästa värdena från hexadecimal till decimala tecken för lättare avläsning. Detta underlättade arbetet väldigt mycket, gjorde felsökningarna lättare och gav även användaren feedback.

5.3 Resultat från tester

I Bilagor-avsnittet (figur A.1 till A.8) ses resultat på alla de olika sätt projektets skåpssensor kan registrera öppning och stängning av skåp, dörr eller låda. Resultaten visas i form av grafer. Graferna visar tydligt hur det ser ut i stängt läge och hur det kan se ut när man öppnar och stänger dörren/locket. För att lättare hänga med i dessa grafer, hänvisas det till förra kapitlet (4.2.4) där man kunde se figur 4.4 som visade de olika sätt skåpssensorn kunde registrera rörelse.
Dessa grafer kommer ifrån när alla olika lägen testades efter varandra med den lilla lådan. I detta fall fungerade allting bra, och den kunde tydligt avgöra när locket på lådan var öppet och stängt i alla olika lägen. De störningar man kan se i graferna i olika lägen beror på svårigheten att hålla lådan stilla, när man till exempel skall öppna den i luften (som en taklucka i figur 4.4 (d)).

Under testning på olika ytor då batteriet användes som strömförsörjning kunde inga grafer skapas, utan istället användes den inbyggda LED-lampan och Bluetooth för att avgöra om den lyckades upptäcka öppning eller inte. Under dessa tester fungerade läge a-d i figur 4.4, alltså alla lägen där pitch var den avgörande faktorn, som förväntat utan fördröjningar eller förskjutningar. För läge “e” går själva upptäckandet av öppning “långsammare”, alltså att det har en viss fördröjning, även om resultatet (att den registrerar en öppning och stängning) fortfarande är stabilt, och den lyckas varje gång.

Någonting som inte hanns med och därför inte fungerar är fästande av prototypen på en dörr/låda/lock som inte har gångjärn, det vill säga något som går på räls, till exempel en byrålåda. Prototypen kan därför inte registrera denna typ av öppning och stängning.

6
Diskussion och Slutsats

Syftet med detta arbete var att skapa en generisk och portabel skåpssensor, vars uppgift var att upptäcka och registrera rörelser. Den skulle gärna också vara liten och enkel samt kunna fästas på många ytor. Helhetsprototypen av skåpssensorn blev någorlunda lyckat. Den kan känna av rörelse och dess position samt registrera detta. Samtidigt kan den lokalisera dess ursprungliga läge samt räkna upp antalet gånger ytan den är fäst på har använts. Vid utvärdering av vilka sensorer som skulle passa detta projekt visade det sig att en accelerometer, gyroskop och magnetometer fungerar väldigt bra tillsammans för upptäckt av position och orientering och hittas bland annat i flygplan, mobiler och bilar.

Skåpssensorn kan även med hjälp av sin inbyggda LED-lampa signalera och ge feedback till användaren att ytan är öppen eller stängd, genom att lampan lyser då exempelvis dörren är öppen, och släcks när dörren är stängd. Ett redan färdigt applikationsprogram har också kunnat användas på mobilen och genom att ansluta Arduinokortets Bluetooth med mobilens, kan användaren se hur många gånger dörren har öppnats. Detta var verkligen något som underlättade all sorts testning under arbetet och det var lite synd att detta gjordes mer i mitten eller mot slutet av arbetet istället för tidigare. Detta hade kunnat göras tidigare om jag hade haft mer förkunskaper om användandet av IMU-sensorer och dess filtrering, något som tog mycket av tiden.

6.1 Återkommande problem

Något som har kvarstått i prototypen och under arbetets gång är förskjutning och fördröjning av värden. Det har lagts väldigt mycket tid och energi på att åtgärda dessa problem. Ena sekunden fungerar skåpssensorn väldigt bra vid testningar på olika ytor och vid olika sätt av öppningar/stängningar av ytor, men andra sekunden har det eventuellt uppstått någon typ av fördröjning eller förskjutning. Det kunde vara så att när man testade skåpssensorn på den lilla lådan inkopplad till datorn som strömkälla, registrerade IMU-sensorn rörelser och eventuella snabba rörelser rätt bra. Men när det testades på en annan yta med annan strömförsörjning, så som powerbank eller batteri, uppstod fördröjningar och eller någon typ av förskjutning av värdena. Det var till en början väldigt frustrerande att förstå varför detta uppstod, särskilt innan man kunde ansluta kortet via Bluetooth och se vilka värden som påverkade resultatet mest, och hur de betedde sig vid olika sätt av öppningar. Ytterligare en nackdel till att dessa problem ständigt uppkom och ändringar be-

hövde göras var att kompilering av kod tog tid. Programmeringsmiljön tillsammans med detta Arduinokort gjorde att kompilering av kod och uppladdning av kod tog väldigt mycket av ens tid då den var väldigt långsam.

Ett annat vanligt exempel på hur problemen med fördröjningar/förskjutningar kunde se ut var att om man hade problem med yaw (horisontella rörelser) men inte med pitch (vertikala rörelser) eftersom den registrerade helt korrekt och man åtgärdade detta problem, så kunde resultatet av ändringen bli omvänt. Det vill säga, att pitch istället fick en fördröjning och yaw fungerade.

Dessa problem kvarstår inte helt och hållet i dagsläget och skåpssensorns tillförlitlighet är ändå acceptabel, även om en fördröjning återstår. Detta för att fördröjningen inte är så pass stor/hög att resultatet skulle bli otillförlitligt. Vissa fördröjningar går att åtgärda, men när man gör det fås istället en rejäl översväng hos ett annat värde, därför kom man till en punkt då man fick acceptera att det är några små kvarstående problem som kanske inte påverkar registreringen och uppräkningen ändå. Däremot är en förskjutning av värde inte lika acceptabelt, då det skulle kunna resultera i felregistrering, att den helt enkelt inte längre känner igen ursprungsläget (det vill säga stängt läge).

6.2 Tänkbara orsaker

Dessa problem kan bero på flera anledningar. Kanske är det så att det finns bättre filter och algoritmer än Madgwickfiltret. Madgwickfiltret ska ju filtrera störningarna som de olika sensorer kan ha var för sig och få de att tillsammans kunna utföra denna typ av orienteringsfiltering på mindre system, som denna skåpssensor. Men det kan möjligtvis vara så att denna IMU (LSM9DS1) inte är tillräckligt kraftfull eller har hög noggrannhet/precision vid mätning och att det inte beror på Madgwickfiltret. Man kanske behövde en annan IMU-sensor för detta projektet. Exempelvis ett Arduinokort som är kraftfullare och en extern IMU-sensor som kanske är bättre och ger mer exakta resultat. Eller möjligtvis en helt annan mer kraftfull mikrokontroller. Ett tydligt tecken på att det kan ha varit processorkraft som var den strypande faktorn var att när man kopplade upp sig mot prototypen med Bluetooth gick allt mycket långsammare och fördröjningar blev mycket kraftfullare. Denna extra processorkraft som Bluetooth hanteringen tog påverkade resultatet mycket, och tyder då på att snabbare filtrering med en starkare processor samt snabbare sensorer skulle kunna ge mer responsiva och stabilare värden.

Ytterligare en anledning kan vara att det råkar finnas någon typ av störning i form av till exempel en annan elektronisk apparat vilket påverkar resultatet. Man kan inte heller påverka omgivningen för prototypen då den också skulle kunna användas på elektroniska apparater såsom kylskåp, mikrovågsugn och diskmaskin. Den skall ju självklart även fungera i dessa fall. För att motverka effekten av näraliggande elektroniska apparater skulle mer tid kunna läggas på att läsa om olika typer av magnetometrar, samt det bästa sättet att hantera dessa störningar. Detta hanns inte med i detta projekt.

6.3 Förslag på förbättringar

Det har heller inte utförts några längre tester under detta arbete, det vill säga att man testar skåpssensorn under en längre period och ser om den fungerar efter några timmar eller om värdena som den registrerar börjar förskjutas. Det skulle kunna finnas en väldigt liten förskjutning som inte märks av förrän sensorerna har fått stå på en lång stund. Detta hade varit en förbättring av prototypen. Fler tester, på olika ytor i olika miljöer skulle också vara värdefullt att testa. Alla tester utfördes i min lägenhet, något som då ger en viss brist i variation på testerna. Då den till viss del använder sig av magnetometer för att rätta till förskjutningar, skulle det vara intressant att stresstesta den i olika miljöer med olika mycket elektronik. Några förslag på miljöer för detta:

- Kontorslandskap
- Matbutik
- Bil/buss
- "Elektronik fritt" rum som kanske ett sovrum

Hade tester som dessa gjorts hade man haft ett bättre underlag för att säga hur generisk och hur väl den fungerar.

Tillräcklig tid har inte lagts på strömförsörjning och därför har det inte tagits reda på hur strömeffektiv lösningen kan bli. Däremot är denna prototyp en strömsnål lösning då detta Arduinokort drar generellt mindre ström än många av de andra modeller. Det har i detta projekt påvisats att denna lösning på en portabel och trådlös skåpssensor är ett fungerande sätt. Detta baseras på egna försök i projektet där test gjordes på de flesta typer av olika luckor och dörrar, även om resultatet inte alltid varit tillförlitligt. Men det finns en till form av öppning av dörrar/luckor, och det är de som går på räls (skjutdörrar) som inte har gångjärn. Just nu fungerar inte denna prototyp på sådana exempel på dörrar, vilket är tråkigt. Detta var något som tid ville läggas på men blev mindre och mindre viktigt när det uppstod små problem med de andra typerna av dörrar under arbetets gång.

Baserat på projektets resultat samt en djupare analys av dessa typer av IMU-sensorer kan man dra slutsatsen att denna typ av lösning definitivt är möjlig och kan fungera mycket bra efter fortsatt arbete enligt ovanstående förbättringar.

6.4 Miljömässig aspekt

Fördelen med skåpssensor prototypen är att den kan användas när man behöver den samt på flera olika apparater, produkter, skåp och dörrar. Ur miljösynpunkt är detta ett bra alternativ att tillföra en redan befintlig produkt eller apparat som man redan äger.

Referenser

[1] Arduino, *"What is Arduino?"*, 2020. [Online]. Tillgänglig: URL: https://www.arduino.cc/en/Guide/Introduction. Hämtad: 2020-02-10.

[2] Arduino, *"Arduino Software (IDE)"*, 2015. [Online]. Tillgänglig: URL: https://www.arduino.cc/en/Guide/Environment. Hämtad: 2020-02-10.

[3] Arduino, *"Arduino Nano 33 BLE Sense"*, 2020. [Online]. Tillgänglig: URL: https://store.arduino.cc/arduino-nano-33-ble-sense. Hämtad: 2020-02-10.

[4] Arduino, *"Arduino LSM9DS3 library"*, 2019. [Online]. Tillgänglig: URL: https://www.arduino.cc/en/Reference/ArduinoLSM9DS1. Hämtad: 2020-02-10.

[5] Wikipedia, *"Inertial measurement unit"*, 2020. [Online]. Tillgänglig: URL: https://en.wikipedia.org/wiki/Inertial_measurement_unit. Hämtad: 2020-02-10.

[6] STMicroelectronics NV, *"LSM9DS1, iNEMO inertial module:3D accelerometer, 3D gyroscope, 3D magnetometer"*, 2015. [Online]. Tillgänglig: URL: https://content.arduino.cc/assets/Nano_BLE_Sense_lsm9ds1.pdf. Hämtad: 2020-02-20.

[7] W3C, *"High-level Sensors"*, 2017. [Online]. Tillgänglig: URL: https://www.w3.org/TR/motion-sensors/#highlevel-sensors. Hämtad: 2020-02-24.

[8] W3C, *"Motion Sensors Explainer"*, 2017. [Online]. Tillgänglig: URL: https://www.w3.org/TR/motion-sensors/#index. Hämtad: 2020-02-24.

[9] Sparkfun, *"Accelerometer Basics"*, [Online]. Tillgänglig: URL: https://learn.sparkfun.com/tutorials/accelerometer-basics/all. Hämtad: 2020-04-06.

[10] National Geographic, *"Coriolis Effect"*, 2011. [Online]. Tillgänglig: URL: https://www.nationalgeographic.org/encyclopedia/coriolis-effect/12th-grade/. Hämtad: 2020-04-20.

[11] Sparkfun, *"Gyroscope"*, [Online]. Tillgänglig: URL: https://learn.sparkfun.com/tutorials/gyroscope/all. Hämtad: 2020-04-09.

[12] Bryan Siepert, *"LIS3MDL Triple-axis Magnetometer"*, 2020. [Online]. Tillgänglig: URL: https://learn.adafruit.com/lis3mdl-triple-axis-magnetometer. Hämtad: 2020-04-09.

[13] *"Magnetometer"*, 2019. [Online]. Tillgänglig: URL: https://www.w3.org/TR/magnetometer/#intro. Hämtad: 2020-04-09.

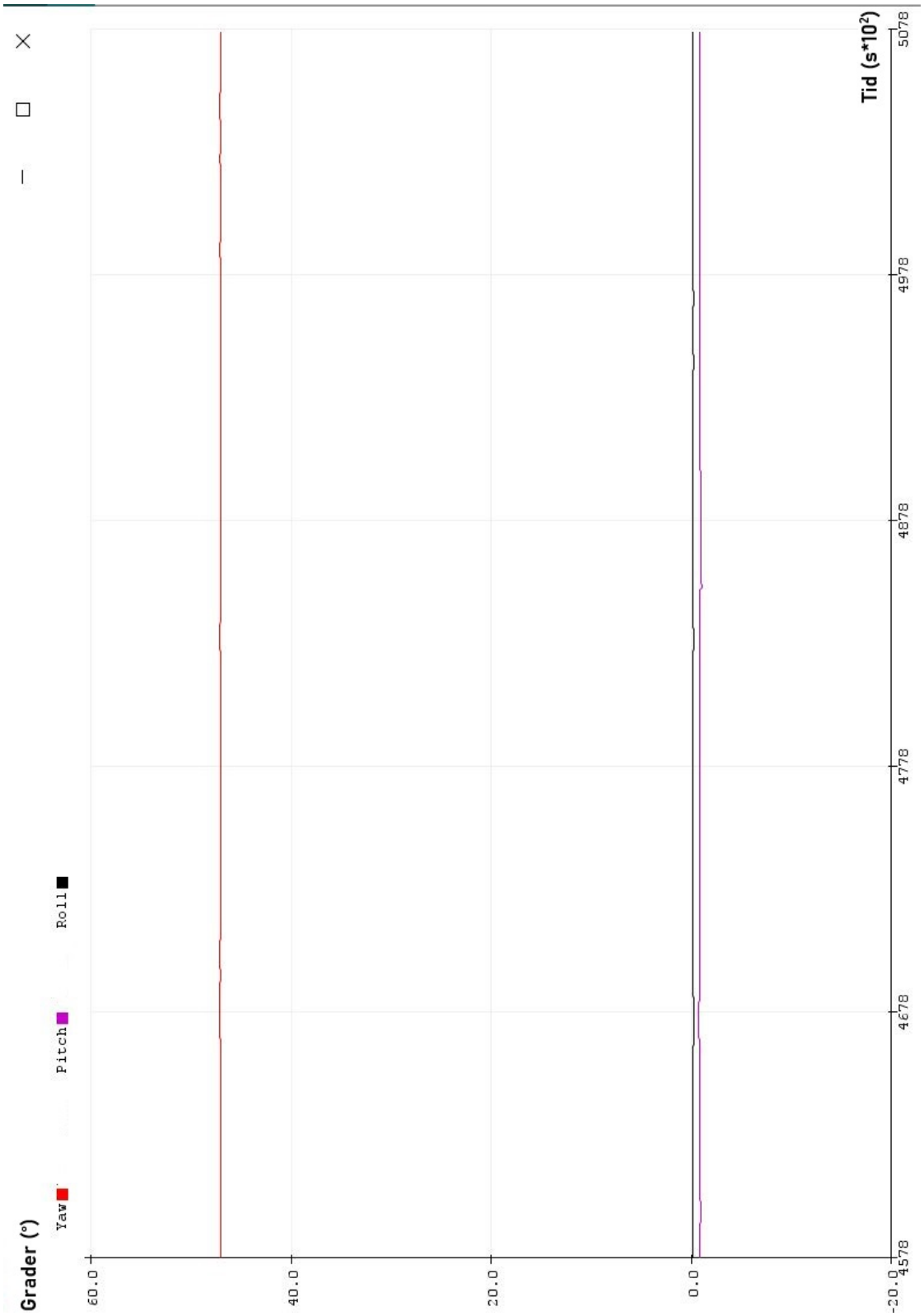
[14] Arduino, *"Arduino/Genuino 101 CurieIMU Orientation Visualiser"*, 2020. [Online]. Tillgänglig : URL: https://www.arduino.cc/en/Tutorial/Genuino101CurieIMUOrientationVisualiser. Hämtad: 2020-04-09.

[15] Sebastian O. H. Madgwick, "An efficient orientation filter for inertial and inertial / magnetic sensor arrays". I: 2010. URL: https://www.x-io.co.uk/res/doc/madgwick_internal_report.pdf.

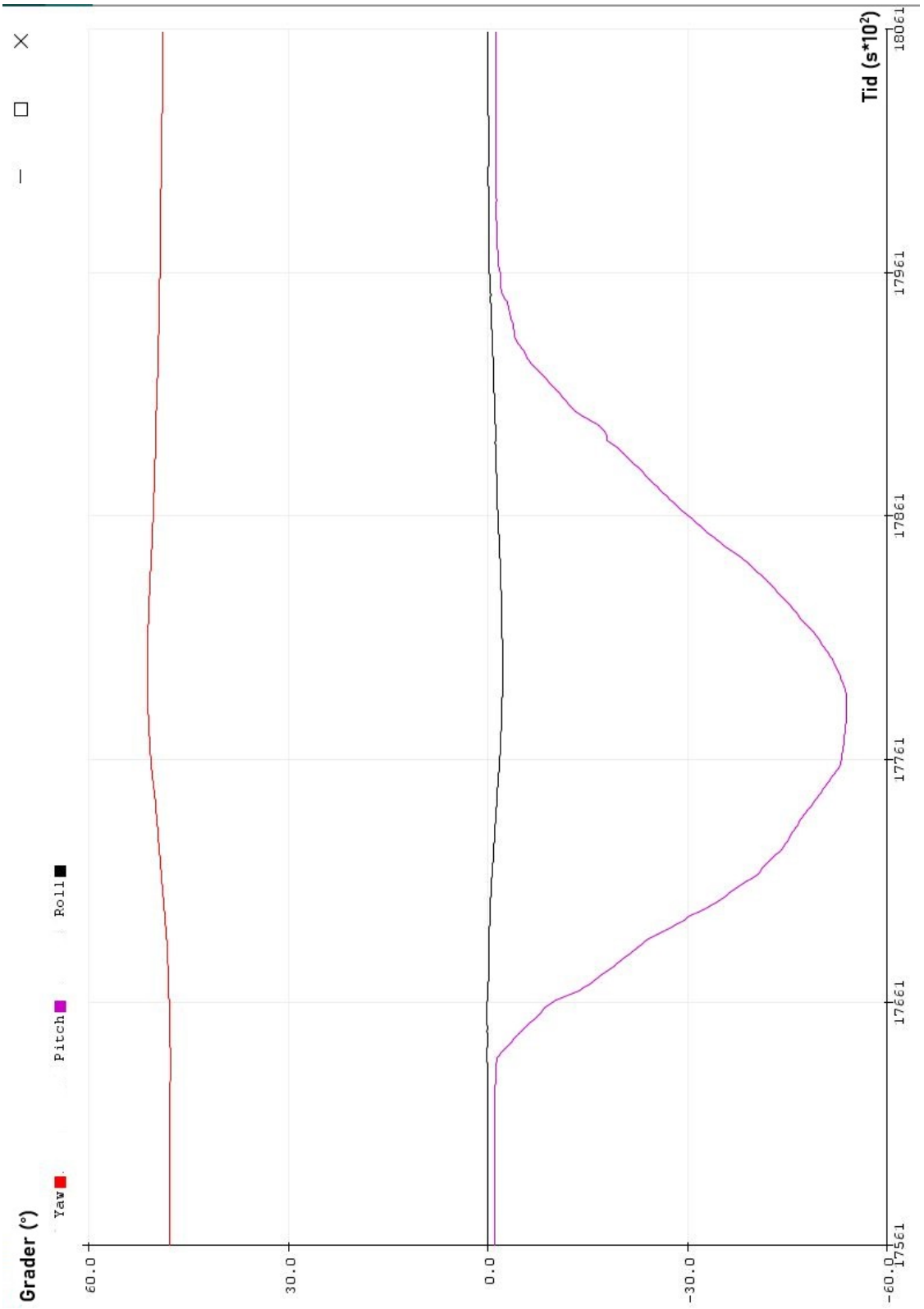
[16] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems". I: *Journal of Basic Engineering* 82.1 (mars 1960), s. 35–45. ISSN: 0021-9223. DOI: 10.1115/1.3662552. URL: https://doi.org/10.1115/1.3662552.

A
Bilagor

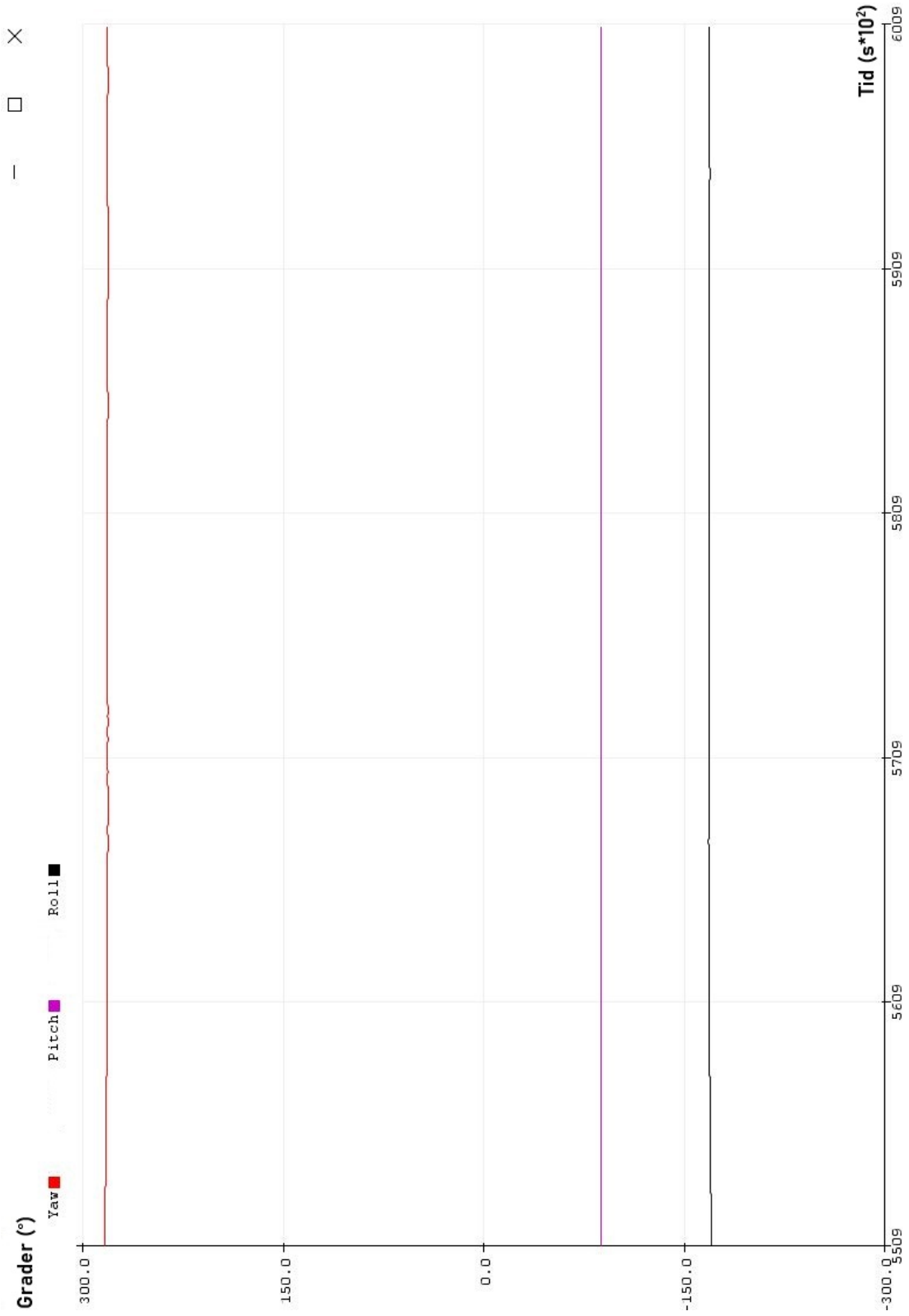
A.1 Grafer



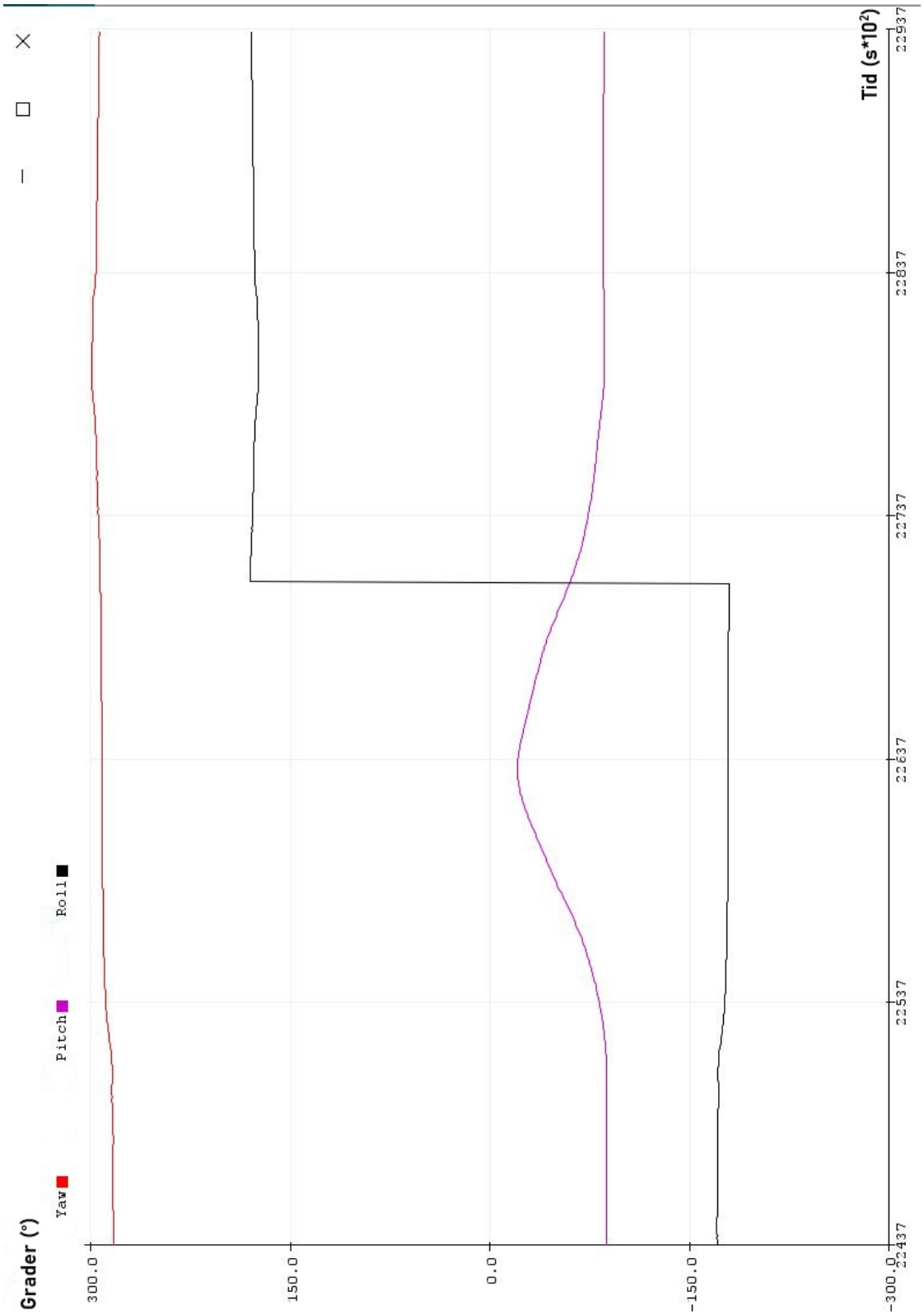
Figur A.1: *Stängt läge/ursprungsläge för registrering av pitch (registrering av vertikalled). Efterliknar Figur 4.4 (a).*



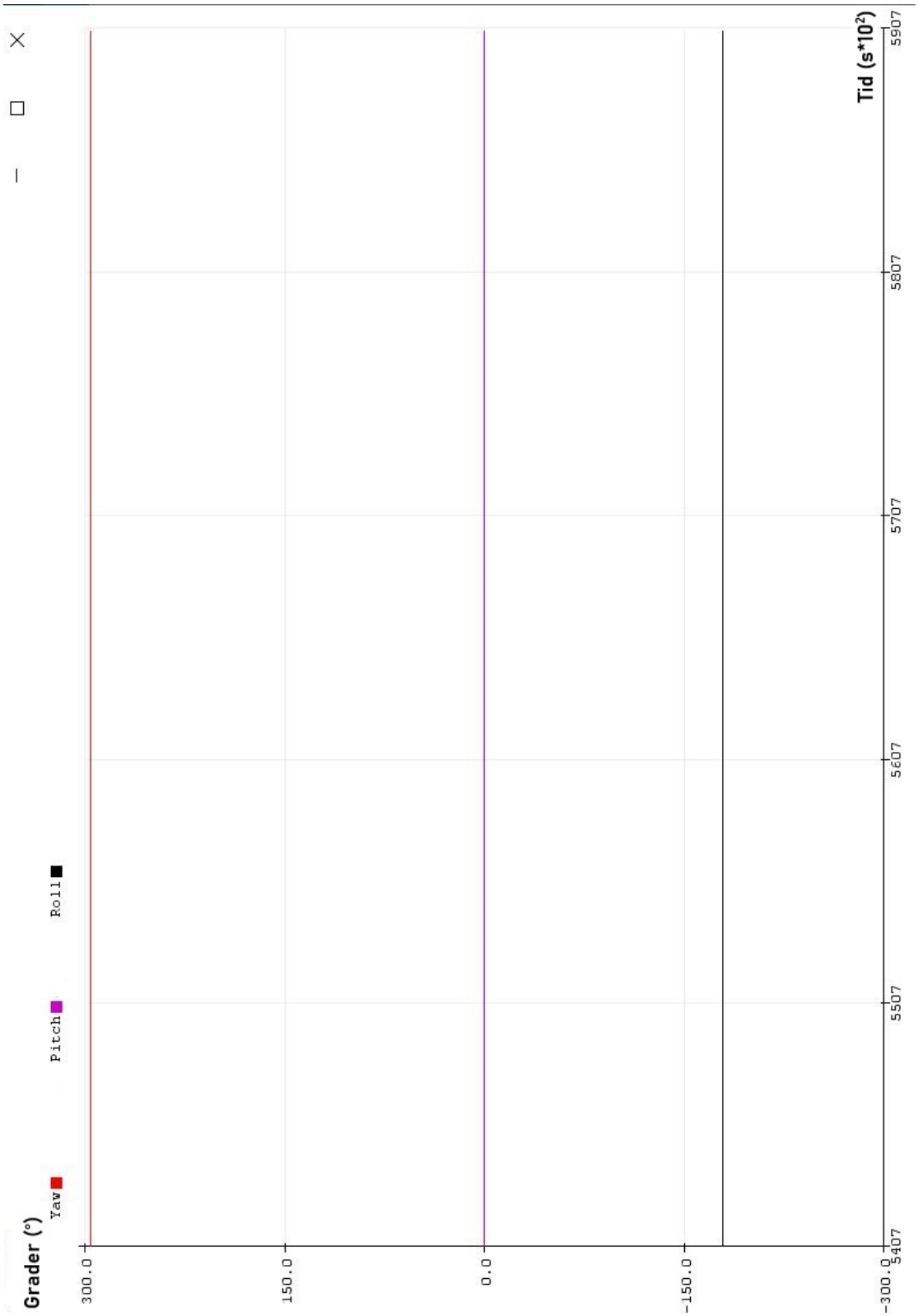
Figur A.2: *Öppet läge för registrering av pitch (registrering av vertikalled). Efterliknar Figur 4.4 (a).*



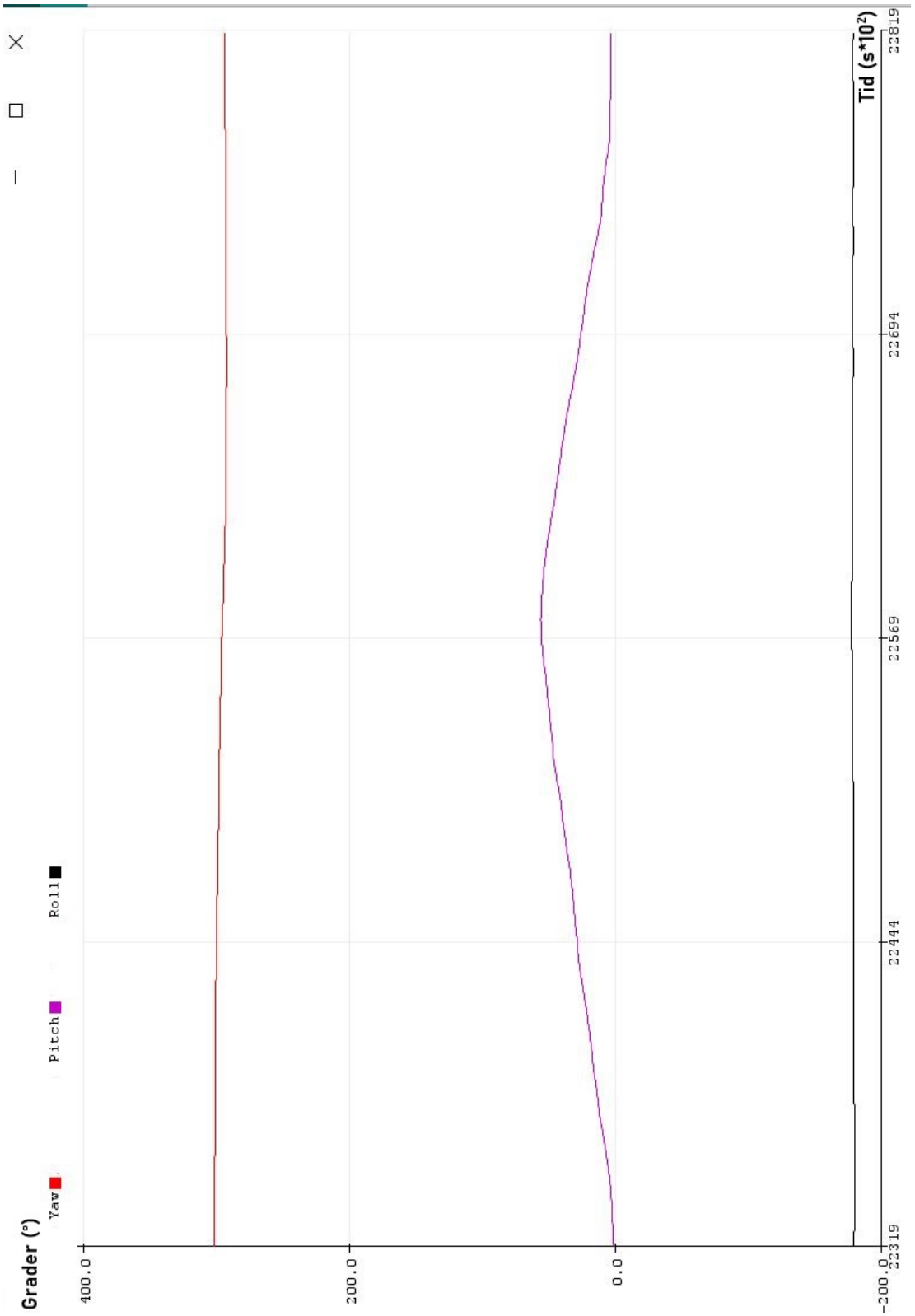
Figur A.3: *Stängt läge/ursprungsläge för registrering av pitch (registrering av vertikalled). Efterliknar Figur 4.4 (b).*



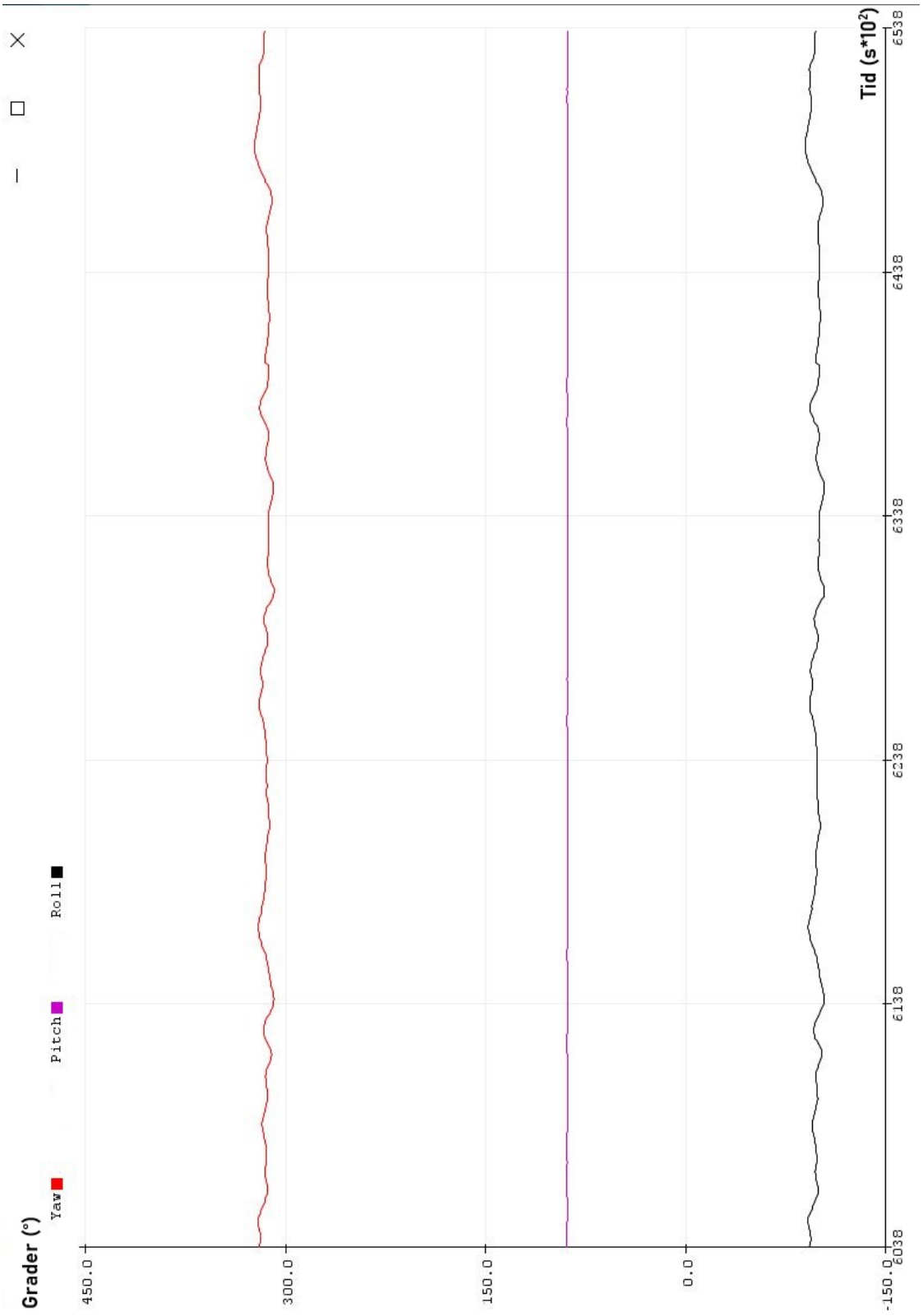
Figur A.4: *Öppet läge för registrering av pitch (registrering av vertikalled). Efterliknar Figur 4.4 (b).*



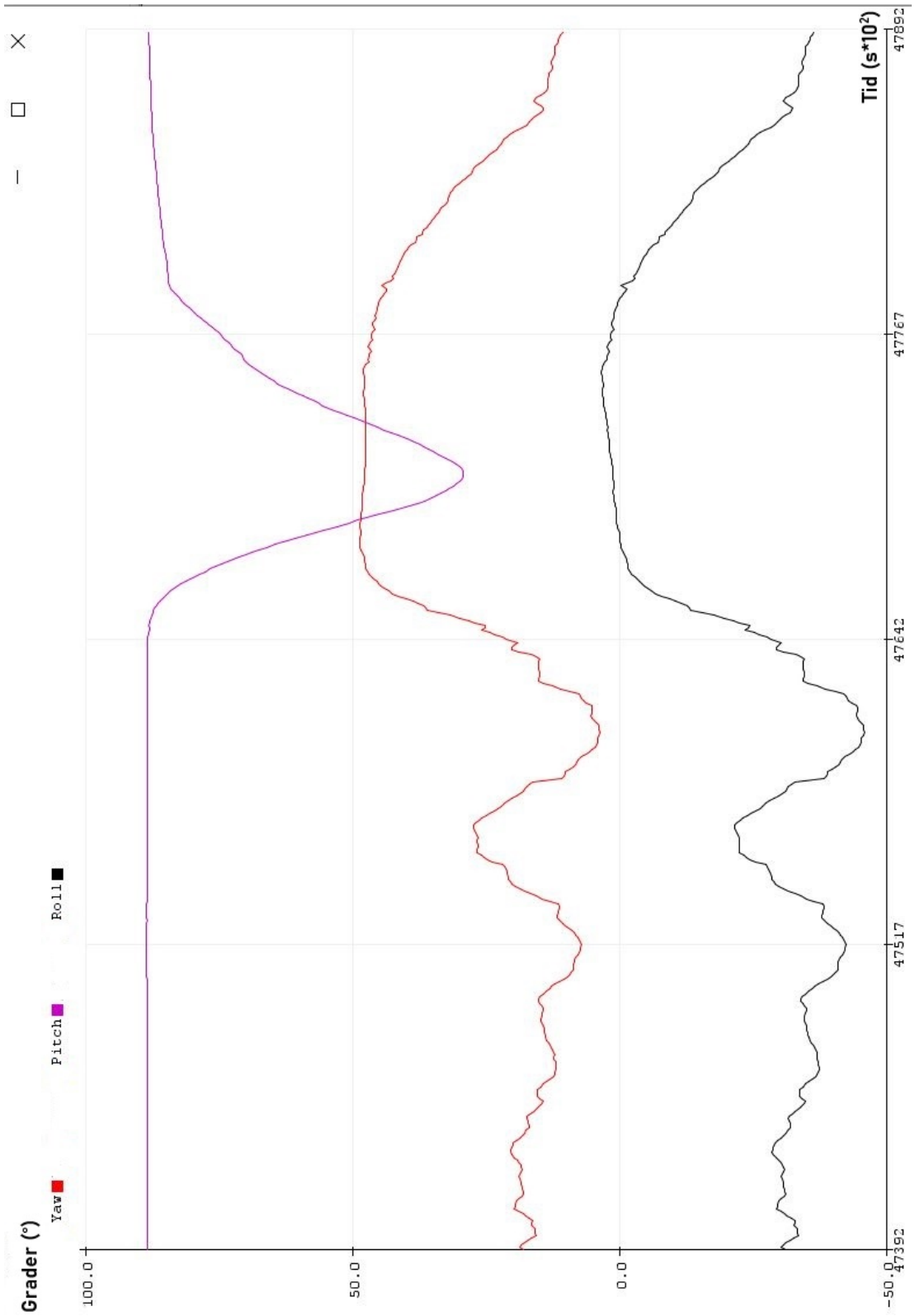
Figur A.5: *Stängt läge/ursprungsläge för registrering av pitch (registrering av vertikalled). Efterliknar Figur 4.4 (c).*



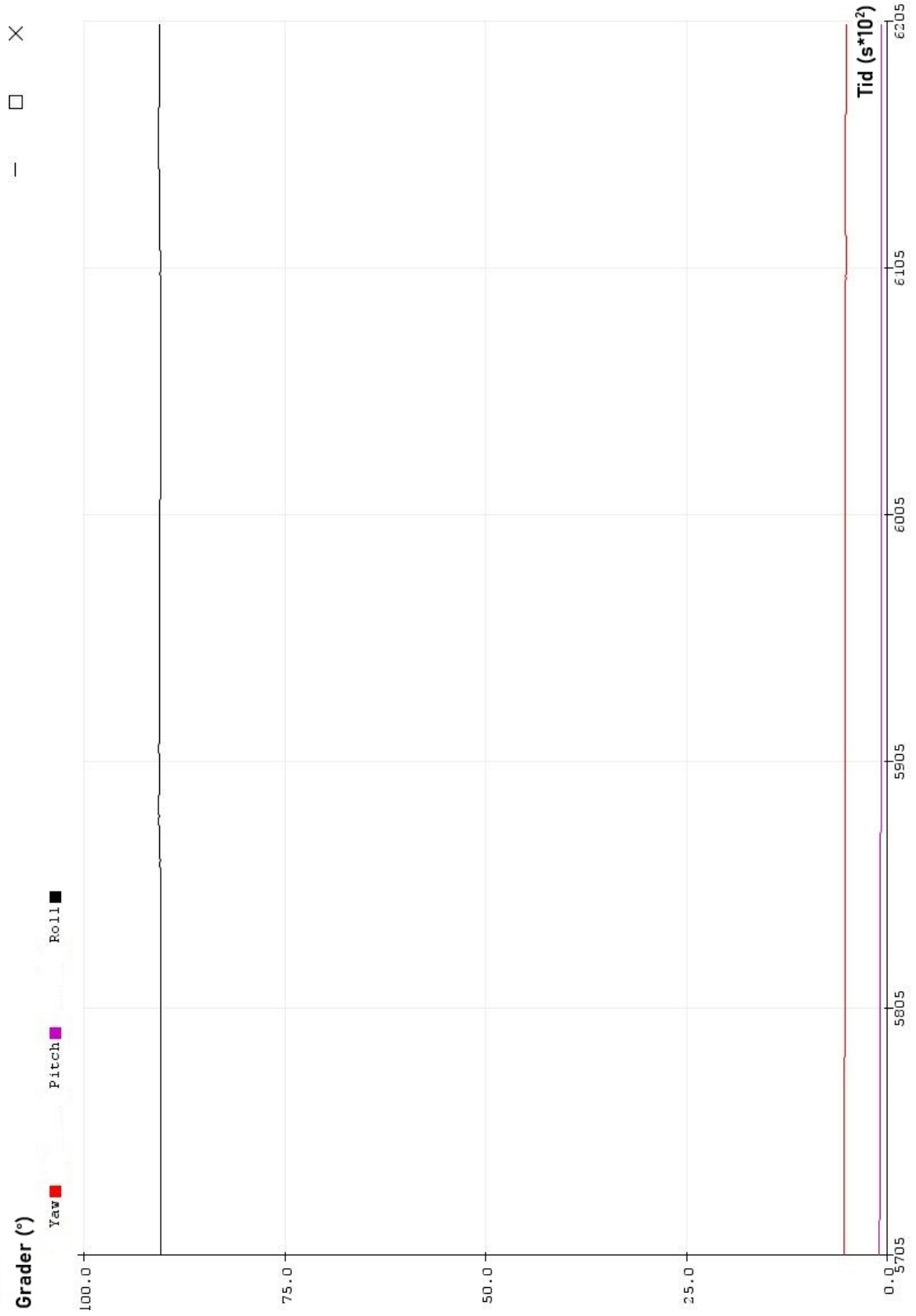
Figur A.6: *Öppet läge för registrering av pitch (registrering av vertikalled). Efterliknar Figur 4.4 (c).*



Figur A.7: *Stängt läge/ursprungsläge för registrering av pitch (registrering av vertikalled). Efterliknar Figur 4.4 (d).*



Figur A.8: *Öppet läge för registrering av pitch (registrering av vertikalled). Efterliknar Figur 4.4 (d).*



Figur A.9: *Stängt läge för registrering av yaw (registrering av horisontalled). Efterliknar Figur 4.4 (e).*

Figur A.10: *Öppet läge för registrering av yaw (registrering av horisontalled). Efterliknar Figur 4.4 (e).*

A.2 Arduino kod

```
#include <Arduino_LSM9DS1.h> // IMU library
#include <MadgwickAHRS.h>    // Madgwick library
#include <ArduinoBLE.h>      // Bluetooth library

Madgwick filter;
unsigned long microsPerReading, microsPrevious;
const float sensorRate = 104.00;                  // Samplerate
int nrOpen = 0;                                   // Number of times opened
bool isOpen = false;                              // Keep track if open
float pitchClosed;                                // Starting position
float yawClosed;                                  // Starting position
bool alreadyOpen = false;                         // Keep track if been open before
bool isDoor;                                      // Door or not

//---------------------- Bluetooth readings -------------------------
BLEService imuService("1101");
BLEIntCharacteristic customYawChar("2101", BLERead | BLENotify);
BLEIntCharacteristic customPitchChar("2102", BLERead | BLENotify);
BLEIntCharacteristic customRollChar("2103", BLERead | BLENotify);
BLEIntCharacteristic customnrOpenChar ("2104", BLERead | BLENotify);
//---------------------------------------------------------------------

void setup() {
  Serial.begin(9600);           // baud rate
  float roll, pitch, yaw;       // yaw, pitch, roll
  float xAcc, yAcc, zAcc;       // acceleration values
  float xGyro, yGyro, zGyro;    // gyroscope values
  float xMag, yMag, zMag;       // magnetometer values



  // start the IMU and filter
  if (!IMU.begin()) {
    Serial.println("Failed to initialize IMU!");
    while (1);
  }

  filter.begin(sensorRate);

  // initialize variables to pace updates to correct rate
  microsPerReading = 1000000 / sensorRate;
  microsPrevious = micros();

  //---------------------------------------------------------------------

   if (!BLE.begin()) {
    Serial.println("starting BLE failed!");
    while (1);
   }

    // Setup bluetooth
    BLE.setLocalName("ArduinoIMU");
    BLE.setAdvertisedService(imuService);

    imuService.addCharacteristic(customYawChar);
    imuService.addCharacteristic(customPitchChar);
    imuService.addCharacteristic(customRollChar);
    imuService.addCharacteristic(customnrOpenChar);

    BLEDescriptor("2101", "Yaw");
    BLE.addService(imuService);
```

```
  // start advertising
  BLE.advertise();
//-----------------------------------------------------------------
unsigned long microsNow;

int starttime = millis();
int endtime = starttime;
while ((endtime - starttime) <=60000)                // do this loop for up to 60000ms
{
 // check if it's time to read data and update the filter
  microsNow = micros();
  if (microsNow - microsPrevious >= microsPerReading) {
      // read raw data
      IMU.readAcceleration(xAcc, yAcc, zAcc);
      IMU.readGyroscope(xGyro, yGyro, zGyro);
      IMU.readMagneticField(xMag, yMag, zMag);

      // update the filter, which computes orientation
      filter.update(-xGyro, -yGyro, -zGyro, xAcc, yAcc, zAcc, -xMag, yMag, zMag);

      // print the yaw, pitch and roll
      roll = filter.getRoll();
      pitch = filter.getPitch();
      yaw = filter.getYaw();

      Serial.print(" Yaw: ");
      Serial.println(yaw);
      Serial.print(" ----- Pitch: ");
      Serial.print(pitch);
      Serial.print(" ----- Roll: ");
      Serial.println(roll);
  '




      // increment previous time, so we keep proper pace
      microsPrevious = microsPrevious + microsPerReading;
    //}
  }
  endtime = millis();
}

//-----------------------------------------------------------------

  // read raw data
  IMU.readAcceleration(xAcc, yAcc, zAcc);
  IMU.readGyroscope(xGyro, yGyro, zGyro);
  IMU.readMagneticField(xMag, yMag, zMag);

  // update the filter, which computes orientation
  filter.update(-xGyro, -yGyro, -zGyro, xAcc, yAcc, zAcc, -xMag, yMag, zMag);

  // print the yaw, pitch and roll
  pitchClosed = filter.getPitch();
  Serial.print(" PitchClosed: ");
  Serial.println(pitchClosed);

  rollClosed = filter.getRoll();
  Serial.print(" RollClosed: ");
  Serial.println(rollClosed);

  yawClosed = filter.getYaw();
  Serial.print(" YawClosed: ");
  Serial.println(yawClosed);
```

```
    // If it's a door, then the roll must be ~90 degrees
    if(((rollClosed >= 80) && (rollClosed < 100)) || ((rollClosed <= -80) && (rollClosed > -100)) )
    {
      // If it's a door, the pitch should be ~0 degrees
      if (((pitchClosed >= -10) &&  (pitchClosed < 10))){
        Serial.println(" IS A DOOR");
        isDoor = true;
      }
      else {
        Serial.println(" IS NOT A DOOR");
        isDoor = false;
      }
    }
    else {
      Serial.println(" IS NOT A DOOR");
      isDoor = false;
    }
    //delay(10000);

    // Initialize digital pin LED_BUILTIN as an output.
    pinMode(LED_BUILTIN, OUTPUT);

    // Blink three times after 1 minute after pressing the reset-button
    digitalWrite(LED_BUILTIN, HIGH);
    delay(500);
    digitalWrite(LED_BUILTIN, LOW);
    delay(500);
    digitalWrite(LED_BUILTIN, HIGH);
    delay(500);
    digitalWrite(LED_BUILTIN, LOW);









    delay(500);
    digitalWrite(LED_BUILTIN, HIGH);
    delay(500);
}

void loop() {

  float xAcc, yAcc, zAcc;
  float xGyro, yGyro, zGyro;
  float xMag, yMag, zMag;
  float roll, pitch, yaw;
  unsigned long microsNow;
    // check if it's time to read data and update the filter
    microsNow = micros();
    if (microsNow - microsPrevious >= microsPerReading) {
      if (!isDoor || IMU.accelerationAvailable() && IMU.gyroscopeAvailable() && IMU.magneticFieldAvailable()) {

        IMU.readAcceleration(xAcc, yAcc, zAcc);
        IMU.readGyroscope(xGyro, yGyro, zGyro);
        IMU.readMagneticField(xMag, yMag, zMag);

        // update the filter, which computes orientation
        filter.update(-xGyro, -yGyro, -zGyro, xAcc, yAcc, zAcc, -xMag, yMag, zMag);

        // print the yaw, pitch and roll
        roll = filter.getRoll();
        pitch = filter.getPitch();
        yaw = filter.getYaw();

        Serial.print(" Yaw: ");
        Serial.println(yaw);
```

```
        Serial.print(" ----- Pitch: ");
        Serial.print(pitch);
        Serial.print(" ----- Roll: ");
        Serial.println(roll);

        Serial.print(" yawClosed: ");
        Serial.println(yawClosed);

        //-------------------------- Bluetooth ----------------------------

        // wait for a BLE central
        BLEDevice central = BLE.central();

        // if a BLE central is connected to the peripheral:
        if (central) {
          Serial.print("Connected to central: ");
          // print the central's BT address:
          Serial.println(central.address());

          // while the central is connected:
          if (central.connected()){
            customYawChar.writeValue(yaw);
            customPitchChar.writeValue(pitch);
            customRollChar.writeValue(roll);
            customnrOpenChar.writeValue(nrOpen);
          }
        }

        //-----------------------------------------------------------------



      // increment previous time, to keep proper pace
      microsPrevious = microsPrevious + microsPerReading;

      // Check if it's a door, counts, LED-light turns on when open
      if (((isDoor && (yaw > (yawClosed + 2.0))) || (isDoor && (yaw < (yawClosed - 2.0))))
      || ((!isDoor && (pitch > (pitchClosed + 2.0))) || (!isDoor && (pitch < (pitchClosed - 2.0))))){  // Is door open?
          if (isOpen == false){
             nrOpen++;
             isOpen = true;
          }
          digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
          alreadyOpen = false;
          Serial.print(" Open: ");
          Serial.print(nrOpen );
          Serial.println(" times ");
      }
      else {
          isOpen = false;

          if(alreadyOpen == false)
          {
            digitalWrite(LED_BUILTIN, LOW);   // turn off the LED by making the voltage LOW
            alreadyOpen = true;
          }
    }
  }
 }
}
```