NTNU

Norwegian University of
Science and Technology

Low Power Inertial Measurement Unit for Internet of Things Applications

Erlend Hestnes

Master of Science in Electronics
Submission date: June 2016
Supervisor: Per Gunnar Kjeldsberg, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Master Assignment

Candidate Name:

Erlend Hestnes

Assignment Title:

Low Power Inertial Measurement Unit for Internet of Things Applications

Assignment Description:

Internet of Things (IoT) applications very often involve the use of sensors. Motion sensing can be used in a vast number of applications and are therefore very interesting in high volume IoT products.

An object's motion can be decomposed as translation along and rotation about each of the X, Y and Z-axes. An inertial measurement unit (IMU) is a sensor package that usually consists of an accelerometer, a gyroscope and sometimes a magnetometer. With these sensors, an IMU is able to provide information on both translation and rotation for an object. The introduction of micro-electromechanical systems (MEMS) has enabled IMUs to undergo miniaturization as well as significant cost reduction. However, there is still an inherent problem with power consumption for such devices. While both MEMS accelerometers and magnetometers are able to operate in the microampere range, gyroscopes usually require a few milliampere. Several publications have showed that it is possible to forgo the gyroscope in an IMU by using a combination of multiple accelerometers placed in a special cube configuration.

This thesis aims to study the possibility of creating a Gyro-Free IMU (GF-IMU) using linear MEMS accelerometers. The design goal is to create a low-cost solution that occupies a small area, and consumes significantly less power than gyro-based solutions.

Assignment Proposer/Co-Supervisor:

Øystein Moldsvor - Disruptive Technologies AS

Supervisor:

Per Gunnar Kjeldsberg

Abstract

Motion sensing is an interesting area for Internet of Things (IoT) applications. Today, motion sensing for embedded applications is usually achieved by using an inertial measurement unit (IMU). An IMU is a sensor package, which normally consists of an accelerometer, gyroscope and sometimes a magnetometer. The introduction of micro-electromechanical systems (MEMS) has enabled IMUs to undergo miniaturization as well as significant cost reduction. There is however still an inherent problem with power consumption for these devices. While both accelerometers and magnetometers are able to operate in the microampere range, gyroscopes usually require a few milliampere. Several publications have showed that it is possible to create a Gyro-Free IMU (GF-IMU) by using a combination of multiple accelerometers placed in a special cube configuration.

Most of the accelerometer based GF-IMU solutions that exists are impractical due to the amount of sensors used and the size of the cube. The practical implementation of a GF-IMU based on two tri-axial accelerometers is relatively new. As such, there are many open questions regarding its true potential.

Both a simulation-based approach and a practical implementation have been used to investigate an existing GF-IMU solution based on two tri-axial accelerometers. The simulation has been used for a theoretical analysis to better illustrate the relationship between different cube geometries and the parameters of precision and power consumption. The practical implementation has been used to provide actual data on how a GF-IMU compares to a conventional gyro-based IMU.

The simulation has successfully been used to show the relationship between different cube configurations and the parameters of precision and power consumption in a GF-IMU. The results from the practical testbed implementation has shown that it is possible to use a GF-IMU configuration to provide an estimate of angular velocity with an average mean squared error (MSE) of 64.68 $(deg/s)^2$. The angular velocity estimate can further be used to provide an angle of rotation with an accuracy of ± 10 degrees over a short period of time. At best, the implemented GF-IMU consumes 79.6% less current than a conventional gyro-based IMU. The testbed results have also revealed a severe limitation with the GF-IMU implementation. Namely, that there is hidden dependency in the equations used for calculating rotation about each axis. This dependency makes it impossible to calculate rotation about the X, Y and Z-axis at the same time.

From the acquired results, we have concluded that the area of applications between the implemented GF-IMU and a conventional gyro-based IMU are different. While a conventional IMU is precise enough to be used for inertial navigation applications, a GF-IMU can only be accurate over a short period of time. A GF-IMU is therefore more suited for applications that only require simple trajectory estimation, such as gesture detection and attitude estimation.

Sammendrag

(Abstract in Norwegian)

Å bestemme hvordan en gjenstand beveger seg er av stor interesse for Tingenes Internett (IoT) applikasjoner. For dagens innvevde systemer er det vanlig å bruke en treghets måleenhet (IMU) for slike formål. En IMU er en sensorpakke som vanligvis består av et akselerometer, gyroskop og eventuelt et kompass. Introduksjonen av mikromaskinerte-elektromekaniske komponenter (MEMS) har gjort det mulig for IMUer å gjennomgå en miniatyrisering og en signifikant kostnadsreduksjon. Fremdeles så er det likevel et stort problem med effektforbruk for slike enheter. Mens akselerometer og kompass er i stand til å operere i mikroampere området, så bruker gyroskop gjerne noen milliampere. Flere publikasjoner har vist at det er mulig å konstruere en gyroskopfri IMU (GF-IMU) ved å bruke en kombinasjon av flere akselerometre plassert i en spesiell kube konfigurasjon.

De fleste GF-IMU løsningene som finnes i dag er upraktiske med hensyn på antall sensormoduler som må til, og selve størrelsen på kuben. Den praktiske realiseringen av en GF-IMU basert på to tre-akse akselerometre er relativt ny. Det finnes derfor flere spørsmål knyttet til potensialet for en slik løsning.

Både en simuleringsbasert fremgangsmåte og en praktisk implementering har blitt brukt for undersøke en eksisterende gyroskopfri løsning basert på to akselerometre. Simuleringen har blitt brukt for en teoretisk analyse for å bedre illustrere forholdet mellom ulike kubekonfigurasjoner opp i mot parameterne presisjon og effektforbruk. Den praktiske implementeringen har blitt brukt til å tilegne faktiske data på hvordan en gyroskopfri IMU yter i forhold til en konvensjonell gyroskopbasert IMU.

Simuleringen har blitt brukt til å vise forholdet mellom ulike kubekonfigurasjoner opp i mot parameterne presisjon og effektforbruk i en GF-IMU. Resultatene fra den praktiske implementeringen har vist at det er mulig å bruke en GF-IMU til å gi et estimat på vinkelhastighet med en gjennomsnittlig kvadratisk rot feil (MSE) på 64.68 $(deg/s)^2$. Vinkelhastighetsestimatet kan videre bli brukt til å finne en rotasjonsvinkel med en nøyaktighet på ± 10 grader over et kort tidsrom. På sitt beste, så bruker den implementerte GF-IMUen 79.6% mindre strøm enn en konvensjonell gyroskopbasert løsning. Resultatene fra den praktiske implementeringen har også avslørt en stor begrensning ved den implementerte GF-IMU løsningen. Nemlig at det er en skjult avhengighet i likningene brukt for å regne ut rotasjon rundt hver akse. Denne avhengigheten gjør det umulig å regne ut rotasjon rundt hver av de tre aksene (X, Y og Z) samtidig.

Fra de tilegnede resultatene, så har vi konkludert med at applikasjonsområdet mellom vår implementerte GF-IMU og en konvensjonell IMU er ulike. Mens en konvensjonell IMU er presis nok til å brukes for treghetsnavigasjons applikasjoner, så kan vår implementerte GF-IMU kun være nøyaktig over et kort tidsrom. Vår GF-IMU er derfor bedre egnet for enkel baneestimering, som for eksempel gest-deteksjon og kursestimering.

Preface

The presented dissertation is an original, unpublished, independent work conducted by the author, E. Hestnes

The idea behind this project came from a specialization project in the NTNU subject TFE4520, conducted during the Autumn semester of 2015. In this project, five ultra-low power MEMS accelerometers was investigated for usage in potential IoT applications. Results from that report yielded valuable insight into the physical limitations of MEMS technology, with regards to mechanical noise, precision and power consumption. From this project, it was also found that inertial sensing would be a very interesting area to explore for potential IoT applications. This idea was further developed in collaboration with Øytein Moldsvor from Disruptive Technologies AS. The result was to study the possibility of creating an low power inertial measurement unit (IMU) using only linear accelerometers.

During the implementation of the EcoIMU algorithm in this thesis, it was discovered a severe limitation with the system. Namely, that there is hidden dependency in the equations used for calculating rotation about each axis. This dependency makes it impossible to calculate the angular velocity about the X, Y and Z-axis at the same time. It was attempted to contact the authors of the EcoIMU publication, asking them about this issue. Prof. Pai H. Chou, which was the supervising professor for the publication, responded a few weeks before the submission deadline. He did not have a clear answer to my specific problem, but he was able to provide me with their original python code. No countermeasures for removing the axis dependency was found in this code, although it was not time to do a thorough investigation.

Trondheim, 2016-06-15

Erlend Hestnes

Acknowledgements

I would first and foremost like to acknowledge the great support my supervisor, Per Gunnar Kjeldsberg, has given me during the writing of this thesis. Per Gunnar has contributed by giving me guidance on the overall report structure, as well as given me useful tips with regards to the created Matlab simulation.

Secondly, I would like to thank Øystein Moldsvor and Disruptive Technologies for helping me concretize the idea behind this dissertation. Øystein has also given me several interesting ideas on possible practical implementations of a GF-IMU.

I would also like to acknowledge my friend Morten Olsen Lysgaard. Morten has a comprehensive background in mathematics from NTNU, and much experience in working with inertial navigation systems. He has therefore been a very useful resource throughout the writing of this thesis.

Finally, I would like to acknowledge Ville Kaajakari, the author of the book Practical MEMS, for personally providing me with two of the original figures from his book.

E.H.

Contents

List of Figures

List of Tables

Abbreviations

ADC Analog Digital Converter
AHRS Attitude Heading Reference System
BW Bandwidth
DK Development Kit
DoF Degrees-of-Freedom
DSP Digital Signal Processing
ENOB Effective Number Of Bits
FIFO First-In First-Out
FPU Floating Point Unit
FSR Full Scale Range/Measurement Range
GF-IMU Gyro-Free Inertial Measurement Unit
IC Integrated Circuit
IMU Inertial Measurement Unit
IoT Internet of Things
LDO Low-Dropout Regulator
LP Low Power
LSB Least Significant Bit
MEMS Microelectromechanical System
MSE Mean Squared Error
ODR Output Data Rate
PSD Power Spectral Density
RMS Root-Mean-Square
SDK Software Development Kit
SNR Signal-to-Noise Ratio
SoC System-on-Chip
SPI Serial Peripheral Interface
TWI Two-Wire Interface

1 Introduction

1.1 Topic

Measuring the motion of an object is regarded as a fundamental function in many of today's sensing applications. An objects motion in a three dimensional space has what is called six degrees of freedom (DoF). These DoF can be decomposed as translation along and rotation about each of the X, Y and Z-axes. These physical quantities can either be measured by external observation or by in-situ motion sensors. External observation includes video motion capture systems, as well as radar and sound measurement. Such systems usually require several sensors placed around the objects viewing angles for a sufficient accuracy [25, p.207]. Another problem with external observation is that an objects motion is constrained by the perimeter of the external sensors. In this dissertation, in-situ motion sensors called inertial measurement units (IMU) are considered. An IMU is a sensor package that usually consists of an accelerometer, a gyroscope and sometimes a magnetometer [26, p.5]. With these sensors, an IMU is able to provide information on both translation and rotation for an object. The first IMUs that existed where large and expensive mechanical contraptions, primarily reserved for military applications and the aerospace industry [1]. The introduction of micro-electromechanical systems (MEMS) has enabled IMUs to undergo miniaturization as well as significant cost reduction [26, p.5]. As a result of this, MEMS IMUs has started to appear in several consumer grade applications.

1.2 Problem Description

The biggest problem with MEMS IMUs that exist on the market today is that they have a reasonably high power consumption. While both MEMS accelerometers and magnetometers are able to operate in the microampere range, gyroscopes usually require a few milliamperes [25, p.212]. This is an unacceptable current consumption for an application that is intended to last a few years on a limited energy supply. Due to the high current consumption in gyro-based IMUs, there has been conducted much research on so-called Gyro-Free IMUs (GF-IMU).

1.3 Justification and Motivation

Several publications has shown that it is possible to forgo the gyroscope by using a combination of multiple accelerometers placed in a special cube configuration [3, 21, 24, 25]. Most of the publications only present theoretical solutions, and the few practical implementations that exist are often too large and impractical to deploy in an actual, commercial product. This thesis aims to address this issue by exploring the possibility of creating a GF-IMU that only consists of a pair of tri-axial accelerometers. The design focus is to create a low-cost solution that occupies a small area, and consumes significantly less power than gyro-based solutions. Being able to implement such a device would have several use-cases for a broad range of IoT applications.

1.4 Research Questions

The accelerometer based GF-IMU solutions that exists today use a combination of multiple accelerometers placed in a special cube configuration. Most of these solutions are impractical due to the number of sensors used and the size of the cube. The practical implementation of a GF-IMU based on two tri-axial accelerometers is relatively new. As such, there are many open questions regarding its true potential.

Three research questions are presented here, and addressed chronologically throughout this work.

1. *What is the relationship between cube geometry and the parameters of precision and power consumption in a GF-IMU configuration?*
2. *How precise and how energy efficient is a GF-IMU implementation compared to a conventional gyro-based IMU?*
3. *Can a GF-IMU be used for any practical IoT applications?*

1.5 Summary of Claimed Contributions

Both a simulation-based approach and a practical implementation have been used to answer the research questions stated above. The main contributions of this thesis are.

- Implemented a Matlab simulation that is able to analyze a GF-IMU configuration over a wide variety of cube geometries.
- Developed a practical testbed of a GF-IMU based on two tri-axial accelerometers. The testbed has been used to compare the precision and energy efficiency between an accelerometer based GF-IMU and a conventional gyro-based IMU.
- Discovered a fundamental limitation with a GF-IMU based on two tri-axial accelerometers. This limitation makes it impossible to calculate rotation about all three axes at the same time using only two accelerometers.
- Developed an improved algorithm for calculating rotation about one free axis of choice.
- Proposed a practical IoT application for a GF-IMU on a conceptual level.

1.6 Thesis Outline

The rest of this thesis is organized as follows:

Chapter 2 - Background Theory: This chapter presents information that is necessary to ease the understanding of the rest of this thesis. This includes a summary of some general knowledge about MEMS accelerometers, gyroscopes and inertial measurement units.

Chapter 3 - Previous Work: Presents a coverage of existing work that was relevant for this dissertation. The chapter emphasizes describing the working principle behind an existing GF-IMU solution called EcoIMU.

Chapter 4 - Methodology: This chapter starts by presenting a theoretical analysis of the EcoIMU implementation. This analysis is further used to describe the design choices behind the created Matlab simulation and practical testbed. The chapter concludes by describing how the simulation and testbed was used to provide results, as well as how the current consumption was measured for the testbed.

Chapter 5 - Results and Discussions: This chapter first presents a theoretical analysis of different cube configuration using the designed Matlab simulation. The testbed is then used to compare the precision between the implemented GF-IMU and a conventional gyro-based IMU. The current consumption between these two configurations is presented at the end.

Chapter 6 - Applications: Here, a practical IoT application for the implemented GF-IMU is presented on a conceptual level.

Chapter 7 - Conclusion: The three research questions from the introduction are answered.

Chapter 8 - Further Work: Presents further improvements for the implemented GF-IMU system, with regards to both precision and power consumption.

2 Background Theory

A summary of some general knowledge about MEMS accelerometers, gyroscopes and inertial measurement units is presented in this chapter. This is necessary to ease the understanding of the rest of this thesis.

2.1 Microelectromechanical Systems (MEMS)

Microelectromechanical Systems (MEMS) are embedded systems involving one or many micromachined components or structures [15, p. 2-3]. The technique is used for a vast number of applications ranging from sensors such as accelerometers, gyroscopes and barometers to solely mechanical structures such as fluid nozzles on an inkjet printer [15, p. 4].

2.1.1 MEMS Accelerometers

The working principle of a MEMS accelerometer can be viewed as a simple mechanical system, as illustrated in Figure 2.1. A proof mass m is connected to a frame by a flexible spring with a spring constant k. When acceleration is imposed along the X-axis, the proof mass will begin to move. Due to Newton's law of inertia, the motion of the proof mass X_m will lag the frame motion X_f [13, p.34]. This will cause an oscillation of the proof mass inside the frame. To prevent excessive oscillation, the mass vibrations are normally damped by introducing a dampening material (such as gas or liquid) inside the package. This is represented by a dashpot γ in Figure 2.1. The motion of the proof mass can then be modeled as a damped harmonic oscillator, and can thus be described by Equation 2.1 [13, p.35]. The acceleration can easily be derived from this equation.

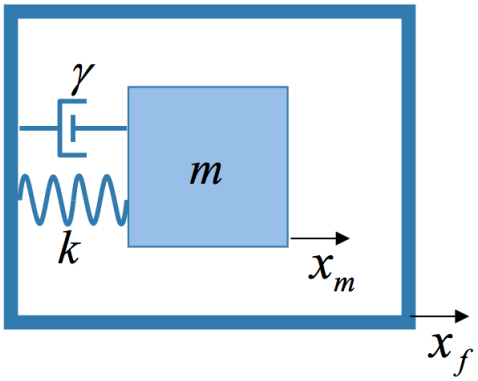

Figure 2.1: Working principle of a MEMS accelerometer [13, p.34]

$$m\frac{\partial^2 x}{\partial t^2} + \gamma\frac{\partial x}{\partial t} + kx = \vec{F} = m\vec{a}_x \tag{2.1}$$

Mechanical Noise in MEMS Accelerometers

The mechanical noise in a MEMS accelerometer is usually specified by the power spectral density (PSD) of the noise equivalent acceleration $\bar{a_n}$, given by Equation 2.2. Here, $4k_bT$ is a constant that represents the thermal noise energy, ω_0 is the mechanical resonant frequency of the system, m denotes the proof mass from Figure 2.1 and Q is the quality factor represented by Equation 2.3 from [13, p.36]. A closer analysis of Equation 2.2 reveals that the thermal noise energy $4k_bT$ is a constant that is unaffected by the system size [13, p.13]. This means that the thermal noise induced in mechanical vibrations sets a lower limit for the smallest, measurable acceleration [13, p.41]. One can also see from Equation 2.2 that increasing the proof mass m and reducing the resonant frequency reduces the overall noise in the system. This observation illustrates one of the main challenges with further miniaturization of MEMS sensors [13, p.42].

$$\bar{a_n} = \sqrt{\frac{4k_bT\omega_0}{mQ}} \tag{2.2}$$

$$Q = \frac{\omega_0 m}{\gamma} \tag{2.3}$$

2.1.2 MEMS Accelerometer Parameters

This section gives an overview, as well as an explanation, of the most common characteristics to look for when selecting a MEMS accelerometer for an embedded application.

Measurement Range

For an analog-to-digital converter (ADC), the full scale range (FSR) is defined as the difference between the maximum V_{max} and minimum V_{min} analog input values that the ADC is able to measure without clipping the signal [14, p.133]. In context with accelerometers, the FSR is usually referred to as measurement range and is defined as the level of acceleration that is supported by the sensor's output signal specification [2]. This range is normally listed as a number n times the earth's gravity (i.e. ±2g , ±4g etc.) in the component datasheet. This number is an upper limit for the accurate measurement range of the part. For example, if the part is rated with a measurement range of ±2g, it means that the part can measure an acceleration accurately up +2g and a retardation down to -2g. If the part is accelerated above this limit, the output might rail or be distorted at the output [2]. Most digital MEMS accelerometers have the ability to select between a set of different measurement ranges, in order to support a wider range of applications. For convenience, the accelerometer measurement range is going to use the FSR abbreviation for the rest of this thesis.

Cross-Axis Sensitivity

A tri-axial accelerometer is able to measure acceleration along the Cartesian coordinate axes of X, Y and Z. When moving such a device along the X-axis, one should ideally only see acceleration on the X-axis. However, due to a combination of misalignment errors, etching inaccuracies and circuit crosstalk there is going to be a small coupling between the different axes. This coupling is referred to as cross-axis sensitivity, and is usually specified as a percentage number in the component datasheet [2].

Sensitivity

The accelerometer sensitivity is defined as the sensor's ratio of change in mechanical input to change in electrical output. Ideally, the ratio between the acceleration and the sensor output should be linear. In practice, all MEMS accelerometers suffers from non-linearity due to mechanical stresses and circuit temperature coefficients [2]. For digital accelerometers, the sensitivity is usually specified at a specific FSR as units of mg/LSB or counts/g. The sensitivity is dependant on two factors, the selected FSR and the number of bits n in the embedded ADC. The relationship can be seen in Equation 2.4 and Equation 2.5.

$$\text{Sensitivity} = \frac{FSR}{2^n} = \text{mg/LSB} \tag{2.4}$$

$$\text{Sensitivity} = \frac{2^n}{FSR} = \text{counts/g} \tag{2.5}$$

The sensitivity provides information on the smallest measurable acceleration step. For example, if an accelerometer has a rating of 1mg/LSB it means that a change in the LSB corresponds to an acceleration of 0.001 g's (1mg). The sensitivity also acts as a scale factor between the accelerometer raw data and the number of g's

Output Data Rate

The output data rate (ODR) defines the data sample rate for digital accelerometers. The bandwidth (BW) is defined as the highest frequency signal that can be sampled without any aliasing by the specified ODR. As specified by the Nyquist criterion, the bandwidth is half the output data rate [2], as seen in Equation 2.6. Digital MEMS accelerometers usually have the ability to change between ranges of different ODR. Selecting a lower ODR usually translates into lower power consumption for the part.

$$BW = \frac{ODR}{2} \tag{2.6}$$

Most digital accelerometers have the ability to improve their precision by using something called oversampling. In oversampling mode, the accelerometer uses an average of several samples in order to reduce the noise [9, p.6]. An apparent drawback with this approach is that it introduces a latency in the reaction time. The reaction time slows down with a rate proportional to the number of samples taken for the average. Oversampling also requires the device to use an ODR that is more than twice the bandwidth of the sampled motion signal. Oversampling can therefore be used as a means to trade precision for power [9, p.6-7].

Noise

The noise in a MEMS accelerometer comes from mechanical noise in the MEMS element, as described in Section 2.1.1, and as electronic switching noise in the integrated circuit (IC). The overall noise in the system is commonly given by the power spectral density (PSD) in the accelerometer datasheet, usually in the units of $\mu g\sqrt{Hz}$. The PSD captures the frequency content of a stochastic process (noise) and describes how the power is distributed across different frequencies. The

noise in an accelerometer is predominantly considered to be Gaussian white noise, and is thereby a constant value across all frequencies [9, p.3-4]. The relationship between the PSD and the root-mean-square (RMS) noise is given by Equation 2.7, and can be simplified to 2.8 [9, p.3].

$$N_{\text{rms}}^2 = \int_0^{BW} PSD(f)df \tag{2.7}$$

$$N_{\text{rms}} = \sqrt{PSD \cdot BW} \tag{2.8}$$

According to [8, p. 3], the PSD noise value that is listed in the component datasheets is commonly listed as the square root PSD from the derivation in Equation 2.8, which in turn produces Equation 2.9.

$$N_{\text{rms}} = PSD \cdot \sqrt{BW} \tag{2.9}$$

The full derivation of Equation 2.7 to Equation 2.8 can be found in Appendix A.

Digital Resolution

The digital resolution of an accelerometer is normally specified as the number of bits in the embedded ADC. The devices that are available today typically range between 12 and 16-bits [7, p.16]. Although, this number might sometimes be misleading. Every MEMS accelerometer suffers from system noise to a certain extent, which in turn will limit the number of effective bits in the ADC [9, p.3]. In other words, it does not help to have a 16-bit ADC if four of the lower bits are full of noise. The number of effective bits (ENOB) is possible to calculate if the system noise is specified in the datasheet. One first needs to calculate the system noise for a specific measurement range (FSR) and bandwidth (BW). This can be done by using Equation 2.10 together with Equation 2.9. By combining Equation 2.11 with Equation 2.10 one is able to calculate the number of effective bits [9, p.5].

$$SNR(\text{dB}) = 20\log\frac{\frac{FSR}{2\sqrt{2}}}{N_{\text{rms}}} \tag{2.10}$$

$$ENOB = \frac{SNR(\text{dB}) - 1.76}{6.02} \tag{2.11}$$

The full derivation of Equation 2.11 can be found in Appendix A.

2.1.3 MEMS Gyroscopes

A traditional mechanical gyroscope consists of a spinning disc mounted on two gimbals, as seen in Figure 2.2. The two gimbals allow the spinning disc to rotate freely about all three axes. By the laws of conservation, the disc will try to keep its angular momentum at all times. As an effect of this, the spinning disc will try to resist any change in orientation. Hence, if the disc is spinning and the system is exposed to a rotation, only the angles between the adjacent gimbals will change. The

disc itself will remain at a constant orientation. The angles between adjacent gimbals can be used to measure the orientation [26, p.8-9].

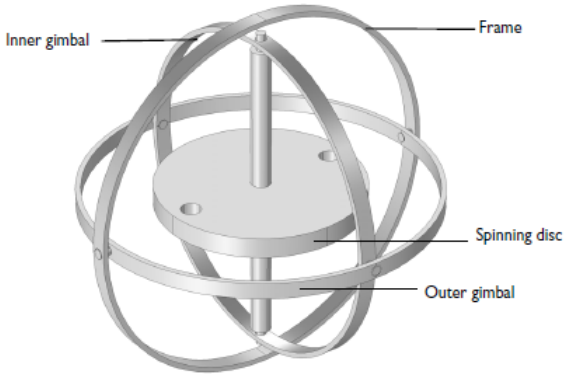

Figure 2.2: Illustration of a conventional mechanical gyroscope [26, p.8]

While conventional gyroscopes are used to measure orientation, all MEMS gyroscopes measures angular velocity by making use of the Coriolis force [26, p.8]. The Coriolis force affects all objects that move in a rotating frame [13, p.346]. Figure 2.3 illustrates a mass m that is thrown out in straight line with a velocity $\vec{v}$ from the origin of a disc with a direction towards point B. If the disc is not rotating, and no other forces are acting on the mass, the mass will travel in a straight line and hit point B. Now, if we let the disc rotate in a counterclockwise direction with an angular velocity $\vec{\omega}$, point B would have moved by the time the mass reaches the circumference of the disc. So the mass would hit point A on the disc instead. In the eyes of an observer on the spinning disc, the path of the mass appears curved, as illustrated by the dotted line in Figure 2.3. An external observer would on the other hand see the path as being straight. Because the Coriolis force is only observed in the rotating frame, it is sometimes referred to as being a fictional force. The Coriolis force is however measurable within the rotating frame, and there is therefore nothing fictional about it [13, p.347].

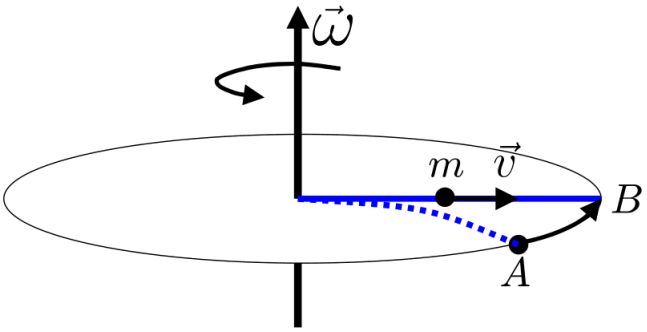

Figure 2.3: The Coriolis force [13, p.346]

In essence, the Coriolis effect states that in a frame of reference rotating at angular velocity $\vec{\omega}$, a mass m moving with velocity $\vec{v}$ experiences a force $\vec{F}_c$ [26, p.8]. This relationship can be seen in Equation 2.12.

$$\vec{F}_c = -2m(\vec{\omega} \times \vec{v}) \tag{2.12}$$

MEMS Gyroscope - Working Principle

The working principle of most MEMS gyroscopes can be viewed as a simple mechanical system consisting of two orthogonal vibration modes, as illustrated in Figure 2.4. A proof mass m is connected by a two-spring configuration inside a frame. The proof mass is then free to move along the X_1-axis. Movement along this axis is referred to as a drive motion. The frame itself is also connected in a two-spring configuration that is orthogonal to the drive-motion. By such, the frame is free to move along the X_2-axis. Movement along this axis is referred to as sense vibrations. During measurement mode, the proof mass is excited to vibrate in the drive-mode along the X_1-axis. This is usually done via electrostatic actuation [13, p.349]. If no angular rotation is applied to the device, movement along the sense-mode direction is essentially zero. During a rotation, the Coriolis force will act on the frame along the sense-mode direction. The sense-mode vibrations amplitude then becomes proportional to the angular velocity $\vec{\omega}$ [13, p.348].

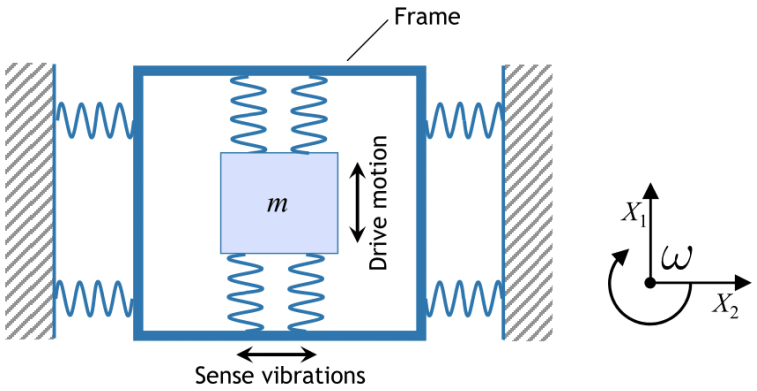

Figure 2.4: The working principle of a vibrating two-mode gyroscope [13, p.348]

2.1.4 MEMS Error Sources

This section presents the most common errors that arise in MEMS gyroscopes and accelerometers, as well as their effect on the integrated signal. The presented theory in this section is solely based on [26, p.10-17].

Constant Bias Error

The output value from a tri-axial MEMS gyroscope should ideally be zero for all axes when it is stationary, as it is not undergoing any rotation. In practice, all MEMS based gyroscopes output a constant offset value when stationary. This constant offset is referred to as a bias error ϵ, and is often a result of manufacturing tolerances and temperature sensitivity in the materials. Since the angular velocity from a gyroscope usually is integrated to get the angle of orientation, ϵ will grow linearly with time t. This will produce an angular error $\theta_\epsilon(t)$ as seen in Equation 2.13.

$$\theta_\epsilon(t) = \epsilon \cdot t \tag{2.13}$$

Fortunately, the bias error can easily be removed by taking a long-term average of the gyro's output whilst the gyroscope is not undergoing any rotation. Once the bias value is obtained, it can easily be subtracted from the measurements before integration.

The output value from a tri-axial MEMS accelerometer should ideally be zero on the X and Y-axis, and $1g$ on the Z-axis when lying perfectly flat. The contribution on the Z-axis is due to the constant pull from the earth's gravity. As with MEMS gyroscopes, MEMS accelerometers are also affected by a bias error ϵ on all axes. If the acceleration is integrated to find the velocity, the error will grow linearly with time, as seen in Equation 2.14. If the linear acceleration is integrated twice in order to find position, the error will grow with a rate proportional to the time squared, as seen from Equation 2.15. The same approach used for removing gyroscope bias can be used for accelerometers as well. That is, compute an initial bias when the device is stationary and subtract it from the accelerometer readings.

$$v_\epsilon(t) = \epsilon \cdot t \tag{2.14}$$

$$s_\epsilon(t) = \epsilon \cdot \frac{t^2}{2} \tag{2.15}$$

Temperature Effects

All materials are affected by temperature changes to some extent, due to the fact that the kinetic energy between molecules increases as the temperature increases. Both external environmental heating as well as internal self-heating of the MEMS element are some of the biggest contributors to the bias error in Equation 2.13. As mentioned in Section 2.1.4 it is easy to remove the bias error by taking an initial long term average of the samples, then subtracting it before integration. However, this approach only works well if the temperature remains somewhat constant. If the MEMS element is exposed to a drastic temperature change, the bias is going to change, thus rendering the bias correction value useless. To compensate for temperature changes, it is common for modern IMUs to include an on-board temperature sensor. This enables either the user or the hardware to perform temperature corrections on the output data.

Calibration Errors

Calibration errors are a collective term that refers to errors associated with scale factors, sensor alignments and the linearity of the MEMS sensor. A big problem with calibration errors is that they tend to produce errors whilst the device is moving. This makes it hard to compensate for such errors. It is difficult to describe this error using mathematical equations, as it is dependent on so many factors. However, the component manufacturer usually specifies sensor tolerances and linearity.

2.2 Digital Compass

The compass is one of the world oldest navigational instruments. It is used to show orientation relative to the earth's magnetic field. The traditional compass consists of a magnetic needle attached to a pivot point. The needle aligns itself with the horizontal component of the earth's magnetic field, thus providing a heading relative to the magnetic north [22].

Modern digital compasses, often called magnetometers, utilize solid-state technology instead of moving parts. In the recent years, these devices have undergone a miniaturization and are now available as very low cost integrated circuits. Most of the electronic compasses that exists today

either uses a Hall-effect sensor or a magnetoresistive material to sense the magnetic field lines. A Hall effect sensor is essentially a transducer that varies its output voltage in response to a magnetic field. The sensed voltage can then be converted to a digital signal, and further used to represent the magnetic field intensity. Magnetoresistivity is an ability some material have to change resistance under the influence of a magnetic field [11, p.1-2]. The change in resistance can be measured and used to represent the magnetic field intensity.

Hall effect sensors responds to magnetic field lines that are perpendicular to the sensor, while magnetoresistive sensors responds to field lines that are parallel to the sensor, as illustrated in Figure 2.5. This is the main application difference between a Hall-effect sensor and a magnetoresistive sensor [11, p.2].

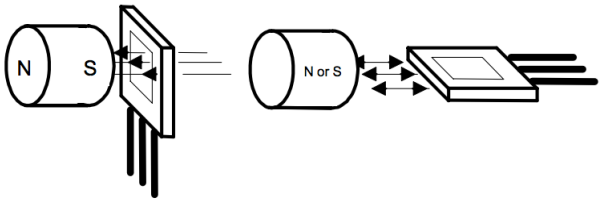

Figure 2.5: Hall-effect sensor vs magnetoresistive sensor [11, p.2]

2.3 Inertial Measurement Unit

An inertial measurement unit (IMU) is a system that is used for measuring the motion of an object in free space relative to an inertial frame [25, p.1]. A typical IMU can be regarded as a sensor package consisting of an accelerometer, a gyroscope and sometimes a magnetometer. Each of these sensors are usually tri-axial, meaning that they are able to output values either along or about three different axes (X, Y and Z) [26, p.5]. It is common for IMUs to denote each of these measurement axes as a degree of freedom (DoF). An IMU that only consists of a tri-axial accelerometer and gyroscope is therefore denoted as a 6 DoF device, while an IMU that also includes a tri-axial magnetometer is denoted as a 9 DoF device.

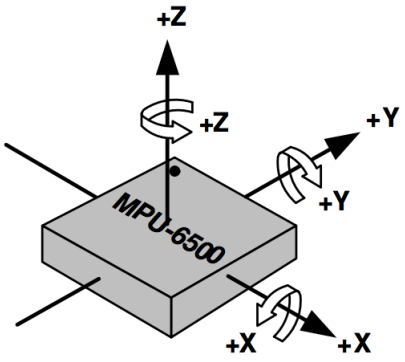

Figure 2.6: Illustration of a 6 DoF IMU from Invensense [12]

The first IMUs that existed where large and expensive mechanical contraptions, primarily reserved for military applications and the aerospace industry [1, p.1-2]. The introduction of MEMS has enabled IMUs to undergo miniaturization as well as a significant cost reduction [26, p.5].

2.3.1 IMU - Working Principle

By definition, an IMU must be able to provide the motion of an object [25, p.1]. An objects motion can be decomposed into two kinematic components, translation and rotation. The following sections presents the most common approach for measuring these components using a conventional 6 DoF IMU, like the one illustrated in Figure 2.6.

Rotation

The gyroscope is used to compute rotation in an IMU. A tri-axial gyroscope outputs angular velocity $\vec{\omega}$ about the X, Y and Z-axis. Angular velocity is defined as the rate of change in angular displacement, as seen in Equation 2.16, and is represented as a vector quantity [10, p.314].

$$\vec{\omega}(t) = \frac{d\theta}{dt} \tag{2.16}$$

By using integration on the angular velocity, and assuming that the initial velocity is zero, one is able to obtain the angle of rotation $\vec{\theta}$, as seen in Equation 2.17.

$$\vec{\theta}(t) = \int_0^t \frac{d\theta}{dt} dt \tag{2.17}$$

The angle of rotation about the X, Y and Z-axis is more commonly denoted as roll ψ, pitch θ and yaw ϕ. This is illustrated in Figure 2.7. With roll, pitch and yaw one has sufficient information to find an objects orientation (attitude). If one thinks of the paper plane in Figure 2.7 as a vector, then this vector becomes the attitude.

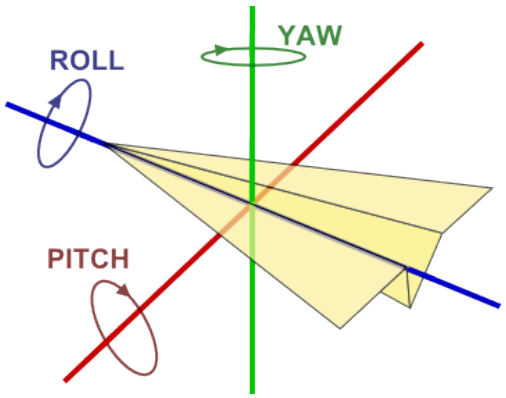

Figure 2.7: Rotation about the X, Y and Z-axis [27]

Linear Translation

The accelerometer in an IMU is used to compute linear translation. A tri-axial accelerometer outputs linear acceleration $\vec{a}$ along the X, Y and Z-axis. Acceleration is defined as the rate of change in velocity with respect to time. It is given by a vector with both magnitude and direction relative to a reference frame. The SI-units are length divided by time squared [10, p.412-413].

By integrating the linear acceleration with respect to time one is able to obtain the linear velocity, as seen in Equation 2.18.

$$\vec{v}(t) = \int_0^t \vec{a}(t)dt \tag{2.18}$$

Integration of the linear velocity with respect to time yields the linear position, as seen in Equation 2.19. By such, one is able to determine an objects displacement, as illustrated with the dotted line in Figure 2.8.

$$\vec{s}(t) = \int_0^t \vec{v}(t)dt \tag{2.19}$$

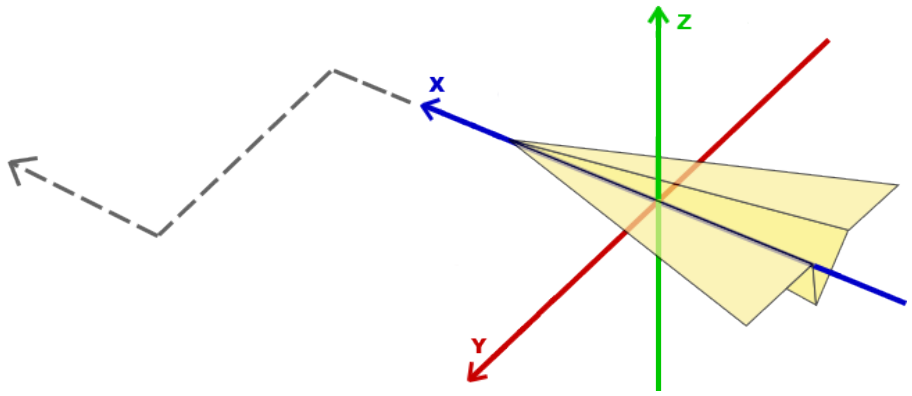

Figure 2.8: Linear translation along the X, Y and Z-axis [27]

Calculating Roll and Pitch Using Gravity

As mentioned, a single tri-axial accelerometer is commonly used to measure linear translation along the X, Y and Z-axis. But, it is also possible to calculate roll and pitch from just one tri-axial accelerometer [18]. This is achieved by making use of the constant gravitational pull on the device. By looking at the gravitational contribution along the different axes of the accelerometer, denoted here as G_x, G_y and G_z, it is possible to calculate an angle between them. The angle between Y and Z corresponds to roll ψ, while the angle between X and Z corresponds to pitch θ. This approach does have some limitations. First off all, the approach assumes that the gravitational pull is the only acceleration affecting the device, meaning that it also assumes that the device is undergoing zero linear translation. The pitch angle is also limited to only measure between -90 and +90 degrees, and roll cannot be calculated if the pitch angle is either +90 or -90 degrees. The approach is also not able to provide any information about the yaw angle. The formulas for calculating roll ψ and pitch θ from gravity can be seen in Equation 2.20 and 2.21 [18, p13-16].

$$\psi = \arctan\left(\frac{G_y}{\sqrt{G_x^2 + G_z^2}}\right) \tag{2.20}$$

$$\theta = \arctan\left(\frac{-G_x}{\sqrt{G_y^2 + G_z^2}}\right) \tag{2.21}$$

2.3.2 AHRS - Attitude Heading Reference System

Even though it is possible to obtain an attitude vector (roll, pitch and yaw) from just integrating the angular velocity from a single gyroscope, it is rarely used. This is because gyroscopes suffers from several error sources, as listed in Section 2.1.4. These error sources will produce inaccuracies in the data provided by the gyroscope. Since the attitude vector is calculated by integration, these errors will accumulate. Over time, this will cause the integrated attitude vector to eventually drift away. To mitigate for this problem, it is common to combine sensor data from a gyroscope together with data provided by an accelerometer. Data from both sensors are then used to produce a more accurate attitude estimate. Combining sensor data this way, is usually referred to as sensor fusion [26, p.33]. When sensor fusion is used to provide an attitude estimate, it is commonly referred to as being an attitude heading reference system (AHRS) [20].

The simplest form of an AHRS only use data from a tri-axial gyroscope and a tri-axial accelerometer. MEMS gyroscopes respond fast, which makes them well suited for capturing rapid changes in angle [20]. On the downside, they suffer from several error sources which causes their integrated angle to eventually drift over time, as discussed in the previous section. This makes them unsuited for determining the angle of rotation over an extended period of time. From Section 2.3.1 we know that a MEMS accelerometer can be used to provide a fairly accurate reading of roll and pitch, as long as there is no linear translation applied to the system. An AHRS uses the gravity calculated roll and pitch from the accelerometer as a reference to correct the roll and pitch estimate from the gyroscope. This is usually done once the rotation has settled. For a slightly better attitude estimation, it is common for an AHRS to also include a tri-axial magnetometer. As discussed in Section 2.2, a magnetometer outputs a heading relative to the earth's magnetic field. This can be used as a reference for the yaw estimate, as this is an angle that cannot be calculated from just using an accelerometer.

3 Previous Work

This chapter presents results from the literature study carried out during the initial phase of this thesis. This includes the most relevant results from the specialization project, as well as a study of existing GF-IMU solutions based on linear MEMS accelerometers.

3.1 TFE4520 - Specialization Project

The presented dissertation is a continuation of a specialization project in the NTNU subject TFE4520, conducted during the autumn semester of 2015. In this project, five ultra-low power MEMS accelerometers were investigated for usage in potential IoT applications, these devices are shown in Table 3.1.

Device	**ADXL362**	**MMA8491Q**	**MC3610**	**LIS3DH**	**KX123**
Manufacturer	Analog Devices	Freescale Semiconductor	mCube	STMicroelectronics	Kionix
Supply Voltage	1.6-3.5V	1.95-3.6V	1.6-3.6V	1.71-3.6V	1.71-3.6V
Shutdown current	10nA , $V_{DD} = 2.0V$	1.8nA , $V_{DD} = 2.8V$	400nA , $V_{DD} = 1.8V$	500nA , $V_{DD} = 2.5V$	900nA , $V_{DD} = 2.5V$
ODR current	1.8/13µA [2] $V_{DD} = 2.0V$, 100Hz ODR	20µA [1] $V_{DD} = 2.8V$, 100Hz ODR	6/9/26µA, (FIFO On) [4] 3/6/17 µA, (FIFO Off) [4] $V_{DD} = 1.8V$, 100Hz ODR	20/10µA [3] $V_{DD} = 2.5V$, 100Hz ODR	21µA $V_{DD} = 2.5V$, 100Hz ODR
Spectral Noise (X,Y)	550 / 250µg/$\sqrt{Hz}$ [2] $V_{DD} = 2.0V$,100Hz ODR	406µg/$\sqrt{Hz}$ $V_{DD} = 2.8V$,100Hz ODR	560/400/204µg/$\sqrt{Hz}$ [4] $V_{DD} = 1.8V$,100Hz ODR	220 / N.A. µg/$\sqrt{Hz}$ [3] $V_{DD} = 2.5V$,100Hz ODR	106 µg/$\sqrt{Hz}$ $V_{DD} = 2.5V$,100Hz ODR
Digital Resolution	12-bit	14-bit	14-bit	16-bit	16-bit
Measurement range	±2,4,8g	±8g	±2,4,8,12,16g	±2,4,8,16g	±2,4,8g

Table 3.1: Comparison of five ultra-low power accelerometers [7, p.16].

Results from this report yielded valuable insight into the physical limitations of MEMS technology, with regards to mechanical noise, precision and power consumption. From this project, it was also found that inertial navigation would be a very interesting area to explore for potential IoT applications. From this finding, it was decided to further investigate the possibility of creating a GF-IMU using linear accelerometers.

3.2 GF-IMU Using Linear Accelerometers

As mentioned in Section 2.3.2, most conventional consumer grade IMUs today use a gyroscope to measure rotation and an accelerometer to measure linear translation. Data from these sensors are usually processed further by using an AHRS to produce an attitude estimate. With regards to precision, this is an approach that works very well. In terms of power consumption and price, it is not an optimal solution.

Traditionally, gyroscopes have always been more complex devices than accelerometers, and have therefore been more difficult to manufacture. As a result of this, gyroscopes have traditionally

[1] Specified at 400nA/Hz in datasheet
[2] Normal mode/Ultra-low noise mode
[3] Normal mode/Low power mode
[4] Ultra-low power mode/Low power mode/Precision mode
[5] Low power mode/High resolution mode

been more expensive than accelerometers. Even though the price of gyroscopes has dropped significantly in recent years, one can still get a tri-axial MEMS accelerometer at a fraction of the price compared to a MEMS gyroscope [4, 5]. The biggest difference between the two is however the power consumption. While a typical MEMS accelerometer can be in the microampere range when measuring, a gyroscopes is usually in the milliampere range [12, p.8]. From Section 2.1.3 we know that MEMS gyroscopes requires electrostatic actuation of the proof mass to take measurements. This actuation requires a substantial amount of energy, and is the main reason that gyroscopes consumes much more power than accelerometers [13, p.349].

Because of the apparent price and power drawback of using a gyroscope in a IMU, there has been done much research on so called Gyro-Free IMUs (GF-IMU) using only linear accelerometers. From a mathematical point of view, it was known for a long time that minimum of six uni-axial accelerometers was required to get a complete description of a rigid body motion [25, p.207-208]. Although, this was not realized until Chen et al. [3] proposed a practical working solution using a total of six uni-axial accelerometers placed at the center on each face of a bounding cube, as seen in Figure 3.1. The sensing axis of each accelerometer is placed along the diagonal of each face on the cube; by such the diagonals form a regular tetrahedron.

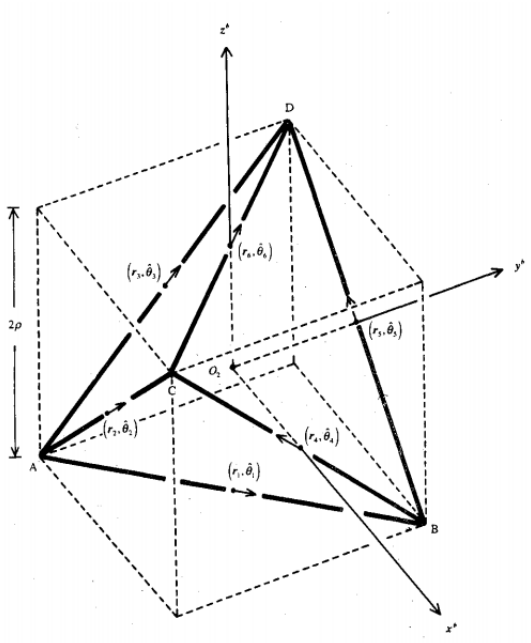

Figure 3.1: Chen's GF-IMU configuration [3].

3.3 EcoIMU

Since Chen's publication, there has been proposed several models that aim to both simplify and improve the accuracy of the cube model, some of which are found in [3, 21, 24, 25]. One of the more practical schemes are found in a IEEE publication called EcoIMU [25]. The paper was published in the 2010 International Conference on Body Sensor Networks. In the EcoIMU publication, a model using only two tri-axial accelerometers placed at opposite corners of a bounding cube are proposed, as seen in Figure 3.2. By reducing the number from six uni-axial accelerometers down to a pair of tri-axial accelerometers makes for a solution that is more convenient to deploy in a practical application. The EcoIMU publication claims to be able to make a solution that in some applications are more accurate than a conventional gyro-based IMU. The EcoIMU implementation will be discussed further in the following sections. All of the presented theory and equations in these sections are from [25], although the mathematical notation and figures has been slightly changed for convenience.

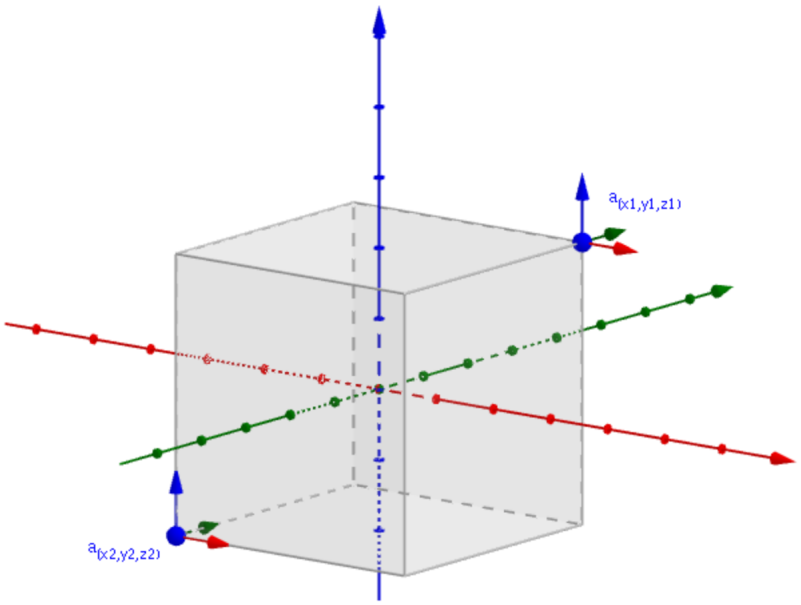

Figure 3.2: EcoIMU cube configuration.

3.3.1 Working Principle

The basic working principle behind the EcoIMU algorithm is that the corner accelerometers in Figure 3.2 experiences an acceleration that consists of a translational component $\bar{a}$ and rotational component a_d. The total acceleration from each device can therefore be given by Equation 3.1 and 3.2.

$$a_{x1} = \bar{a}_x + a_{dx} \quad a_{y1} = \bar{a}_y + a_{dy} \quad a_{z1} = \bar{a}_z + a_{dz} \tag{3.1}$$

$$a_{x2} = \bar{a}_x - a_{dx} \quad a_{y2} = \bar{a}_y - a_{dy} \quad a_{z2} = \bar{a}_z - a_{dz} \tag{3.2}$$

The center point of the cube does not experience any acceleration contribution from rotation, which means that the center point only experiences acceleration from translation. The center point translation can easily be calculated by taking the average of the acceleration vectors from both accelerometers, as seen in Equation 3.3.

$$\bar{a}_{x,y,z} = \left(\frac{a_{x1} + a_{x2}}{2}, \frac{a_{y1} + a_{y2}}{2}, \frac{a_{z1} + a_{z2}}{2} \right) \tag{3.3}$$

The authors behind the EcoIMU implementation further states that the angular acceleration about the X, Y and Z-axis can be found by respectively creating enclosing cylinders around the XY, YZ and XZ projections of the bounding cube. An example of the enclosing cylinder in the XZ-projection plane can be seen in Figure 3.3. The diameter of each bounding cylinder is given by the Pythagorean theorem in Equation 3.4.

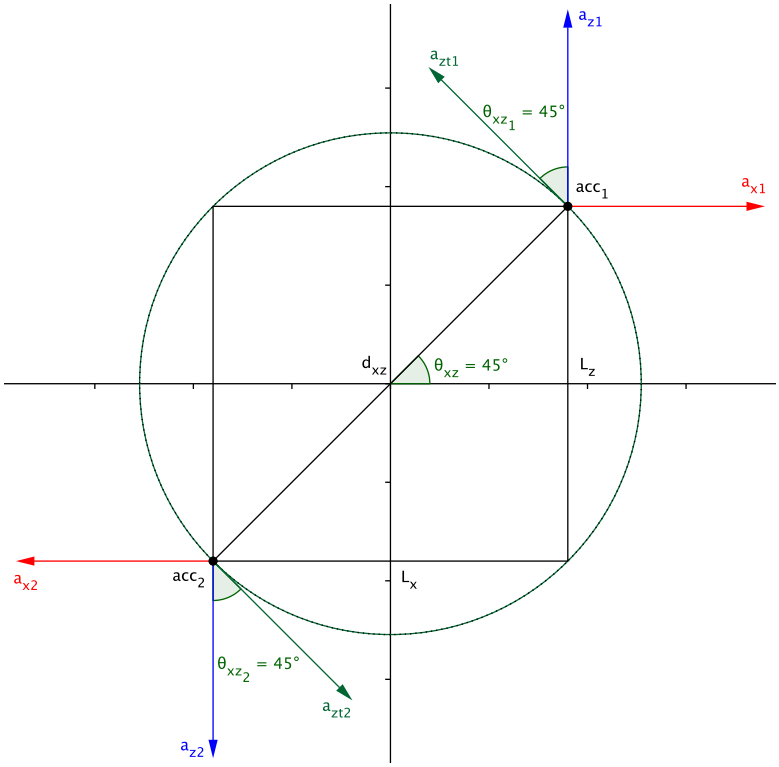

Figure 3.3: Illustration of bounding cylinder in the XZ-projection plane.

$$d_{xy,yz,xz} = \left(\sqrt{L_x^2 + L_y^2}, \sqrt{L_y^2 + L_z^2}, \sqrt{L_x^2 + L_z^2} \right) \tag{3.4}$$

The tangential acceleration is then found by simply decomposing the acceleration vector along the circumference of the enclosing cylinder, as illustrated with the green arrows in Figure 3.3. A different acceleration axis is assigned to each enclosing cylinder. In the EcoIMU implementation, they decide to use the X-axis for the XY-plane, the Y-axis for the YZ-plane and the Z-axis for the XZ-plane. Although, other combinations should also be possible, as two acceleration vectors are available for each projection. The tangential angle for each axis is determined from the shape of the projection plane, and is calculated by using Equation 3.5

$$\theta_{xy,yz,xz} = \left(\arcsin\left(\frac{L_x}{d_{xy}}\right), \arcsin\left(\frac{L_y}{d_{yz}}\right), \arcsin\left(\frac{L_z}{d_{xz}}\right)\right) \tag{3.5}$$

The tangential acceleration for the XY, YZ and XZ-plane can then be calculated using Equation 3.6, 3.7 and 3.8.

$$a_{xy_{t1}} = a_{x1}\cos(\theta_{xy}) \quad a_{xy_{t2}} = a_{x2}\cos(\theta_{xy}) \tag{3.6}$$

$$a_{yz_{t1}} = a_{y1}\cos(\theta_{yz}) \quad a_{yz_{t2}} = a_{y2}\cos(\theta_{yz}) \tag{3.7}$$

$$a_{xz_{t1}} = a_{z1}\cos(\theta_{xz}) \quad a_{xz_{t2}} = a_{z2}\cos(\theta_{xz}) \tag{3.8}$$

The linear acceleration from the rotation then becomes the difference between the two tangential accelerations, as seen in Equation 3.9.

$$a_{d_{xy}} = \frac{a_{xy_{t1}} - a_{xy_{t2}}}{2} \quad a_{d_{yz}} = \frac{a_{yz_{t1}} - a_{yz_{t2}}}{2} \quad a_{d_{xz}} = \frac{a_{xz_{t1}} - a_{xz_{t2}}}{2} \tag{3.9}$$

The angular acceleration is the linear acceleration normalized to unit radius. The radius in this case will be half the diameter of each bounding cylinder. The angular acceleration about the X, Y and Z-axis is then given by Equation 3.10.

$$\alpha_{zxy} = \left(\frac{2a_{d_{xy}}}{d_{xy}}, \frac{2a_{d_{yz}}}{d_{yz}}, \frac{2a_{d_{xz}}}{d_{xz}}\right) \tag{3.10}$$

By integrating the angular acceleration with respect to time, and assuming zero initial velocity, one is able to obtain angular velocity ω, as seen in Equation 3.11. This is the same information that a gyroscope provides. The angular velocity can be integrated again to provide an angle of orientation, as seen in Equation 3.12, 3.13 and 3.14. This angle is more commonly known as roll ψ, pitch θ and yaw ϕ.

$$\omega_{zxy}(t) = \omega_{zxy}(t - \Delta t) + \Delta t \alpha_{zxy}(t) \tag{3.11}$$

$$\psi(t) = \psi(t - \Delta t) + \Delta t \omega_x(t) \tag{3.12}$$

$$\theta(t) = \theta(t - \Delta t) + \Delta t \omega_y(t) \tag{3.13}$$

$$\phi(t) = \phi(t - \Delta t) + \Delta t \omega_z(t) \tag{3.14}$$

From the equations above, it is important to note that the proposed GF-IMU requires two integration steps to compute the angle of rotation. A conventional gyro-based IMU only requires one, as the gyroscope by default outputs angular velocity. It is therefore highly unlikely that the angle estimate produced from the GF-IMU is going to be more accurate than that of a conventional IMU. Remember from Section 2.1.4 that the bias error is going to be proportional to the time squared for a double integration of acceleration.

3.3.2 Hardware Implementation

The EcoIMU publication also presents a hardware implementation on which they run their GF-IMU algorithm. The EcoIMU hardware implementation consists of two wireless sensor nodes called Econodes, depicted in Figure 3.4. Each sensor node is comprised of a Hitachi Metals H34C tri-axial analog accelerometer, a Nordic nRF24E1 integrated radio and 8051-based microcontroller and an 80mAh rechargeable lithium polymer battery. The 12-bit, 143samples/sec internal ADC of the nRF24E1 is used to sample the accelerometer. Data from the sensor nodes is transmitted wirelessly to a gateway, which is connected to a computer. All of the computation is done on an external computer using Python. The Econode only occupies $1cm^3$ in volume and weighs less than 2 grams.

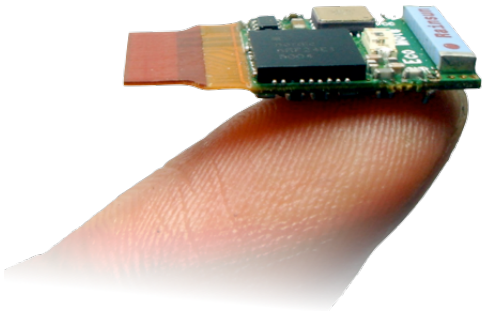

Figure 3.4: Picture of an Econode [6]

The authors behind the EcoIMU publication suggests in their future work section to move the algorithm processing onto the embedded board, thereby removing the need for an external computer.

4 Methodology

This chapter first presents a theoretical evaluation on how different cube geometries affects the parameters of the EcoIMU implementation from Section 3.3. This consecutively leads to a section about the design considerations behind the created GF-IMU Matlab simulation. The chapter continues by describing how the GF-IMU testbed was implemented. At the end, the chapter gives a description on how the simulation and testbed was used to provide results.

4.1 Theoretical Evaluation

IoT applications often have strict requirements when it comes to parameters such as physical size, power consumption and cost. For an accelerometer based GF-IMU configuration, all of these parameters have an impact on each other. This section will therefore provide a further investigation on how the physical parameters of the cube configuration in Figure 3.2, such as size and placement of the accelerometers, impacts the overall precision and power consumption of the system.

4.1.1 The Gravity Component

From Section 2.3.1 we know that all MEMS accelerometers experiences a constant contribution of 1g from the earth's gravitational pull. If the sensor is lying perfectly flat, only the Z-axis of the device is going to be affected by this. However, once the device is tilted about either the X- or Y-axis, the gravity vector is going to be decomposed along the other two axes as well.

The EcoIMU publication from Section 3.3 does not include any information about the gravity component in their equations. It was therefore necessary to perform a theoretical analysis on these equations in order to better understand what would happen to the gravity component in the EcoIMU algorithm.

The gravity component can be regarded as a constant offset that is present for all three axes (G_x, G_y and G_z) on both of the accelerometers on the bounding cube. Thus, Equation 3.1 and 3.2 can be remodeled as Equation 4.1 and 4.2

$$a_{x1} = \bar{a}_x + a_{dx} + G_x \quad a_{y1} = \bar{a}_y + a_{dy} + G_y \quad a_{z1} = \bar{a}_z + a_{dz} + G_z \tag{4.1}$$

$$a_{x2} = \bar{a}_x - a_{dx} + G_x \quad a_{y2} = \bar{a}_y - a_{dy} + G_y \quad a_{z2} = \bar{a}_z - a_{dz} + G_z \tag{4.2}$$

From Equation 3.6, 3.7 and 3.8 we can see that the angular acceleration is calculated by taking the difference between the tangential acceleration vectors of both accelerometers. This means that the gravitational contribution is going to disappear in this stage. Although, it is worth noting that the gravity component might affect the angular velocity estimate if the two sensors are misaligned, as this would cause the gravity offset to be different on both devices.

4.1.2 Measured-, Tangential- and Angular Acceleration

The EcoIMU implementation differentiates between three types of acceleration. The first term is referred to as measured acceleration, denoted as $\vec{a}$, and is the raw data that is read directly from the MEMS accelerometer. The second term is the tangential acceleration $\vec{a}_t$, and is the measured acceleration decomposed along the circumference of the enclosing cylinder of the bounding cube. Both the measured acceleration (blue arrow) and the tangential acceleration (green arrow) are shown for the XZ-plane in Figure 3.3. The last type of acceleration is referred to as angular acceleration α, and is the tangential acceleration normalized to unit radius.

What Affects Measured Acceleration?

Now, let's continue in describing how the shape of the bounding cube affects these three types of acceleration. Consider the situation in Figure 4.1, where two accelerometer pairs facing an opposite direction from each other are placed on opposite sides of a plane. Now assume that the plane rotates 90 degrees in a counterclockwise direction. In this case, the accelerometers at the edge would need to traverse a longer path in the same period of time compared to the accelerometers closer to the center. Because of this, the accelerometers at the edge would experience a larger measured acceleration than the ones placed closer to the center. This example clearly illustrates the relationship between measured acceleration during a rotation and the sensors distance from the origin of the cube.

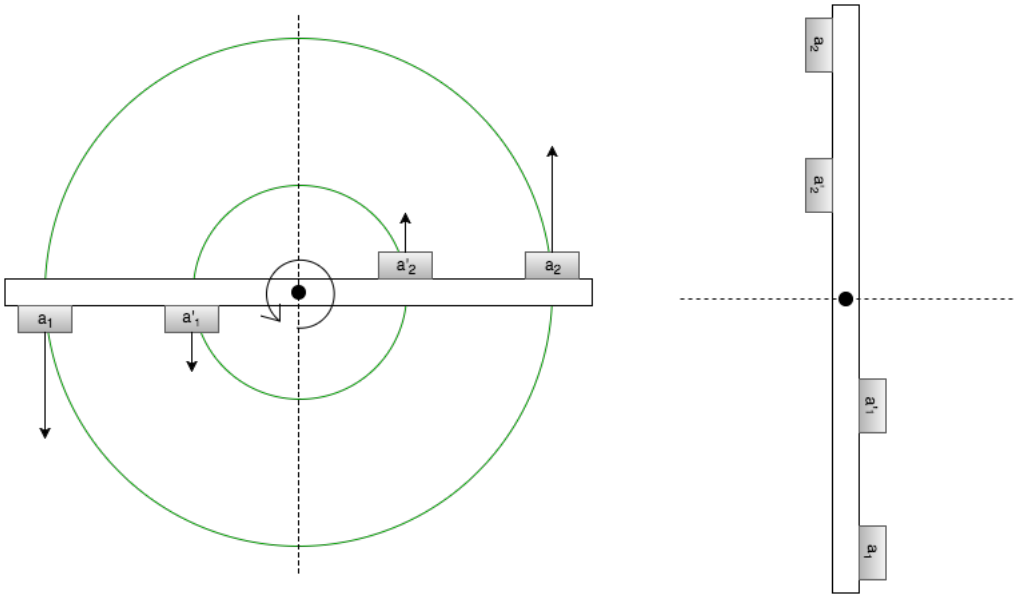

Figure 4.1: Measured acceleration on a rotating plane.

For the cube configuration in Figure 3.2 there are three distances or diameters to consider, namely $d_{xy,yz,xz}$. From Equation 3.4 we know that these diameters corresponds to the hypotenuse of each of the three cube projection (XY, YZ and XZ). The EcoIMU implementation assigns a different acceleration axis to each enclosing cylinder. This means that each of the three diameters is going to affect their assigned acceleration axis. Which means that d_{xy} is going to influence the measured X-acceleration, d_{yz} is going to affect the measured Y-acceleration and d_{xz} is going to affect the measured Z-acceleration.

What Affects Tangential Acceleration?

Ideally, one would like the axes of the accelerometers to be aligned with the circumference of their assigned bounding cylinder. If this were the case, the measured acceleration would directly yield the tangential acceleration. This can easily be seen from Equations 3.6-3.8. If the tangential angle becomes zero, the tangential angle equals the measured acceleration. However, this would require the accelerometers to be mounted on the plane with an incline, which is not a good solution from a practical point of view. It is therefore far easier to just decompose the measured acceleration based on the tangential angle of the bounding cube. This works well as long as the tangential angle does not become too large. From this, it can be seen that the tangential acceleration is dependent on the geometry of the bounding cube, more specifically the cosine of the tangential angles θ_{xy},θ_{yz} and θ_{xz}. The θ_{xz} angle is illustrated in Figure 3.3. One important thing to note here is that the tangential angles are linked. Meaning that a change in the geometry of the XY-plane is going to affect the tangential angle in the YZ and XZ-plane.

What Affects Angular Acceleration?

The angular acceleration is the tangential acceleration normalized to unit radius. Considering the situation in Figure 4.1 again, this would mean that all of the accelerometers on the plane would have the same angular acceleration. This is quite intuitive, seeing that they all have traversed the same angle during the rotation. This is an important observation, as it effectively makes the angular acceleration independent of the cube geometry.

Table Summary

Provided in Table 4.1 is a summary of the three different acceleration types presented in this thesis.

Notation	Description	Formulae	Units
$\vec{a}$	Measured acceleration	$\vec{a}$	m/s^2
$\vec{a}_t$	Tangential acceleration	$\vec{a}_t = \vec{a} \cdot cos(\theta)$	m/s^2
$\vec{\alpha}$	Angular acceleration	$\vec{\alpha} = \frac{\vec{a}_t}{r}$	rad/s^2

Table 4.1: Three different types of acceleration

4.2 Precision and Power Consumption

As discussed in the previous sections, both the measured acceleration and the tangential acceleration are affected by the cube geometry. Now, let's continue in describing how the cube configuration in a GF-IMU affects parameters such as precision and power consumption. To get a better understanding on how these parameters are correlated, one need to take a better look at the constraints of using a MEMS accelerometer for this type of application.

4.2.1 Precision - MEMS Accelerometers

From Section 2.1.2 we know that digital MEMS accelerometers are limited by having both a finite digital resolution and ODR. This imposes limitations on how small and how fast an acceleration can be in order for the device to be able to measure it.

Ideally, the smallest acceleration that a MEMS accelerometer is able to measure is given by its sensitivity, as specified in Section 2.1.2. To get the most out of the finite digital resolution of the ADC, it is beneficial to use the lowest measurement range (FSR), as seen from Equation 2.4. On most MEMS accelerometers, the smallest selectable measurement range is $\pm 2g$. From the specialization project in [7], the typical digital resolution of MEMS accelerometers range between 12 and 16-bit. This gives a minimum measurable acceleration of $0.0096 m/s^2$ and $0.00059 m/s^2$ respectively, as seen from Equation 4.3 and 4.4.

$$\text{Sensitivity} = \frac{2 \cdot 9.81 - (-2 \cdot 9.81)[m/s^2]}{2^{12}} = 0.0096[m/s^2] \tag{4.3}$$

$$\text{Sensitivity} = \frac{2 \cdot 9.81 - (-2 \cdot 9.81)[m/s^2]}{2^{16}} = 0.00059[m/s^2] \tag{4.4}$$

It is however important to note that the theoretical sensitivity specified in the component datasheet does not take noise into consideration. The RMS noise in an accelerometer is dependent on the bandwidth BW as well as the characteristic PSD of the device, as seen in Equation 2.9. It follows from this equation that if the PSD is constant and the sensitivity is constant that the RMS noise should decrease as the bandwidth decreases. As described in Section 2.1.2, it is more correct to use the effective number of bits (ENOB) instead of the specified number in the component datasheet, as the ENOB includes the system noise. Equation 4.5 is therefore a more accurate estimate of the actual sensitivity.

$$\text{Sensitivity} = \frac{FSR}{2^{ENOB}} \tag{4.5}$$

The PSD value is highly dependent on which part that is used, but it is usually in the area of 200-500$\mu g/\sqrt{Hz}$ [7]. An example is a good way to show how noise affects the ENOB. Assume that a device has a digital resolution of 16-bit and a PSD of $450\mu g/\sqrt{Hz}$. Further assume that the device uses a measurement range of $\pm 2g$ and an ODR of 100Hz (50Hz BW). By using Equation 2.9, 2.10 and 2.11 it gives us an ENOB of 15.35. The true sensitivity of the device can then be seen in Equation 4.6. The full calculation can be found in Appendix A.

$$\text{Sensitivity} = \frac{2 \cdot 9.81 - (-2 \cdot 9.81)[m/s^2]}{2^{15.35}} = 0.00093[m/s^2] \tag{4.6}$$

Notice how the smallest measurable acceleration increased from $0.00059 m/s^2$ in the ideal case in Equation 4.4 to $0.00093 m/s^2$ in Equation 4.6. This example clearly illustrates that noise is an important factor that must be considered in such systems.

Since MEMS accelerometers are limited by having a finite sensitivity, there is going to be a physical limitation on how small the cube can be. Remember from Section 4.1.2 that the magnitude of measured acceleration is dependent on the distance from the cube origin. If the cube becomes too small, the measured acceleration is going to be too small in order for the accelerometer to be able to measure it correctly. To get the best results, the cube should therefore be so large that the measured acceleration will fill the measurement range of $\pm 2g$. This will cause the noise to become a less predominant factor during quantization.

4.2.2 Power Consumption - MEMS Accelerometers

From Section 2.1.2 we know that a lower ODR usually translates into a lower power consumption for the part, so one would ideally like to use the lowest possible ODR for best possible energy efficiency. The frequency of the motion signal itself is not dependent on the cube shape; rather it is dependent on the change in speed of the imposed rotation or translation. In order to know which ODR to select, one therefore needs to study the frequency components of a typical motion signal. That being said, there can still be a relationship between power consumption and the cube configuration. Remember from the previous section, that the measured acceleration is dependent on the cube size and shape. A smaller cube causes the measured acceleration to be smaller, thereby making noise a more predominant factor during quantization. From Section 2.1.2, we know that oversampling can be used to trade precision for power. Oversampling could therefore be necessary to use if the cube becomes too small, which effectively would introduce a relationship between the power consumption and the cube size.

4.3 Simulation

From the previous sections, we have now established that there is a relationship between the cube configuration and parameters of power consumption and precision. Since the parameters are correlated, it is convenient to be able to simulate them over different cube configurations. A Matlab simulation was therefore created in order to provide such simulation results. The following sections will discuss how this simulation was created.

4.3.1 Functional Description

The initial idea for the simulation was to use captured accelerometer data from a smartphone to calculate angular velocity based on a predefined cube shape. Data was first captured in a .CSV format using a smartphone app called SensorKinetics Pro. This app captures raw data from the phone's internal sensors. The accelerometer data from this app was then imported into Matlab. A virtual cube configuration consisting of the line segments L_x, L_y and L_z was then created, as depicted in Figure 4.2. The tangential angle $\theta_{xy,yz,xz}$ as well as the diameters for the enclosing cylinders $d_{xy,yz,xz}$ was then calculated. This was done by using Equation 3.5 and Equation 3.4 respectively. The imported accelerometer data was then decomposed onto the virtual cube to obtain the tangential acceleration a_t, this was done by using Equations 3.6, 3.7 and 3.8. The tangential acceleration was further used to calculate the angular acceleration $\alpha_{x,y,z}$ using Equation 3.9 and Equation 3.10. Integration was then used to calculate the angular velocity $\omega_{x,y,z}$ from the angular acceleration.

A fundamental problem with this approach is that the applied accelerometer data from the smartphone does not take the virtual cube shape into consideration. Meaning that a change in the simulated cube shape is not going to alter the applied accelerometer data. Remembering the situation in Figure 4.1, where the accelerometers placed closer to the edge of the plane would experience a larger measured acceleration than accelerometers placed closer to the center. Since the applied accelerometer data was independent on the cube shape, the angular acceleration became dependent on the cube shape, as seen from Equation 3.10.

Naturally, the applied simulation data needed to be independent of the cube geometry. One way to achieve this, was to go in the reverse direction by starting of with a rotational motion about a

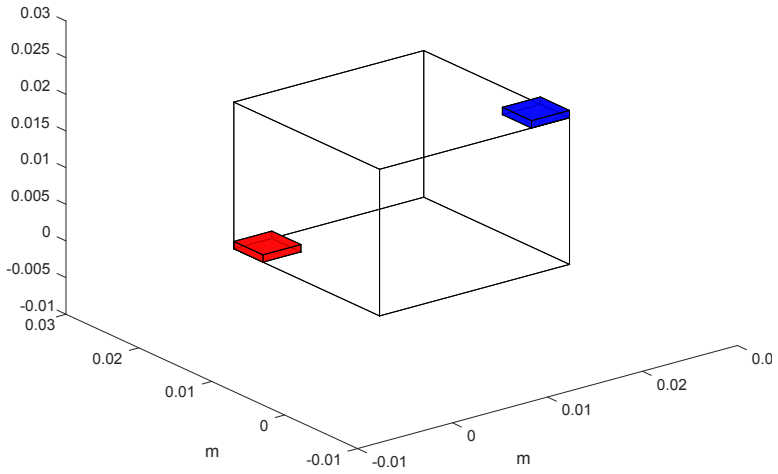

Figure 4.2: Matlab cube configuration

fixed point. Then use this data to calculate what each accelerometer should experience in terms of measured acceleration depending on the cube geometry. To achieve this a slightly different approach was used. This is illustrated conceptually in Figure 4.3. Instead of using captured accelerometer data from the smartphone, captured gyroscope data was used instead. This data was then imported into Matlab, where a Fast-Fourier Transform (FFT) analysis was performed to see which frequency components the motion was made up of. Based on this information, it was possible to tell something about the minimum ODR that should be used on the MEMS accelerometers in order to sample the motion correctly. The angular velocity from the gyroscope was further derived into angular acceleration. The steps from the initial approach were then used in a reverse order to calculate what each accelerometer on the cube would experience in terms of measured acceleration. From this, it was possible to see how the cube shape affected the measured acceleration. The possibility of adding signal degrading factors was then added to the simulation. This included decimation, which is the process of reducing the sampling rate of a signal, white noise, cross axis-sensitivity and quantization errors. The white noise was calculated from a specified PSD value, the bandwidth of the decimated signal and a selectable measurement range by using Equation 2.10. The cross-axis sensitivity was applied by adding a coupling, specified by a percentage number, between X, Y and Z-axis. The quantization error was applied by first dividing the data on the sensitivity from Equation 4.6 and rounding to the nearest integer. The result was then multiplied with the accelerometer scale factor in order to get the measured acceleration. The calculated measured acceleration was then used to calculate back to the angular velocity, in order to show how the applied error sources affected the integrated angular velocity estimate.

From Section 4.1.1 we know that the gravity component can be regarded as constant offset that is going to disappear from the angular velocity calculation. For simplicity, the gravity component was therefore omitted from the simulation.

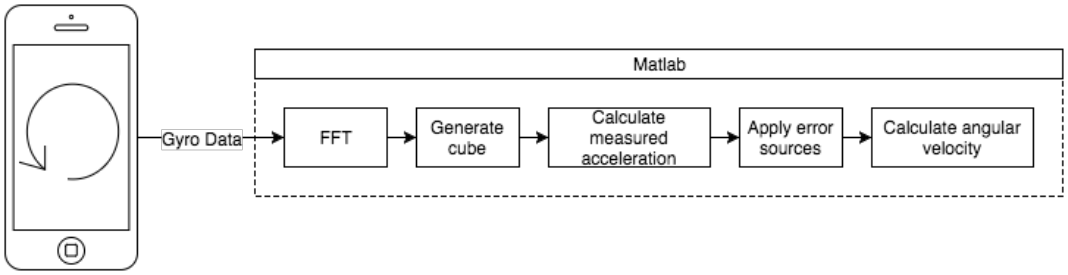

Figure 4.3: Matlab simulation architecture

4.4 GF-IMU Testbed

A custom testbed was constructed as part of this thesis in order to further investigate the EcoIMU configuration from Section 3.3. The testbed consists of a Nordic nRF52 development kit (DK) and three GY-6500 breakout modules. Each of these components, as well as the software architecture for the system will be further discussed in the following sections.

4.4.1 nRF52

The nRF52 is a Bluetooth System-on-Chip (SoC) from the company Nordic Semiconductor. The nRF52 has a ARM Cortex-M4F core clocked at 64MHz in addition to a broad range of peripherals designed specifically for low power autonomous operation [17]. Nordic has designed their SoC around a concept that is often referred to as race to halt operation. The principle behind this is to finish the computation as fast as possible so that the core can go to sleep as fast as possible. The 'F' in the CPU core name stands for floating point unit (FPU). Nordic claims that doing floating point operations with the FPU is roughly 20 times faster than doing them without an FPU [16]. This means that the core can spend more time in idle mode, which in turn will lead to a lower average power consumption. The task of calculating angular velocity from accelerometer data involves digital signal processing (DSP) operations like integration and filtering. It is difficult to avoid floating point numbers for such operations, as fixed-point numbers simply does not offer the required precision. The incorporation of an FPU was therefore one of the main reasons behind choosing the nRF52 for the testbed.

4.4.2 GY-6500 Breakout Module

The main component of the GY-6500 breakout module, shown to the right in Figure 4.4, is the MPU-6500 from a company called Invensense. The MPU-6500 is a 6 DoF IMU that consists of a tri-axial accelerometer and a tri-axial gyroscope all in one single 3x3x1mm QFN package [12]. The MPU-6500 is regarded as the de-facto standard IMU amongst hobbyists, as it offers a relatively good performance at a reasonable price. Due to the popularity of this sensor module, there exist many open-source drivers for the device on the Internet. The MPU-6500 is therefore an excellent starting-point for rapid prototyping of applications that require inertial sensing. This was the main reason behind choosing the module for the testbed.

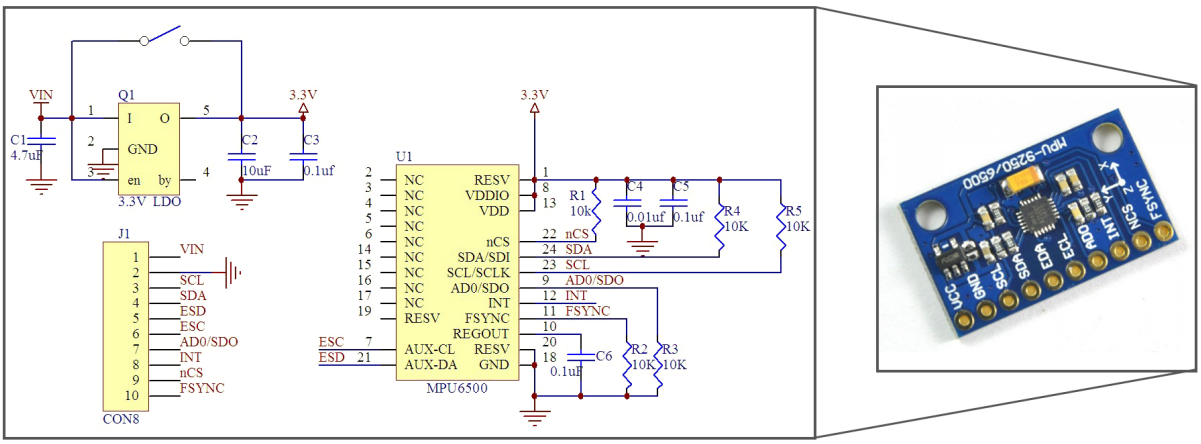

Figure 4.4: GY-6500 schematic and module

As seen from the schematic in Figure 4.4, the GY-6500 breakout board also features a 3.3V low-dropout regulator (LDO) as well as the necessary passive components required for running the MPU-6500. The testbed was designed to operate on a supply voltage of 3.3V, which means that the on-board LDO on the GY-6500 breakout board was unnecessary. Fortunately, the breakout board was designed in such a way that a solder bridge could be shorted to completely bypass the LDO. The solder bridge is modeled as a switch in the schematic in Figure 4.4. The solder bridge was shorted for all of the modules.

4.4.3 Testbed

For the testbed it was deemed important to have the flexibility to test the system over a wide range of different cube configurations. At the same time, it would be equally important to be able to fasten the modules properly, as a misalignment of the sensors would be a critical factor in terms of achieving good system accuracy. The solution can be seen in Figure 4.5. The nRF52 DK was mounted on an acrylic sheet, whereas the upper half of this sheet was perforated with 3mm holes in a 14x11 grid configuration with 10mm spacing. The 10mm spacing was chosen as it matches the hole spacing of 20mm on the GY-6500 module. Hexagonal brass spacers with a height of 6mm were used to elevate each module such that they would have a different height from each other. 2.54mm Dupont cables were used to attach the GY-6500 modules to the nRF52 DK. A small breadboard was used as a cable hub between the nRF52 and the GY-6500 modules. The testbed was designed in such a way that current consumption could easily be measured for each of the relevant modules. This includes the current consumption for all of the GY-6500 modules as well as the system total.

The testbed gives a great deal of flexibility with regards to testing the EcoIMU algorithm over a wide range of different geometric structures. As an initial configuration, a 4x4x1.2cm cube was chosen. It was difficult to mount the modules any closer than this due to the cabling and the size of the modules. Figure 4.6 illustrates the initial cube configuration of the testbed.

To test the EcoIMU implementation, it would have been sufficient to just use the accelerometers on two GY-6500 modules, placed in separate corners of a cube. But, it would then be difficult to know if the produced angular velocity estimate was correct. It was therefore decided to add a third module, placed at the center of the cube. This module would be configured to only use the gyroscope, and would act as a reference to test the correctness of the angular velocity estimate produced by the EcoIMU algorithm.

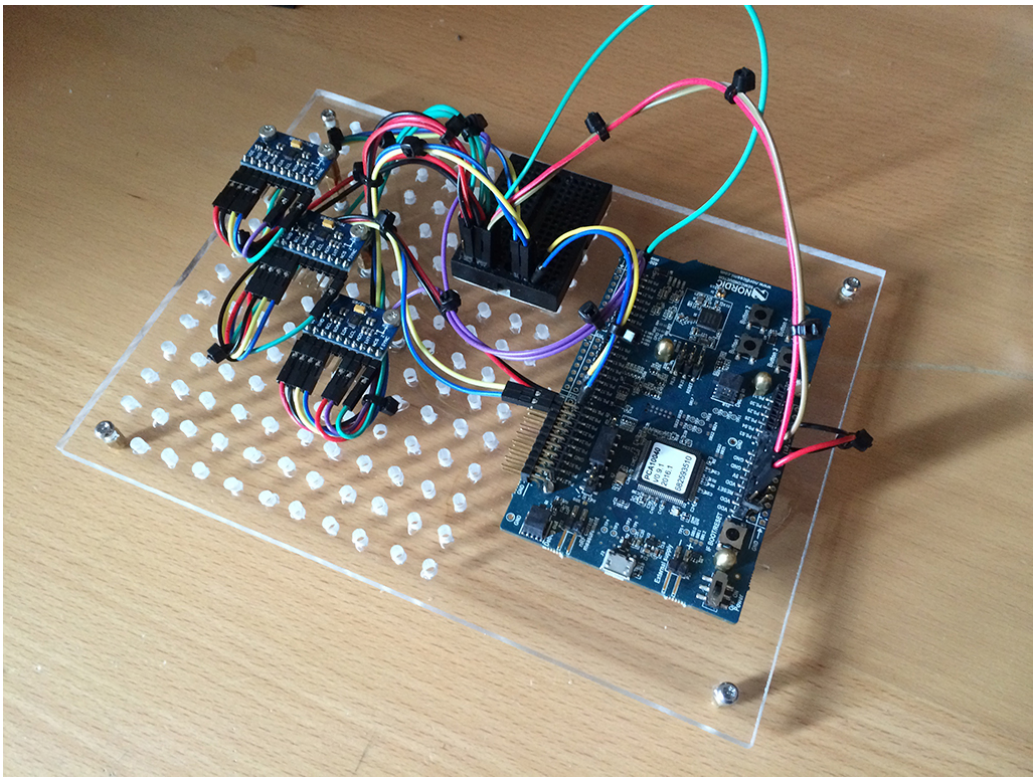

Figure 4.5: GF-IMU testbed

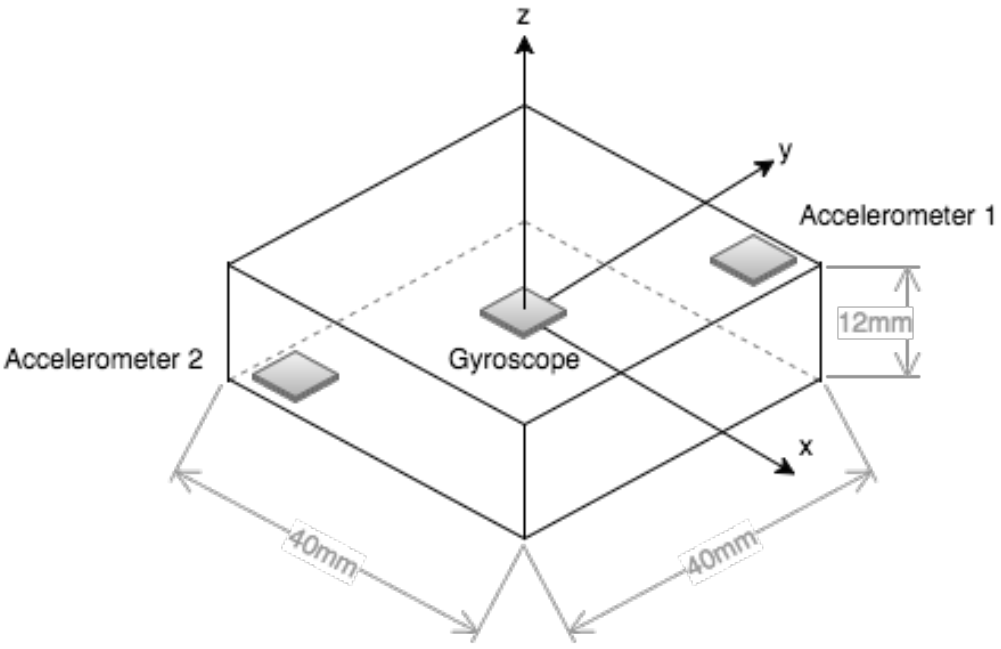

Figure 4.6: Initial 4x4x1.2cm testbed configuration

4.5 Testbed Software Architecture

As with the EcoIMU hardware implementation, the software architecture of the testbed was first divided into two stages, a simple data acquisition stage on the nRF52 DK and a real-time data processing stage on a computer running Matlab. This is illustrated conceptually in Figure 4.7.

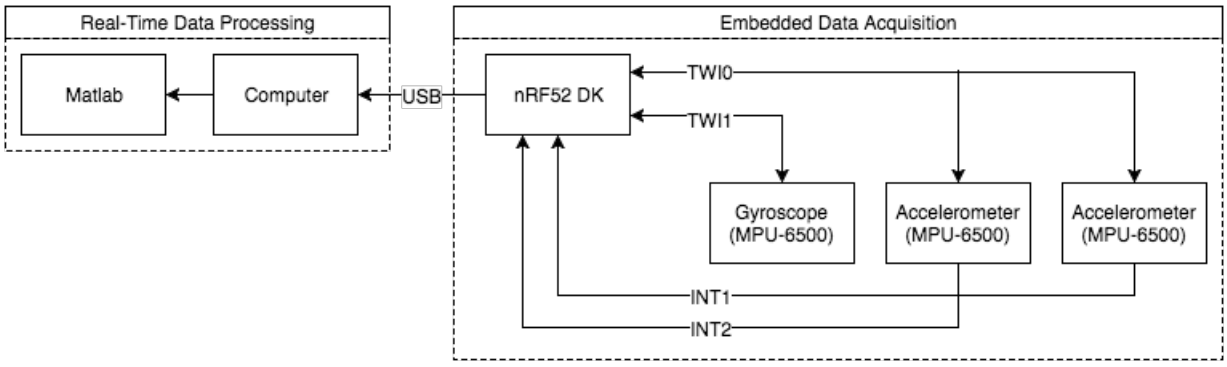

Figure 4.7: Initial testbed software architecture

4.5.1 Embedded Data Acquisition - Timer-Based Approach

The code on the embedded side was written in ANSI C and compiled using ARM Keil uVision v5. A C++ library for the MPU-6500 written by Jeff Rowberg and Nicolas Baldeck [23] was ported to C. The library was further fused with the Nordic Software Development Kit (SDK), which includes drivers for all of the peripherals on the nRF52.

A conceptual overview of the initial firmware running on the nRF52 is shown in Figure 4.8. A step-by-step list on how the initial data acquisition scheme was configured is presented below. See Appendix C for more information regarding the embedded firmware source code.

1. A UART serial interface was first configured in order to transfer data from the embedded testbed to the computer. The nRF52 DK has an on-board debugger chip, which has the ability to forward UART data from the nRF52 over USB, thereby acting as a virtual USB COM port. A baudrate of 230400bps was selected.

2. The TWI (Two-Wire Interface) interface on the nRF52 was then configured in order to communicate with the GY-6500 modules. The MPU-6500 is capable of using either TWI at 400kHz or SPI (Serial Peripheral interface) at 20MHz for data communication [12, p.8]. TWI was chosen as communication interface, as the ported MPU-6500 library was intended to be used with this communication interface. TWI is a bus protocol that supports having up to 127 devices on the same bus, as long as each device on the bus use a unique address [19]. The MPU-6500 has the ability to change between two addresses, depending on whether the physical pin AD0 is pulled to a logical low or high. This meant that two separate TWI buses were needed in order to support three GY-6500 modules. Fortunately, the nRF52 had hardware support for using two separate TWI buses. The two corner accelerometer modules were configured to use the TWI0 bus, and the gyroscope on the center module was configured to use the TWI1 bus. This is illustrated in Figure 4.7.

3. At first, the two MPU-6500 accelerometer modules were configured to use their normal mode, which uses the maximum ODR of 4kHz and a measurement range of $\pm 2g$. The gy-

roscope was disabled for both of these modules. The reference module in the middle was configured to only use the gyroscope with a measurement range of 250deg/s. The plan was to first start with the highest precision for the accelerometers, then try to gradually reduce this once a working setup was up and running. An initial bias value was also calculated for both accelerometers and the gyroscope using an average of 1000 samples. This number was subtracted from the accelerometer data in order to remove the bias error, as mentioned in Section 2.1.4.

4. A timer on the nRF52 was configured to trigger a periodic interrupt at a predefined time interval of 10ms (100Hz). Each time the interrupt was triggered, the nRF52 would execute an interrupt service routine (ISR) that would perform a read on each of the three sensor modules. The nRF52 was configured to sleep between each interrupt, in order to save power. Since data is only read out from the modules inside the ISR the actual ODR for the timer-based approach is only 100Hz, even though the accelerometers are configured to sample at a frequency of 4kHz. A benefit with this approach is that oversampling can be added with only some minor adjustments. For a 2x oversampling, one would only need to double the timer frequency and buffer alternate samples.

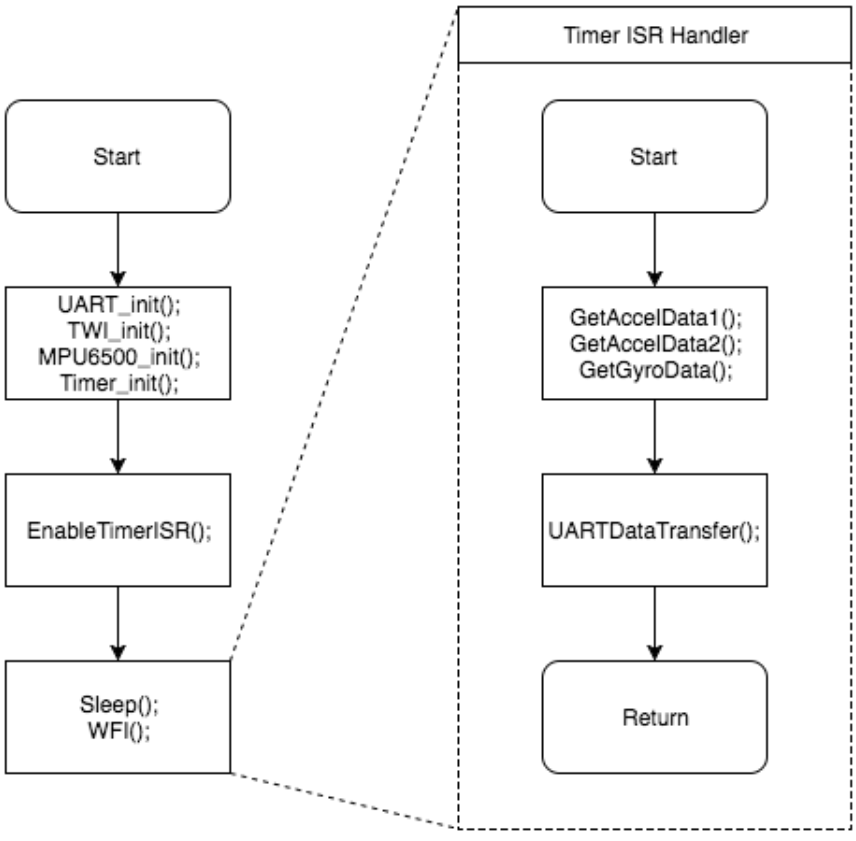

Figure 4.8: Initial embedded firmware architecture

4.5.2 Embedded Data Acquisition - Low Power Mode

As previously mentioned, the initial embedded data acquisition scheme was primarily designed around being simple and not particularly low power. A second scheme optimized for energy efficiency was therefore devised. A conceptual overview of this data acquisition scheme is shown in Figure 4.9.

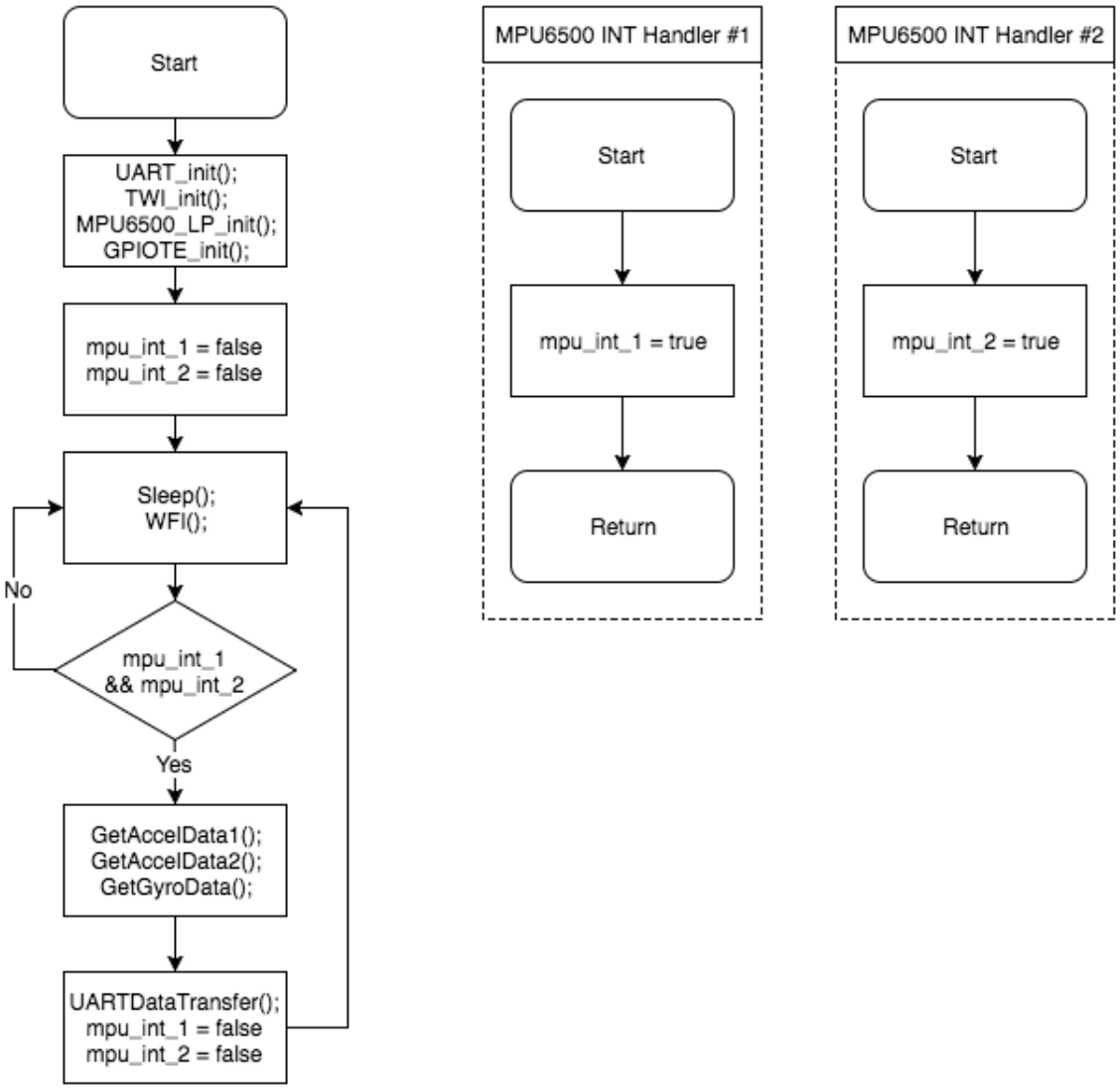

Figure 4.9: Embedded firmware architecture for the LP-mode

This scheme exploits the low power accelerometer mode (LP-mode) on the MPU-6500. In this mode, the accelerometers are duty-cycled to only take a sample at a predefined frequency [12, p.10]. The selectable frequencies in this mode are shown in Figure 4.10. When data is ready, the MPU-6500 can generate a interrupt on the physical INT pin to notify the host controller. The accelerometer is in idle between each sample. The LP-mode enables both the host controller and the accelerometers to sleep in-between consecutive samples, thereby saving a considerable amount of power.

An inherent problem with the LP-mode is data synchronization. Even though the LP-mode is initiated at exactly the same time for each of the MPU-6500 modules, the data samples are at some point going to drift out-of-sync due to internal clock inaccuracies. Out-of-sync data will not be a

LPOSC_CLKSEL	Output Frequency (Hz)
0	0.24
1	0.49
2	0.98
3	1.95
4	3.91
5	7.81
6	15.63
7	31.25
8	62.50
9	125
10	250
11	500
12-15	Reserved

Figure 4.10: Selectable frequencies in LP-mode [12]

critical factor if the ODR is high enough. However, as this implementation aims to achieve ultra-low power consumption, one would at some point try to reduce the ODR to a working minimum. When reducing the ODR, the time between each sample becomes longer, effectively making the synchronization between frames more important.

The MPU-6500 sensors have a dedicated pin called FSYNC. Initially, it was thought that this pin could be used as a trigger to synchronize samples between two MPU-6500. However, after talking to engineers from Invensense, it was discovered that this was not the purpose of this pin. The FSYNC pin is intended to be used in digital image stabilization applications, where it is useful to know which data sample that exactly corresponds to the time the image was taken. If the FSYNC pin is asserted, a watermark can be added to the LSB of a predefined data register. From this, it is possible to know which sample that corresponds to the event. In LP-mode, only one sample is taken at a predefined interval. If this sample time is out-of-sync between the modules, it does not help to add a watermark to the sample. This situation is illustrated in Figure 4.11. In this figure, the gray boxes illustrates the current sample, while the white boxes corresponds to previous or future samples. Note how the FSYNC pin only contributes in marking samples that are already out-of-sync.

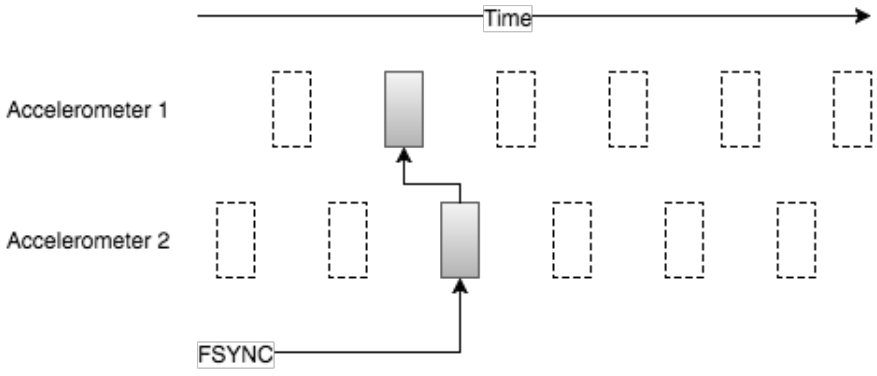

Figure 4.11: Using FSYNC in LP-Mode

The initial timer-based approach also suffers from synchronization issues. Although, they are less severe and far more deterministic compared to the LP-mode. For the timer-based approach, the MPU-6500 modules are configured to run at their highest frequency of 4kHz. This means that

there is not going to be any big synchronization issues between the modules, at most 250µs [1]. However, since the two GY-6500 modules are connected on the same TWI bus, data from them needs to be read in a successive fashion. This means that by the time data is read from the first accelerometer module, the data on the second accelerometer module will have changed. This time offset between samples will be fixed, and since the TWI bus runs on a frequency of 400kHz it will also be quite small. Using SPI as serial interface for the modules could have mitigated the data acquisition latency further, as this interface is able to run at 20MHz.

A more advanced scheme could also have been used to mitigate the synchronization error for the timer-based approach. The MPU-6500 has an embedded FIFO capable of holding 512-samples of accelerometer and gyro-data. This FIFO could be used in conjunction with the FSYNC pin. The approach is illustrated in Figure 4.12. Here, both accelerometers are configured to use their highest frequency of 4kHz. Data from the accelerometers are put in the FIFO as they are acquired. When the nRF52 timer triggers an interrupt, the FSYNC pin is asserted. This marks the most recent data sample in both FIFO's. During data readout in the ISR, the nRF52 reads out the entire FIFO from both accelerometers. Once the data is acquired, the nRF52 can iterate through the data samples from both FIFO's and find each sample marked with the FSYNC watermark. With this approach one is able to remove the time offset between samples caused by the TWI read latency. However, this approach would require that more information be read from the accelerometers, as well as additional computational overhead in searching for the watermarked sample. The approach would trade an increase in power consumption for slightly better synchronized data. It was therefore decided not to implement this scheme.

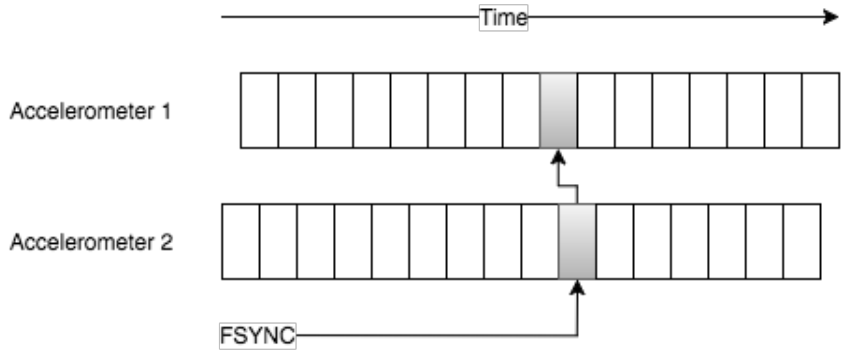

Figure 4.12: Using FSYNC in FIFO Mode

[1] $1/4kHz = 250\mu s$

4.5.3 Matlab Real-Time Processing

Writing code in Matlab is generally much faster than writing code in C for an embedded platform such as the nRF52. Each time the code needs to be updated on the embedded platform, it needs to be compiled then flashed to the chip. In comparison, Matlab just requires the user to hit run, and the program will start almost immediately. Matlab does also have a quite comprehensive set of built-in library functions, such as advanced filters and data integration techniques. Since the EcoIMU implementation would require some initial experimenting, it was decided that it would be most effective to first do this part in Matlab. A Matlab script performing real-time data processing on the data from the embedded platform was therefore written. A conceptual overview of the real-time data processing script is shown in Figure 4.13. See Appendix C for more information regarding the real-time processing code.

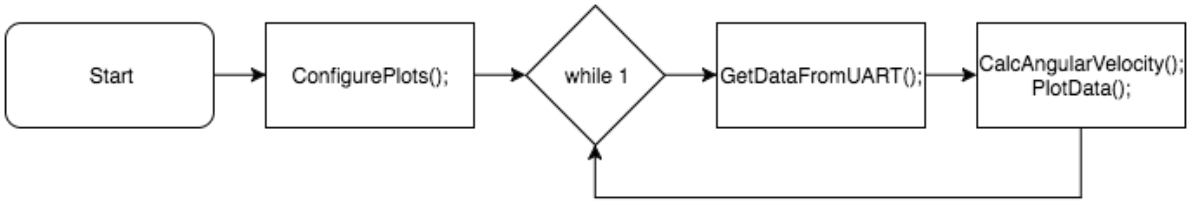

Figure 4.13: Real-time data processing in Matlab

After a working GF-IMU setup was up and running in Matlab, the algorithm was implemented in C for the embedded platform. This was an important step to do before current measurement, as it would be important to also consider the cost of additional data processing required by the GF-IMU algorithm. The real-time Matlab script was still used, but only to plot the data. A conceptual overview of the real-time data plotting script is shown in Figure 4.14.

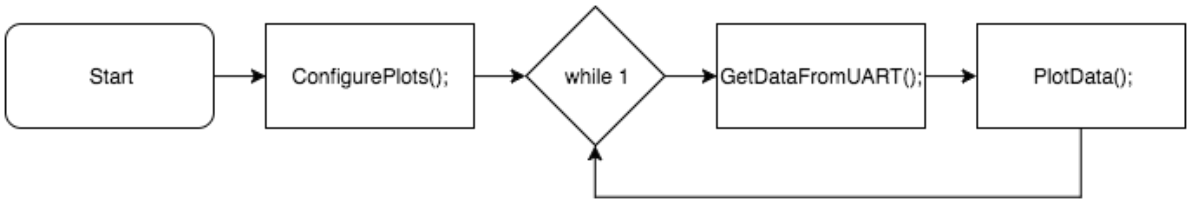

Figure 4.14: Real-time data plotting in Matlab

Calculating Angular Velocity

The calculation of angular velocity from the two accelerometer was initially done by using the EcoIMU equations from Section 3.3 directly. First, this involves decomposing the measured acceleration along the circumference of the enclosing cylinders in each cube projection. The tangential acceleration is then used to compute angular acceleration. The angular acceleration was then integrated using first order Euler integration, as seen in Equation 4.7. Here, dt denotes the time step between each sample.

$$\omega_{xyz}(n) = \omega_{xyz}(n-1) + \alpha_{xyz}(n) \cdot dt \tag{4.7}$$

Since the angular velocity estimate is based on integration, data errors are going to accumulate over time. In order to keep the angular velocity estimate from drifting away, it was necessary to

implement a heuristic to keep this from happening. This was done by simply computing an average of the angular acceleration about each axis. If the angular acceleration was below a predefined threshold for an extended period of time, the angular velocity was reset back to zero.

Calculating Rotational Angle

The angular velocity estimate was further used to calculate angle of rotation (roll, pitch and yaw). This was also performed by doing Euler integration on the angular velocity estimate, as seen from Equation 4.8, 4.9 and 4.10.

$$\psi(n) = \psi(n-1) + \omega_x(n) \cdot dt \tag{4.8}$$

$$\theta(n) = \theta(n-1) + \omega_y(n) \cdot dt \tag{4.9}$$

$$\phi(n) = \phi(n-1) + \omega_z(n) \cdot dt \tag{4.10}$$

From Section 2.3.2, we know that an AHRS uses the gravity calculated roll and pitch from an accelerometer to compensate for drift in the angle estimate calculated from a gyroscope. The same heuristic can also be used for a GF-IMU. There is however no way to correct the yaw-estimate by using this approach.

4.6 Testing

The following sections describe how the testbed and simulation was used to provide data, as well as how the current consumption for the testbed was measured.

4.6.1 Simulation

The simulation parameters was taken from the MPU-6500 datasheet [12], as this was the module of choice for the testbed. The parameters are listed in Table 4.2.

Device	Resolution	ODR	PSD	FSR	Cross-axis sensitivity
MPU-6500	16bit	100Hz	300µg/$\sqrt{Hz}$	±2g	2%

Table 4.2: Simulation parameters [12]

As described in Section 4.3.1, the internal gyroscope of a smartphone was used to provide data for the Matlab simulation. Two different sets of motion sequences were generated for the simulation. The first sequence consisted of a few consecutive yaw motions, as depicted in Figure 4.15. The rotation speed of the yaw motion was gradually increased from slow to fast, in order to cover a typical IMU use-case. From this we define our typical motion to be at most 400 deg/s, as seen from Figure 4.15. This is a reasonable rotational speed for applications that are designed around human interaction. The first motion sequence is used for a theoretical analysis that aims to better illustrate how the measured acceleration is affected by different cube geometries.

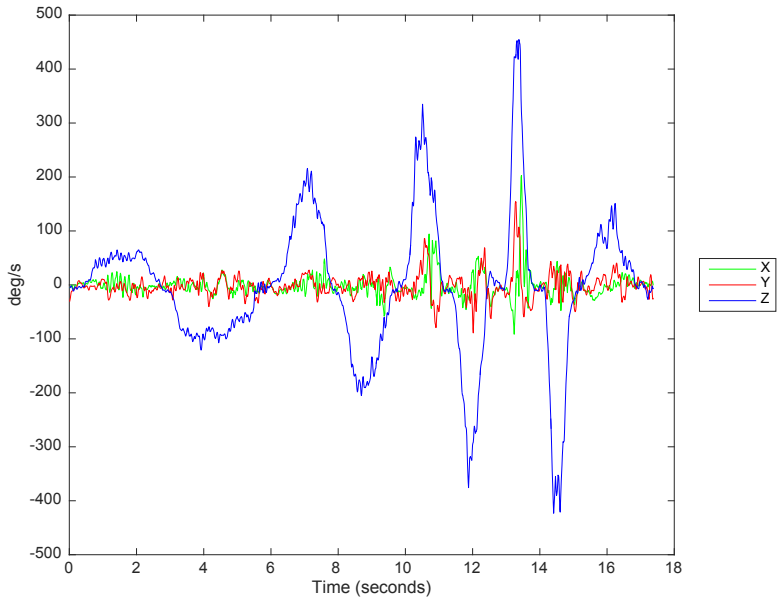

Figure 4.15: Yaw motion from smartphone gyroscope

The second sequence of motion data consisted of a series of yaw, pitch and roll motions, as depicted in Figure 4.16. This motion sequence is used to test the angular velocity estimate for all of the three axes on a cube configuration that matches the initial 4x4x1.2cm testbed from Figure 4.6.

The Matlab simulation first calculates the measured acceleration for all three axes on both accelerometers. Component specific error sources are then imposed on to the measured acceleration of each accelerometer. This includes white noise, quantization errors and cross-axis sensitivity. The error sources are calculated from the parameters in Table 4.2. The measured acceleration is then used to compute the angular velocity about the X, Y and Z-axis. The estimate is then compared against the original data from the gyroscope.

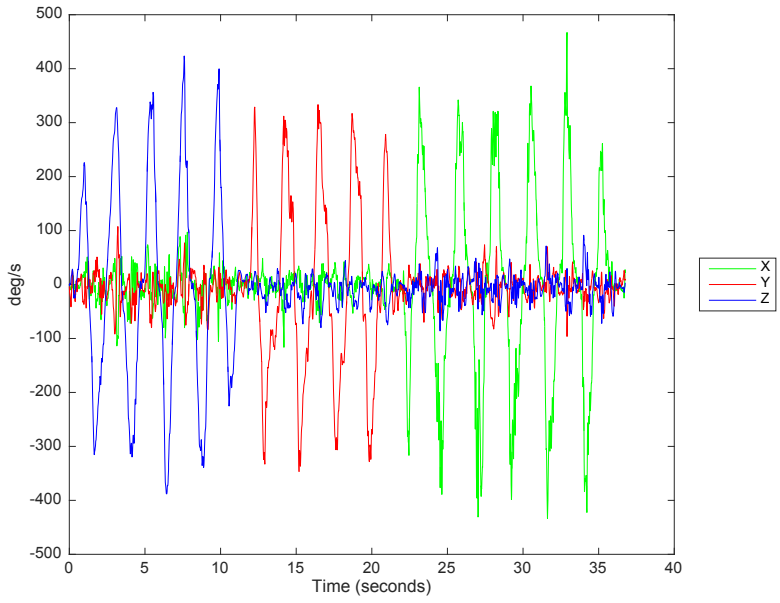

Figure 4.16: Mixed motion signal from smartphone gyroscope

4.6.2 Testbed

From Section 2.3 we know that an IMU by definition must be able to provide rotation about- and translation along the X, Y and Z-axes. Most conventional IMUs use an accelerometer to measure the linear translation and a gyroscope measure rotation. From this, we already know that an accelerometer can be used to measure linear translation. The interesting thing here is therefore to see how well two accelerometers in a GF-IMU configuration perform in computing angular velocity and angle of rotation (roll, pitch and yaw). It was therefore decided to focus the GF-IMU testing around rotation, and not linear translation.

For a GF-IMU to be practical in a real-world application, it is important that it does not occupy too much space. For the testbed, it was therefore decided to start with the smallest cube configuration that the testbed would allow. The initial 4x4x1.2cm testbed configuration can be seen in Figure 4.6.

Angular Velocity

The testbed was first used to provide a comparison between the angular velocity estimates from the corner accelerometers against data from the center gyroscope. As such, the gyroscope is used as a reference to see how well the angular velocity estimate from the two accelerometers performs. The angular velocity was computed and plotted in real-time by using the designed Matlab script from Section 4.5.3. Four measurements on each of the three axes were performed. Each measurement includes readings from the gyroscope and the angular velocity estimate from the accelerometers. The mean squared error (MSE) was computed for each test, in order to get an exact figure on the deviation between the angular velocity estimate from the accelerometers and the gyroscope.

The timer-based data acquisition scheme from Section 4.5.1 was initially used to provide data from the testbed. This scheme configures the accelerometers to use their highest ODR of 4kHz. Although, data is only read out from the sensors with a frequency of 100Hz. The accelerometer LP-mode from Section 4.5.2 was then tried with the frequencies of 125Hz, 62.5Hz and 31.25Hz.

The applied rotations was performed by holding the testbed platform from Figure 4.5 in air while turning it 90 degrees back and forth about the relevant measurement axis. The turning motion was adjusted from slow to fast. No specific motion pattern was chosen for the rotation, but it was attempted to vary the rotation speed as much as possible. Since the applied motion was performed by hand, some measurement inaccuracies between tests are to be expected. This was the motivation behind performing four measurements for each axis.

Angle of Rotation

As with the angular velocity estimate, the angle of rotation was calculated and plotted using the real-time Matlab script. As before, the testbed was held in the air by hand and turned 90 degrees back-and-forth about the relevant measurement axis. The timer-based data acquisition scheme was first used, followed by a test using the accelerometer LP-mode with the frequencies of 125Hz, 62.5Hz and 31.25Hz.

4.6.3 Current Consumption

The testbed current consumption was measured using a µCurrent measurement device, the red board in Figure 4.17. The µCurrent consists of a range of selectable high quality shunt resistors and a high quality comparator configured as a non-inverting amplifier. Thus, the current consumption can be measured as a voltage drop over the shunt resistor. The µCurrent has the ability to select between three different ranges; 1mV/nA, 1mV/µA and 1mv/100µA. The voltage was measured using a HMO2024 oscilloscope, as depicted in Figure 4.17. The oscilloscope was configured use an oversampling (averaging) factor of 32x, in order to reduce random noise. The mean value from the oscilloscope was used to determine the average current consumption for the measurements.

The current consumption was measured for both of the corner GY-6500 modules, the reference GY-6500 module in the middle and the nRF52 DK. See Figure 4.5 for reference. The corner GY-6500 modules were configured to only use the accelerometers. Together with the nRF52 DK they constitute the GF-IMU configuration. The center GY-6500 module was configured to use both the gyroscope and the accelerometer. Thus, it constitutes a conventional 6 DoF IMU. The UART data transmission from the nRF52 was disabled for the current measurements, as this is something that would not be used in an actual application. Disabling the UART transmission significantly reduces the nRF52 on time, hence it also makes a big difference in the average current consumption for the nRF52.

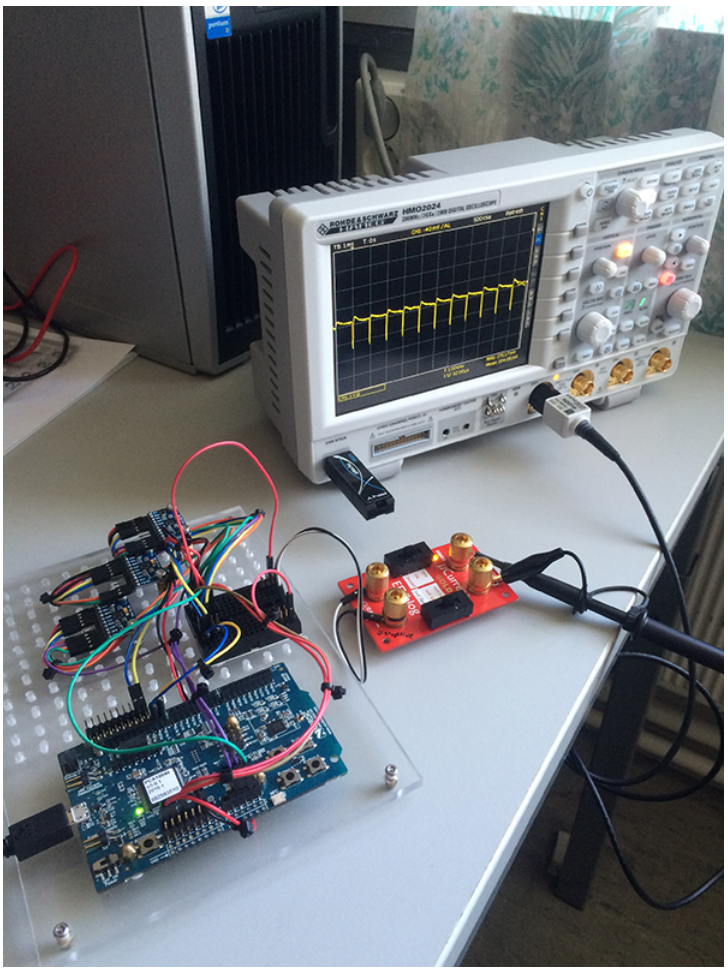

Figure 4.17: Measuring the current consumption.

5 Results and Discussions

In this chapter, results obtained from the Matlab simulation and the testbed will be presented. The first part of the chapter focuses on the simulation, and aims to analyze how different cube configurations and bandwidth affects the overall performance of the system.

The next part of the chapter focuses on the testbed. This part provides actual data on how the angular velocity estimate of GF-IMU configuration compares to data from a gyroscope. The angular velocity estimate is then used to see how accurate it is possible to compute an angle of rotation. A comparison of the power consumption between the two configurations is presented at the end.

5.1 Theoretical Analysis

The following sections present a theoretical analysis that aims to give a better understanding on which bandwidth and measurement range to use for the GF-IMU configuration. The Matlab simulation together with the yaw-motion from Figure 4.15 is used for this purpose.

5.1.1 Bandwidth

As discussed in Section 4.2, there is a trade-off between power consumption and precision when it comes to selecting an ODR rate for the MEMS accelerometers. As specified by the Nyquist criterion, the ODR needs to be at least twice as high as the highest frequency component in the signal that one is trying to sample. Using an ODR rate of 100Hz for a signal that has maximum frequency of 10Hz is excessive. To get at better understanding of which ODR to use for the GF-IMU, it was necessary to know something about the frequency components of a typical motion. A FFT analysis of the yaw-motion signal in Figure 4.15 was therefore performed using the designed Matlab simulation. The frequency components of the motion signal are shown in Figure 5.1. Notice how most of the frequency components of the yaw-motion are located below 15Hz. This suggests that the ODR should at least be 30Hz. Although, this assumes that the accelerometer is able to quantize the signal in the first place. As mentioned in Section 4.2, the system noise will become a more predominant factor as the cube becomes smaller. For such situations it was mentioned that oversampling could be used to improve the precision. 30Hz might therefore not be sufficient if the cube becomes too small. To better know which ODR to use, we therefore need to take a closer look on how the measured acceleration is affected by a change in cube size.

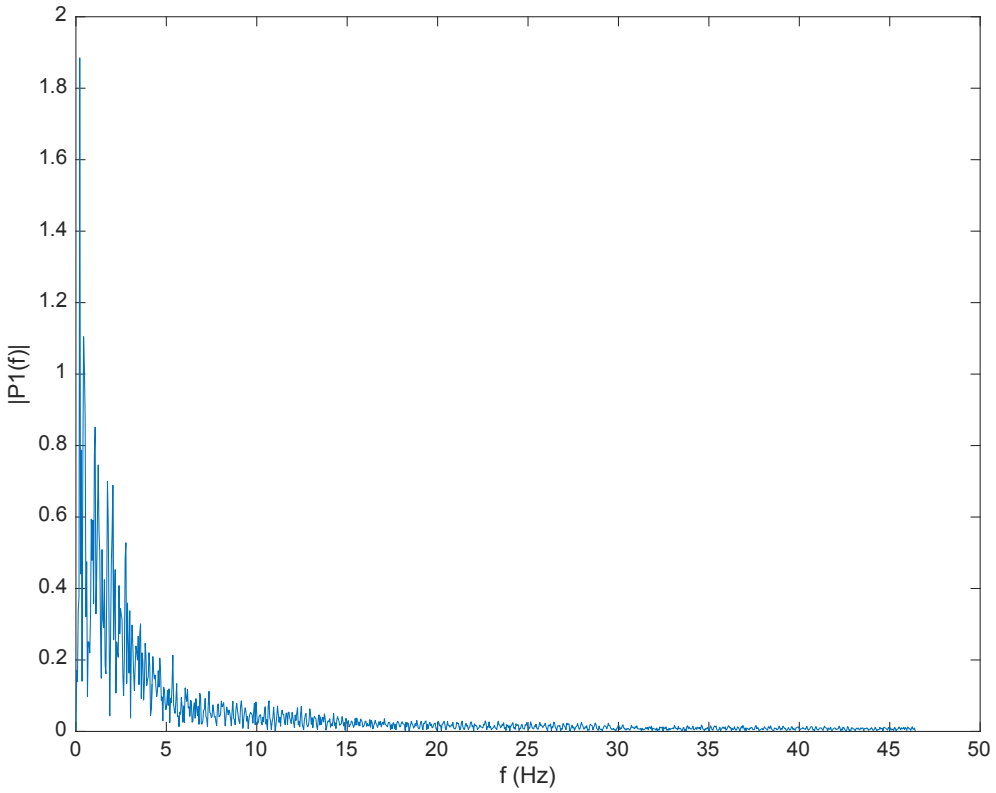

Figure 5.1: FFT analysis of yaw motion

5.1.2 Measurement Range

From Section 4.2, we know that in order to get the best sensitivity from a MEMS accelerometer one should use the smallest measurement range of $\pm 2g$. However, it might not be feasible to use this measurement range. If the magnitude of the measured acceleration during a typical motion is larger than $\pm 2g$, a larger measurement range needs to be selected. In Section 4.6.1, we defined our typical motion as series of consecutive yaw-motions, which magnitude was at most 400 deg/s. This can be seen in Figure 4.15. This motion is going to be our starting base for this analysis. By using this typical motion, we aim to analyze how different cube sizes affect the measured acceleration.

From Section 3.3, we know that the EcoIMU implementation uses the decomposed tangential X-axis to determine angular velocity about the Z-axis (yaw). From Section 4.1.2 we know that the measured X-acceleration is only dependent on parameters in the XY-plane, namely the diameter d_{xy} and the tangential angle θ_{xy}. These parameters are further calculated from the width L_x and length L_y of the cube. This means that the height of cube L_z will not have any effect on the angular velocity estimate about the Z-axis. It follows from this analysis, that the YZ-plane is not going to be affected by a change in L_x, and that the XZ-plane is unaffected by a change in L_y. That being said, a change in one cube face is effectively going to change the parameters of the other two faces. So in that way, they are all connected.

A cube size of 2x2x2cm was first created in the Matlab simulation, as seen in Figure 5.2. The measured X-acceleration for the yaw-motion in Figure 4.15 was then calculated using the simulation. This is shown in Figure 5.3. The Y-axis of Figure 5.3 is scaled to show the lowest measurement range of $\pm 2g$, or in this case $\pm 20 m/s^2$. It was previously mentioned that the measured accelera-

tion should try to fill the measurement range, in order for the noise to become less predominant during quantization. As seen from Figure 5.3, the measured acceleration during the yaw-motion is between $\pm 2m/s^2$, which suggests that the cube could be larger than this in order for the measured acceleration to better fill the measurement range.

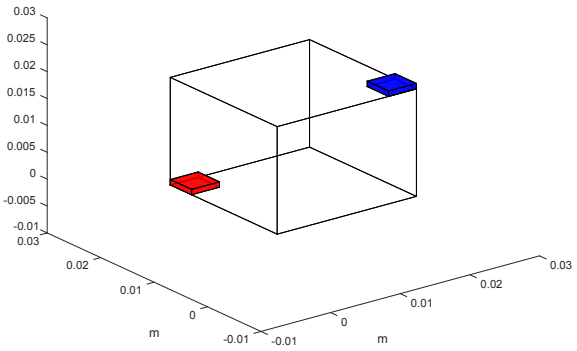

Figure 5.2: 2x2x2cm cube

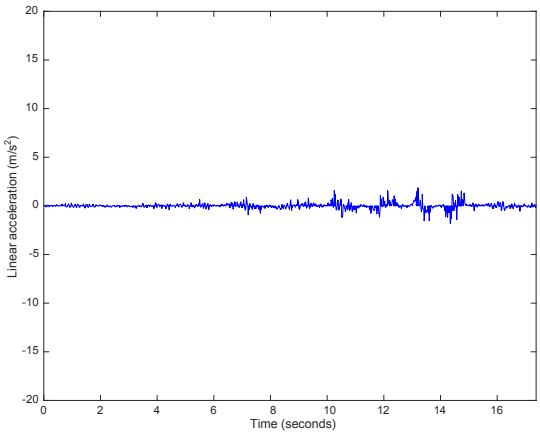

Figure 5.3: a_{x1} during a yaw

By increasing the cube size to 20x20x20cm, the measured acceleration is able to completely fill the measurement range of $\pm 2g$. This is illustrated in Figure 5.4. A cube size of 20x20x20cm is obviously impractical for most real-world applications. However, a quite useful result has already been found from this simple investigation. Namely, as long as a cube plane is below 20x20cm it can safely use the lowest measurement range of $\pm 2g$ for our typical motion.

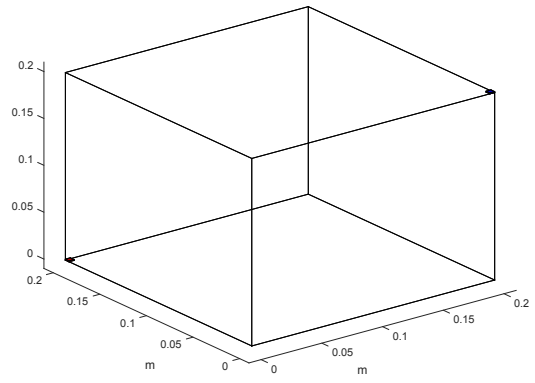

Figure 5.4: 20x20x20cm cube

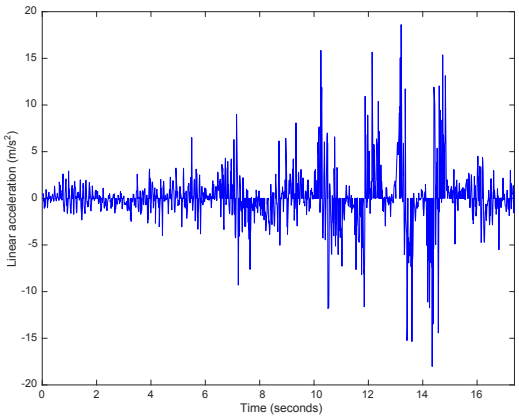

Figure 5.5: a_{x1} during a yaw

Both of the above simulation examples uses a perfect cube, with three square faces. As stated above, the measured X-acceleration is solely dependent on the XY-projection of cube. Thus, a cube of 20x20x1cm will give the same measured X-acceleration as the one in Figure 5.4. This is shown in Figure 5.6 and 5.7.

A change in the cube height would effectively make the XZ and YZ-projection of the cube rectangular, as seen in Figure 5.6. This would make the hypotenuse of XZ and YZ-projection smaller. Thus, also making their enclosing cylinder smaller. As we know from Section 3.3, a rotation about Y-axis is determined from the measured Z-acceleration decomposed along the enclosing cylinder in the XZ-plane. From Figure 3.3 and Equation 3.5, we can see that the tangential angle for the XZ-plane

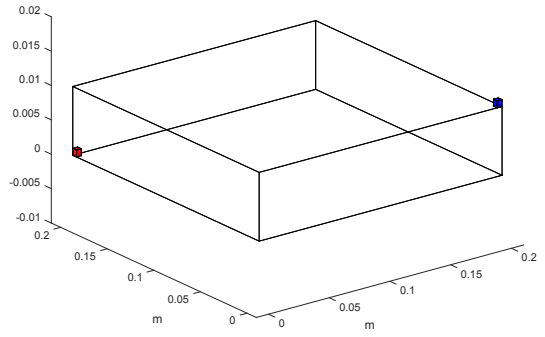

Figure 5.6: 20x20x1cm cube

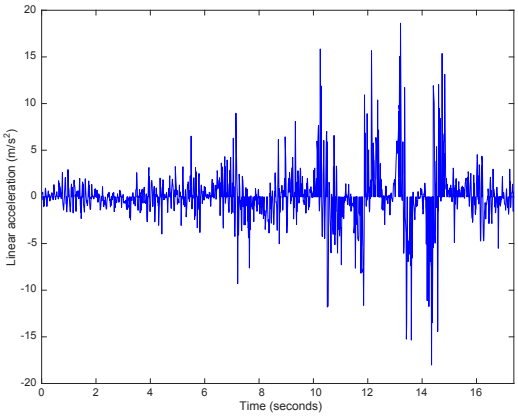

Figure 5.7: a_{x1} during a yaw

(θ_{xz}) becomes smaller if the height (L_z) becomes smaller. Thus, the measured acceleration is going to be closer to the tangential acceleration vector of the bounding cylinder in XZ-plane. This means that even though the diameter gets smaller, the measured acceleration is going to increase. Thus, reducing the height of the cube is going to have a positive impact for a rotation about the Y-axis.

Unfortunately, this is not the case for the YZ-plane. From Section 3.3, we know that a rotation about the X-axis is determined from the measured Y-acceleration decomposed along the enclosing cylinder in the YZ-plane. From Equation 3.5, we can see that a decrease in the height (L_z) of the cube effectively makes tangential angle of the YZ-plane (θ_{yz}) larger. A larger tangential angle leads the measured Y-acceleration vector further away from the tangential acceleration vector of the bounding cylinder in the YZ-plane. Thus, the measured Y-acceleration is going to decrease. As we can see here, there is a trade-off between measured acceleration and the overall shape of the cube. The cube can be tweaked in order to give a higher measured acceleration for a rotation about one axis, at the penalty of a lower measured acceleration for the rotation about the other two axes.

5.2 Simulation Results

We have now used the simulation to perform a theoretical analysis on how measured acceleration is affected by the cube shape. It is now interesting to see how the simulation can be used to calculate angular velocity.

Calculating Angular Velocity

The Matlab simulation was configured to use a 4x4x1.2cm cube, in order to match the initial testbed configuration from Figure 4.6. A new set of motion data was generated using the Sensor Kinetics Pro app, as seen from Figure 4.16. The new data set consists of a series of yaw, pitch and roll motions.

The new motion data was applied to the Matlab simulation. The simulation first calculates the measured acceleration for all three axes on both accelerometers. Component specific error sources are then imposed on to the measured acceleration of each accelerometer. This includes white noise, quantization errors and cross-axis sensitivity. The error sources are calculated from the parameters in Table 4.2. The measured acceleration is then used to compute the angular velocity

about the X, Y and Z-axis. The estimate is then compared against the original data from the gyroscope. The simulation results for the X, Y and Z-axis are respectively shown in Figure 5.8, 5.9 and 5.10. From these figures, one can immediately see that all the angular velocity estimates (red line) follows the data from the gyroscope (green line). Thus, it seems that the accelerometer sensitivity should be good enough to quantize the signal, even though the measured acceleration does not fill the entire measurement range of $\pm 2g$ at a 4x4x1.2cm cube configuration. One can however notice that the angular velocity estimate about the X-axis in Figure 5.8 is starting to drift away at the end of applied motion data. From the theoretical analysis in the previous section, we know that reducing the cube height L_z makes the measured Y-acceleration smaller. Thus, it is going be more affected by noise during quantization. This is likely the explanation for the observed drift in angular velocity estimate about the X-axis in Figure 5.8.

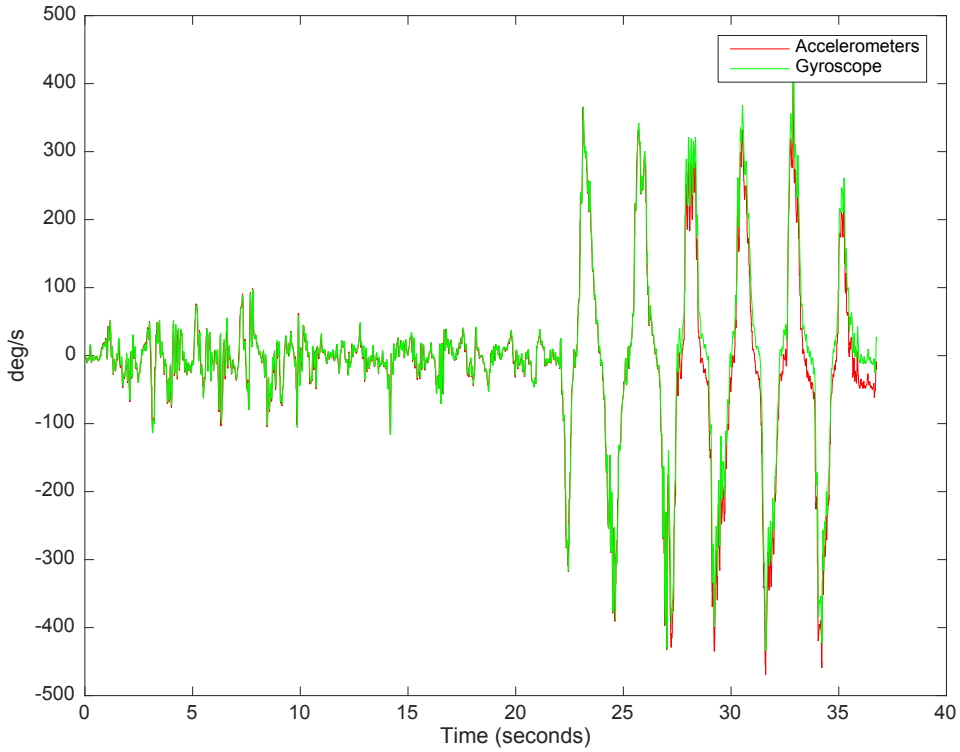

Figure 5.8: Simulated angular velocity about the X-axis (ω_x)

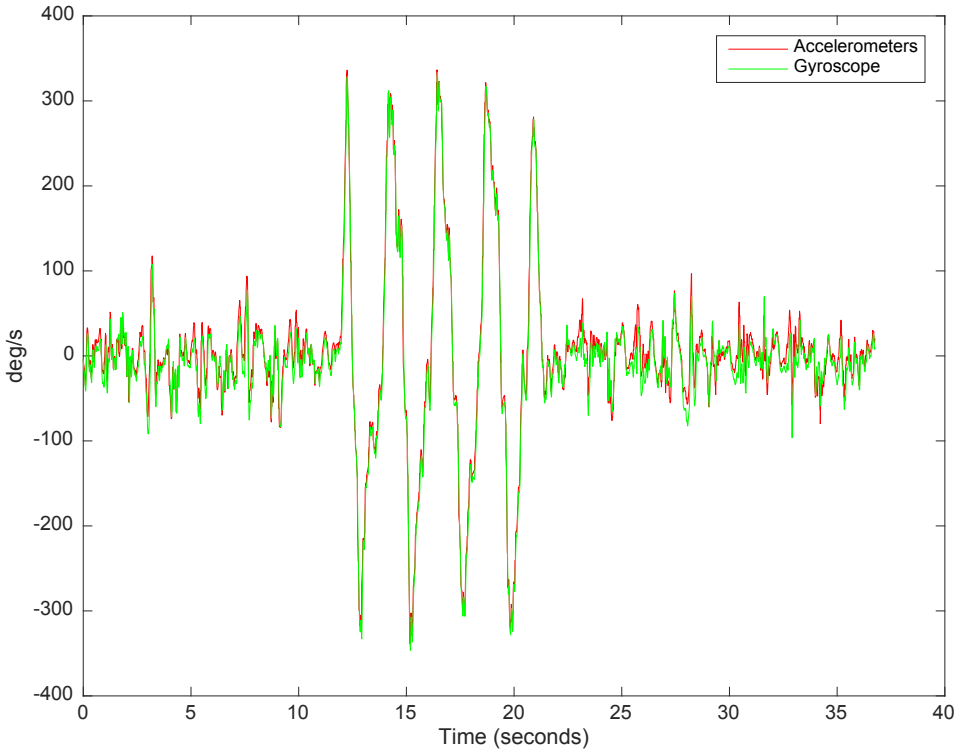

Figure 5.9: Simulated angular velocity about the Y-axis (ω_y)

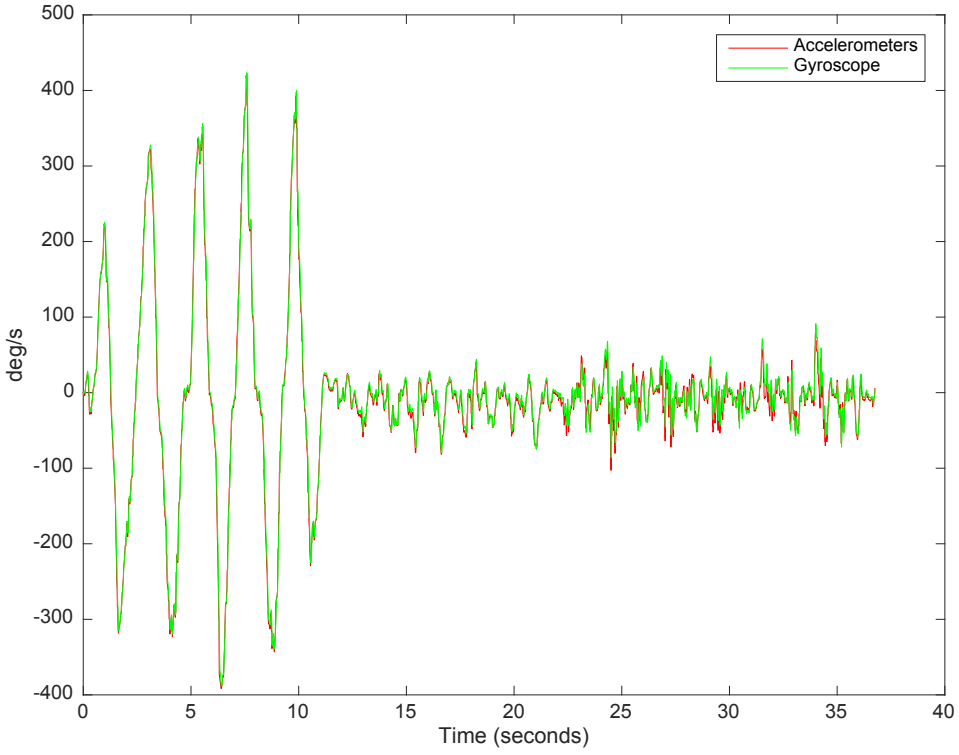

Figure 5.10: Simulated angular velocity about the Z-axis (ω_z)

5.3 Testbed Results

The following sections will present and discuss the results obtained from the testbed implementation. This includes a comparison between the angular velocity estimate from the GF-IMU configuration and the reference gyroscope. The angular velocity estimate is further used to see how accurate it is possible to compute the angle of rotation. An analysis of the current consumption between the two solutions is presented at the end.

5.3.1 Angular Velocity - EcoIMU Equations

Results from using the EcoIMU algorithm from Section 3.3 with the timer-based data acquisition approach at 100Hz directly yielded the results shown in Figure 5.14, 5.15 and 5.16. These figures respectively shows the angular velocity about the X, Y and Z-axis from the accelerometer estimate (red line) and the reference gyroscope (green line). The X-axis on these plots shows the sample number. In order to convert this into time, simply multiply the sample number n with the sampling period dt ($t = n \cdot dt$).

In order to know exactly how much the two graphs deviated from each other, it was decided to compute the mean squared error (MSE) between the two graphs. Four angular velocity measurements were performed for each of the three rotational axes. The MSE was calculated for each measurement, as shown in Table 5.1. The average MSE of the four measurements in each of the three columns is presented at the bottom row of the table.

As seen from the figures, none of the angular velocity estimates from the accelerometers are exceptionally good at following the gyroscope. The estimate about the Y-axis and Z-axis are the best ones, with an average MSE of 730.54 $(deg/s)^2$ and 570.06 $(deg/s)^2$ respectively. The estimate about the X-axis has an average MSE of 1.43e+03 $(deg/s)^2$, which is nearly twice as much as the estimate about the Y-axis.

Measurement	**MSE(ω_x)** $(deg/s)^2$	**MSE(ω_y)** $(deg/s)^2$	**MSE(ω_z)** $(deg/s)^2$
1	1.58e+03	758.87	501.58
2	1.27e+03	665.26	526.27
3	1.19e+03	779.22	687.77
4	1.68e+03	718.80	564.61
Avg	1.43e+03	730.54	570.06

Table 5.1: MSE - EcoIMU Equations

This observation is to some extent similar to that of the simulation results, and can be explained from looking at the cube geometry of the initial testbed configuration. The first quadrant of the XY, YZ and XZ-projection of testbed cube from Figure 4.6 is shown in Figure 5.11, 5.12 and 5.13. The calculated diameter and tangential angle of each cube projection are respectively presented in Equation 5.1 and 5.2.

We know from Section 3.3, that the angular velocity estimate about the Z-axis is calculated from decomposing the X-axis along the enclosing cylinder in the XY-plane. This is also illustrated in Figure 5.11. Notice from Equation 5.1 that the XY-plane has the largest diameter, and thereby the largest enclosing cylinder of the three projections. From the theoretical analysis, we know this

means that the measured acceleration is going to be larger for a rotation about the Z-axis compared to a rotation about the X and Y-axis. The noise is therefore going to be a less predominant factor during quantization for a rotation about the Z-axis. This explains why the angular velocity about the Z-axis appears less jagged than the other two estimates.

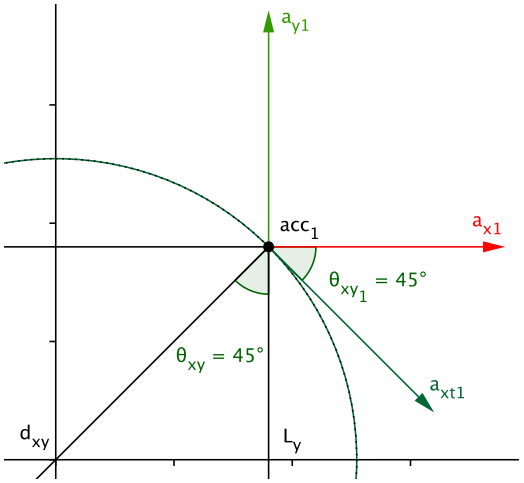

Figure 5.11: First quadrant of XY-projection of testbed cube

$$d_{xy} = 5.66cm \quad d_{yz} = 4.18cm \quad d_{xz} = 4.18cm \tag{5.1}$$

$$\theta_{xy} = 45° \quad \theta_{yz} = 73.3° \quad \theta_{xz} = 16.69° \tag{5.2}$$

The angular velocity estimate about the Y-axis is calculated from decomposing the Z-axis along the enclosing cylinder of the XZ-plane. As seen in Figure 5.12. From Equation 5.2 we can see that the XZ-plane has the smallest tangential angle. This means that the Z-axis is closer to being aligned with the tangential acceleration vector of the enclosing cylinder in the XZ-plane, which is also seen from Figure 5.12. Thus, the measured Z-acceleration is going to be larger for a rotation about Y-axis than the X-axis, even though they both have the same diameter.

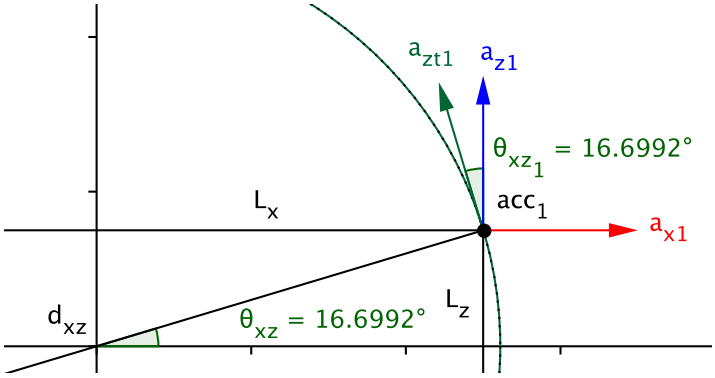

Figure 5.12: First quadrant of XZ-projection of the testbed cube

A rotation about the X-axis is calculated from the enclosing cylinder in YZ-plane. From Equation 5.1 and 5.2, we can clearly see that this plane has the largest tangential angle and the smallest

diameter. The measured Y-acceleration is therefore going to be much smaller and more prone to quantization errors during a rotation about the X-axis.

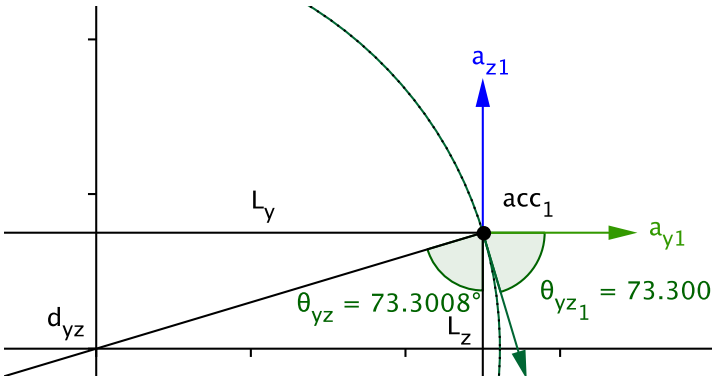

Figure 5.13: YZ-projection of testbed cube

The simulation results from Section 5.2 predicted that the angular velocity estimate about the X-axis would be the worst one, however one can clearly see that the actual estimates from the testbed are much worse than what the simulation predicted. From Figure 5.15, one can observe that the angular velocity estimate from the accelerometers is not able to recover from a rotation at several points (i.e. at sample 300, 400, 500 and 750). This observation cannot be explained from quantization noise alone. Rather, this observation might be related to a fundamental difference in working principle between a MEMS gyroscope and MEMS accelerometer. Namely, that the proof mass inside a MEMS accelerometer lags the frame motion.

It is worth mentioning that oversampling was explored as a means to improve the EcoIMU results. Using an average of two samples made the estimate about the X-axis less jagged. However, the estimate was still not able to follow the data produced by the gyroscope.

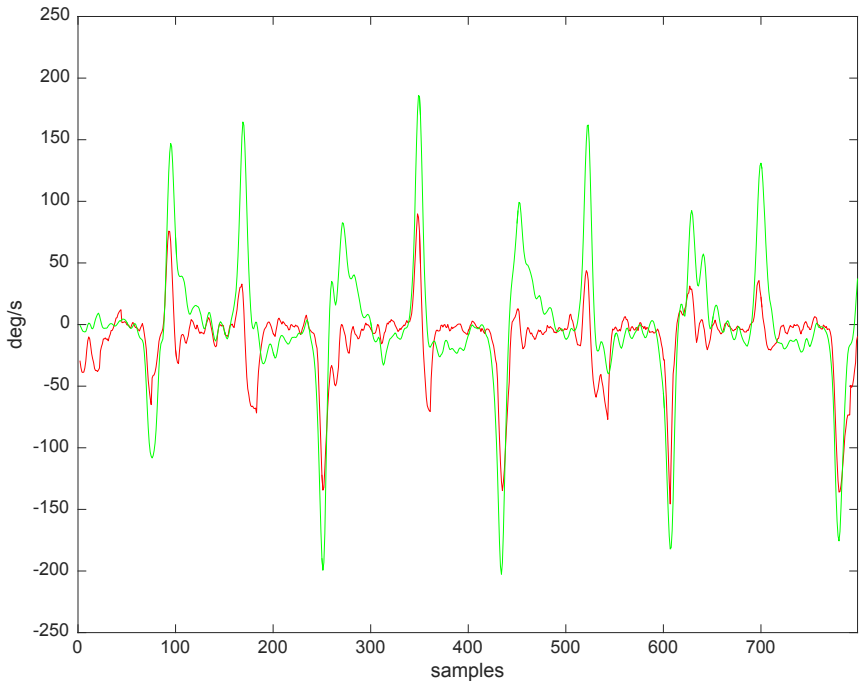

Figure 5.14: Angular velocity about the X-axis (ω_x) using EcoIMU equations at 100Hz

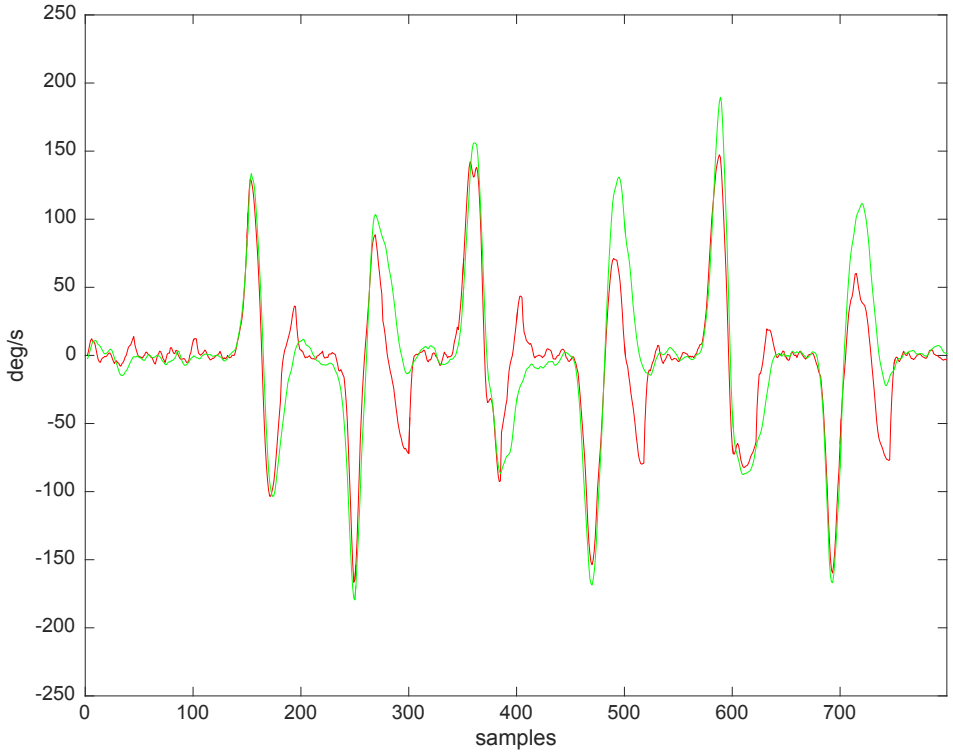

Figure 5.15: Angular velocity about the Y-axis (ω_y) using EcoIMU equations at 100Hz

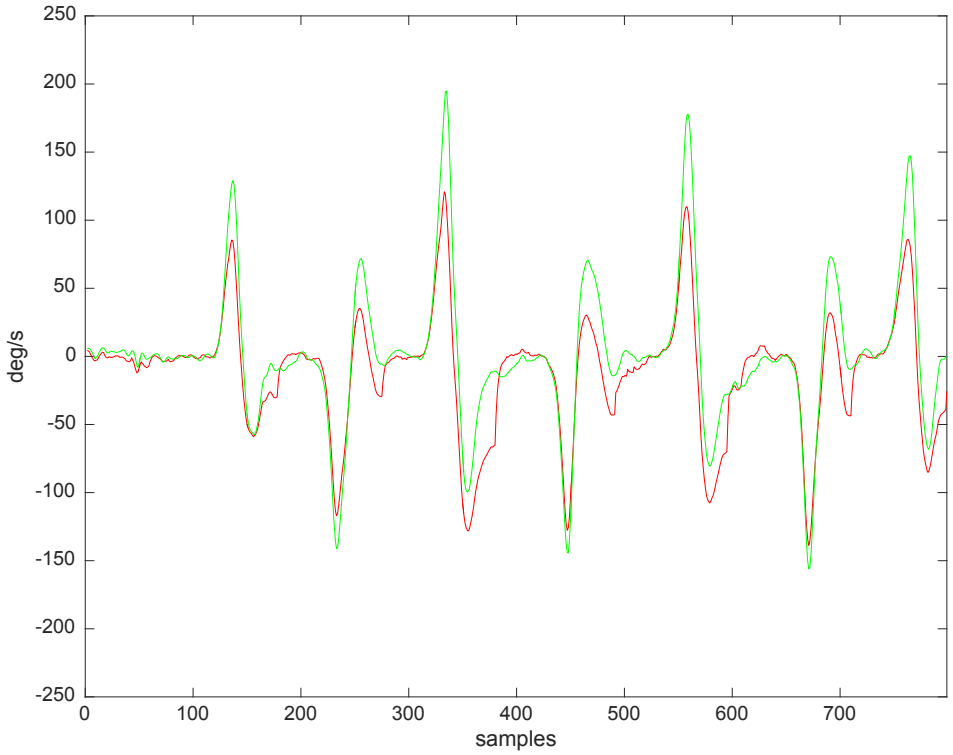

Figure 5.16: Angular velocity about the Z-axis (ω_z) using EcoIMU equations at 100Hz

5.3.2 Alternative Implementation

All of the angular velocity estimates from the EcoIMU algorithm had a relatively high MSE value. In order to get better results, it was attempted to optimize the EcoIMU algorithm. From Figure 3.3, it is shown that two tangential vectors can be used to to calculate the angular velocity for each projection. The hypothesis was that maybe the angular velocity estimate could be improved by using an average of both tangential vectors instead of just using one. Equation 3.6, 3.7 and 3.8 from Section 3.3 was therefore replaced with Equation 5.4, 5.5 and 5.6. A set of complementary angles was also necessary in order to decompose the perpendicular acceleration axis for each projection, as seen in Equation 5.3. The rest of the EcoIMU equations were the same as before.

$$\theta_{xy',yz',xz'} = \left(\arcsin\left(\frac{L_y}{d_{xy}}\right), \arcsin\left(\frac{L_z}{d_{yz}}\right), \arcsin\left(\frac{L_x}{d_{xz}}\right)\right) \tag{5.3}$$

$$a_{xy_{t1}} = \frac{a_{x1}\cos(\theta_{xy}) - a_{y1}\cos(\theta_{xy'})}{2} \quad a_{xy_{t2}} = \frac{a_{x2}\cos(\theta_{xy}) - a_{y2}\cos(\theta_{xy'})}{2} \tag{5.4}$$

$$a_{xz_{t1}} = \frac{a_{x1}\cos(\theta_{xz}) - a_{z1}\cos(\theta_{xz'})}{2} \quad a_{xz_{t2}} = \frac{a_{x2}\cos(\theta_{xz}) - a_{z2}\cos(\theta_{xz'})}{2} \tag{5.5}$$

$$a_{yz_{t1}} = \frac{a_{y1}\cos(\theta_{yz}) - a_{z1}\cos(\theta_{yz'})}{2} \quad a_{yz_{t2}} = \frac{a_{y2}\cos(\theta_{yz}) - a_{z2}\cos(\theta_{yz'})}{2} \tag{5.6}$$

5.3.3 Angular Velocity - Improved Equations

Results from using the improved equations are shown in Figure 5.17, 5.18 and 5.19. These figures respectively shows the angular velocity about the X, Y and Z-axis from the accelerometer estimate (red line) and the reference gyroscope (green line). Four MSE measurements was also performed for each axis, as shown in Table 5.2.

From the figures, one can immediately see that all of the angular velocity estimates are significantly improved compared to the results from using the original EcoIMU equations. Take special notice on how good angular velocity estimate about the Z-axis in Figure 5.19 is, with an average MSE of only 64.68 $(deg/s)^2$.

Measurement	**MSE(ω_x)** $(deg/s)^2$	**MSE(ω_y)** $(deg/s)^2$	**MSE(ω_z)** $(deg/s)^2$
1	620.30	517.57	53.54
2	776.77	520.45	57.87
3	701.19	480.55	102.26
4	548.05	471.40	45.03
Avg	661.58	497.49	64.68

Table 5.2: MSE - Improved Equations

The percentage decrease between the average MSE of the EcoIMU equations and improved equations is presented in Table 5.3.

Measurement	**MSE**(ω_x) $(deg/s)^2$	**MSE**(ω_y) $(deg/s)^2$	**MSE**(ω_z) $(deg/s)^2$
EcoIMU	1.43e+03	730.54	570.06
Improved Equations	661.58	497.49	64.68
%-Decrease	53.7%	31.9%	88.7%

Table 5.3: MSE - Percentage Decrease

From Figure 5.17, 5.18 and 5.19, one can clearly see that the improved equations are able to better recover after a rotation. The issue is still noticeable in the angular velocity estimate about the Y-axis in Figure 5.18, but less severe compared to the one in Figure 5.15 using the original EcoIMU equations.

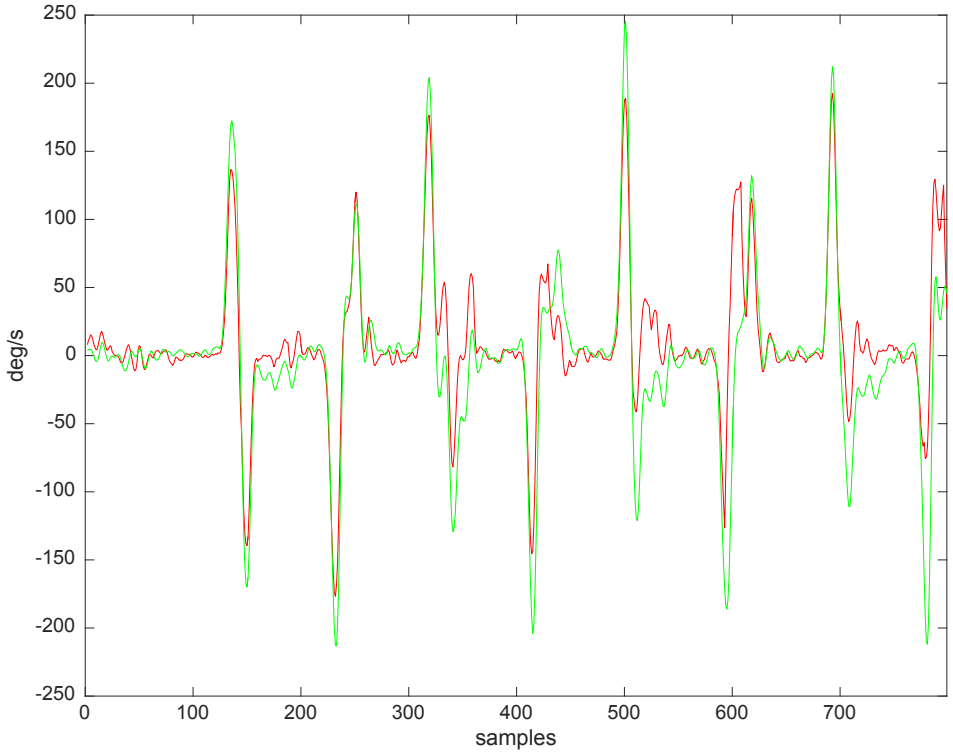

Figure 5.17: Angular velocity about the X-axis (ω_x) using improved equations at 100Hz

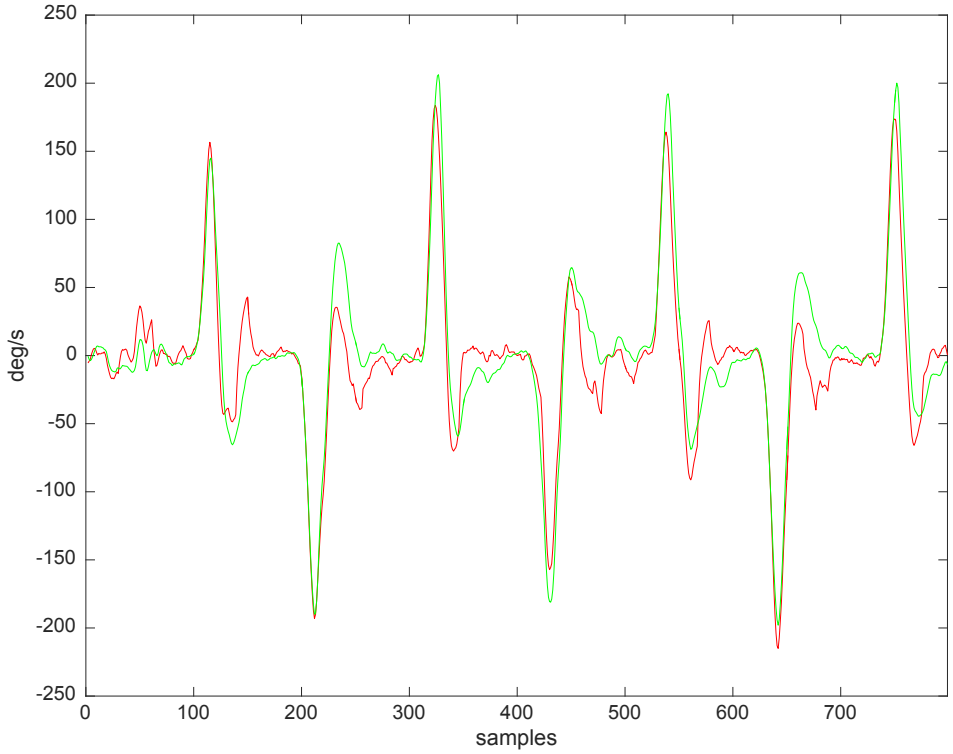

Figure 5.18: Angular velocity about the Y-axis (ω_y) using improved equations at 100Hz

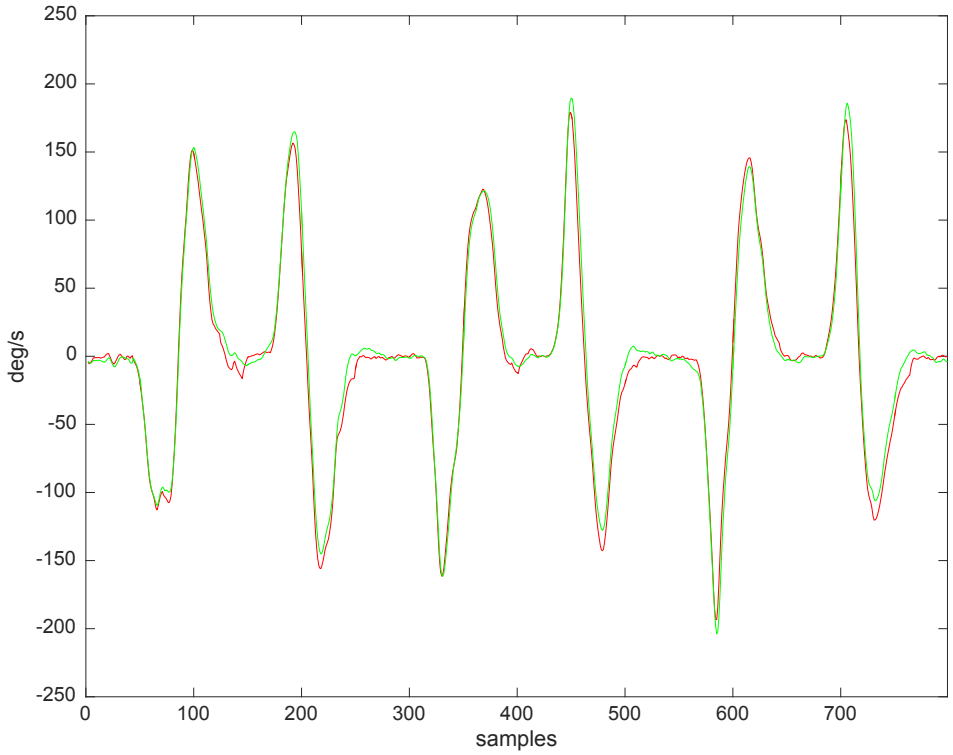

Figure 5.19: Angular velocity about the Z-axis (ω_z) using improved equations at 100Hz

5.3.4 Limitations

After implementing the original EcoIMU algorithm, it was soon discovered some unforeseen limitations with the system. The issue is shown in Figure 5.20 and 5.21. Figure 5.20 shows the angular velocity estimate about the Y-axis while a rolling motion is applied to the system. Notice how little the gyroscope (green line) is affected compared to the accelerometer configuration (red line). The same problem also applied to the Z-axis, as seen in Figure 5.21. This figure shows angular velocity about the Z-axis while a pitching motion is applied to the system. The same problem was also present using the improved equations.

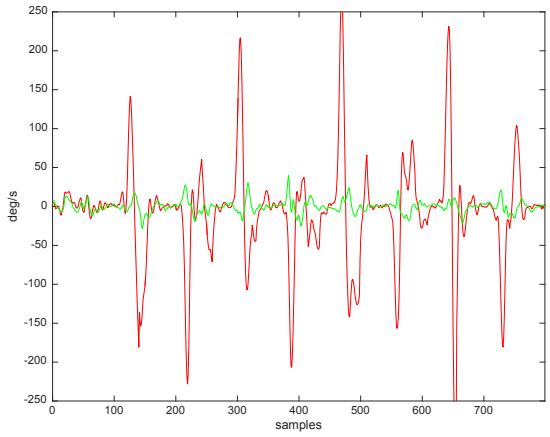

Figure 5.20: ω_y while rolling

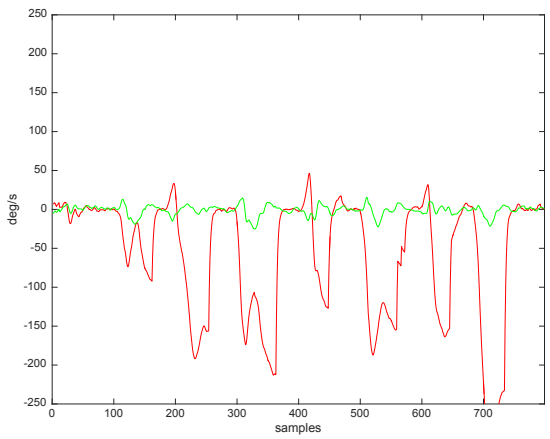

Figure 5.21: ω_z while pitching

At first, it was assumed that there was some error in the testbed implementation. But, after cross checking with the EcoIMU implementation several times, it was discovered that this had to be a physical limitation with the EcoIMU algorithm.

Problem Analysis

We know that the EcoIMU publication computes angular velocity about the X, Y and Z-axis by creating enclosing cylinders around each face of a bounding cube. A separate acceleration axis is decomposed and used as a tangential acceleration vector for each cylinder. Seemingly, the presented equations for this approach should work on paper, as shown from the Matlab simulation results. The problem is that there exists a dependency between the acceleration axes that are used for each bounding cylinder, thereby making it impossible to calculate the angular velocity about each of the three axes at the same time. The issue is illustrated in Figure 5.22, which shows the bounding cube projected in the XY, XZ and YZ plane. From Figure 5.22, it can be seen that a rotation about the Z-axis can be measured using either the tangential Y-axis or the tangential X-axis from the corner accelerometers in the XY-plane. A rotation about the Y-axis can be measured using either the tangential Z-axis or tangential X-axis in the XZ-plane. Notice how the X-axis is present in both the XY and the XZ-projections. This means that the X-axis is going to measure an acceleration when there is a rotation about the Z-axis and when there is a rotation about the Y-axis. There is also a dependency between the Y-axis in the XY-projection and the Y-axis in the YZ-projection. Essentially, this makes it impossible to compute angular velocity about all three axes at the same time. This is a fundamental flaw with the EcoIMU approach.

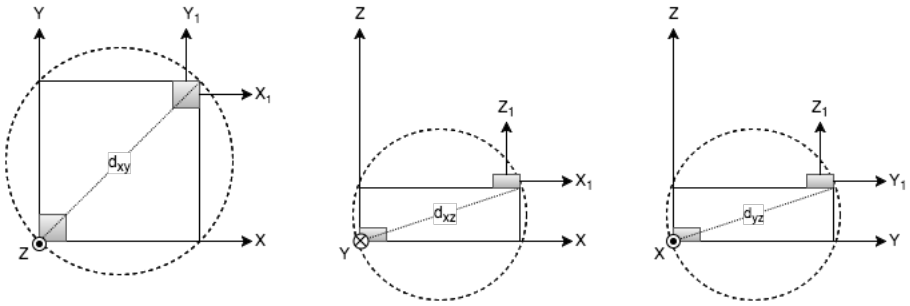

Figure 5.22: Cube projected in the XY, XZ and YZ plane

The axis dependency issue is not noticeable in the Matlab simulation. This can simply be explained from the working principle behind the simulation. From Section 4.3.1, we know that the simulation works by deriving and decomposing angular velocity from a smartphone gyroscope down to measured acceleration onto a virtual bounding cube. The simulation does this by using the original EcoIMU equations, which assigns a separate acceleration axis to each of the three enclosing cylinders. This approach does not take the axis dependency into account, thus the issue is not going to be visible in the end simulation results.

The axis dependency issue was not stated explicitly anywhere in the EcoIMU report. On the contrary, the publication gave the impression that two tri-axial accelerometers would be sufficient to create a fully functional IMU.

5.3.5 Low Power Optimization

Albeit the axis dependency issue, the improved equations yielded satisfactory performance in terms of determining angular velocity about the X, Y and Z-axis separately. It was now interesting to see how the system would perform in the accelerometer LP-mode from Section 4.5.1.

Synchronization Problems

We know that there is an inherent synchronization problem with using the LP-mode. To better illustrate this, it was decided to do a simple investigation. A Saleae Logic analyzer tool was connected to the INT pin of each GY-6500 module as well as a GPIO on the nRF52 DK. The GPIO was configured to toggle after both interrupts from the modules had been detected. The accelerometers was first configured to sample at a frequency of 125Hz. Figure 5.23 and 5.24 clearly shows the synchronization issue in the LP-mode. At the beginning in Figure 5.23, both accelerometers are close to being perfectly synchronized. Figure 5.24 shows the same configuration after a few seconds. Notice from this figure how the interrupt signal from both accelerometers has drifted apart from each other.

As mentioned in Section 4.5.1 the synchronization problem only becomes important when the sampling frequency is low. Reducing the LP-frequency from 125Hz down to 31.25Hz clearly illustrates this. Figure 5.25 shows the initial in-sync interrupt timing, while Figure 5.26 shows the same configuration after a few seconds. Since the time between samples becomes larger when the frequency goes down, synchronization suddenly becomes more important.

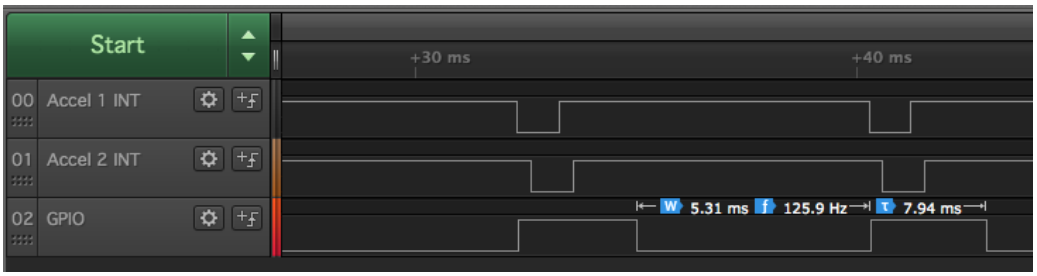

Figure 5.23: 125Hz LP-mode in-sync

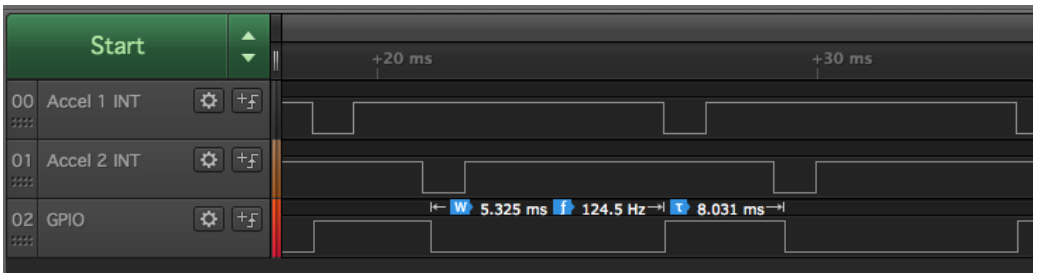

Figure 5.24: 125Hz LP-mode out-of-sync

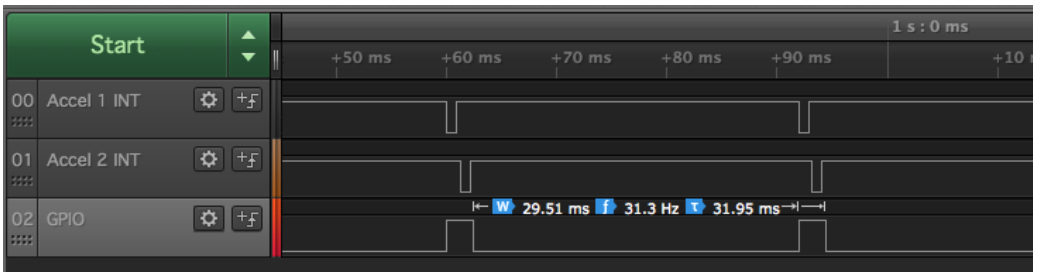

Figure 5.25: 31.25Hz LP-mode in-sync

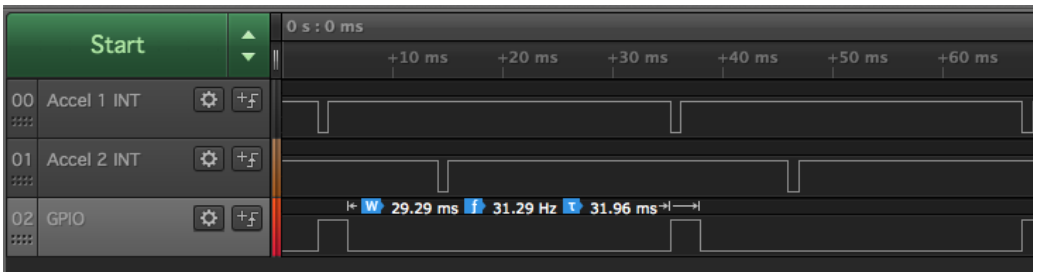

Figure 5.26: 31.25Hz LP-mode out-of-sync

Angular Velocity - LP-Mode

Despite the synchronization problem, it was decided to test how the LP-mode would compare to the initial timer-based approach. Only the angular velocity about the Z-axis was considered for these measurements, as this estimate was proven to be the most accurate.

The results from running the improved equations in the LP-mode is shown in Figure 5.27, 5.28 and 5.29. These figures respectively shows the angular velocity about Z-axis for 125Hz, 62.5Hz and 31.25Hz. Table 5.4 shows the MSE about the Z-axis for four measurements at 125Hz, 62.5Hz and 31.25Hz.

From these figures, we can see that the LP-mode at 125Hz is comparable to the timer-based approach at 100Hz. This suggests that the LP-mode synchronization issues at this frequency are negligible. For the measurements at 62.5Hz in Table 5.4, one can clearly see that there is a large variance in the data. The best MSE value is 150.18 $(deg/s)^2$ and the worst is 410.31 $(deg/s)^2$. It is suspected that this is a result of the synchronization issues in the LP-mode. We see the same situation for the LP-mode down at 31.25Hz, a large variance in the MSE.

Measurement	**MSE(ω_z) at 125Hz**	**MSE(ω_z) at 62.5Hz**	**MSE(ω_z) at 31.25Hz**
1	60.90	150.18	274.85
2	86.60	410.31	599.52
3	40.09	242.53	531.72
4	79.07	218.25	541.63
Avg	66.67	255.32	486.93

Table 5.4: MSE - LP-Mode

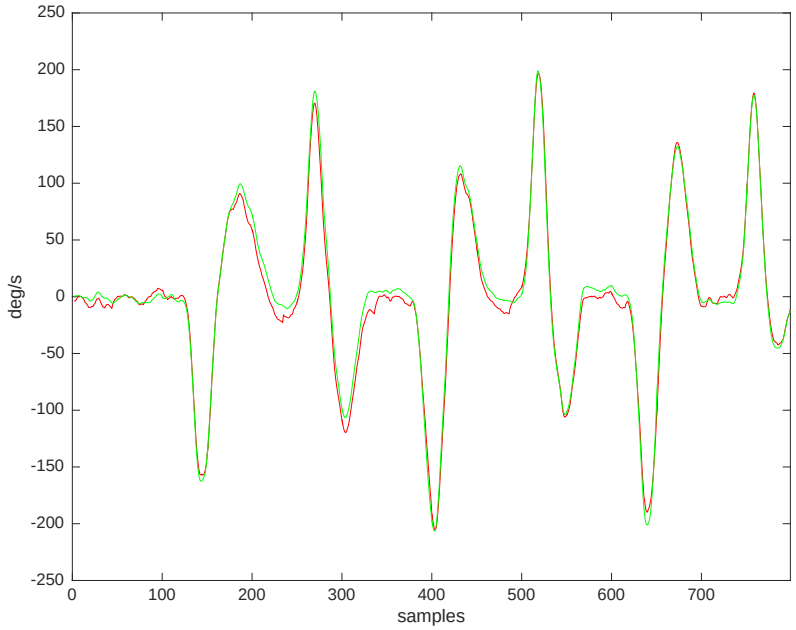

Figure 5.27: Angular velocity about Z-axis (ω_z) at 125Hz ODR

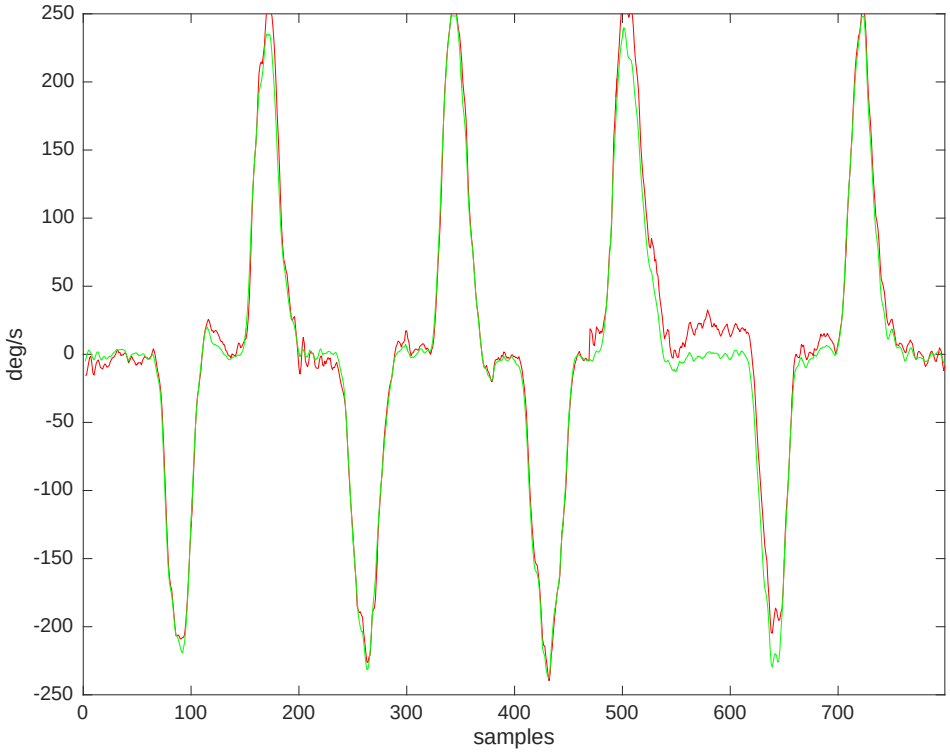

Figure 5.28: Angular velocity about the Z-axis (ω_z) at 62.5Hz ODR

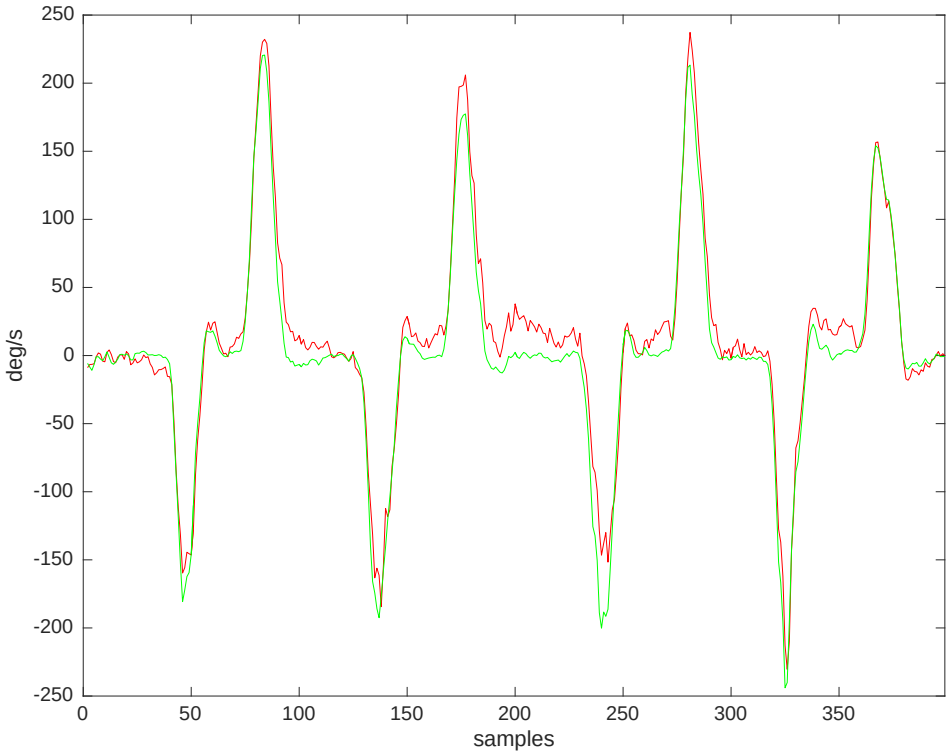

Figure 5.29: Angular velocity about the Z-axis (ω_z) at 31.25Hz ODR

5.3.6 Angle Measurement

From Section 2.3, we know that data from an IMU is often used compute an attitude consisting of roll, pitch and yaw. Up until now, the focus has been on comparing the angular velocity estimate from the accelerometer configuration against that of a conventional gyroscope. With the improved equations, it has been shown that it is possible to obtain an angular velocity estimate that is close to that from a gyroscope. In this section we will look at how the angular velocity estimate can be used to compute the rotational angle. It was decided to do this only for the Z-axis, as the results clearly show that it is the most precise estimate of the three.

The angle estimate was first calculated using the initial timer-based approach, since these measurements had the lowest MSE. Figure 5.30 shows the yaw angle plotted as degrees. The angle estimate was found to be accurate within $\pm 10 deg$ for the first rotations, which is surprisingly good considering that the approach is based on double integration. As predicted, the angle estimate would eventually drift away as more time went on. This is clearly illustrated at the end of Figure 5.30.

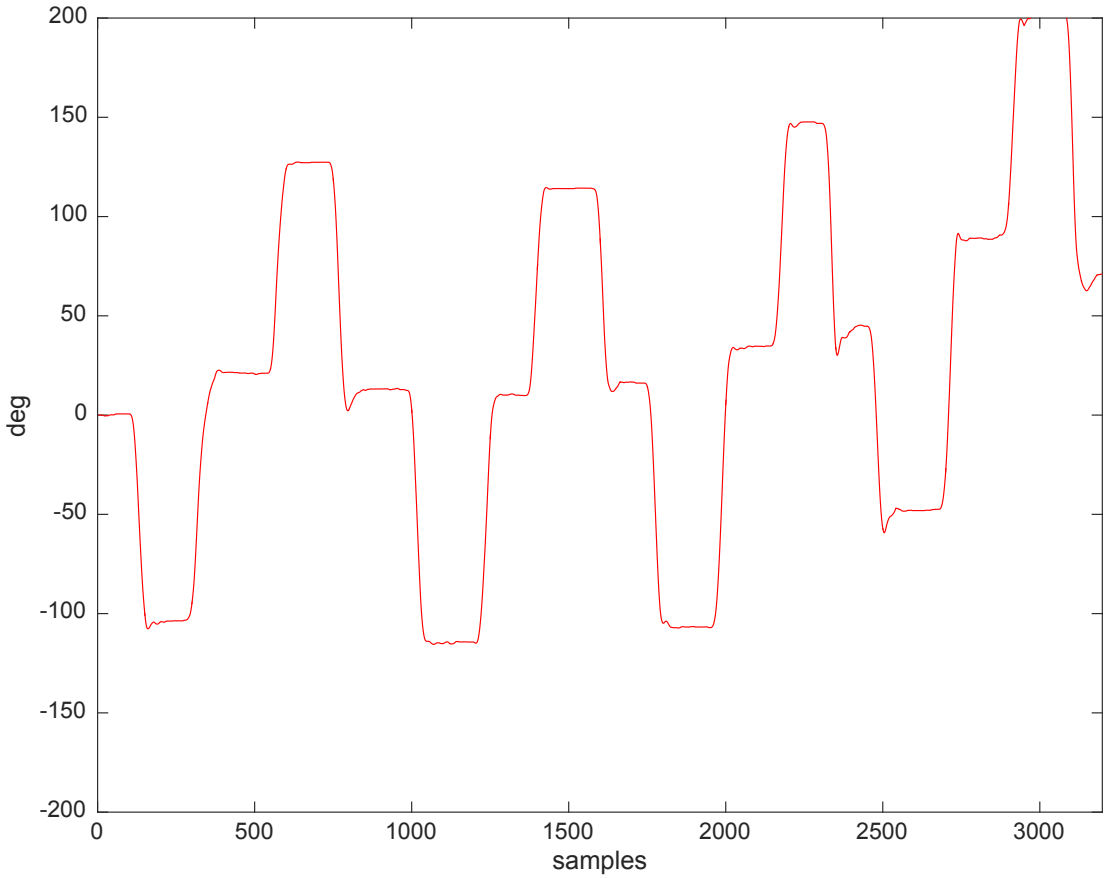

Figure 5.30: Yaw estimate at 100Hz using the timer-based approach

It was then decided to test the yaw estimate using the accelerometer LP-mode. The LP-mode was first configured to use a frequency of 125Hz. The angle estimate for this frequency is shown in Figure 5.31. One can immediately see that the estimate is less clean than the one for the timer-based approach in Figure 5.30. Apart from that, the two estimates are close to being identical at this frequency.

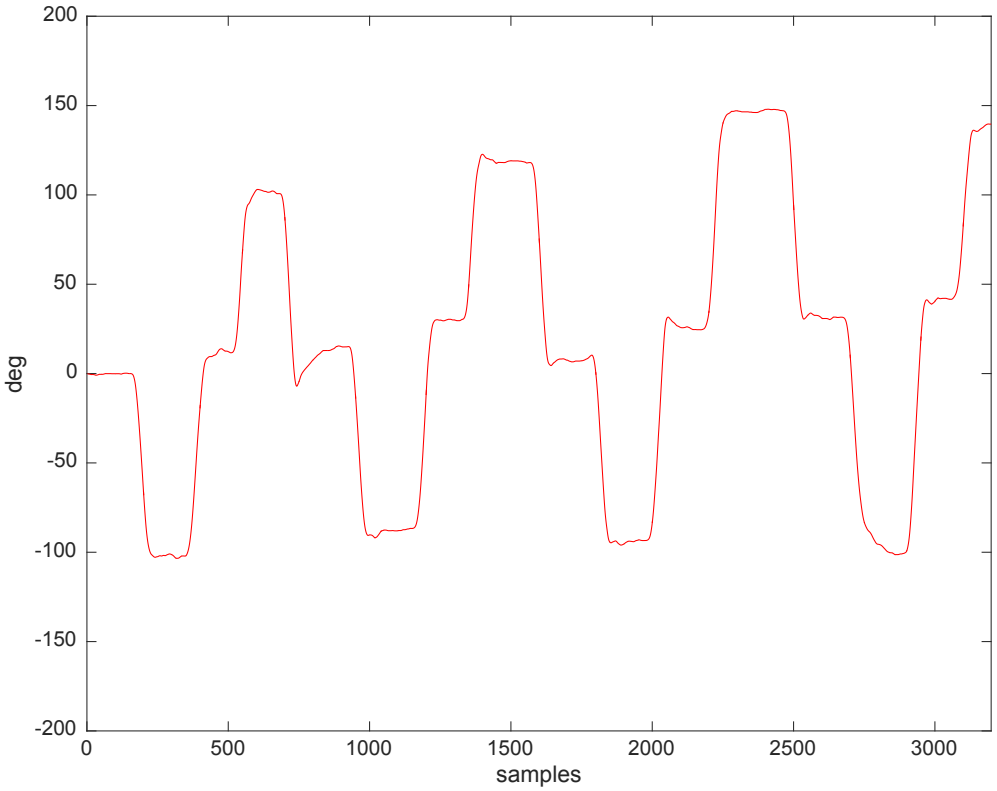

Figure 5.31: Yaw estimate at 125Hz using the LP-mode

The angle of rotation about the Z-axis was further computed for an LP-mode frequency of 62.5Hz, as seen in Figure 5.32. From this figure, one can see that the angle estimate is surprisingly good during the first three rotations, still within $\pm 10 deg$. For the last rotation, the angle is starting to diverge rather drastically.

Figure 5.33 shows the angle estimate down at 31.25Hz. Even at this frequency, the angle estimate is surprisingly accurate for the first two rotations. It is suspected that the angle estimate at 62.5Hz and 31.25Hz could have worked better if the synchronization issues in the LP-mode had been resolved.

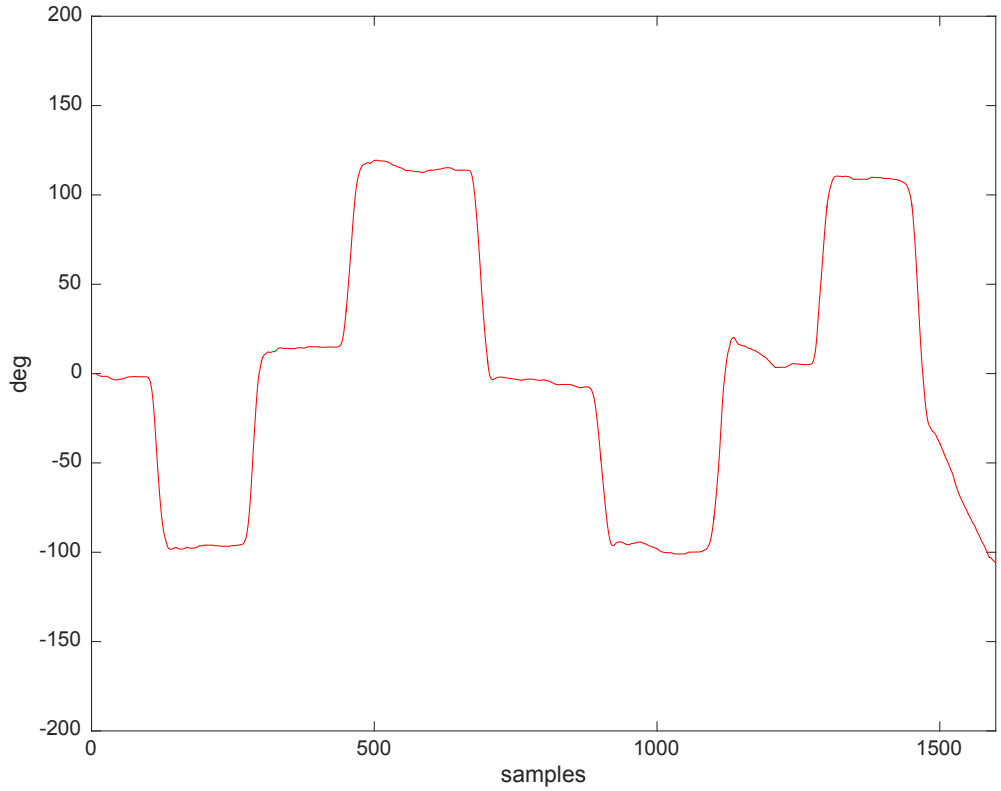

Figure 5.32: Yaw estimate at 62.5Hz using the LP-mode

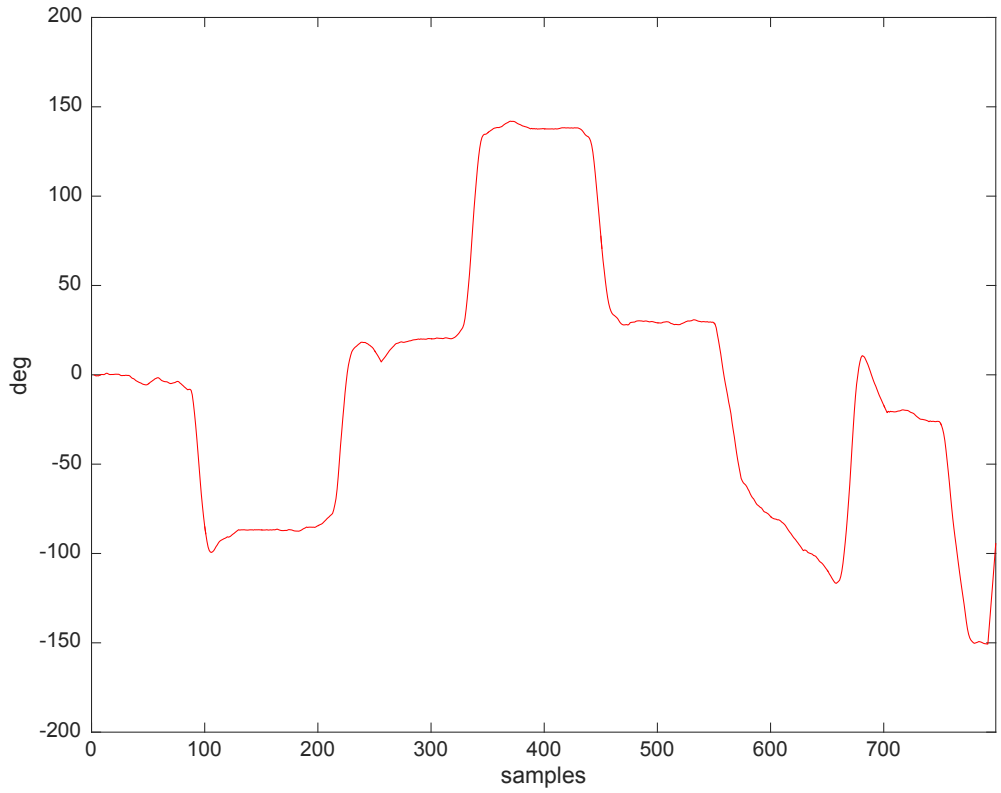

Figure 5.33: Yaw estimate at 31.25Hz using the LP-mode

5.3.7 Current Consumption

The whole motivation behind creating a GF-IMU using accelerometers is the potential gain in energy efficiency. This section presents results from the current consumption measurements for the GF-IMU testbed.

The results are shown in Table 5.5. The first two columns in the table respectively show the current consumption for both corner accelerometers and the nRF52. These two numbers are added together in the third column. Thus, they constitute the current consumption for the GF-IMU configuration. The fourth column shows the current consumption for the center GY-6500 module, using both the gyroscope and the accelerometer. This column constitutes the current consumption of a conventional IMU. A percentage saving between the GF-IMU and the conventional gyro-based IMU is presented in the fifth column. A screen capture from the oscilloscope for each measurement can be found in Appendix B.

Configuration	**2x Accel**	**nRF52**	**GF-IMU**	**IMU (Gyro + Accel)**	**%-Saving**
100Hz Timer ISR	906µA	548µA	1.45mA	3.1mA	53.2%
125Hz LP-mode	244µA	567µA	811µA	3.1mA	73.8%
62.5Hz LP-mode	174µA	550µA	724µA	3.1mA	76.6%
31.25Hz LP-mode	128µA	504µA	632µA	3.1mA	79.6%

Table 5.5: Current consumption for GF-IMU and conventional IMU

The results from Table 5.5 clearly shows how energy efficient a GF-IMU configuration is compared to a conventional gyro-based IMU. When using the timer-based data acquisition approach, the GF-IMU has a 53.2% reduction in current consumption compared to that of a conventional IMU. Using the accelerometer LP-mode further reduces this number. At 31.25Hz, the GF-IMU configuration has a 79.6% reduction i current consumption compared to that of a conventional IMU.

6 Applications

From the previous chapter, we have now seen how a GF-IMU performs compared to a conventional gyro-based IMU with regards to both precision and current consumption. The following chapter aims to further discuss some practical applications for a GF-IMU implementation.

At this point, it has already been established that a GF-IMU configuration using only two tri-axial accelerometers is less precise than a conventional gyro-based IMU. We have also discovered a limitation with the GF-IMU system, which effectively makes it impossible to compute the rotational angle about all three axes at the same time. On the pluss side, we have found that a GF-IMU configuration is far more energy efficient than gyro-based solutions. The area of application for the two solutions is therefore different. While a conventional IMU is precise enough to be used for inertial navigation applications, a GF-IMU would be better suited for applications that only require angle estimation over a short duration of time. A typical application here would be gesture detection. In the following sections we are going to investigate how a GF-IMU can be used exactly for this purpose.

6.1 Gesture Detection Example

Let's envision an IoT sensor module that is hermetically sealed inside a small plastic container. The package has been sealed in order to protect the inside electronics from environmental factors such as rain and dust. We further assume that this module is going to be used for control input. It would be very impractical to fit such a product with mechanical switches, as it would both occupy too much space and also risk compromising the hermetical seal. Thus, the control input would need to reside inside the package. A GF-IMU based gesture detection scheme could be an excellent solution to this.

From Figure 5.33 in Chapter 5 we know that the yaw estimate from the GF-IMU can be computed with an accuracy of ± 10 degrees for the first two 90 degrees rotations using the LP-mode at a frequency of 31.25Hz. This level of precision would be more than good enough for a system that only needs to determine whether the user turns the device in clockwise or counterclockwise direction, as this would only require the estimate to be accurate over short duration of time. This application would also only require rotation about the Z-axis, so the axis dependency issue between the other axes would not be of concern.

We divide the rotations in Figure 5.33 into three sectors, as illustrated in Figure 6.1. The first sector illustrates the first gesture, and is comprised of a 90 degrees rotation in a counterclockwise (CCW) direction and a rotation back to zero. The second sector illustrates the second gesture, which is comprised of a 90 degrees rotation in a clockwise (CW) direction and a rotation back to zero. In the third sector, the angle estimate is starting to diverge. The gesture detection system is therefore reset back to zero at this point. Two detectable rotations would give us a total of four possible gestures, namely CW-CW, CCW-CCW, CW-CCW and CCW-CW. These gesture combinations could for instance be used to control indoor lighting appliances.

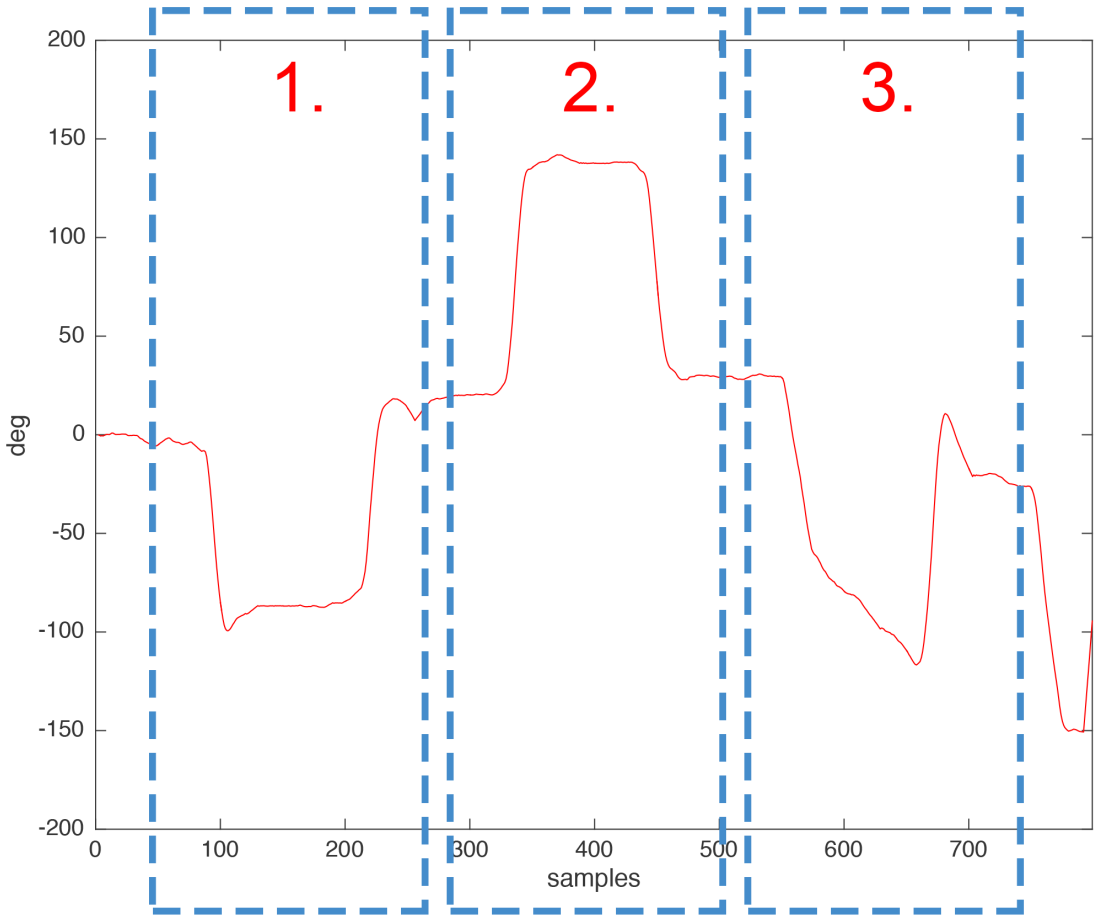

Figure 6.1: Gesture Detection Example

7 Conclusion

The red line of this thesis has been to study the possibility of creating a GF-IMU based on two tri-axial accelerometers. As a basis, we have used an existing solution named EcoIMU. Both a simulation based approach and a practical implementation has been used to further investigate this system. To conclude the work of this thesis, we revisit the research questions from Section 1.4, with conclusions on our findings.

1. *What is the relationship between cube geometry and the parameters of precision and power consumption in a GF-IMU configuration?*

   The designed Matlab simulation has been used to provide a theoretical analysis on how different cube configurations affects the parameters of precision and power consumption in a GF-IMU. This analysis has shown that both the accelerometer distance from cube center and the angle between the measurement axis and the tangential acceleration vector of the bounding cylinder affects the overall precision of the system. By making the cube rectangular, it is possible to optimize the precision for a rotation about one axis, at the cost of a lower precision about the other two axes. With regards to power consumption, we have found that there is a relationship between the applied motion and the selected ODR of the accelerometers. A simple investigation suggests that the ODR should at least be 30Hz for applications that are based on human interaction.

2. *How precise and how energy efficient is a GF-IMU implementation compared to a conventional gyro-based IMU?*

   A practical testbed has been used to compare the precision and current consumption between a GF-IMU and conventional gyro-based IMU. By using our improved set of equations for the EcoIMU algorithm, we have shown that the testbed can be used to determine angular velocity about one free axis of choice for a cube configuration of 4x4x1.2cm. The angular velocity estimate about the Z-axis is the best one, as it is calculated from largest cube projection. Using the timer based approach from Section 4.5.1, we have shown that angular velocity about the Z-axis can be computed with an average mean squared error (MSE) of 64.68 $(deg/s)^2$. The angular velocity estimate about the Z-axis can further be used to provide an angle of rotation with an accuracy of ± 10 degrees over a short period of time. We found this to valid even for the accelerometer LP-mode down at 31.25Hz. In this mode, the implemented GF-IMU consumes 79.6% less current than a conventional gyro-based IMU. The testbed results have also revealed a severe limitation with the EcoIMU implementation. Namely, that there is hidden dependency in the equations used for calculating rotation about each axis. This dependency makes it impossible to calculate rotation about the X, Y and Z-axis at the same time.

3. *Can a GF-IMU be used for any practical IoT applications?*

   From Chapter 5, we have found that the implemented GF-IMU is less precise than a conventional IMU. We have also discovered a limitation with the implemented system, which

effectively makes it impossible to compute rotation about all three axes at the same time. On the plus side, we have found that a GF-IMU configuration is far more energy efficient than gyro-based solutions. The area of application between the two is therefore different. While a conventional IMU is precise enough to be used for inertial navigation applications, our system can only be accurate over a short period of time. The implemented system is therefore more suited for applications that only require simple trajectory estimation. From Chapter 6, we have found gesture detection to be a suitable IoT application for our implementation.

8 Further work

During this work, it has been discovered several techniques and approaches that could have been used to improve both the current consumption and precision of the implemented GF-IMU system. These suggestions are presented in the list below.

8.0.1 Improvements - Current Consumption

- A supply voltage of 3.3V was used for the testbed. Both the nRF52 and the MPU-6500 are able to operate at a lower supply voltage than this. Reducing the operating voltage to a working minimum would contribute in reducing the overall power consumption.
- As seen from Table 5.5, the current consumption for the nRF52 is relatively stable around 500µA. Using a faster data acquisition technique for the MPU-6500 accelerometers could have optimized this number further. Remember from Section 4.5.1, that MPU-6500 has the ability to use SPI at 20MHz instead of TWI at 400kHz as serial interface. By using a faster serial interface to acquire data from the modules, the nRF52 would be able to finish the data acquisition faster, thereby enabling it to sleep for a longer period of time. This would contribute in lowering the average current consumption even further.
- The MPU-6500 IMU is not particularly designed for being low power. Rather, it is designed to be a low-cost, general-purpose sensor module that can be used for a broad range of different applications. By using one of the ultra-low power accelerometers from the specialization project, see Table 3.1, instead of the MPU-6500 one could have saved a considerable amount of power.

8.0.2 Improvements - Precision

- As the yaw estimate in a GF-IMU is based on double integration of angular acceleration, drift over time is inevitable. By adding a magnetometer to the system, one could mitigate this drift by using the heading provided by the magnetometer as a point of reference. The magnetometer would not require frequent sampling, so the solution would still be more energy efficient than a conventional IMU.
- Both the timer-based data acquisition approach and the LP-mode in this thesis suffers from synchronization issues. Using MEMS accelerometers with synchronization capabilities would most likely improve the overall precision of the system. Synchronized accelerometers would also give an opportunity to experiment with lower ODR values.

Bibliography

[1] A.D. King, "Inertial Navigation – Forty Years of Evolution," *GEC Review*, vol. 13, no. 3, 1998, (Visited Apr. 2016). [Online]. Available: http://imar.de/downloads/papers/inertial_navigation_introduction.pdf

[2] "Accelerometer Specifications - Quick Definitions," http://www.analog.com/en/products/landing-pages/001/accelerometer-specifications-definitions.html, Analog Devices, 2015, (Visited Oct. 2015).

[3] J.-H. Chen, S.-C. Lee, and D. B. DeBra, "Gyroscope free strapdown inertial measurement unit by six linear accelerometers," in *Journal of Guidance, Control, and Dynamics*, vol. 17, no. 2. American Institute of Aeronautics and Astronautics, March 1994, pp. 286–290. [Online]. Available: http://dx.doi.org/10.2514/3.21195

[4] "MEMS Accelerometer - Price list," http://www.digikey.com/product-search/en/sensors-transducers/accelerometers/1966355?k=accelerometer, Digi-Key, (Visited May. 2015).

[5] "MEMS Gyroscope - Price list," http://www.digikey.com/product-search/en/sensors-transducers/gyroscopes/1967243?k=gyroscope, Digi-Key, (Visited May. 2015).

[6] "Econode - The ultra-compact wireless sensing platform," http://eco.epl.tw/images/main_promo.png, Embedded Platform Lab, (Visited Feb. 2015).

[7] Erlend Hestnes, "Sensor Data Acqusition for Ultra-Low Power Applications," NTNU, December 2015.

[8] "Quick Reference Guide - Accelerometer Terminology Guide," http://cache.freescale.com/files/sensors/doc/support_info/SENSORTERMSPG.pdf, Freescale Semiconductor, 2007, (Visited Oct. 2015).

[9] "Application Note - How many bits are enough?" http://cache.freescale.com/files/sensors/doc/app_note/AN4075.pdf, Freescale Semiconductor, 2010, (Visited Dec. 2015).

[10] R. Hibbeler, *Engineering Mechanics: Dynamics*, 13th ed. Prentice Hall, 2012.

[11] "Application Note - Magnetoresistive Sensors," http://www.arauzsl.com/productos/SENSORES%20DE%20POSICION%20HONEYWELL/HOJAS%20DE%20APLICACION.pdf, Honeywell, (Visited Mar. 2016).

[12] "MPU-6500 Product Specification," https://store.invensense.com/datasheets/invensense/MPU_6500_Rev1.0.pdf, Invensense, (Visited May. 2016).

[13] V. Kaajakari, *Practical MEMS*, v1.03 ed. Small Gear Publishing, 2009.

[14] W. Kester, *Data Conversion Handbook*. Elsevier, 2004.

[15] N. Maluf and K. Williams, *An introduction to microelectromechanical systems engineering*, 2nd ed. Boston : Arteh House, 2004.

[16] "Nordic Developer Zone - FPU Current," https://devzone.nordicsemi.com/question/71533/fpu-current/, Nordic Semiconductor, (Visited May. 2016).

[17] *nRF52832 - Product Specification*, v1.0 ed., Nordic Semiconductor, 2016, (Visited May. 2016).

[18] "Tilt sensing using a three-axis accelerometer," https://www.nxp.com/files/sensors/doc/app_note/AN3461.pdf, NXP Semiconductors, March 2013, (Visited Feb, 2016).

[19] *I2C-bus specification and user manual*, v6.0 ed., NXP Semiconductors, 2014, (Visited May. 2016).

[20] Olli W., "IMU Data Fusing: Complementary, Kalman, and Mahony Filter," http://www.olliw.eu/2013/imu-data-fusing/, (Visited Apr. 2016).

[21] A. J. Padgaonkar, K. W. Krieger, and A. I. King, ""Measurement of Angular Acceleration of a Rigid Body Using Linear Accelerometers"," vol. 42, no. 3. ASME, 09 1975, pp. 552–556. [Online]. Available: http://dx.doi.org/10.1115/1.3423640

[22] Paul J. Gans, "Compass," http://scholar.chem.nyu.edu/tekpages/compass.html.

[23] J. Rowberg and N. Baldeck, "I2CDevLib," https://github.com/jrowberg/i2cdevlib, (Visited May. 2016).

[24] C.-W. Tan, S. Park, K. Mostov, and P. Varaiya, "Design of gyroscope-free navigation systems," in *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*, 2001, pp. 286–291.

[25] Y. L. Tsai, T. T. Tu, and P. H. Chou, "EcoIMU - A compact, wireless, gyro-free inertial measurement unit based on two triaxial accelerometers," in *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, April 2011, pp. 133–134.

[26] O. J. Woodman, "An introduction to inertial navigation," *University of Cambridge*, August 2007, (Visited Mar. 2016). [Online]. Available: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf

[27] "Theory about quaternions," http://www.xojo3d.com/tut012.php, Zoclee, (Visited Apr. 2016).

A Mathematical Deductions

Relationship between power spectral density (PSD) and root-mean-square (RMS) [9].

$$N_{rms}^2 = \int_0^{\infty} PSD(f)df \tag{A.1}$$

$$N_{rms}^2 = \int_0^{BW} PSD(f)df \tag{A.2}$$

$$N_{rms}^2 = PSD \cdot (BW - 0) \tag{A.3}$$

$$N_{rms} = \sqrt{PSD \cdot BW} \tag{A.4}$$

$$PSD = \frac{N_{rms}}{\sqrt{BW}} \tag{A.5}$$

Calculating the effective number of bits (ENOB), from [9].

$$\frac{S_{rms}}{N_{rms}} = \frac{\frac{2^n}{2\sqrt{2}}}{\frac{1}{\sqrt{12}}} = 2^n \cdot \frac{\sqrt{3}}{\sqrt{2}} \tag{A.6}$$

$$(\frac{S_{rms}}{N_{rms}})dB = 20\log 2^n \cdot \frac{\sqrt{3}}{\sqrt{2}} \tag{A.7}$$

$$(\frac{S_{rms}}{N_{rms}})dB = 20\log 2^n + 20\log 1.225 \tag{A.8}$$

$$SNR(dB) = (\frac{S_{rms}}{N_{rms}})dB = 602n + 1.76 \tag{A.9}$$

$$ENOB = n = \frac{SNR(dB) - 1.76}{6.02} \tag{A.10}$$

Calculating the ENOB as an example.

$$N_{\text{rms}} = 4 \cdot 450[\mu g/\sqrt{Hz}] \cdot \sqrt{50}[\sqrt{Hz}] = \frac{9g}{500 \cdot \sqrt{2}} \tag{A.11}$$

$$SNR(\text{dB}) = 20\log \frac{\frac{2g-(-2g)[m/s^2]}{2\sqrt{2}}}{N_{rms}} = 94.21 \tag{A.12}$$

$$ENOB = \frac{94.21 - 1.76}{6.02} = 15.35 \tag{A.13}$$

B Current Measurements

B.1 Gyroscope + Accelerometer (Conventional IMU)

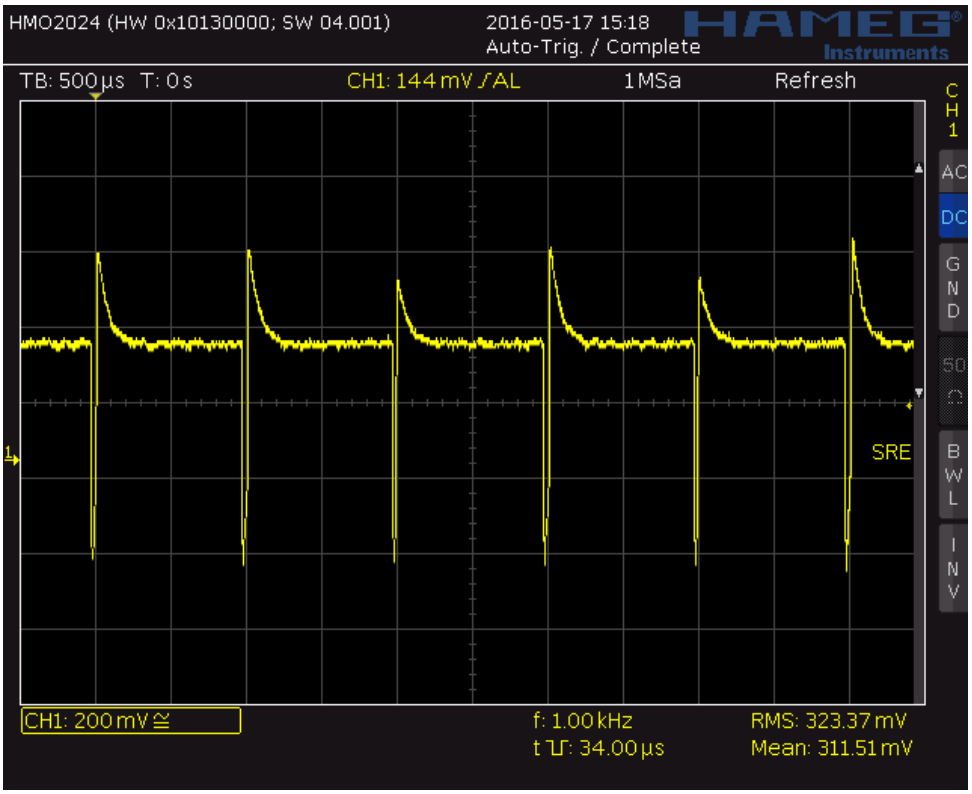

Figure B.1: Current consumption for gyroscope (1mv/µA)

B.2 Timer-Based Approach

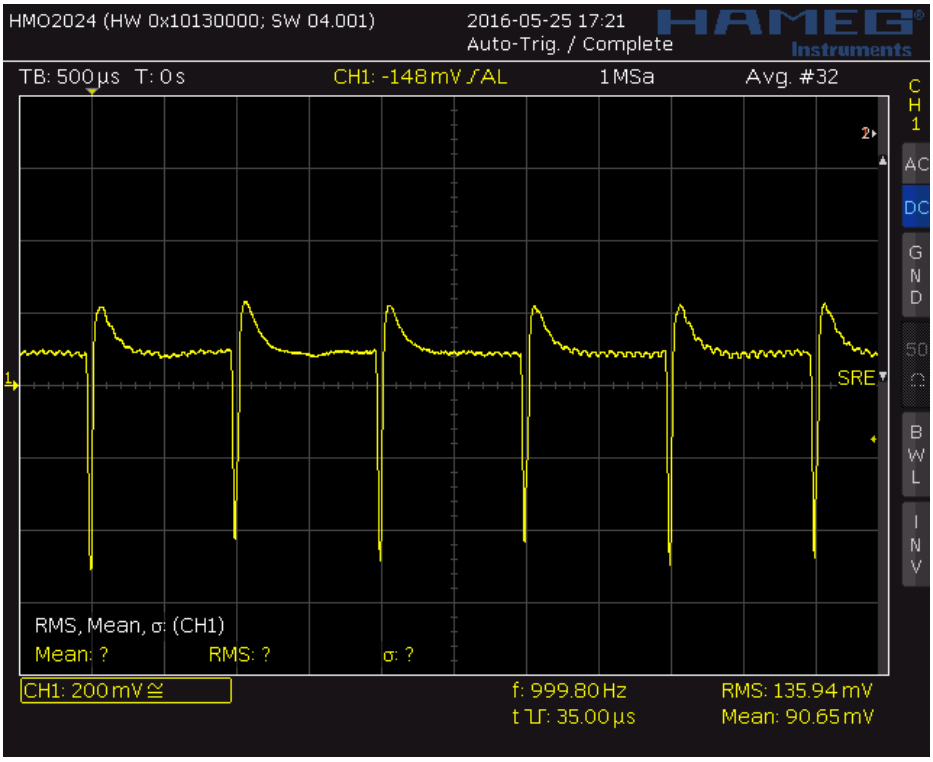

Figure B.2: Current consumption for accelerometers at 4kHz (1mv/100µA)

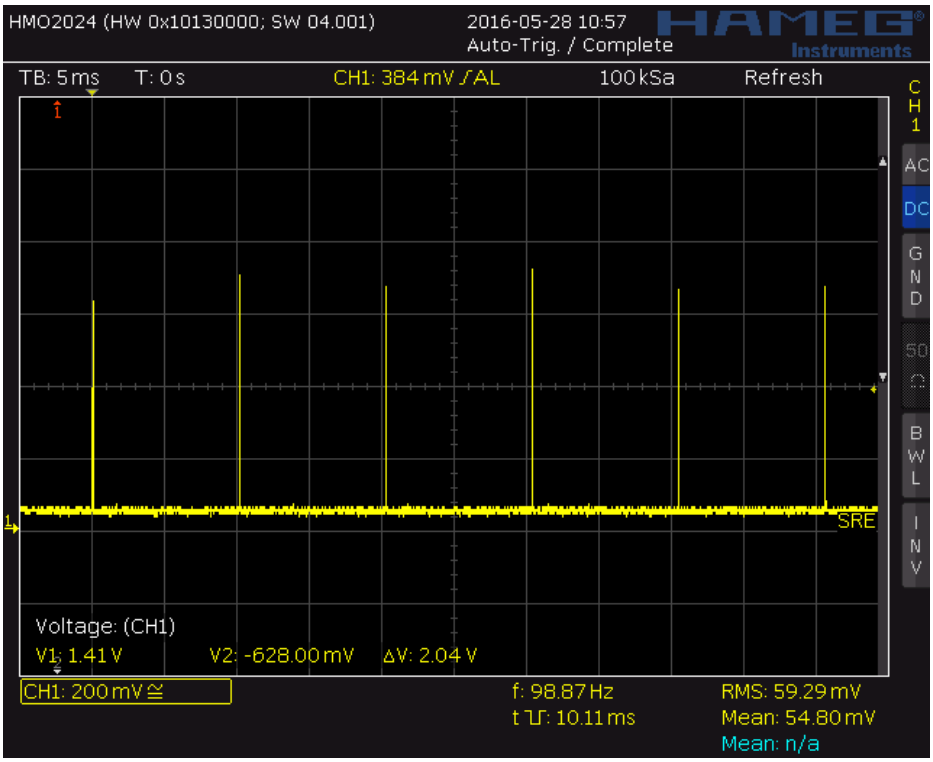

Figure B.3: Current consumption for nRF52 using the timer based approach (1mv/100µA)

B.3 LP-Mode at 125Hz

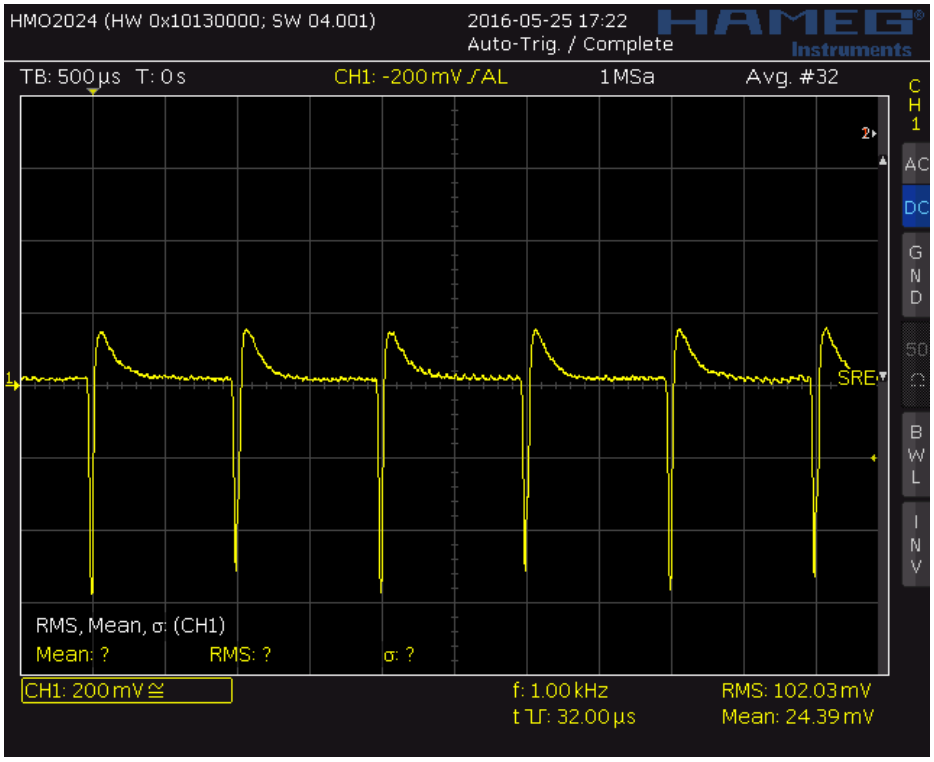

Figure B.4: Current consumption for accelerometers in LP-mode at 125Hz (1mv/100µA)

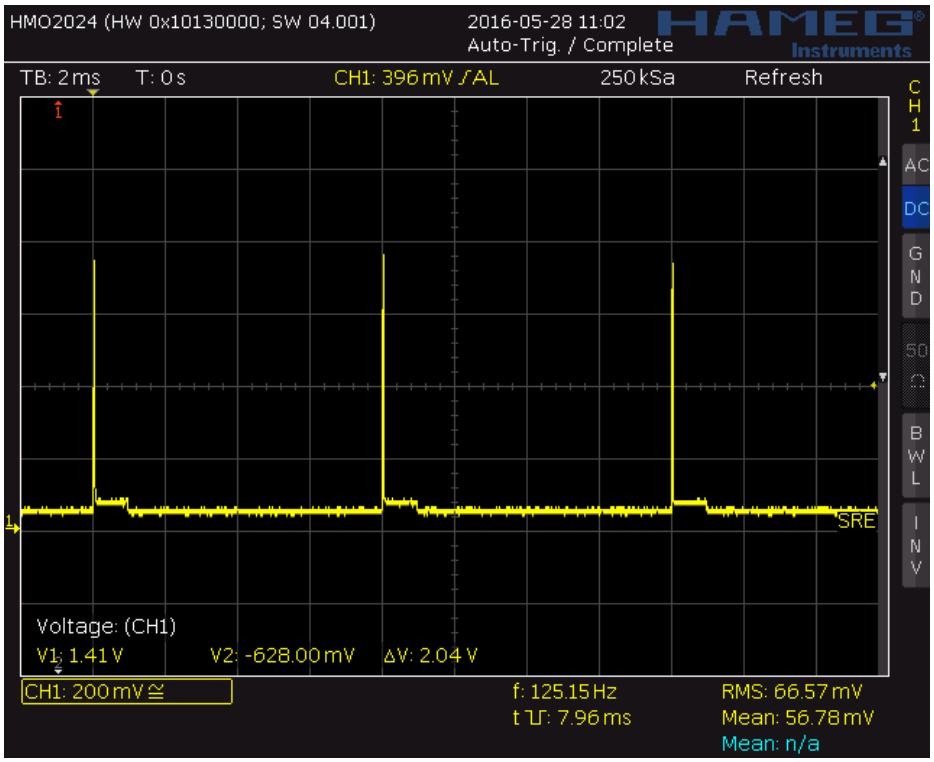

Figure B.5: Current consumption for nRF52 using the LP-mode at 125Hz (1mv/100µA)

B.4 LP-Mode at 62.5Hz

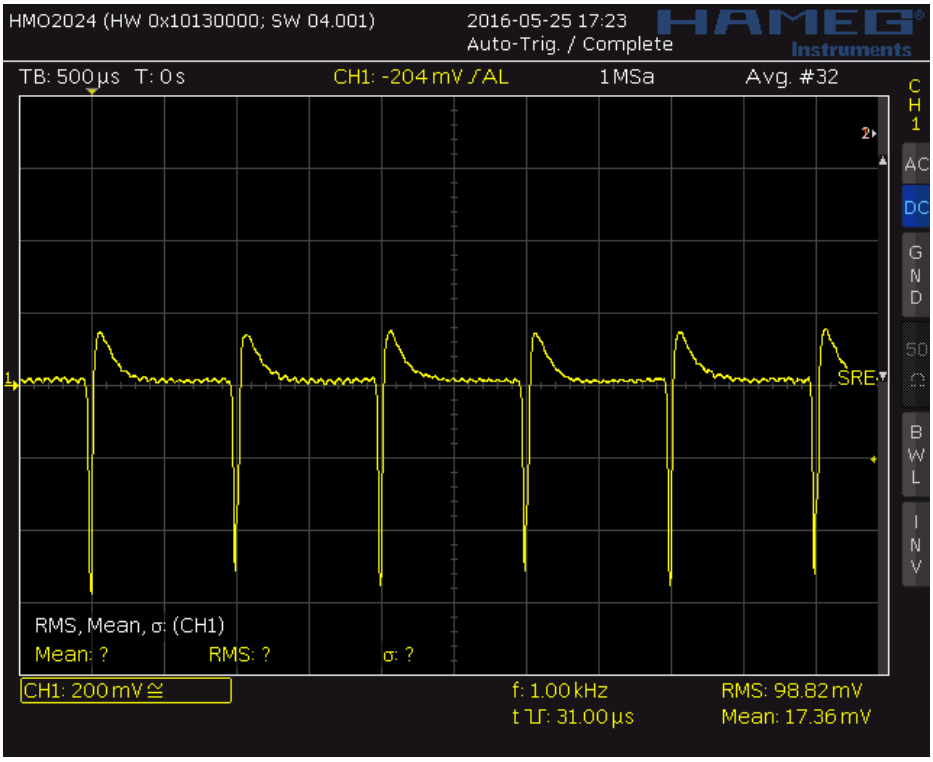

Figure B.6: Current consumption for accelerometers in LP-mode at 62.5Hz (1mv/100µA)

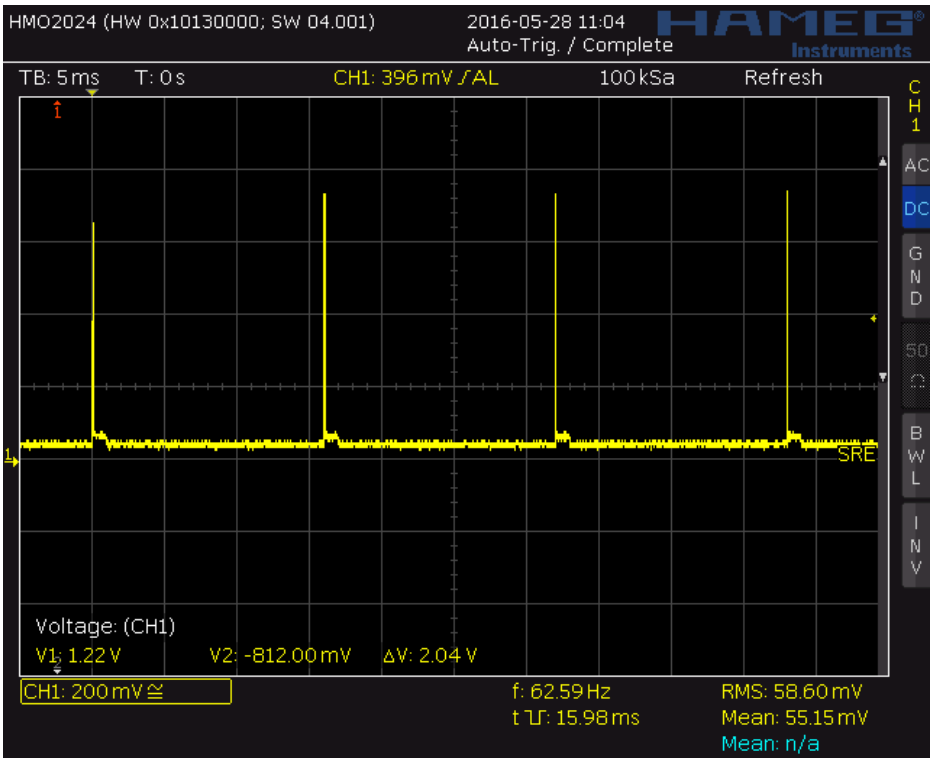

Figure B.7: Current consumption for nRF52 using the LP-mode at 62.5Hz (1mv/100µA)

B.5 LP-Mode at 31.25Hz

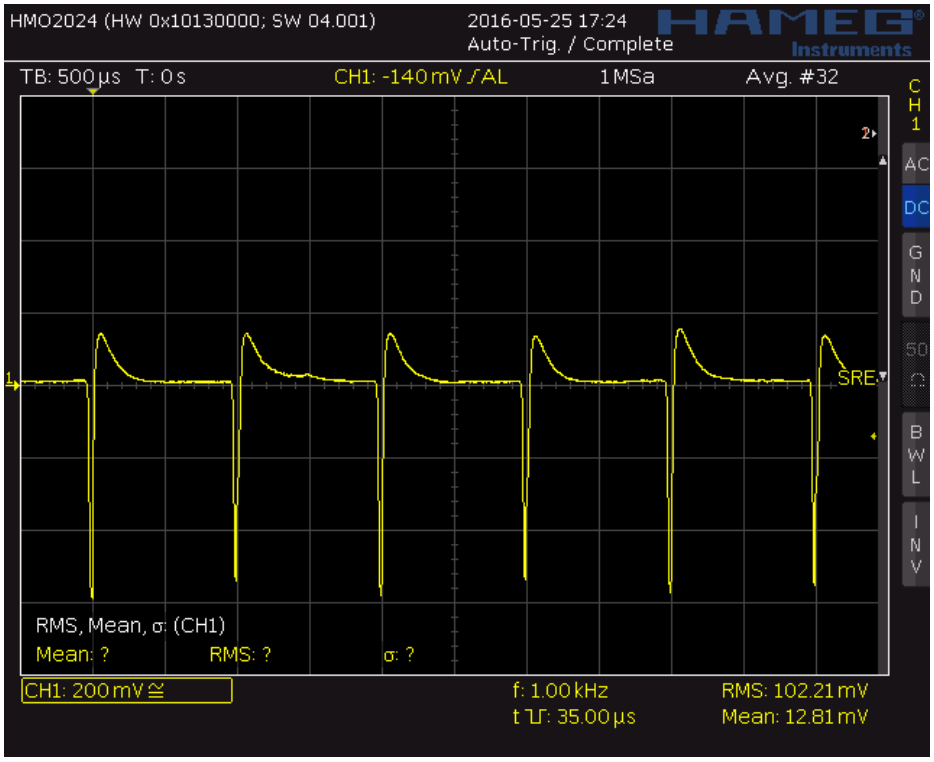

Figure B.8: Current consumption for accelerometers in LP-mode at 31.25Hz (1mv/100µA)

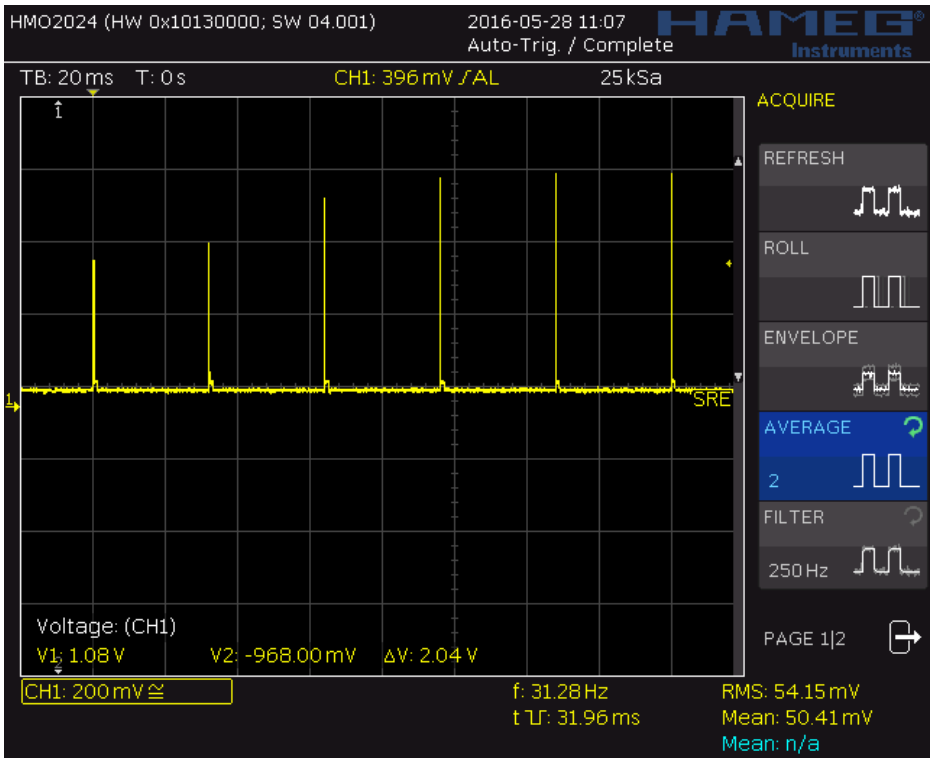

Figure B.9: Current consumption for the nRF52 using the LP-mode at 31.25Hz (1mv/100µA)

C Software and Firmware

All of the code created during this work can be found in the attached .zip file. The below sections gives a brief description of the file organization in this .zip file.

C.1 Matlab Simulation

A tree structure of the Matlab simulation files is presented below. This includes the simulation script, as well as the applied simulation data.

```
simulation/
├── gf_imu_simulation.m
└── Sensor Kinematics Pro data
    ├── mixed_motion_2_gyr.csv
    └── yaw_slow_to_fast_3_gyr.csv
```

C.2 Real-Time Matlab Processing

A tree structure of the Matlab scripts used for real-time data processing is presented below.

```
real-time matlab processing/
├── calculate_angular_velocity.m
└── real_time_plot_script.m
```

C.3 Embedded Firmware

A tree structure of the embedded C-files for the nRF52 is shown below. The Nordic SDK is not included in this tree structure. However, a guide on how to fuse the code with the Nordic SDK is provided in the readme.rtf file.

```
testbed firmware/
├── main.c
├── Drivers/
│   ├── i2cdev.c
│   ├── mpu6500.c
│   └── filter.c
└── readme.rtf
```