



JÖNKÖPING UNIVERSITY
School of Engineering

Accurate Fall Detection and Identification: Surveillance-Integrated Self-Reporting System

Main subject area: *Computer Engineering*
Author: *Jakob Freij and Oscar Lindberg*
Supervisor: *Andreas Axelsson*
JÖNKÖPING *Feb 2026*

This final thesis has been carried out at the School of Engineering at Jönköping University within Computer Engineering. The authors are responsible for the presented opinions, conclusions and results.

Examiner: Anders Carstensen
Supervisor: Andreas Axelsson
Scope: 15 hp
Date: 2025-05-19

Abstract

This is a report investigating fall detection using inertial measurement units. The report consists of a background and problem statement, the scope of the research, methodology, and theoretical framework. The proposed method consists of using a setup made up of a microcontroller and an IMU. We have determined a theoretical threshold, tested and adjusted it to verify the fall detection system's accuracy, allowing us to uncover the possibilities of threshold-based fall detection.

Contents

1	Introduction	5
1.1	Problem statement	5
1.2	Purpose and research questions	5
1.2.1	Purpose	5
1.2.2	Research questions	6
1.3	Scope and limitations	6
1.4	Disposition	7
2	Method	8
2.1	Theoretical method description	8
2.2	Data collection	8
2.2.1	Medical	8
2.2.2	Sensor Technologies	9
2.2.3	Microcontroller	9
2.2.4	Detection Method	9
2.3	Hardware Methods	10
2.4	Data analysis	10
2.4.1	IMU & Sensor Evaluation	10
2.4.2	Detection Analysis	11
2.5	Threshold	11
2.5.1	Gyroscope	12
2.5.2	Accelerometer	12
2.6	Software Methods	12
2.7	Validity and reliability	13
2.8	Considerations	13
3	Theoretical framework	14
3.1	Theoretical background of fall detection	14
3.1.1	Fall types	14
3.1.2	Sensor based fall detection	14
3.2	Classification outcomes	15
3.3	Hardware setup	15
3.3.1	ICM-20948	16
3.3.2	BMI160	16
3.3.3	LSM9DS1	16
3.3.4	MPU6050	16
3.3.5	Conclusion	16
3.4	Hardware implementation	17
3.4.1	Interface	17
3.4.2	Hardware	17
3.5	Software Implementation	18
3.5.1	Data Processing	19
3.6	Testing And Visualization	21
3.6.1	Testing	21

3.6.2	Visualization	23
4	Analysis	26
4.1	Hardware Setup Analysis	26
4.2	Sensor Performance Evaluation	26
4.3	Environmental and Practical Limitations	27
4.4	Data Discrepancy	27
4.5	Difficulties	27
5	Results	28
5.1	Results discussion	28
5.2	Method discussion	31
6	Conclusions and recommendations	32
6.1	Practical implications	32
6.2	Ethical and Legal Implications	33
6.3	Scientific implications	33
6.4	Further research	34
7	References	35

1 Introduction

1.1 Problem statement

Until recently, reports on isolated incidents involving elderly people have increased. Approximately 28-35% of elderly people fall at least once a year [5]. When these incidents occur, elderly people cannot self-report these incidents due to severe injuries and/or being immobile. Elderly individuals may suffer severe injuries and/or loss of consciousness from falling [2]. In many cases the delay of treatment for falls increases risk of complications over time and emotional trauma such as fear of falling again. Which could lead to reduced social and physical activity [29].

Traditional solutions like emergency buttons are often inadequate because they rely on the individual being conscious and mobile which may not always be the case [30]. Some additional downsides of using emergency buttons are that the user might not want to wear them because of the appearance or discomfort. This is a big problem worldwide, as falls have shown to be the second most common cause of fatal injuries among the global elderly population [1]. A good and reliable way to detect falls could be integrated into a reporting system and make sure that the patient receives help as soon as possible. Combining sensor technology and a microcontroller to effectively determine that a fall has happened.

These sensors can monitor body movement through analysis of acceleration and angular velocity data. Despite the rise of technical solutions nowadays, many fall detection systems suffer from issues related to accuracy, false positives, limited mobility, and hardware implementations. This study will go in depth about how we will analyze the sensor technology to detect falls accurately and then evaluate the methods in order to detect falls. The ultimate goal is to contribute to the development of a safer living for individuals who may suffer the most when a fall incident becomes unreported.

1.2 Purpose and research questions

1.2.1 Purpose

As the elderly population continues to grow, it is our duty to protect them using the resources available. Therefore, utilizing technology to enhance the self-reporting system is essential. Sensor technology has been an important aspect for FD (fall detection) [10]. The goal is to research and explore how sensor surveillance can improve incident reporting. This includes identifying effective ways to use sensor technology while also analyzing their limitations in terms of accuracy related to falls. The purpose is to find the more accurate sensors and compare them to each other, considering both the advantages and disadvantages of each sensor technology.

Additionally, by analyzing falls, we can detect and categorize different type of falls. Different types of falls might also lead to different types of injuries [9]. Our vision is to effectively identify these falls using sensor technology to provide more detailed reports. This would lead to a more informative report in the end node of the reporting chain.

By detecting a fall we can obtain information regarding changes in sensor data that could lead to the possibility of an injury.

We have opted to use an Inertial Measurement Unit (IMU), which is a sensor module that can measure acceleration and orientation using embedded gyroscopes and accelerometers. An IMU combined with a compact, low-power microcontroller will hopefully be a satisfactory solution to detect when a person is falling.

Wearable device-based fall detection is the second least researched fall detection method, with only about 4% of published research papers being on this method [31]. This shows that there is probably more knowledge to be uncovered in this field, and it is a reason to make some additional research on it.

We will do a comparison of the different available techniques that could be used for fall detection. And rate the efficiency of our deemed technique compared to the others.

1.2.2 Research questions

RQ1: How accurate is the identification of falls using an Inertial Measurement Unit?

This research question covers how effective this technology would be at solving the issues we have stated. It also looks at what the limitations and challenges are, and how we can use this technology to acquire the most accurate data.

RQ2: What fall detection methods are possible to use and what are their limitations?

After obtaining the data, we will also need a way to process it and determine a fall event. This research question will make us explore the different possibilities and venture into their pros and cons. Eventually, we will decide on one that fits our application and discover its possibilities through testing and evaluation.

1.3 Scope and limitations

The research aims to discover the overall possibilities to accurately identify and report different type of falls. This includes researching and experimenting with different types of technologies, in different type of settings. To acquire the necessary knowledge to identify falls, we have tied engineering knowledge with abstract medical knowledge to effectively communicate between professions. Since

this research is about using sensor technology to detect falls, we have not been researching deeply into the medical aspect of the study. Yet we have strived to understand surface-level medical terms to effectively communicate.

The goal of this research is finding a FD method that is relatively simple, compact, and has a low power consumption, and evaluating its effectiveness. We will not go out of our way to find the most effective method for detecting falls. We do have in mind that the technology we are researching should have a real world use case. For example, it should be possible to integrate the hardware into a wearable device, and when a fall is detected, it should be possible to send a report (hence, our focus on low power consumption and compactness). However, we will not implement this kind of functionality ourselves.

1.4 Disposition

This chapter provides an overview of the structure of the report. The report is organized as follows:

1. *Introduction*
The first chapter describes the problem we are aiming to solve. It also includes purpose, research questions as well as the scope of what is to be included in the research.
2. *Method and implementation*
This section describes the methodology that is to be used for this research. It covers data collection and analysis techniques and discusses the choice of hardware and software. It includes a subsection about validity and reliability.
3. *Theoretical framework*
This section discusses in detail how our hardware and software setups are implemented. It also explains our choice of components and overall architecture.
4. *Analysis*
This section we'll analyze and evaluate our hardware setup, practical methods, and discrepancy in data performance.
5. *Results*
This section reveals the results regarding our work, It also includes discussions on the results and the methods that were used.
6. *Conclusion and recommendations*
In this section we'll uncover the practical and scientific implications of our work, and the further research that could be done to extend the the knowledge on the thesis subject.
7. *References*
The references section includes all references used in the report.

2 Method

2.1 Theoretical method description

The methodological approach for this thesis follows a structured, quantitative research process designed to evaluate the feasibility of using IMU-based threshold methods for fall detection. The work combines theoretical analysis, hardware selection, experimentation, evaluation. In broad terms, the method consists of four overarching steps:

1. **Theoretical review**

A literature review was conducted to identify relevant fall properties, biomechanical metrics, sensor technologies, and previously suggested thresholds for acceleration and angular velocity. This established a theoretical foundation for selecting appropriate hardware and estimating an initial fall-detection threshold.

2. **System design**

Based on the theoretical review, a prototype system was constructed consisting of a microcontroller and an IMU. The purpose of this stage was to make a proof-of-concept, not to build a finalized wearable device.

3. **Experimental data collection**

A quantitative research method was applied where fall and non-fall events were performed in a controlled environment using a realistic humanoid dummy. Each test event produced time-series sensor data that could later be classified into true positives, false positives, true negatives, and false negatives. This process generated the dataset used for threshold evaluation.

4. **Data analysis and threshold evaluation**

The theoretical thresholds identified in the literature review were compared against measured data, and alternative thresholds were tested to balance sensitivity and specificity. The analysis focused on determining how well a simple threshold-based method can distinguish between fall and non-fall events.

2.2 Data collection

2.2.1 Medical

The data collection of the research work include acquiring knowledge about used technology in the industry and the medical implementations our work could have. This includes finding the different categories of falls to be able to determine possible injuries, which could help sending a more informative report. Slips and trips are most common causes of falls among the elderly. These types of falls has all the different orientation outcomes. The different directional orientations are forward, backwards and lateral (sideways) [9].

Studies have shown that falls can be classified by different metrics. When measuring a high trunk movement by degrees per second ($^{\circ}/s$), studies have shown that ADL's (activities of daily living) for e.g sitting down and lying down, the biomechanical metrics such as degrees per second seem to differentiate to a slip-induced backwards fall. Where ADL activities peaked at peak average $84.1^{\circ}/s$ compared to the studied falls of $139.7^{\circ}/s$. [25] Other studies have shown different outcomes. Nyan [27] Observed high-trunk falling movements, the peak velocity was approximately $450^{\circ}/s$. which is a vast difference between the two studies. Such discrepancy in angular velocity could be due to different factors. Like environmental, angular momentum, and placement of the sensor. This study shows the potential to detect falls using metrics that depend on body movement.

2.2.2 Sensor Technologies

Discovering the potential technologies we could use was our top priority. The main method we used to uncover the more accurate sensor was quantitative. Therefore, by using potential technologies already used in the industry regarding object and accelerator tracking, we unveiled the sensor with the more accurate result. [12] States how important it is to select and manage different sensors. IMUs (Inertial measurement units) are widely used in the medical-engineering sector. An IMU is a device used to measure the acceleration and angular velocity of an object. Another option was using acoustic sensors [4]. These sensors can detect sounds of impacts. FD can also be done with the help of body part tracking using a depth camera [6]. With consistent camera setups, surveillance is always active. PIR (Passive infrared) sensor is another sensor technology used for human detection among many surveillance systems and has potential for FD [13].

2.2.3 Microcontroller

The system we have developed involves networks. When testing, it was appropriate to use technology that was able to communicate over a network. However, our main purpose was not to involve ourselves in the reporting over a network. Rather, our main objective was to detect falls and determine whether a person had fallen. Nonetheless, we needed to use appropriate hardware that was compact and capable of network communication.

These devices are usually called IoT (Internet of Things) devices. There are many development boards that are applicable for IoT development. The ESP32 is a fitting example for IoT systems [14]. Other boards such as the STM32 have a wide variety of boards to select from with network capabilities and they are well used in industry today.

2.2.4 Detection Method

These are the type of technologies used to collect data. There are different approaches to analyze readings to determine whether or not a fall has occurred.

These are a 1) threshold-based, 2) conventional machine learning-based, and 3) deep learning-based [16].

2.3 Hardware Methods

Our components were assembled on a smaller breadboard, even though we had other options such as designing and printing a PCB with an ESP32 as the SoC or using a perfboard to solder our components. However, we chose the simplicity of the breadboard so that we could spend more time evaluating test falls and determining a threshold, rather than optimizing the compactness of the system, something that isn't particularly relevant to the research aspect on FD.

We were also concerned about whether the test dummy was going to fall onto the system, resulting in the system becoming pressured and destroyed. Therefore we mounted the system in a more consistent and safe way to avoid collisions.

For the non-falls that were tested, the system was simply held the same way it was mounted onto the dummy simply going from a standing position to a sitting position causing light knee flexion. And also testing a free fall down onto the sitting position causing variety in speed and positioning in non-fall data.

2.4 Data analysis

2.4.1 IMU & Sensor Evaluation

We have found the different technologies used in FD have different advantages and disadvantages. Acoustic sensors can be used to detect falls [4]. A drawback of using an acoustic system to detect falls is that it is difficult to obtain realistic fall sounds. Therefore it can be challenging to train and test the system. The sensors ability to detect fall sounds can also be obstructed by background noise. FD can also be done with the help of body part tracking using a depth camera [6]. The problem with this however, is that it can be very invasive to set up cameras in people's homes and similar areas.

Camera surveillance is rather inconsistent, also due to how they are placed. They also have less accuracy than other technologies. PIR sensors undergo the same fate as camera technologies. They are advantageous for detecting humans but disadvantageous at identifying human behavior [13].

This left us with IMU sensors left to be reviewed. Since the device is wearable, we could be more consistent at tracking events compared to PIR and camera technologies, as we could follow the body and measure at the edge.

An IMU is made up of an accelerometer and a gyro sensor. The accelerometer tracks sudden movements and changes in acceleration, therefore tracking the sudden changes in body positioning. The gyro, on the other hand, tracks the positioning of the system-for example, comparing the differences between lying down and standing up. The IMU can draw conclusions regarding whether a fall has happened or not.

2.4.2 Detection Analysis

After doing a theoretical analysis of different possible IMUs, we decided on one that we thought was superior according to some set metrics.

In the event of an emergency, the system will conclude an FD based on IMU sensor readings.

1. **A: threshold-based**

Threshold-based is a method used to define fall events. It is not computationally intensive, very dynamic and it has a fast response time. It would analyze the sensor readings and if it passes over the pre-determined threshold, the system would self-report in the event of an emergency. Using training data, a predetermined threshold would be implemented with the aim of accurately filter events. [16] While this method is not complex compared to other methods it also has disadvantages in terms of accuracy. Due to it having a certain predetermined threshold, many false alarms or unreported incidents may occur depending on how you set the threshold. Adjustments and experiments must be done before establishing a threshold. Sensitivity and specificity values may be still be affected negatively.

2. **B: conventional machine learning-based**

Conventional machine learning-based utilizes supervised learning therefore trains on data to categorize data received. Upon suspicion an event triggers that will analyze the sensor readings in a heuristic function determining the potential event. The disadvantage with this technique is the slow response time. And that it is computationally intensive which makes it unreliable to run at the edge which would not suit a small wearable device.

3. **C: deep learning-based** deep learning-based explores the potential of deep neural networks to detect falls. [17] Introduced a deep learning method called CMFALL. Which is a complete FD system built from ground up using deep learning. This technique shows the best accuracy rate [16]. However, it is computationally intensive compared to a threshold-based approach, but it is still capable of running on certain MCU's (microcontrollers). The downside is that such a system needs large training sets, which require a lot of memory and a long training time on those data sets.

2.5 Threshold

This study aims to investigate further into the possibilities of using a threshold to determine a fall. The biggest reason being that the subject is not researched enough compared to the other techniques. It also has many developing benefits, such as simplicity, adaptability, and fast response time. While it has its downsides we strive to complete and evaluate a threshold-based FD algorithm

to acquire the most accurate results as possible. There are many ways to set up thresholds. We will primarily be looking at the gyroscope and accelerometer values and if they reach a certain threshold - the system will simply categorize a fall. The threshold will be set according to past studies and the data that has shown to categorize a fall. This way we can theoretically determine a threshold and therefore also test it out practically resulting in determining a threshold based on theory but also make it fit the real world.

2.5.1 Gyroscope

Like mentioned, passed studies have shown that the high-trunk in GLF (ground-level fall) reaches an angular velocity high of $139.7^\circ/\text{s}$. While other ADL reached a high of 84.1 [25]. We also observed a study on which the angular velocity reached a high of $450^\circ/\text{s}$. In theory regarding the threshold we can never really be secure about our choice of it. Which highly suggest the importance in practical testing. Considering the difference in values over a period of time we can extract the substantial temporal change over a set period. If we consider a threshold to not include ADL, according to [25]. We will need to effectively filter out events that are less than $84.1^\circ/\text{s}$. Since values are returned as a signed integer meaning the values can be extracted the positive limit down to the negative limit in each axis. Meaning the difference in angular velocity exceeds the polarities. That is, the effective threshold to filter out ADL would be:

$$168.2^\circ/\text{s} = 84.1^\circ/\text{s} * 2$$

This is our estimated gyroscope threshold based on theory.

2.5.2 Accelerometer

Likewise for the accelerometer, we are in need of a set threshold to effectively categorize falls. [27] suggest that past studies have shown to prove a falling speed of a range of $4\text{--}11\text{ m/s}^2$ and that backwards fall has an increased speed of 25 percent compared to others. With that data in mind, the estimated threshold difference factor would be 8m/s^2 . With backwards fall to be 10m/s^2 with an increase in speed of 25 percent.

2.6 Software Methods

To capture a fall using an IMU we must process the data that we receive. We will use Espressif's own extension in Visual Studio Code (ESP-IDF) [32], which will simplify the development process since it is made by the same creator. Here, we will also use the programming language C, since it is what ESP-IDF is built upon and it is often used in other fall detection systems [33]. We will develop our system by interacting with the sensor registers.

The software will be built to interact with the IMU. There are two ways the ESP32 can interact with the IMU: reading and writing data. We will write our

software based on the foundations of the serial protocols. We will separate our software into interface and implementation. Which is a typical setup often used in C programming, that contains header and source files [34].

We will also need to process the data acquired and represent it to the threshold determining algorithm. Since this research focuses on threshold-based fall detection, the determining algorithm is not complex. We will use a simple logic statement that monitors two values, the gyro value and the accelerometer value.

To verify and judge the data manually, we will also need a way to read the data ourselves after a simulated fall. Therefore we will need to print out the values for representation, and this will be done using Python. It is simple to use for user interaction and is a powerful language used for simplicity and tests [35].

2.7 Validity and reliability

The aim of the thesis project was to correctly identify different types of falls according to data. Having acknowledged the possible outcomes was extremely crucial. We had to consider the potential miscalculations and the setbacks of the project. Identifying the issues we may encounter and make choices accordingly. In high accelerative activity false alarms may happen. We might have had to consider other activities and work our way around those.

By exploring the different ways one could fall, we could also in the process have identified other periods of accelerative events that may have led to false alarms. E.g if a person involved themselves in a certain behavior that triggered a front fall report. We may even have been able to identify the false alarm behavior more clearly. Detection of falls on its own without directional analysis may not be possible to research.

In the aspects of reliability, we needed to ensure consistent training data, and acquire a clear understanding of the human body's motion when suffering a fall. Ensure that the IMU operates under the same condition to not collect data which is categorized faulty, such as the position of the IMU.

2.8 Considerations

Hence, using body measurements of a dummy, we have certain things to take into account regarding our data. There are many ways to alter or miscalculate test results. Due to the fact that we are using a dummy instead of real people, we need to create the most realistic falling scenarios possible to accurately test a real fall.

Another important point to consider regarding the test data is the risk of overfitting. We may acquire high accuracy within our dataset by developing specifically around those tests. This can lead to poor performance when the system

is exposed to new, unseen data, due to overfitting. To avoid this, we need to gain knowledge about biomechanical metrics related to the body’s movement during a fall and form thresholds based on these values, rather than tailoring the system to the specific test data we have collected.

We need to perform tests with different types of falls and outcomes, such as varying directions of fall, different degrees of impact, and landing positions. Other considerations regarding the data include the fact that the falls are replicated and tested on a dummy. Although it is a realistic human dummy, it can still not fully replicate all aspects of a real human fall, and data may have been lost due to untested mispredictions.

3 Theoretical framework

3.1 Theoretical background of fall detection

Fall detection is a well-studied area within medical engineering, and the theoretical basis is important to understand how sensor-based systems can distinguish fall events from normal activities. A fall is commonly defined as an unintentional event where a person ends up on a lower level than intended, often due to slips, trips, or balance loss [3]. Among elderly individuals, falls are one of the most frequent and dangerous types of injuries, making reliable detection methods highly relevant [5][29].

3.1.1 Fall types

Previous research shows that falls can be categorized based on their direction, cause, and body orientation. Common fall types include forward falls, backward falls, and sideways falls [9]. These categories differ in trunk rotation, acceleration patterns, and impact profiles. Slip-induced backward falls, for example, usually involve a rapid increase in angular velocity, while forward and side falls often show higher linear acceleration values during impact.

Several biomechanical studies have quantified these movements. Liu & Lockhart [25] recorded trunk angular velocity during backward falls and found average peak values around 139.7 dps, which differ significantly from activities of daily living such as sitting or lying down, which peaked at around 84.1 dps. Other studies have reported even higher angular velocity up to 450 dps during rapid fall events [27].

3.1.2 Sensor based fall detection

A variety of sensor technologies have been explored for fall detection. Acoustic sensors can recognize impact sounds but suffer from high noise sensitivity and difficulty collecting realistic fall sounds [4]. Depth cameras and vision-based systems can track body posture and motion [6], but raise privacy concerns.

PIR motion sensors can detect human presence but cannot classify behavior or distinguish fall events [13].

Wearable sensors combining accelerometers and gyroscopes are among the most promising technologies because they allow direct measurement of body motion in real time and operate independently of the environment [12][31]. IMUs capture:

- Linear acceleration
- Angular velocity
- Motion patterns

3.2 Classification outcomes

A quantitative method for the research was used, where we examined the data we obtained from a large number of test falls.

Either a fall is detected or not. These were divided into four sub cases [15]:

1. True positive (TP) - a fall is correctly identified.
2. False positive (FP) - the device detects a fall that did not happen.
3. True negative (TN) - no fall occurred and no fall was detected.
4. False negative (FN) - a fall occurred but was not identified by the device.

The following two metrics will be evaluated:

- Sensitivity - the devices capacity to detect a fall.

$$\frac{TP}{TP + FN}$$

- Specificity - the devices capacity to *not* detect a fall when there is not one.

$$\frac{TN}{TN + FP}$$

3.3 Hardware setup

The hardware setup features an ESP32 [21] microcontroller. The ESP32 was chosen because it is small and compact. This would potentially make it more user friendly and allow us to embed it inside of a compact device. It also has integrated Wi-Fi and Bluetooth capabilities. This would in theory allow a smooth way to implement a self-reporting system that could send an alert if a fall was detected. However, a self-reporting system is outside of our scope, but may be viable for future research.

The properties we need in an IMU are the following:

- Low power consumption
- Accelerometer and gyroscope
- Small size/compactness

The IMUs that were considered are ICM-20948, BMI160 [19], LSM9DS1 [20], and MPU-6050 [23].

3.3.1 ICM-20948

A valuable of the ICM-20948 is its low noise floor, which means that it can detect movement more precisely. This is a valuable property for fall detection, as a slight change in acceleration patterns could push its output values over the threshold of what counts as a fall.

3.3.2 BMI160

The BMI160 has a 3-axis accelerometer and a 3-axis gyroscope. It has an extremely small power consumption which makes it suitable for wearable devices. It is a small piece of hardware, which would make it suitable for use inside of a bracelet for example.

3.3.3 LSM9DS1

The LSM9DS1 also includes a 3-axis magnetometer in addition to a 3-axis accelerometer and a 3-axis gyroscope. The magnetometer gives us the possibility to measure absolute orientation which allows us to better detect rotational falls. However, the magnetometer has a low data output of 80 hz which might not be ideal for fast motion tracking.

3.3.4 MPU6050

The MPU6050 was excluded early in our evaluation. The only benefits of it are that it is affordable and widely supported. It has a power consumption of 500 μ A which is the highest among the candidates, and it is also the largest of them. It contains outdated sensors with inferior noise performance and accuracy than its competitors.

3.3.5 Conclusion

Specification	ICM-20948	BMI160	LSM9DS1
Power Consumption	8 μ A	18 μ A	450 μ A
Components	Accel., Gyro., Magnetometer	Accel., Gyro.	Accel., Gyro.
Size	3x3x1 mm	3x2.5x0.8 mm	3.5x3x1 mm

Table 1: Comparison of IMUs for Fall Detection

Our decision landed on the BMI160. It stood out as the best all-round choice due to its balance between power consumption, performance, and size. Its power consumption of $18\mu\text{A}$ is far superior to the LSM9DS1's $450\mu\text{A}$, and not far behind the ICM-20948's $8\mu\text{A}$. It is the smallest of the three which makes it easier to seamlessly integrate into a wearable device. It does not feature a magnetometer, however. We have come to the conclusion that a magnetometer will not be able to contribute much to the fall detection system, and we value other aspects over it.

3.4 Hardware implementation

3.4.1 Interface

The BMI160 comes with two different serial interfaces: SPI and I2C. When choosing a serial protocol, we considered a key factor. Which was hardware space. While I2C only requires 2 wires compared to SPI's requirement of at least 4. I2C was the obvious superior.

3.4.2 Hardware

After we have determined MCU, IMU and serial interface. It was time to assemble the circuit (Fig 1). Where the V_{in} , Ground, SDA (I2C serial data) and SCL (I2C serial clock) connect to each component respectively

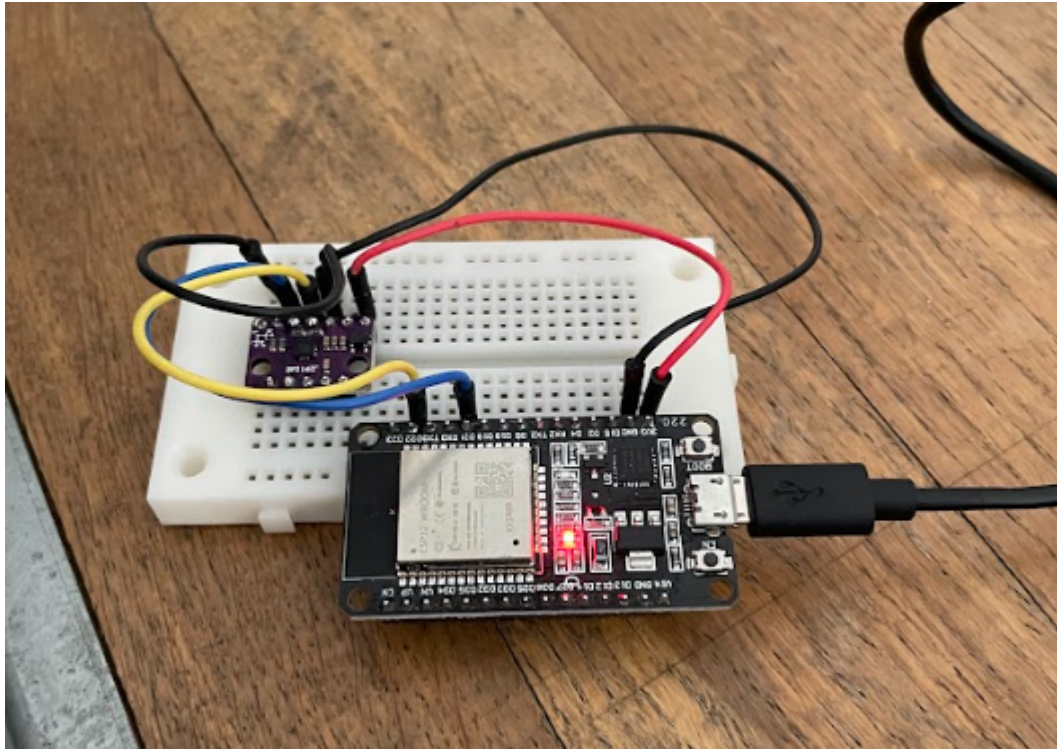


Figure 1: Breadboard wiring

3.5 Software Implementation

We did not only need a hardware setup, but also a software implementation that goes appropriately along with the hardware. The implementation can be broken down into different steps:

1. Writing to the IMU
2. Reading from the IMU
3. Processing sensor data
4. Threshold check for falls
5. Data visualization

The IMU operates by communicating over a serial protocol, and in our case it is over I2C. Both writing and reading to the IMU involve data registers, which can be found in the BMI160 datasheet [19]. ESP-IDF has a complete I2C peripheral API [36], which we use to both write and read to interact with the BMI160.

There are different registers within the BMI160, and they store different types of information. First, in the software, we initialize the BMI160 by powering it on using a specified register. We also specify our settings regarding the accelerometer and gyro values, such as which sensitivity and g-range we want to acquire. These are covered in the data processing chapter. A header file was developed to keep track of all the registers that the BMI160 has, along with the function definitions for the communication used when writing to and reading from the IMU. A source file was also created where the implementations of these functions were done. They worked as follows:

- **BMI160_Init** - Executes during initialization of the fall detection system, ensuring power-on settings and sensor configurations.
- **BMI160_WriteRegister** - Writes data to a specific register address on the BMI160 sensor.
- **BMI160_ReadRegister** - Reads data from a specific register address on the BMI160 into a buffer.
- **BMI160_ReadAccel** - Reads accelerometer data from a selected axis.
- **BMI160_ReadGyro** - Reads gyroscope data from a selected axis.

Now that the fundamental functions were implemented for the system regarding system initializing and data retrieval. We call these function in our main loop.

The function `detector_task` is our main loop. This function manages the continuous operation of the fall detection system. It begins by initializing the I2C bus and creating a device handle for the BMI160, followed by sensor initialization. The function then enters an infinite loop where accelerometer and gyroscope values are read from all axes and converted into real physical units. These processed values are printed for evaluation during testing, and a short delay is applied before repeating the cycle.

By reading the printed values with a Python program called `ser`, which is short for serial, we are able to format the readings in a more readable manner, where the sensor logs have timestamps included. This file also converts the values into a file with a CSV format. These CSV files were evaluated and analyzed using matplotlib to create graphs that visually represented the test falls. All the code could be found on the fall detection GitHub repository. [37]

3.5.1 Data Processing

The BMI160 provides raw accelerometer and gyroscope readings as 16-bit signed integers through its ADC. To interpret these readings, the raw values must be converted into physical units according to the sensitivity parameters defined in the BMI160 datasheet [19].

After reading the accelerometer registers of every axis on the BMI160 with ADC, we receive a 16-bit signed integer. Then we set this value to be in a range of $\pm 2g$. Which is a higher precision compared to other ranges. The accelerometer is calculated as follows:

$$a_{\text{real}} = \frac{\text{Raw Value}}{\text{Sensitivity (LSB/g)}}$$

The sensitivity is calculated as the 16-bit integer range (32768) divided by the set range ($\pm 2g$), which returns acceleration values between $\pm 2g$.

$$\text{Sensitivity} = \frac{32768 \text{ LSB}}{2g} = 16384 \text{ LSB/g}$$

For a complete analysis, we also calculate the SI units (m/s^2) value:

$$a_{\text{m/s}^2} = a_{\text{real}} \times 9.81 \text{ m/s}^2$$

The use of $g \rightarrow \text{m/s}^2$ scaling follows standard inertial sensor practice as described by Zhou et al. [12] and Nyan et al. [27].

The BMI160 gyroscope values from the ADC are also stored in 16 bit signed integer values for angular velocity for each axis as followed:

- **Roll** (ω_x): Rotation around the X-axis
- **Pitch** (ω_y): Rotation around the Y-axis
- **Yaw** (ω_z): Rotation around the Z-axis

The sensitivity is determined by the precision required to track the human body's angular movements. The BMI160 gyroscope has a default range of $\pm 250^\circ/\text{s}$, which is commonly used in IMUs. This range offers a resolution of $0.0076^\circ/\text{s}$, which is higher than that of wider ranges such as $\pm 500^\circ/\text{s}$ or $\pm 1000^\circ/\text{s}$. Although narrower ranges like $\pm 125^\circ/\text{s}$ appear more precise, they may result in data loss. This is because slip-induced falls have been observed to peak at an average of $139.7^\circ/\text{s}$ [25], potentially exceeding the $\pm 125^\circ/\text{s}$ range and causing values to clip. The sensitivity setting of 131 is used to convert raw sensor data into degrees per second. This is based on the fact that 131 multiplied by the resolution (0.00763) equals approximately $1.0^\circ/\text{s}$. Therefore, angular velocity is calculated using this sensitivity value. So, the real $^\circ/\text{s}$ value is calculated as follows:

$$a_{\text{real}} = \frac{\text{raw_value}}{\text{Sensitivity}}$$

3.6 Testing And Visualization

3.6.1 Testing

Initially, we performed 30 test falls. The tests would be performed by a human replicate dummy and we placed the hardware on its high-trunk. The threshold values were then evaluated on these data sets.

The dummy is a realistic human dummy that we were fortunate enough to borrow to collect falling data (Fig 2). It is roughly 170cm tall and 70kg in weight, so it perfectly replicates the human body movement during a fall. The dummy was held up and dropped in a simulated way. we strived to replicate a real scenario fall.



Figure 2: Dummy

The system was mounted along the high-trunk of the replicates body. Ensuring the system stays intact using silver tape and soft plastics around it (Fig 3).



Figure 3: Mounting

3.6.2 Visualization

Upon our choice of hardware, we will evaluate our results and adjust accordingly. With the help of Python libraries, we can ensure graphical visualization of the fall, therefore, obtain an idea of how the fall has occurred. By creating a virtual space and a rectangular block representing the falling body that visualizes the sensor readings when they are altered, we can deepen our understanding of how sensor reading is interpreted in the digital world (Fig 4). Now we have two representations of our fall and if they are equivalent we can ensure that the real world fall is interpreted correctly by the digital threshold. Since it is a Python library, it will be easy to interpret and use during our research, and it will not cause unnecessary time spent solely on visualization due to the low entry barrier of Python.

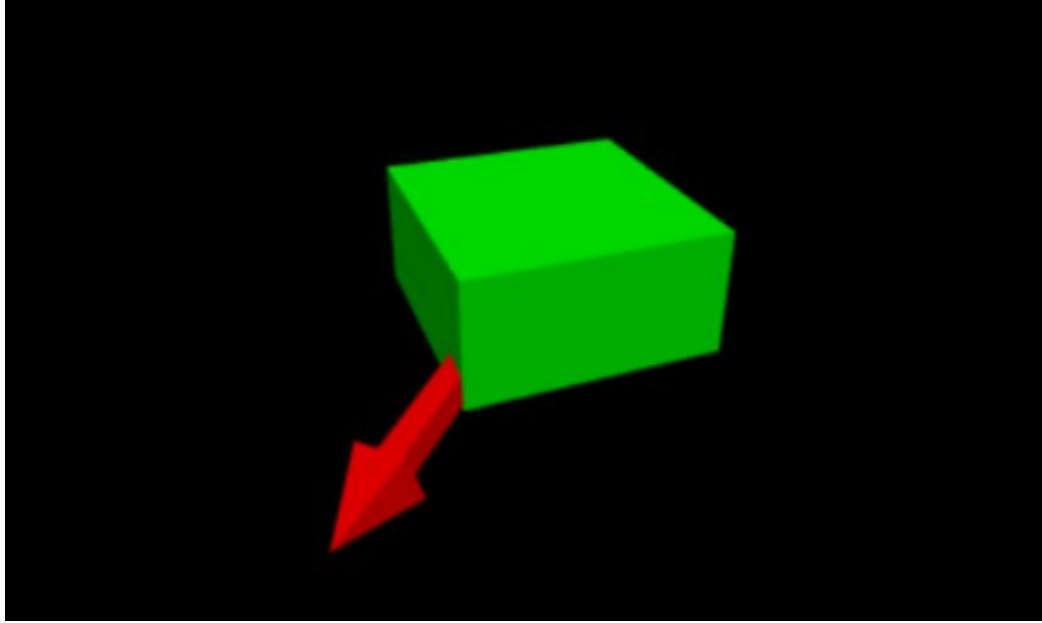


Figure 4: Visualization

We obtained data for roll, pitch and yaw from the IMU. These values correspond to the x -, y -, and z -axes. The following matrices are then constructed:

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$R_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In this example, ϕ is the roll angle, θ is the pitch angle, and ψ is the yaw angle. To obtain the final rotation matrix R , we multiply the matrices in the order:

$$R = R_z(\psi)R_y(\theta)R_x(\phi)$$

Given a vector $\mathbf{v} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$, we rotate it by applying:

$$\mathbf{v}_{\text{rotated}} = R\mathbf{v}$$

The acceleration vector is:

$$\mathbf{a} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}$$

where a_x, a_y, a_z represent the acceleration in the respective coordinate directions.

The total acceleration is calculated as the Euclidean length of the acceleration vector:

$$|\mathbf{a}| = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

This shows the overall amount of acceleration regardless of direction.

These calculations are incorporated into a python script that visualizes the rotation of the IMU in real time. The purpose of this kind of visualization is partly to demonstrate the accuracy of the system. Seeing a digital representation of the IMU accurately mimicking the IMUs real world movements is a convincing demonstration of the system's accuracy.

Vpython [26] was chosen for the visualization as it is a lightweight framework that is easy to set up and modify. This approach focuses on simplicity and performance and therefore uses a very bare bones design.

4 Analysis

4.1 Hardware Setup Analysis

After conducting our test we can confirm that the hardware in terms of space and performance was efficient enough to collect data from our dummy test falls. We wanted to venture into the different ways the dummy could fall, as the theoretical analysis described, to avoid overfitting the algorithm. Overfitting means specifically tailoring the system to work accurately with the tests that have been done. This means that new data sets are handled by a system that is engineered for a specific test. This will result in an inaccurate system, because it is not developed for the processing of new tests but for a specific case. This is highly relevant for the fall detection subject. Since every fall is different, we cannot tailor our system to a few certain test falls. Instead, we need to develop our system to determine a fall event for all types of falls, because no fall is identical to another. A way of avoiding overfitting is basing our approach on theoretical analysis along with the results that we unveil from the test falls to determine a threshold. While we could try to protect the system more effectively using extensive protection but the risk to reward ratio wasn't worth the attempt. So in summary, we could extend the data collected with more varied data but that would come with sacrifices.

4.2 Sensor Performance Evaluation

IMU's are a great way to detect and trigger certain events like in our case FD. As earlier discussed we have chosen the BMI160 due to the low power consumption and its small size. And after the tests have been conducted we can explore whether it was accurate in terms of readings and performance. Which will play a crucial role in determining how accurate the threshold-based approach is when it comes to FD. Initially when we performed unit tests with the BMI160 it showed stable and consistent sensor readings within expected ranges.

Our next test round was performed with the dummy, where we carefully looked into each test round evaluating whether the data looked reasonable which it did to a degree. Initially we set a certain threshold according to other past studies. But as we kept testing we found out that the $^{\circ}/s$ exceeds the threshold past our limit of 250 $^{\circ}/s$. According to this study the max $^{\circ}/s$ a high-trunk body could have was 139.7. But from our experimentation the values exceed this to a staggering 250.1 $^{\circ}/s$ which is the max value set by the BMI160 configuration. But when we compare ADL's tests, such as sitting, with actual falls we can still depict differences in the metrics but thresholds have to be shifted to fit actual IMU values while the ratio that we had before is still relevant. It still needs to be determined whether the issue stems from errors in past studies or from a faulty IMU setup.

Overall the BMI160 performs well with appropriate proportions and it doesn't seem to pick up unnecessary noise.

4.3 Environmental and Practical Limitations

The tests were performed on ground level onto flat surface, but in real life scenarios this might not always be the case. We need to consider the different outcomes and causes for a fall and that they may not always compare to our tests. Such as falling into a higher surface such as the stairs or falls off a tile or over a carpet on the ground. Which is hard to test and catch all of the different outcomes.

There are many different ways to places to mount the sensor, such as the high-trunk, wrists, ankles and many more. While legacy studies has focused on different mounting points of the body and sometimes multiple at the same time. Values may also differ due to how the sensor is placed.

4.4 Data Discrepancy

Some test files got corrupted or totally ignored leading the system to not include these files for final accuracy calculations, therefore we only obtained 28 tests falls instead of the intended 30. This may affect the percentages across accuracy metrics and inconsistency in fall to non fall comparisons. For some test files, the fall may have happened at a later time compared to the others. Which could cause confusion but the only time extended at that point is the time holding the dummy before letting it fall or waiting to sit down.

4.5 Difficulties

Initially there were difficulties to get IMU to work. The IMU was faulty and tests had to be conducted with a backup IMU of another model (MPU6050), to obtain some start-up testing while waiting for the delivery of a new BMI160.

There was also a confusion regarding which exact calculations to be used with the input data to conduct accurate measurements. We also had to understand how to interpret the measurement data in the visualization part of the code, and what formulas to use to make it easily viewable and understandable.

The main driver and research question for this thesis work was to establish a set threshold based on theory and test result. While digging into theory was an effective start to determine a threshold the values didn't hold up to the expectation. With the use of practical test falls, we were able to identify miscalculations on our behalf regarding the pre-test thresholds. While it might seem effective to establish a threshold based on the test data. This method would be inefficient over time due to overfitting.

5 Results

5.1 Results discussion

We conducted some initial tests with the IMU to make sure it is working. We have concluded that the results roughly correspond to the IMUs movement, and therefore our setup works as intended. We can successfully measure acceleration in m/s^2 and the angular velocity in degrees per second.

A unit test was performed to confirm the IMU functionality (Fig. 5). The output shows accelerometer and gyroscope data in each axis. During the time of the first few printouts, the IMU was resting on a stable surface. Then we altered the IMU's position and angles, perfectly showcasing the change in the IMU's values—confirming the functionality of the IMU.

```
Accel(m/s^2): X: 0.10, Y: -0.94, Z: 9.52 Gyro(dps): X: 0.7, Y: 1.4, Z: 0.5
Accel(m/s^2): X: 0.06, Y: -0.92, Z: 9.59 Gyro(dps): X: 0.6, Y: 1.4, Z: 0.3
Accel(m/s^2): X: 0.15, Y: -0.78, Z: 9.54 Gyro(dps): X: 0.8, Y: 1.3, Z: 0.2
Accel(m/s^2): X: 0.01, Y: -0.89, Z: 9.52 Gyro(dps): X: 0.7, Y: 1.6, Z: 0.4
Accel(m/s^2): X: -0.04, Y: -0.80, Z: 9.52 Gyro(dps): X: 0.5, Y: 1.2, Z: 0.2
Accel(m/s^2): X: 1.22, Y: 6.91, Z: 2.30 Gyro(dps): X: -141.0, Y: 5.4, Z: 32.4
Accel(m/s^2): X: 2.41, Y: 8.44, Z: 4.71 Gyro(dps): X: 250.1, Y: 91.3, Z: -61.4
Accel(m/s^2): X: 1.27, Y: -9.41, Z: 4.08 Gyro(dps): X: 76.5, Y: 8.3, Z: -30.5
Accel(m/s^2): X: 2.21, Y: -8.98, Z: 8.79 Gyro(dps): X: -53.1, Y: -38.6, Z: 58.9
Accel(m/s^2): X: 4.67, Y: -6.06, Z: 13.87 Gyro(dps): X: -214.3, Y: 76.4, Z: 99.4
Accel(m/s^2): X: 7.45, Y: 0.42, Z: 8.05 Gyro(dps): X: 39.8, Y: -70.1, Z: -41.8
Accel(m/s^2): X: -9.82, Y: 0.34, Z: 2.11 Gyro(dps): X: -21.5, Y: -118.9, Z: 13.1
Accel(m/s^2): X: -8.99, Y: 1.27, Z: 4.57 Gyro(dps): X: 18.1, Y: 49.6, Z: -12.5
Accel(m/s^2): X: -9.09, Y: -0.94, Z: 3.24 Gyro(dps): X: -4.9, Y: 11.2, Z: -8.8
Accel(m/s^2): X: -10.34, Y: 0.20, Z: 0.51 Gyro(dps): X: 21.1, Y: 106.9, Z: 21.3
Accel(m/s^2): X: 13.27, Y: 0.83, Z: -0.31 Gyro(dps): X: -71.3, Y: -250.1, Z: -9.1
Accel(m/s^2): X: 1.22, Y: 4.63, Z: 11.69 Gyro(dps): X: 143.5, Y: -105.7, Z: -59.1
Accel(m/s^2): X: -0.76, Y: 0.49, Z: 9.59 Gyro(dps): X: 0.8, Y: 1.1, Z: 0.6
Accel(m/s^2): X: -2.51, Y: 3.12, Z: 9.44 Gyro(dps): X: -0.5, Y: -3.2, Z: 7.6
Accel(m/s^2): X: -0.19, Y: -0.57, Z: 9.52 Gyro(dps): X: 0.5, Y: 1.4, Z: 0.3
```

Figure 5: Output data from initial tests

Before we started conducting the tests we calculated a threshold to test. After we did the tests, we calculated a new threshold that would work better for the data we obtained. The goal was finding thresholds that:

- Most or all frontfalls and sidefalls exceed at least one threshold (or ideally both).
- Most or all sitfalls stay below both of the thresholds.

"Frontfalls" refers to the person falling forward, "sidefalls" is falling sideways, and "sitfalls" refers to the recorded data from a person going from a standing to a sitting position, and should therefore not count as a fall.

Threshold A (Theoretical threshold)

Acceleration: 8 m/s^2
 Angular velocity: $168.2 \text{ }^\circ/\text{s}$

This is the threshold that we determined before conducting the experiments. It proved to be a good balance between detecting real falls and avoiding false positives. Using this threshold, you will capture most frontfalls and sidefalls. It is also relatively good at filtering out the sitfalls, as both the acceleration and angular velocity thresholds are also requirements. It correctly misses sitfalls like:

- Sitfall #8: $\text{acc} = 22.39$ (passes acc), $^\circ/\text{s} = 33.1$ (fails $^\circ/\text{s}$) \rightarrow Not counted

We've plotted the different tests and determined a horizontal line representing the threshold for both the gyro (Fig. 6) and the accelerometer (Fig. 7):

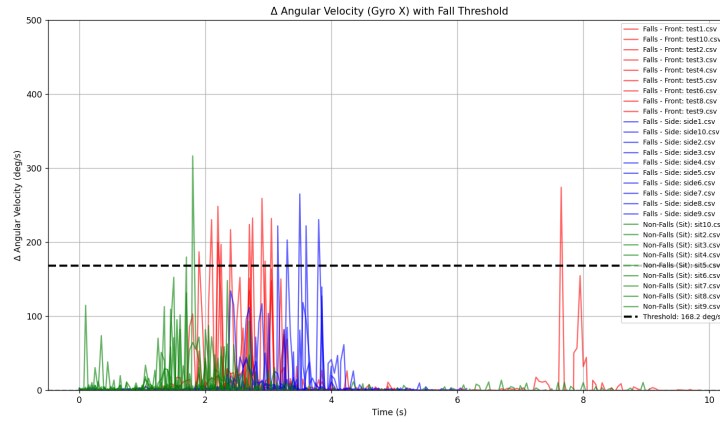


Figure 6: Threshold 1: Δ Angular Velocity over time

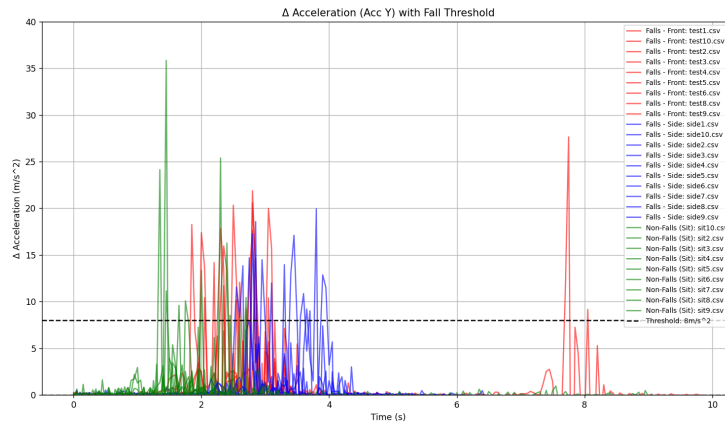


Figure 7: Threshold 1: Δ Acceleration over time

Threshold B (Adjusted for increased sensitivity and specificity)
 Acceleration: 12 m/s^2
 Angular velocity: $168.2 \text{ degrees/second}$
 (Fig 8.)

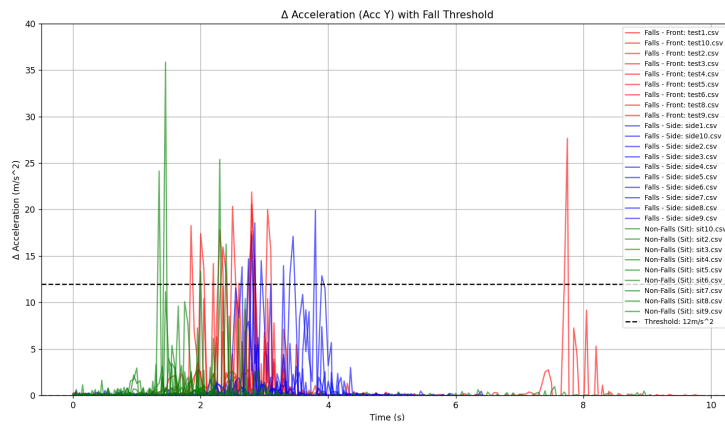


Figure 8: Threshold 2: Δ Acceleration over time

Threshold	Gyro Sensitivity	Gyro Specificity	Acc Sensitivity	Acc Specificity	Average
A	0.6316	0.8889	0.9474	0.1111	0.6448
B	0.6316	0.8889	0.8947	0.4444	0.7149

Table 2: Threshold comparison

Threshold B performs slightly better overall, as seen in the table above. It

is significantly better in terms of specificity but slightly worse sensitivity.

5.2 Method discussion

We built a system consisting of a microcontroller and an IMU. Along with this, we chose a threshold-based approach for the fall detection. A lot of focus was on simplicity, low power consumption, and small size. Hence, our choice of hardware and fall detection method.

Using a threshold is a simple but moderately effective method. A threshold-based model will probably never be able to filter out all false positives and false negatives. When making something as serious as a fall detection system that could be used by elderly people, a threshold-based system might not be enough. In these cases it could perhaps be better to use something like machine learning, deep learning, or even a manually triggered system.

The tests were done with a realistic humanoid dummy. The hardware was attached to the back of the dummy. We then lifted it up to a standing position, and pushed it over whilst running our software to measure the acceleration and angular velocity values. The dummy was very heavy and it was difficult to consistently hold it still in the same position. Using a person to hold it up and throw it on the ground might not be very consistent either. The tests were conducted on both concrete and laminate flooring. These things factor in when evaluating the reliability of this study. On the other hand, it makes every fall unique, which could arguably be more realistic and therefore generate more valuable data. Another thing is that, while realistic in terms of size and weight, the dummy behaves like an unconscious person, which is likely far from reality. If a person were to fall, the person would likely try to soften the fall with their hands, or perform some other kind of movements. This difference could greatly impact output values from the IMU. The dummy is also made to mimic the average person in terms of weight and size. This means that we only get one "person" to conduct the tests on. It would have been better to have a diverse array of dummies of different sizes and weights to test on to get more accurate measurements.

Due to a lack of time and slow communication with the rescue service, we did not get as much time to conduct experiments and test our system as we hoped for. As a consequence of that, we did not get data from as many test falls as we had hoped for. A small sample size can make a dataset less accurate, and it makes the extremes have a greater impact on the dataset as a whole. Ideally, we should have done hundreds of test falls to get as accurate results as we possibly could.

6 Conclusions and recommendations

6.1 Practical implications

Our findings carry several practical implications for the development and deployment of fall detection systems. First, the use of threshold-based methods demonstrates that a fall detection system can be implemented with relatively small and cheap hardware. The microcontroller and IMU used in this study are both compact and energy-efficient, which means they could be integrated into existing wearable devices such as smartwatches. This lowers the barrier for introducing fall detection features into consumer devices, making the technology accessible to a wider audience.

When starting this research, what we had in mind was developing a fall detection system that would be easy to implement in a wearable device. This is so it could be used in devices intended for old people, for example. This way you could make a small and compact piece of hardware that detects if the person wearing it is falling. It could then send a report to emergency services for example, over wifi or bluetooth. The device would be small and would not get in the way, and the user would not have to change batteries or charge it that often, as it has a very low power consumption.

This technology could be implemented in a device that is adapted for non-technical users, which is fitting for use among the elderly population. Theoretically, you could make a wrist or waistband with the IMU and MCU embedded into it, and it would work completely by itself except for the occasional charging or battery change. Many smart watches on the market already feature IMUs, which would potentially make it possible to implement the proposed fall detection method through just an app or a software update. Early fall detection could also reduce time spent laying on the floor, and therefore perhaps prevent unnecessary health risks or injuries.

This work shows the importance of sensor placement and calibration. A device that is not consistently mounted in the same way and on the same place on the body risks producing both false positives and negatives. This implies that wearable devices must also consider things like comfort, and not disrupting everyday activities. This might mean integrating hardware into clothing or accessories that users are already wearing, like smartwatches or belts.

A drawback of using this technology is the power consumption. The user would have to either charge the hardware every once in a while, or change batteries. It would also have to be embedded into some compact wearable device that would be comfortable to wear and that would not get in the way of everyday activities. Another drawback is the inaccuracy of the measurements. No matter where you place the thresholds, you are either going to get false positives or false negatives, or a bit of both.

6.2 Ethical and Legal Implications

As fall detection systems become more integrated in everyday life, especially among the elderly space, ethical and legal considerations must be taken into account. One major concern would be the false positives. This might prompt unnecessary involvement of rescue services and confusion between the victim and the paramedic workers. Over time, when these types of false alarms have been alarmed repeatedly the rescue service might ignore the alerts, reducing overall system reliability. This raises the question of balancing sensitivity (catching as many falls as possible) against specificity (avoiding false alarms). A takeaway from this is that developers must carefully tune thresholds or implement mechanisms for users to confirm the fall, to prevent alarm fatigue.

Another important aspect is the autonomy of the users. Elderly users might not want to wear monitoring devices that might feel intrusive and that they do not understand. It is important to manufacture a device that is small, comfortable, and not too intrusive. At the same time, it must be ensured that users fully understand how their data will be collected, stored, and shared with medical personnel.

There are also legal implications regarding the system. If there were to be a system failure, whose fault is it? The developers, the manufacturers, or the users? Furthermore, data privacy and security is important for many. As data is collected regarding body movement, while IMU's are less invasive than other technologies- legal and ethical actions are determined by how this data is handled since it is often also communicated via a network, while it is measuring body movements. Future research should consider how to design systems that are not only effective and accurate but also legally and ethically appropriate to the best degree.

6.3 Scientific implications

This study contributes to the scientific field of fall detection in several ways. It provides empirical evidence on the limitations of threshold-based methods when applied to realistic fall scenarios. Previous literature [16][17] has often presented threshold methods as either too simplistic or prone to false alarms, but few studies have tested them under realistic conditions with full size humanoid dummies. Our findings confirm that while thresholds can capture general fall tendencies, they fail to adapt to variations such as fall speed, direction, or environmental conditions. This shows the need for more adaptive approaches, such as hybrid models or machine learning-based systems, and provides future researchers with a better idea of what threshold methods can and cannot achieve.

By using a realistic humanoid dummy in combination with an IMU setup, we introduced a testing method that allows for reproducible experiments without risking human injury. Unlike other studies that rely on low-risk volunteer falls,

our approach better mimics the conditions of unconscious or uncontrolled falls. This provides a valuable tool for researchers who want to validate algorithms in a way that bridges the gap between controlled laboratory tests and real world conditions.

Studies made on fall detection for elderly people usually involve experiments in controlled settings, where the person doing the fall is in no risk of hurting themselves. Therefore, it can actually be good that we are using a realistic dummy that we are not afraid to damage, and it could yield better results in areas where other studies have lacked.

Our specific fall detection method, where we measure certain data from a gyroscope and an accelerometer, has not been done before. We have contributed to the scientific field by testing and evaluating this method. Even if it is not that effective compared to other methods then at least we know not to use it and chose a better method instead.

The main thing that this study lacks is large quantities of data. Something that would improve this research would be to conduct a very large amount of tests (several hundreds perhaps), calculate the optimal thresholds, and then do a similar amount of tests again whilst fine tuning the thresholds. This would likely provide more reliable results.

6.4 Further research

This research could be expanded upon by improving the datasets used. One aspect is increasing the sheer amount of data collected. However, the quality of data could also be improved by conducting tests with real people of varying physical attributes. An evaluation of sensor placements could perhaps also make the data more accurate. We placed the sensors on the waist for convenience, but maybe the optimal placement is on the wrist, for example. The best way to do it could also be to have multiple sensors in different parts of the body.

Algorithm improvements could also be researched. One idea is to determine a personal threshold that is unique for every user. This could be based off of parameters such as the user's age, weight, and height. Another idea is to include further context in the measurements. For example, the fall detection could include looking at the posture of the person before the fall takes place, or look at how the person moves right after the fall. Machine learning could be used on a large dataset to determine thresholds.

The setup could be tested in a real setting to optimize things like power consumption, latency, and accuracy. A reliable way to collect fall data is to deploy such a FD system in elderly peoples homes. Over a long time data would be collected under different circumstances during the everyday life. Even more effective would be the collection of data to prevent false positives. The everyday

life of residents of elderly care homes includes different ADL's which could be analyzed to categorize the ADL that are most prone to activate false positives. The system would receive feedback in terms of hardware mounting and placing as well.

A study could also be done on false alarm reduction. You could look at movement patterns after a fall is detected to determine if it actually was a real fall or a false alarm. Another way to do it is to give the user some way to confirm or cancel a registered fall after it has happened.

Taking extensive data sets in different environments and settings could also increase the possibility to analyze and form a more informative report. For e.g the prediction of what type of fall was executed, such as slip induced fall, trip induced fall, etc. A directional analysis could also be conducted. One of the most common fatal falls are caused by a cardiac arrest, By evaluating both fall and post-fall data, further studies may be able to categorize pathological falls into different medical conditions such as strokes or cardiac arrest or any other condition. By combining and studying the characteristics of falls - future system would be able to provide more information to the rescue service regarding the fall.

7 References

- [1] Solomon, C. (2002). Accidental injuries in agriculture in the UK. *Occupational Medicine*, 52(8), 461–466. <https://doi.org/10.1093/occmed/52.8.461>
- [2] Kottari, K. N., Delibasis, K. K., & Maglogiannis, I. G. (2020). Real-time fall detection using uncalibrated fisheye cameras. *IEEE Transactions on Cognitive and Developmental Systems*, 12(3), 588–600. <https://doi.org/10.1109/TCDS.2019.2948786>
- [3] Hauer, K., Lamb, S. E., Jorstad, E. C., Todd, C., Becker, C., & PROFANE-Group (2006). Systematic review of definitions and methods of measuring falls in randomised controlled fall prevention trials. *Age and ageing*, 35(1), 5–10. <https://doi.org/10.1093/ageing/afi218>
- [4] Vallabh, P., Malekian, R. (2018). Fall detection monitoring systems: a comprehensive review. *J Ambient Intell Human Comput* 9, 1809–1833. <https://doi.org/10.1007/s12652-017-0592-3>
- [5] WHO (World Health Organization). (2012). Good health adds life to years: Global brief for World Health Day 2012, WHO reference number: WHO/DCO/WHD/2012.2
- [6] Z. -P. Bian, J. Hou, L. -P. Chau and N. Magnenat-Thalmann. (2015). Fall Detection Based on Body Part Tracking Using a Depth Camera. *IEEE Journal of Biomedical and Health Informatics*, 19(2), 430-439.

<https://doi.org/10.1109/JBHI.2014.2319372>

[7] A. Li et al. (2024). An Integrated Sensing and Communication System for Fall Detection and Recognition Using Ultrawideband Signals. in *IEEE Internet of Things Journal*, 11(1), 1509-1521.

<https://doi.org/10.1109/JIOT.2023.3290421>

[8] P. Rashinkar and V. S. Krushnasamy. (2017). An overview of data fusion techniques, *International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, Bengaluru, India, 694-697.

<https://doi.org/10.1109/ICIMIA.2017.7975553>

[9] Crenshaw, J. R., Bernhardt, K. A., Achenbach, S. J., Atkinson, E. J., Khosla, S., Kaufman, K. R., Amin, S. (2018). The circumstances, orientations, and impact locations of falls in community-dwelling older women. *Archives of Gerontology and Geriatrics*, 74, 38–43.

<https://doi.org/10.1016/j.archger.2017.07.011>

[10] Ferdous, Z., Saadman S. Sakib and M. M. A. Hashem. (2021). Fall Guardian: An Intelligent Fall Detection and Monitoring System for Elderly, *5th International Conference on Electrical Information and Communication Technology (EICT)*, Khulna, Bangladesh, 1-6.

<https://doi.org/10.1109/EICT54103.2021.9733551>

[11] Nahler, C., Steger, C., and Druml, N. (2020). Quantitative and Qualitative Evaluation Methods of Automotive Time of Flight Based Sensors, *23rd Euromicro Conference on Digital System Design (DSD)*, Kranj, Slovenia, 651-659, doi: 10.1109/DSD51259.2020.00106.

[12] Zhou, L., Fischer, E., Tunca, C., Brahms, C. M., Ersoy, C., Granacher, U., Arnrich, B. (2020). How we found our IMU: Guidelines to IMU selection and a comparison of seven IMUs for pervasive healthcare applications. *Sensors*, 20(15), 4090. <https://doi.org/10.3390/s20154090>

[13] Badgujar, S. and Pillai A. S., Fall Detection for Elderly People using Machine Learning, (2020) *11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, Kharagpur, India, 1-4, doi: 10.1109/ICCCNT49239.2020.9225494.

[14] Maier A., Sharp A. and Vagapov Y. (2017). Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things, *2017 Internet Technologies and Applications (ITA)*, Wrexham, UK, 143-148, doi: 10.1109/ITECHA.2017.8101926.

[15] Noury et al, N. (2007). Fall detection - Principles and Methods 2007 *29th Annual International Conference of the IEEE Engineering in Medicine and Bi-*

ology Society, Lyon, France, 1663-1666, doi: 10.1109/IEMBS.2007.4352627.

[16] Jiang et al, Z. (2024) Fall Detection Systems for Internet of Medical Things Based on Wearable Sensors: A Review, in IEEE Internet of Things Journal, 11(21), 34797-34810, doi: 10.1109/JIOT.2024.3421336.

[17] Wang, G., Li, L. Wang, Y. Zhang and Z. Liu. (2020). CMFALL: A Cascade and Parallel Multi-State Fall Detection Algorithm Using Waist-Mounted Tri-Axial Accelerometer Signals, in IEEE Transactions on Consumer Electronics, 66(3), 261-270, Aug. 2020

[18] Ari, N. and Ustazhanov, M. (2014). Matplotlib in python, 2014 11th International Conference on Electronics, Computer and Computation (ICECCO), Abuja, Nigeria, 1-6, doi: 10.1109/ICECCO.2014.6997585.

[19] Bosch. (2020). BMI160: Small, low power inertial measurement unit [Datasheet]. Bosch Sensortec. <https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bmi160-ds000.pdf>

[20] iNEMO inertial module: 3D accelerometer, 3D gyroscope, 3D magnetometer, STMicroelectronics, 2014. [Online]. Available: <https://www.mouser.se/ds/2/389/DM00103319-371319.pdf>

[21] ESP32-WROOM-32, Espressif Systems, 2023. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

[22] ICM-20948, TDK, 2024. [Online]. Available: <https://invensense.tdk.com/wp-content/uploads/2024/03/DS-000189-ICM-20948-v1.6.pdf>

[23] MPU 6050, TDK, 2013. [Online]. Available: <https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf>

[24] Espressif IoT Development Framework, Espressif, 2025. [Online]. Available: <https://www.espressif.com/en/products/sdks/esp-idf>

[25] Liu, J., & Lockhart, T. E. (2014). Trunk angular kinematics during slip-induced backward falls and activities of daily living. *Journal of biomechanical engineering*, 136(10), 101005. <https://doi.org/10.1115/1.4028033>

[26] Vpython, Bruce Sherwood, 2008. [Online]. Available: <https://vpython.org/>

[27] Nyan, M. N., Tay, F. E., Tan, A. W., & Seah, K. H. (2006). Distinguishing fall activities from normal activities by angular rate characteristics and high-speed camera characterization. *Medical engineering & physics*, 28(8), 842-849. <https://doi.org/10.1016/j.medengphy.2005.11.008>

- [28] Kim, T. H., Choi, A., Heo, H. M., Kim, H., & Mun, J. H. (2020). Acceleration Magnitude at Impact Following Loss of Balance Can Be Estimated Using Deep Learning Model. *Sensors*, 20(21), 6126. <https://doi.org/10.3390/s20216126>
- [29] Scheffer, A. C. Schuurmans, M. J., van Dijk N., van der Hooft, T. and Duursma, G. J., (2008). Fear of falling: Measurement strategy, prevalence, risk factors and consequences among older persons, *Age and Ageing*, 37(1), 19–24, doi: 10.1007/s10433-007-0055-6.
- [30] Tolkiehn, M. and McDonagh, D. (2011) User-centred design of a wearable fall detection system for the elderly, *Appl. Therm. Eng.*, 31(14–15), 2444–2450, <https://doi.org/10.1016/j.applthermaleng.2010.09.037>.
- [31] Newaz, N. T., & Hanada, E. (2023). The Methods of Fall Detection: A Literature Review. *Sensors (Basel, Switzerland)*, 23(11), 5212. <https://doi.org/10.3390/s23115212>
- [32] Espressif Systems, "ESP-IDF - Official IoT Development Framework for ESP32," Espressif, [Online]. <https://developer.espressif.com/tags/espressif-ide/>
- [33] Ali, N., Jaafar, A., & Mohamad, N. R. (2025). ESP32-Based IoT Framework for Fall Detection and Caregiver Notification. *International Journal of Research and Innovation in Social Science (IJRISS)*.
- [34] Lin, H.-C., Chen, M.-J., Lee, C.-H., Kung, L.-C., & Huang, J. (2023). Fall Recognition Based on an IMU Wearable Device and Fall Verification through a Smart Speaker and the IoT. *Sensors*, 23(12), 5472. <https://doi.org/10.3390/s23125472>
- [35] Babuji, Y., Woodard, A., Li, Z., Katz, D. S., Clifford, B., Kumar, R., Lacinski, L., Chard, R., Wozniak, J. M., Foster, I., Wilde, M., & Chard, K. (2019). Parsl: Pervasive Parallel Programming in Python. arXiv preprint arXiv:1905.02158. <https://arxiv.org/abs/1905.02158>
- [36] Espressif Systems, "Inter-Integrated Circuit (I2C) — ESP-IDF Programming Guide," Espressif, [Online]. Available: <https://docs.espressif.com/projects/espressif/en/stable/esp32/api-reference/peripherals/i2c.html>.
- [37] jafr03002 (2025) FallDetector (Version: 1.0). <https://github.com/jafr03002/FallDetector>