Pedro Miguel Caiado Ropio

Bachelor's Degree in Electrotechnical and Computer Sciences

Seat Occupancy

Dissertation submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Computer Science and Engineering

Adviser: Rui Santos-Tavares, PhD
Auxiliar Professor,
Universidade NOVA de Lisboa

Examination Committee

Chair:
Rapporteurs:
Members:

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

August, 2020

Seat Occupancy

This document was created using the (pdf)LaTeX processor, based on the "novathesis" template[1], developed at the Dep. Informática of FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

To my parents

Acknowledgements

This work wouldn't have been possible if not for the help of some very special people.

First I would like to thank my adviser teacher, without whom I would have no dissertation or bike to make tests on.

Secondly I would like to thank my parents who always supported me, especially during these last five years, I know it was hard.

Lastly I would like to thank everybody who helped me with this present theses. Thanks sis for reviewing this a few times. Thank you Raquel for being a loyal test subject. And specially thank you Marta and Gil for putting up with me and helping me

while we were stuck at home while I wrote this.

Abstract

Information technology continues to seep into today's world market. Technologies such as the Internet of Things (IoT) continue to gain popularity by helping businesses gather information. This information, if well managed and understood, can greatly increase productivity through efficient use of resources and operations. In this way, the present work offers a possible solution to increase productivity in gyms. By adding sensors in common workout machines, the business manager can understand which machines are the most/less popular, which need more/less maintenance, and even study which areas of the gym need more/less air-conditioning, ventilation or illumination. If information is released to the public, the costumers can also better manage their time by choosing to go to the gym when it is less full.

This work was successful in using accelerometer sensors to gather information about movement in stationary and elliptical bikes. A small study was also conducted to analyze machine usage throughout the day.

Keywords: Internet of things (IoT), accelerometers, seat occupancy

Resumo

As tecnologias da informação continuam a integrar-se nos mercados mundiais. Tecnologias como a Internet das Coisas (IoT) continuam a ganhar popularidade devido à sua capacidade de recolher informação. Esta informação, se for bem compreendida e gerida pode ser utilizada por empresas para aumentar a sua produtividade através de um uso mais eficiente dos seus recursos e das suas operações. Desta maneira, o trabalho aqui apresentado oferece uma possível solução para melhorar a gestão em ginásios. Ao adicionar sensores a máquinas de exercício comuns, o manager do ginásio pode vir a entender melhor quais as máquinas mais/menos populares, quais precisam de mais/menos manutenção ou até quais as áreas do ginásio que precisam de mais ar condicionado, ventilação ou iluminação. Se a informação sobre a utilização das máquinas for partilhada com os utentes do ginásio, estes podem também planear melhor o seu tempo ao escolher só ir ao ginásio quando este está mais vazio.

Este trabalho obteve sucesso na utilização de um acelerómetro na recolha de informação sobre movimento em bicicletas estáticas e elípticas. Um pequeno estudo sobre a utilização destas máquinas ao longo do dia também foi realizado.

Palavras-chave: Internet das coisas (IoT), acelerómetros, Ocupação de assento

Contents

List of Figures

List of Tables

Acronyms

AI	Artificial Intelligence
FaaS	Function as a Service
FTDI	Future Technology Devices International
GPIO	General Purpose Input/Output
HART	Highway Addressable Remote Protocol
HVAC	Heating, Ventilation, and Air Conditioning
IaaS	Infrastructure as a Service
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IP	Internet Protocol
IPv6	Internet Protocol version six
ISA	International Society of Automation
ISP	Internet Service Provider

JSON	JavaScrip Object Notation
LR-WPANs	Low Rate Wireless Personal Area Networks
MAC	Media Access Control
MCU	Microcontroller Unit
MQTT	Message Queuing Telemetry Transport
PaaS	Platform as a Service
RPL	Routing Protocol for Low-Power and Lossy Networks
SaaS	Software as a Service
SPI	Serial Peripheral Interface
TCP	Transmission Control Protocol
UML	Unified Modeling Language
WSAN	Wireless Sensor and Actuator Network
WSN	Wireless Sensor Network

Chapter

1 Introduction

The latest developments in electronics and communication systems of the last decade, particularly in wireless communication, have enabled the emergence and further development of the Internet of Things technologies.

The rise of these technologies and the increasing demand for real-time analytics have created a new paradigm where this information is fundamental in the organization of any system. Many businesses such as gyms can improve their own management by gathering information about themselves, in this example, by collecting information on which machines are most frequently used, gyms can not only make a more effective preventive maintenance of their equipment but also study their own market and make more informed decisions. In this way it is also of great interest that some limited area spaces such as conference halls, libraries, internet cafes or even gyms can display their availability of seats in real-time so that people can arrange their time better and avoid going to a place that's completely full.

Applications like this go in line with the Society 5.0 paradigm, which aims for a human-centered society that balances economic advancement with the resolution of social problems by a system that highly integrates cyberspace and physical space. With the integration of AI, IoT and Big Data technology, into the industry, it's expected that overall productivity and performance can increase even further.

1.1 Internet of Things (IoT)

The proposed objective of this thesis falls into the category of the IoT technologies. These kind of technologies have become much more popular in the last years and are starting to permeate in the industry. Monitoring, collecting and analyzing data in real-time can provide insights into the business, improving efficiency and profitability.

The Internet of Things can be viewed as an extension of the internet into the physical world [6], this extension is made by connecting physical everyday objects such as chairs, tables, lights or even cars to the Internet.

It's possible to improve efficiency and help with everyday work, by giving the objects around us the ability to sense, actuate or share information. This technology has already been greatly studied, discussed and in many ways implemented in the industry such as Siemens' MindSphere. MindSphere [25] is a cloud-based IoT operating system that connects assets such as: products, systems or machines allowing the user to collect, monitor and analyze data in real-time.

Terms such as "things", "nodes", "devices"or "objects"are all frequently used in IoT related documents and frequently share the same meaning in such articles [27]. These terms, with regard to the IoT often refer to an object or piece of equipment of the physical world with the mandatory ability to communicate with other devices, this object might also possess other advantageous capabilities such as the ability to sense, actuate and even the abilities to capture, storage and/or process data.

IoT objects are usually embedded with small, low-power devices in order to minimize production costs [24]. For this reason IoT "things"frequently suffer from low computational power, low memory, short radio signals and a short battery life.

Such constrains generate a need for the creation of new lightweight systems and forces new developments in different areas such as: communication protocols, network access technologies, operating systems and security protocols.

1.1.1 IoT Architecture

IoT applications do not have a clearly defined architecture, and can largely differ from case to case but some author's propose somewhat similar architectures [16, 27, 33–35]. These architectures are all equivalent to each other and are divided into three basic layers. In this paper they will be referred as: the application layer, the network layer and the perception layer.

- **Perception Layer**: This layer consists of all "things"that are sensing, actuating and overall generating data.
- **Application Layer**: This layer is where all the data is stored analysed and processed.
- **Network Layer**: This layer is what gives the "things"their ability to connect to the internet. In this way the network layer bridges the other two layers, allowing the collected data to be stored and analyzed elsewhere.

1.2 Motivation

The main focus of this thesis is the application of a simple accelerometer sensor in various gym machines.

The ability to monitor such machines in real-time can bring various advantages to both the consumer and owner of the gym. If data from every machine is collected, stored and analyzed, owners can verify with certainty which machines are being overused and which are underutilized. This way maintenance can be optimized, rarely used machines can be replaced with more popular ones and graphical information of overall gym utilization can be provided to the costumers throughout the day. This could help costumers decide when to go to the gym based on space availability.

Information about overall gym utilization could also prove beneficial in regards to energy efficiency. The application of WSAN and occupancy-driven demand control measures for lighting and HVAC systems can potentially account

for a decrease of more than 20% of total energy consumed in commercial office buildings [20].

1.3 Objectives

The objective of this study is to create a fully functional system, that can ascertain if accelerometer sensors can be used to detect seat occupancy in real-time, with a good enough degree of certainty. This type of sensor was chosen due to its small form factor, ease of installation and very small price, qualities that can allow for a greater and more diverse number applications in many types of environment.

Chapter

2 State of the Art

This chapter presents the state of the art which includes a small literary review on the subject and discusses some enabling concepts that permit the proper function of this application.

2.1 Literature Review

In this section of the work, a small overview of published papers on the subject of seat occupancy and activity recognition are presented. The purpose of the first overview of this section is to show and analyse some of the differences between the various types of implementations and sensors that are available today on the market. After this first overview, another one is made on the subject of sampling rates, due to a lack of industry standards and the low demands of this particular application. The last overview of this section is on the subject of signal processing.

2.1.1 Seat Occupancy

Seat occupancy detection is a well studied topic with many applications already in service, however these applications are mainly associated with the car industry and most of the patents defined for seat occupancy are intended for vehicle seats. Since seat occupancy detection in vehicles is important due to the high cost of

airbag replacement many different implementations are in use today but none of these are low cost.

The most adopted methods of seat occupancy detection use either resistive pressure sensors or capacitive sensors.

Resistive sensors as the ones in figure 2.1 are not ideal because they can't differentiate between an heavy object and a person, they're not low cost and they need to be embedded in the seat cushion which further complicates the installation process.

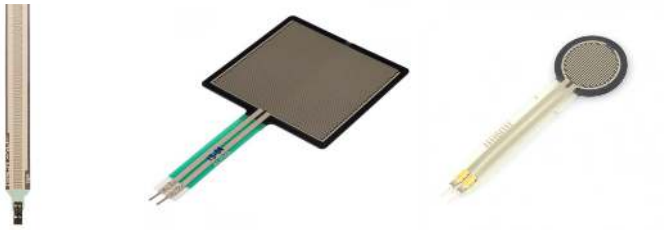


Figure 2.1: Pictures of three force sensing resistors. [1]

In [19] the authors present the results of an experimental evaluation of three different chair sensors. The tested sensors were: a resistance strain gauge that consists of a resistive element that changes its physical dimensions with the application of force, a vibration sensor which was composed of a microcontroller-based tri-axial accelerometer and finally a micro-mechanical switch that is forced into a closed position when occupants are seated.

All three sensors were installed in a common office chair with rollers, the strain gauge and mechanical-switch were placed in a specially constructed holder embedded in the seat pan and the accelerometer was glued under the chair seat. The sensors transmitted data via a wireless low-power transmitter to an online data storage system, the transmitters only sent data out when the sensors state registered a change. The strain gauge was only triggered when a mass of approximately 10 kg was applied to the centre of the seat and the mechanical switch was only triggered by a mass of approximately 11 kg.

The results of [19] were measured in an open-plan office during three different 10 hour tests, conducted over three workdays. The author's observed that the

mechanical-switch sensor had the best performance with 99% of average accuracy, next was the strain gauge sensor which presented an average accuracy of 95% and lastly the accelerometer which presented an average accuracy of 85%. The biggest discrepancy between test days was also measured by the accelerometer which presented an accuracy off 92% and 95% in the first two days and a 76% accuracy in the third day. The low accuracy of the accelerometer sensor was however attributed to the poor detection algorithm, which made use of a large verification window (of approximately 10 minutes) before determining user absence, this verification window helped the sensor to differentiate between vibrations resulting from occupants' presence and vibrations caused by other sources.

Despite the discussed problems the study concluded that all three sensors were capable of providing useful occupancy information and that although the mechanical-switch presented the best performance, this sensor was incapable to detect mobile occupants, presenting false-negatives when seating occupants moved in their chairs, momentarily removing their weight from the sensor, this problem was however solved by the implementation of an algorithm that filtered such quick events and prevented them from hampering the system's performance.

After the discoveries from the previous study, in [20] some of the same authors implement a seat occupancy system that consisted of a passive infrared motion sensor. This system was designed with the intention of automatically control the lighting system, and was installed on the ceiling directly above a seat which had the already mentioned embedded mechanical-switch. The controllability of each luminaire was achieved by using off-the-shelve plug-in switching nodes and the usage of a Z-wave wireless communication protocol. Z-wave protocol was mainly chosen due to interoperability. The experiment was conducted during a three week period, in an open-plan space which had 12 work-spaces and resulted in a 24% average reduction in lighting electrical energy consumption.

In [9] the authors introduced and tested two low cost systems for occupancy detection on five different objects and in a second evaluation tested their classification accuracy on two different beds. The first sensor was an accelerometer that detected entry and exit events and the second was a capacitive proximity

sensor that could detect the presence of a human body over a certain distance. The accelerometer used a threshold-based feature to detect occupancy and the constructed prototype was built with a LightBlue Bean [28], which is a microcontroller with Bluetooth LE communication. The capacitive sensor was attached to an Arduino but its installation included various complications. An initial environment calibration then an occupancy calibration, finally drift compensation and establishment of several thresholds for distinguishing poses, were required before the prototype could be used. An approximation of the build with the capacitive sensor can be seen in figure 2.2.

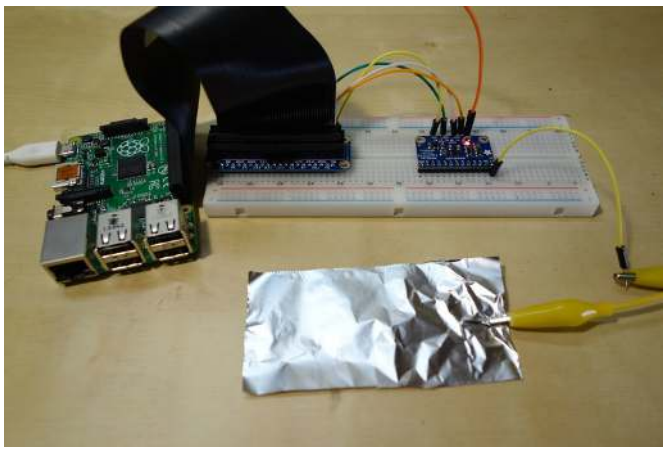


Figure 2.2: Picture of very basic build for a capacitive sensor. [10]

The evaluations were conducted in an office chair, a wooden chair, a wheel chair and two different beds. Two users each performed seat and leave actions five times on every piece of furniture and then ten different users occupied only the two beds and performed the seat and then leave actions ten times each. The results revealed that the best overall detection rate was achieved by the capacitive sensor except when measured in the office chair. The paper also concluded that the accelerometer works better when the seat is mobile and was able to detect entry and leaving events with a good success rate but struggled at differentiating both and when the structure is very rigid, or the person is sitting carefully. The capacitive sensor did not have this limitations but could easily be disturbed by near metal parts. The capacitive sensor prototype was also the most expensive to build and the most difficult to install, due to more calibration phases.

The capacitive sensor was also tested for a library application in [22]. Here the authors used a capacitive sensor to detect both seat occupancy and seat hogging and also used a low cost infrared transmitting diode as a transmitter and a photo-diode as a receiver to detect field obstruction. These sensors were chosen because the authors wanted to create a cheap and non intrusive system that could solve the problem of finding an available seat in a library or event hall. Additionally, they wanted to solve the problem of seat hogging which, in this study, is when students leave their belongings unattended in order to save seats for later.

In order to give the chairs freedom of movement, both sensors were installed under the table, this however difficults occupancy detection due to a greater distance between the people and the sensors, so to solve this problem, the authors fine-tuned the sensor to detect people occupancy using standard deviation.

The setup for the capacitive sensor consisted of a fifteen centimeters foil constructed in a round shape to reduce stray capacitance and a Raspberry Pi with a capacitive touch sensor controller chip that controlled the system and transmitted data. The setup for the infrared sensor consists of a small such sensor connected to the before mentioned Raspberry Pi, this sensor works as a sonar, the infrared radiation emitted by the transmitter is reflected by nearby objects and then detected by the photo-diode.

When analyzing the capacitance of objects that can be put on the table, the authors found that both a book or a laptop on top of the table presented the same approximate capacitance has a human sitting in the chair, depending on the posture. This problem was resolved by adapting the occupancy detection algorithm to notice capacitance variations which were only caused by a person occupying the seat.

Finally in the experiment, the authors concluded that the capacitive sensor was able to reliably detect if the seat was occupied or not, but struggled when there was a laptop nearby and the table moved due to other users, reporting false positives for seat occupancy. The infrared sensor only presented an accuracy of 80%, this might be due to variable lighting conditions and occupants being seated outside the sensor's line of sight. Although the inferior detection accuracy of the

infrared sensor, this sensor presented a faster speed of classification, while the capacitive sensor experienced delays of up to twenty one seconds for detecting seat hogging after a user left.

2.1.2 Sampling Rate

A myriad of applications that use accelerometers for activity recognition are already in use today. Most of these applications tend to use wearable tri-axial accelerometers for diverse implementations and studies, such as novel interaction techniques [18], situated support in smart environments [14], automated health assessments [13, 23] or health care automation [7, 21].

Wearable accelerometers used for human activity recognition require careful configuration and reliable data analysis. This can pose a challenge when battery life and data storage are limited. The amount of time between each data point, i.e., the sampling rate at which data is extracted from the accelerometer sensor is a critical parameter that directly affects power consumption, data storage and bandwidth requirements in case of a wireless transmission application. If a sampling rate is too high then all of these limited resources can be unnecessarily wasted and if the sampling rate is too low, then there will be missing details that needed to be analysed.

In [17] the authors try to find a way to optimize sampling rates for accelerometer-based human activity recognition applications. Previous work in this area suggested that reasonable sampling rates for measuring human activities were approximately of 20Hz. This sampling rate came from the Shannon-Nyquist theorem [15] that states that a sampling rate of a particular signal must be at least twice its highest frequency for a loss-less reconstruction to be achieved. As it is assumed that voluntary human movements do not typically exceed 10Hz [17], then a good enough sampling rate should be of at least 20Hz.

In more recent studies however, accelerometer data is recorded using much higher frequencies. Sampling rates of up to 250Hz [32] were used, this is due to the study of movement that in recent years has been focusing, not only on more complex movements and activities, but also on the quality of the movement itself.

After studying 5 different benchmark datasets and applying their method of analysis to them, the authors of [17] discovered that the now conventionally used sampling rates appear to be too high, and that all 5 benchmark datasets were originally recorded at a much higher sampling rate than needed. The majority of the optimal sampling rates for these datasets was about 45Hz, more than half the usually utilized 100Hz.

2.1.3 Signal Processing

Some older research articles, related to wearable accelerometers, only used simple statistic methods to analyse and classify the sensed data. Statistics like the average value, the root mean square and integrated values were enough to train a neural network for movement classification with a accuracy of around 85-90%.

This level of accuracy is not enough for many applications and so a logical way to improve it could be to use higher level statistics such as: mean, standard deviation, skewness, kurtosis and eccentricity. This way to improve accuracy was tested in [8].

In that study the authors tried to differentiate between eight motion states and found that the mean and standard deviation are the most useful features, not only for distinguishing between static and dynamic states but also to distinguish between different static states like standing, sitting or lying down.

As seen below skewness and kurtosis describe the shape of a distribution of data, skewness refers to the degree of distortion or asymmetry in this distribution, i.e., how much the peak value is deviated from the center, kurtosis measures the peaked value itself at the center of the distribution.

Skewness and kurtosis were used to distinguish between dynamic states, such as walking, running, climbing and descending stairs. The skewness in the X axis of the acceleration data was used to distinguish between horizontal and vertical movement while the skewness in the Y axis could differentiate between walking and running states. The kurtosis in the X axis of the acceleration data was used to differentiate between up and down movements.

All of this statistic values were calculated for only two acceleration axis, but the

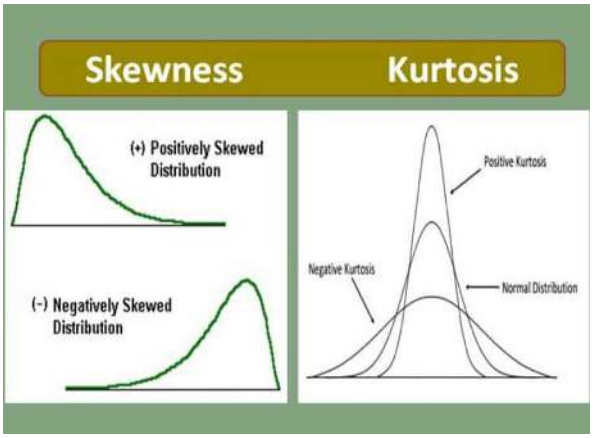


Figure 2.3: Visual representation of skewness and kurtosis. [2]

authors managed to obtain almost perfect classification results. They obtained a correct classification rate of 100% for static, walking and running states, but only an 93% for upstairs movements and a 87% for downstairs movements. This way the authors realized that for simple applications such as distinguishing between static and dynamic states the standard deviation is the most indicated.

2.2 Enabling Concepts

Here, some enabling concepts are discussed in order to better understand and compare the different technologies that will be used latter. First a CLOUD section is presented due to its deep connection with the IoT technologies. Its structure will also be used to describe the application. Then a small overview of various different low power wireless communication protocols is presented due to their necessity when the application passes to a large scale implementation. And finally a comparison between relational and non-relational databases is presented.

2.2.1 CLOUD

Internet of Things (IoT) is a technology that enables the connection of various ordinary objects through the Internet and gives them the ability to transfer data between them without requiring direct human interaction. This new paradigm predicts that many of the everyday objects that surround us today will somehow

be connected to the network. In this context, cloud computing can provide the virtual infrastructure to monitor these devices, store data and more.

Cloud computing is the action of accessing or storing data and programs over the Internet instead of the computer's own hard drive. Cloud services are divided into 4 categories, Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS) and Function as a Service (FaaS).

IaaS is the most basic Cloud service, it allows a costumer to rent IT infrastructure, usually servers, from a Cloud provider.

PaaS serves to supply the costumer with an environment for developing, testing, delivering and managing software applications without needing to set up or manage the underlying infrastructure of servers, storage, network and databases needed for development.

SaaS delivers software applications over the Internet on a subscription basis. This service helps the costumer to host and manage the software application and underlying infrastructure.

Finally FaaS works in the same way as PaaS, however the developers only pay for function execution time which can in turn lower costs and increase scalability at the cost of higher latency.

2.2.2 Wireless Comunications

For the proposed work, a simple and low energy wireless communication technology is needed, for that reason a brief introduction to the main features of various communication technologies is presented below. All the presented technologies work with the IEEE 802.15.4 technical standard, except for the bluetooth Low Energy. The IEEE 802.15.4 standard specifies both physical layer and media access control for Low Rate Wireless Personal Area Networks (LR-WPANs). The main objective of the IEEE 802.15.4 based standards is to provide very low cost communication to low power devices and offers speeds up to 250kbps in the 2.4GHz frequency band [3].

- ***6LoWPAN***: IPv6 over Low Power Wireless Personal Area Networks (*6LoWPAN*) applies the Internet protocol to low power devices that have limited processing capabilities. The *6LoWPAN* working group has been able to define fragmentation, header compression, IPv6 address autoconfiguration and IPv6 neighbour discovery mechanisms that allow IPv6 packets to be traded over IEEE 802.15.4 based networks [31].

- ***Bluetooth Low Energy***: Both the high complexity and high power demanded from the Bluetooth standard make it not suitable for Wireless Sensor Network (WSN) applications. The Bluetooth Low Power on the other hand is an ultra low-power technology that can reach data rates of up to 1Mb/s over distances of up to 10 meters in the 2.45GHz frequency band.

- ***ISA-100***: The ISA-100 standards were especially design for automation and low data rate monitoring applications. Networks using these standards only work using the 2.4GHz radio frequency band and use channel hopping to minimize interference [31].

- ***WirelessHART***: *WirelessHART* is an extension of the HART protocol and was specifically designed for process monitoring and control. This protocol is compatible with most existing devices but can only operate in the 2.4GHz frequency band [11], the protocol also employs frequency hopping, redundant data paths and retry mechanisms to minimize transmission errors. Each device using this protocol can not only transmit its own data but also retransmit data originated in other devices in the network [26].

- ***RPL***: The Routing Protocol for Low Power and Lossy Networks was specifically developed for networks in which both nodes and routers are expected to be power constrained [30]. For this reason, the protocol is a reactive one, it establishes data routes only when needed. This Protocol works with IPv6

and was optimized for one-to-many and many-to-one traffic patterns [11].

- ***Ultra-WideBand***: Ultrawideband is a short range wireless communication technology that transmits data by periodically emitting very short, high peak energy impulses. The main advantages of this technology are its good localization capabilities, the possibility to share previously allocated frequency bands by hiding its signals under noise floor, the ability to transmit high data rates with low power and good security due to its unique mode of operation [12].

- ***ZigBee***: Zigbee is a short range, low cost, low data rate and very low power wireless mesh network standard, targeted at battery powered devices. This technology was designed to be simple, cheap and capable of supporting several different topologies, however the low power limits its range [12]. Due to both the physical and the Media Access Control (MAC) layers of the ZigBee's protocol being defined by the IEEE 802.15.4 standard, this protocol can provide data rates of up to 250kbps in the 868MHz, 915MHz and 2.4GHz frequency bands [11].

Another wireless communications platform that is used for WSNs is Wavenis. This open protocol was specifically design for ultra low-power and long distance two-way transfer of data in wireless communications [11].

This technology is comparable to the IEEE 802.15.4 standard in the way that it lets the user use other applications such as Zigbee on top of it.

The Wavenis Open Standard Alliance was created to manage standardization activities and proposed a new standard for this technology specially design to meet WSN needs, this was because its members believed that the communication standards comercially available in the market didn't meet all their technical needs [29]. The Wavenis Specification covers the Physical, Medium Access Control, Logical Link Control, and Network layers [29].

2.2.3 Relational vs. Non-Relational Databases

Relational databases like MySQL represent and store data in tables and rows. They're based on a branch of algebraic set theory known as relational algebra. Non-relational databases like MongoDB represent data in collections of JSON documents.

Relational databases emphasize the use of referential integrity. Referential integrity is the concept in which multiple database tables share a relationship based on the data stored in the tables, and that relationship must remain consistent. Relational databases facilitate the search, alteration, sorting and analysis of specific pieces of data without having to search sequentially through an entire database.

A non-relational database simply stores data without explicit or structured mechanisms that link data from different tables, which gives it a superior horizontal scalability. This can be extremely advantageous when dealing with great amounts of data. Non-relational databases work very well with object oriented programming languages, such as JavaScript, while the objects represented in a relational database are stored in a format that can't be easily used by the frontend and vice-versa.

Chapter

3 System Modeling

This chapter presents the system modeling. The first section presents two models of the system as a whole, first the built model and second the model when the system is connected to various sensors. The second section of this chapter presents the state and sequence UML diagrams of the system.

3.1 System Model

The basic model of the built system can be observed in figure 3.1. In this implementation, the perception layer consists only of an accelerometer sensor. The network layer uses IPv6 and includes a ESP8266 microcontroller and the MQTT broker. The application layer consists of the MongoDB cloud non-relational database.

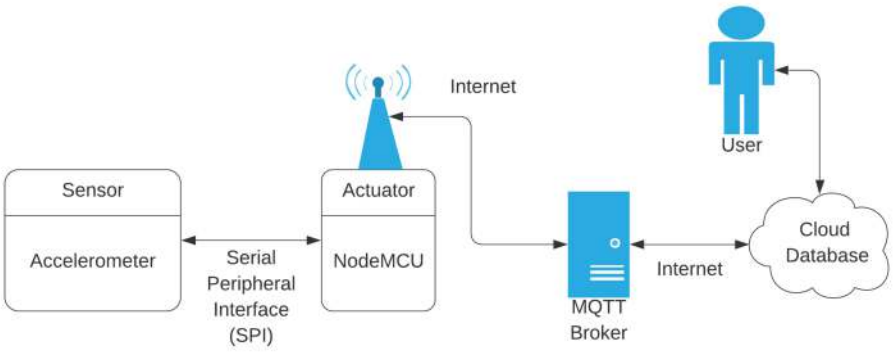


Figure 3.1: Block diagram of the implemented build

A large scale implementation that requires many sensor devices could be easily achieved with the same architecture by simply inserting more sensors and microcontrollers to the MQTT network. Changes to the type of communication between the microprocessors and the broker would also be needed in order to improve battery life. One of the key objectives of this implementation is a quick and easy installation, therefore IPv6 is not a very suitable communications protocol due to its poor energy efficiency. The Zigbee protocol however was design for this kind of applications and the impact of its short range can be minimized by installing the MQTT broker closer to the sensors, a Raspberry Pi could be used to implement this gateway.

A representation of these changes can be seen in figure 3.2, in this figure it's also possible to see the different communication protocols used by the machines.

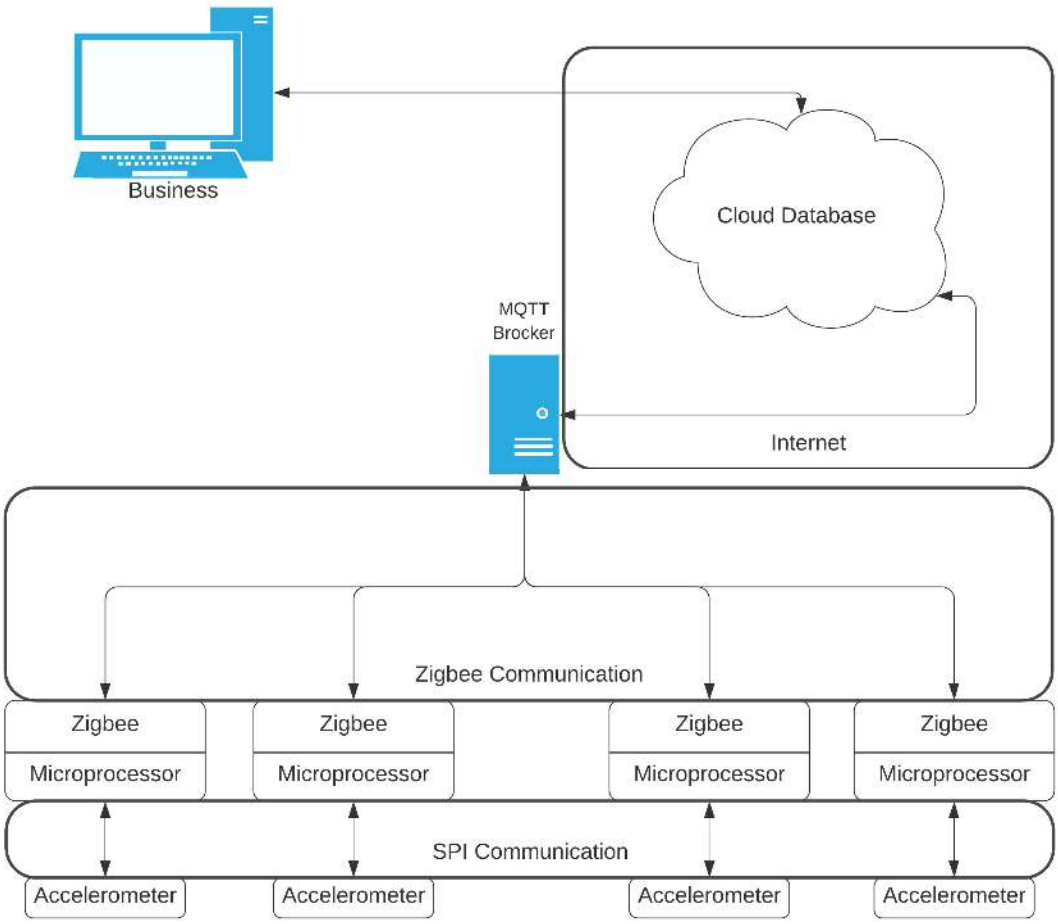


Figure 3.2: Block diagram of the proposed build

3.2 UML Diagrams

The basic state diagram of this system is best exemplified by the state diagram of the NodeMCU, shown in figure 3.3. Because the rest of the system is simple and only forwards messages to the database, as seen in figure 3.4.

It's possible to see in figure 3.3 that the system begins when it's turned on, and it doesn't reach an end state, because it will always function while it is on. When this system begins, the NodeMCU will try to connect to the internet and to the MQTT broker [3.3(a)], if this is successful, then the NodeMCU will try to check the acceleration values that are calculated by the sensor [3.3(b)].

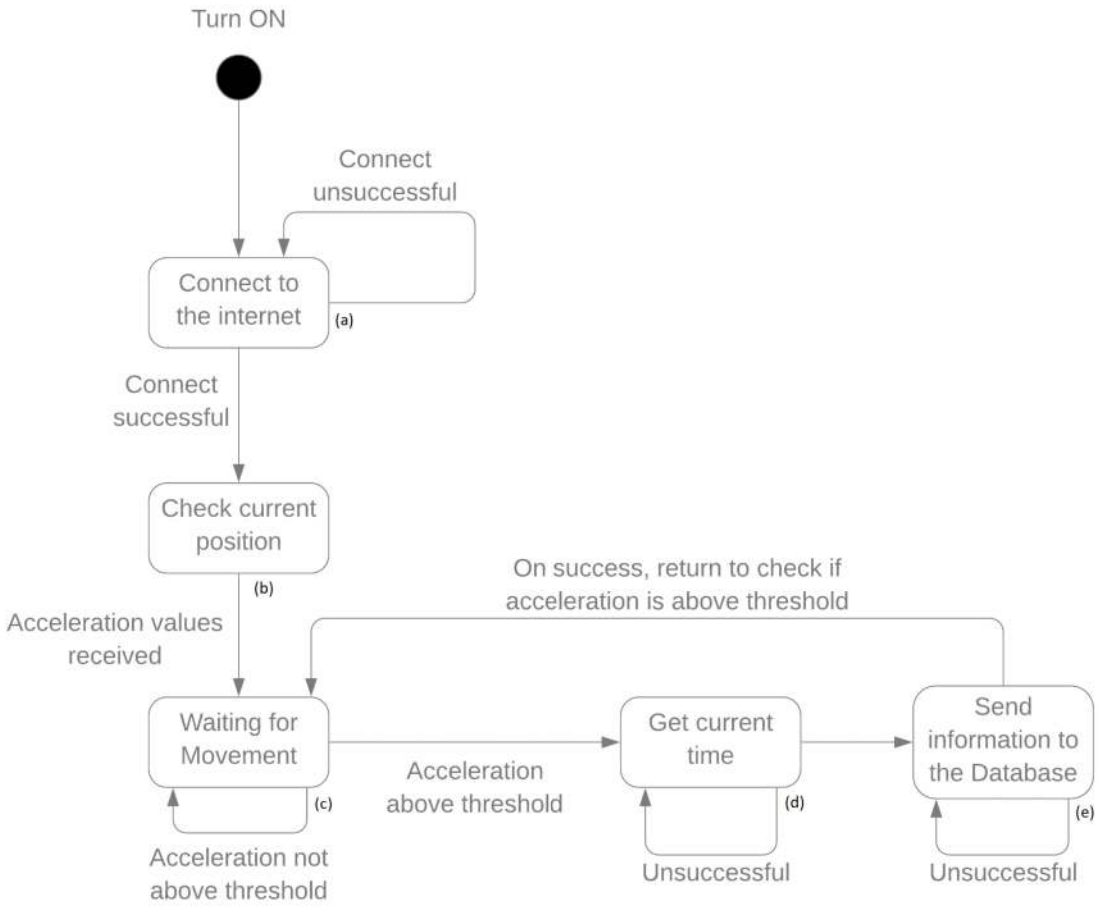


Figure 3.3: State diagram of the system

From here, the NodeMCU can work in two manners, if it is working normally, it will simply repeat the cycle of reading data from the sensor and send it to the MQTT broker that will send it to the database [3.3(c-e)]. But if the NodeMCU is

working with an algorithm that tries to minimize energy consumption, then it will only try to send information to the broker when the bike is in use. The state diagram presented in figure 3.3 shows how the system works when this algorithm is implemented. This new algorithm continues to check the acceleration values at the same frequency but it only will send information to the broker once every second. This way the microprocessor can store and analyse the collected values during each second and determine if the bike was being used during that second or not. Then if this algorithm determines that there was movement during that second, the nodeMCU will request the current real world time [3.3(d)], from the internet, and send information about the movement to the MQTT broker that will then send it to the database [3.3(e)].

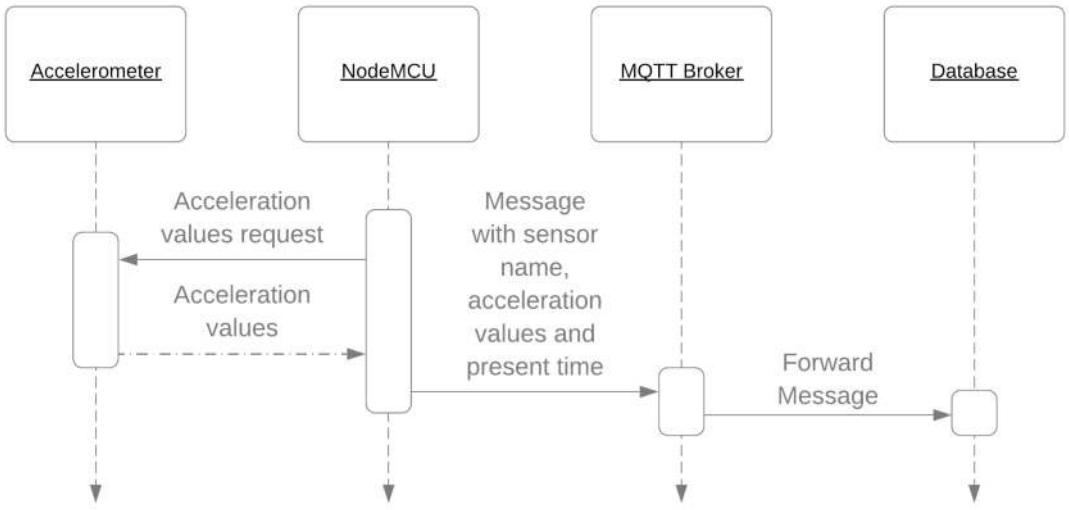


Figure 3.4: Sequence diagram of the system

A sequence diagram depicts the interactions between the different objects of the system in a sequential order, in this case, it depicts how the information passes from the accelerometer sensor to the database, passing by the NodeMCU and the MQTT broker.

Chapter

4 Implementation

This chapter is concerned with how this application was implemented, and is divided in three sections. The first section of this chapter exposes how the application was physically implemented and what hardware was used in it's construction. The second section of this chapter is concerned with the main algorithms used by the application, and the final section discusses some of the problems that the application has and how to minimize them.

4.1 Logistics needed for implementation

The initial proposal for this work was the full development and implementation of a system that could monitor the occupancy of a seat by a person. The system should work by detecting and identifying the movement of a sitting person, and all collected data should be saved on a remote Cloud-based database.

The first implementation of the system started with an Olimex development board called MOD-WIFI-ESP8266-DEV. This module comes with a 16Mb of SPI flash memory and all 10 GPIO resources are available for breadboarding [4]. This module was chosen due to it already being available for testing.

As can be seen in the picture below 4.1, this WiFi module was connected to a power supply and a FTDI through a breadboard. The FTDI is a Serial to USB

interface used to connect the WiFi module to a computer so it can be programmed.

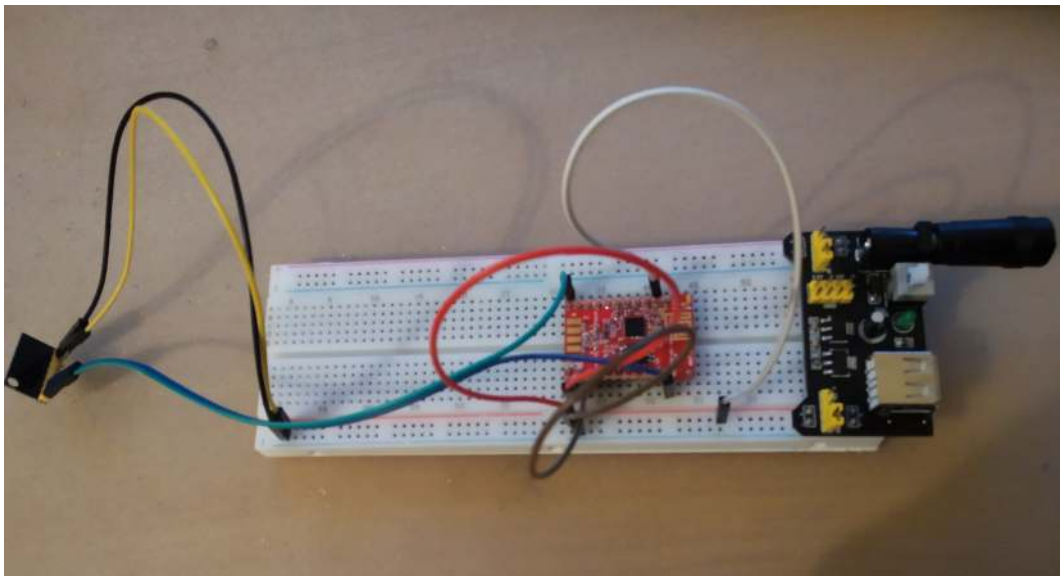


Figure 4.1: Picture of the first build

The ESP8266 was also connected to an ADXL345 which is an ultra low-power, triple axis accelerometer from DFRobot, seen in figure 4.2. This accelerometer has high resolution, up to 16bits and is well suited for tilt-sensing applications, as well as measuring dynamic acceleration. This sensor also provides a low-power mode, which enables an intelligent motion-based power management, with threshold sensing and active acceleration measurements coupled with extremely low power dissipation [5].

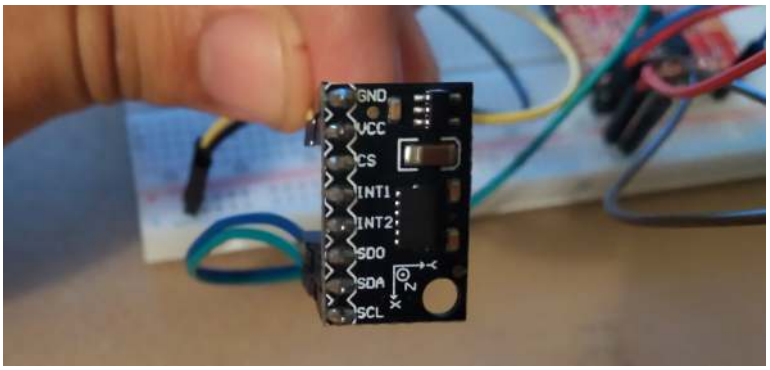


Figure 4.2: Picture the used accelerometer

Due to utilization problems this initial build had to be substituted to another. Instead of using the Olimex development board a different ESP8266 was chosen. A NodeMCU V3 Lua, WiFi ESP8266 - CH340 is a superior development board

that comes with a built in USB connector and 13 GPIO pins available for breadboarding. This module presents dedicated RESET and FLASH buttons, as well as three different 3.3V Voltage supply pins, a 5V supply pin and four GROUND pins. The module has integrated TCP/IP protocol stack and a maximum of five concurrent TCP connections, a maximum of 16MB of flash memory, 512KB normally, and works at 3.3V with a 10uA to 170mA current consumption. The NodeMCU was also partly chosen due to its low price and easy market availability.

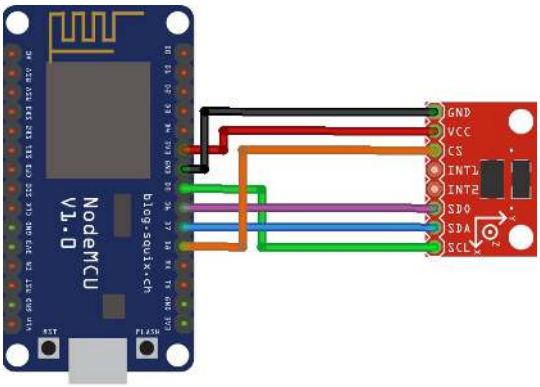


Figure 4.3: Module of the NodeMCU implementation

This system continues working with an ADXL345 accelerometer and power is supplied through an USB cord. In this build the accelerometer communicates with the microprocessor through SPI. Serial Peripheral Interface (SPI) is a synchronous serial communication interface specification that is mainly used in short-distance communications in embedded systems. Devices that adopt SPI communication use a master-slave architecture which employs only one master. Multiple slave devices can be supported due to the slave select lines, which are individual to each slave. This communication protocol uses four different logical signals, the serial clock or SCLK which is an output from the master, the slave select or SS, which is also an output from the master, the master output slave input or MOSI and the master input slave output or MISO. All devices that connect to the same master share the SCLK, MOSI and MISO lines.

After data transmission between the ESP8266 and the sensor as occurred, the microprocessor then processes the received values into a more comprehensible format and sends them to the MQTT broker. That communication is presently

made through IPv6.

4.2 Implemented Software and Code

Presented below are some of the developed algorithms and some small explanations of how they work.

First, excerpts of the code used by the ESP8266 are presented. This algorithm was written in MicroPython and connects the NodeMCU to the accelerometer sensor and to the MQTT broker.

The second algorithm presented partly shows how the MQTT broker works. This algorithm is written in Python and was made with the purpose of creating a gateway between the microcontroller/sensor and the database.

4.2.1 ESP8266 Code

When this algorithm initiates, it creates an SPI object and initialises it with a serial clock of 1MHz, next the slave select pin is defined. Finally it connects the ESP8266 to a preset internet service provider (ISP) and to a preset MQTT broker, then it and retrieves the real world time for later use.

```
def do_connect():
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    if not wlan.isconnected():
        print('connecting to ' + ssid)
        wlan.connect(ssid, pwd)
        while not wlan.isconnected():
            pass
    print('network config:', wlan.ifconfig())
    settime()
    print('current time=%s' % (machine.RTC().datetime(),))
    client = MQTTClient(device_name, mqqt_server, mqqt_port)
    client.connect()
```

Figure 4.4: Function *do_connect()* from the ESP8266 algorithm

Data exchange between the sensor and the microcontroller is made through the *write_register()* and *read_register()* functions. If the NodeMCU is trying to send information to the sensor, the *write_register()* function is called. The *write_register()* function starts by setting the appropriate slave select pin to low. Then the ESP8266 needs to send the address of which register it wants to write into and

only then can it actually write into it. The function ends by resetting the slave select pin to high. If the NodeMCU needs to collect data from the sensor it calls the *read_register()* function.

```
def write_register( registerAddress, value):
# SPI Slave Select LOW
    PinSS.value(0)

    hspi.write(bytes([registerAddress]))

    hspi.write(bytes([value]))
# SPI Slave Select HIGH
    PinSS.value(1)



def read_register( registerAddress, numBytes, values):

###     Since this is the read operation, the most significant bit of
###     the register address should be set.
    address = 0x80 | registerAddress

###     Bit 6 also needs to be set when doing a multi-byte read.
    if numBytes > 1:
        address = address | 0x40

# SPI Slave Select LOW
    PinSS.value(0)

##      Transfer of the starting register address that needs to be read.
    hspi.write(bytes([address]));

    hspi.readinto(values)

## SPI Slave Select HIGH
    PinSS.value(1)
```

Figure 4.5: Function *read_register()* and *write_register()* from the ESP8266 algorithm

SPI communication has a few more particularities when it comes to retrieving data. The *read_register()* function starts by adding a 1 to the beginning of the message it wants to transmit, then the sixth bit of the message is also added if multi-byte read is requested. Finally, the proper slave select pin is set to low and then the altered message, that contains the address of the register from which to read, is sent. The function ends by retrieving the answer from the sensor and resetting the slave select pin to high.

The main function of this algorithm, and the one that is repeated every one hundred milliseconds is the *start_sampling()* function. This function starts by reading the values stored on all axis of the accelerometer. Then it compiles into a single string with, the name of the particular accelerometer, the acceleration

values and the current time. Lastly, the message is published to the broker.

```
def start_sampling():

    #client = MQTTClient(device_name, mqqt_server, mqqt_port)

    #client.connect()


    readRegister(DATAX0, 6, values)

    x = (values[1] << 8) | values[0]
    y = (values[3] << 8) | values[2]
    z = (values[5] << 8) | values[4]

    #f= str(x) + ';' + str(y) + ';' + str(z)

    #Note: message contents are the value in f and time in unix timestamp
    d= machine.RTC().datetime()

    s='{ "device":"' + str(device_name) + '", "valuex": ' + str(x) + '", "valuey": ' + str(y)
    s= s + '", "valuez": ' + str(z) + ', "time":"' + str(d)  + '"}'
    #s='{ "device":"' + str(device_name) + '","value":"' + f + '", "time":"' + str(d)  + '"}'
    #print(s)
    client.publish('/iotdemo/temp',s)
```

Figure 4.6: Function *start_sampling()* from the ESP8266 algorithm

This function was altered in order to lower the energy consumption of the system. Now the function first collects acceleration values during a second and then calculates the population variance of that sample. This way the algorithm can recognize when the bike is in use and therefore only send information to the broker once a second and when it senses movement.

4.2.2 Gateway Code

Message Q Telemetry Transport (MQTT) is a TCP based subscribe and publish messaging protocol, designed for lightweight machine to machine communications.

The MQTT broker serves as a gateway or server. This server allows clients to: connect to it, subscribe to topics and to send/receive short messages, but only if already subscribed to the desired topic.

This code creates an MQTT client and serves as an intermediary between the NodeMCU and the database. After successfully connecting to the broker and subscribing to the right topic, this algorithm loops forever, waiting for a message to be received. When a message arrives, the function *on_message()* is called. This

function reconverts the received message into string, formats the time and sends the slightly altered message to the database.

```
def on_message(client, userdata, msg):
    v=str(msg.payload.decode('utf8'))
    #string comes in as bytes so need to convert it
    sample=json.loads(v)

    t=str(sample['time'])
    t=t[1:]
    t=t[:-1]
    t=t.split(",")

    time_t=datetime.datetime(int(t[0]), int(t[1]), int(t[2]), int(t[4]), int(t[5]), int(t[6]), int(t[7]))
    timestamp_utc = calendar.timegm(time_t.timetuple())

    print('Processing sample : ' + sample['valuex'] +';'+ sample['valuey'] +';'+ sample['valuez'])

    body='{ "device":"' + str(sample['device']) + '", "sample_date" : "' + time_t.strftime("%Y-%m-%d")
    body = body + '", "valuex":"' + str(sample['valuex']) + '", "valuey":"' + str(sample['valuey'])
    body = body + '", "valuez":"' + str(sample['valuez']) + '", "time":"' + repr(timestamp_utc) + '" }'
    secret = b'MongoDB Stitch Webhook password here'
    hash = hmac.new(secret, body.encode("utf-8"), hashlib.sha256)
    url ='https://webhooks.mongodb-stitch.com/api/client/v2.0/app/connecteddevices-hyjxy/'
    url = url + 'service/iotreceivedata/incoming_webhook/savesensordata'
    #enter Stitch Web API URL here
    header={"Content-Type":"application/json","X-Hook-Signature":"sha256=" + hash.hexdigest()  }
    myResponse = requests.post(url,headers=header, data=body )
    print (myResponse.status_code)
```

Figure 4.7: Function *on_message()* from the Gateway.py algorithm

4.3 Problems and Solutions

The purpose of this work is to ultimately install multiple sensors in a single room and have each sensor work for several hours in a closed system, i.e., with their own battery supply. In this case, WiFi and IPv6 become unfeasible due to their high power demand. However LR-WPANs can be implemented in the system as a solution to this problem. Even though battery supplied IoT systems usually have a small range and are not powerfull enough to send data to the CLOUD, an MQTT broker can be installed close to the sensors and be used as an intermediary between the ESP8266 and the database.

An MQTT broker is perfectly capable of handling a large amount of sensors and can: Allow username and password authentication; Eliminate the impacts of a vulnerable or insecure client connection; Manage and track all client connection states, including security credentials and certificates; Is easily scalable; Doesn't compromise security, due to there being no connections between clients; And has a reduced network strain.

Chapter

5 Results

The main tests conducted during this theses are discussed in this chapter. The results of the study conducted to ascertain the best place to install the sensor on the bike are presented on the first section of this chapter. Next, are presented the results on the analysis made to figure out the best sampling rate to be used by this system. The third section of this chapter presents results on various different tests conducted to check how the application responded to noise, incorrect utilization and on different types gym machines. Finally the last section of this chapter presents the results of an analysis of bike utilization throughout different days.

5.1 Sensor Position

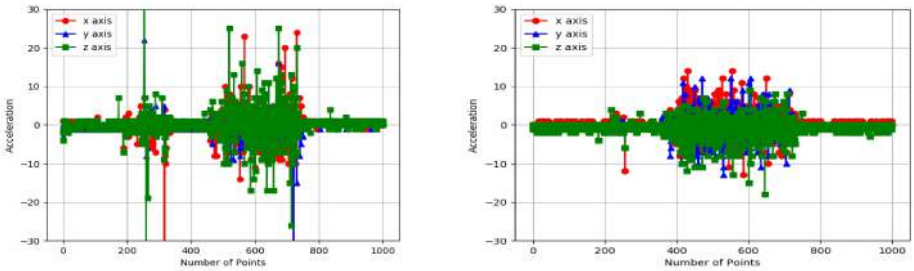
Before any real analysis to movement in a stationary bike could be made, there was a necessity to study and understand where was the best place to install the accelerometer sensor. Various measurements were made with the accelerometer sensor installed in different places, such as the bike's handlebar (5.6a), the seat (5.6b), the front of the bike below the handlebar (5.1c) and on the back of the bike below the seat (5.1d).

Three types of tests were made with each implementation in order to ascertain which provided the most reliable and precise results. The first tests that were

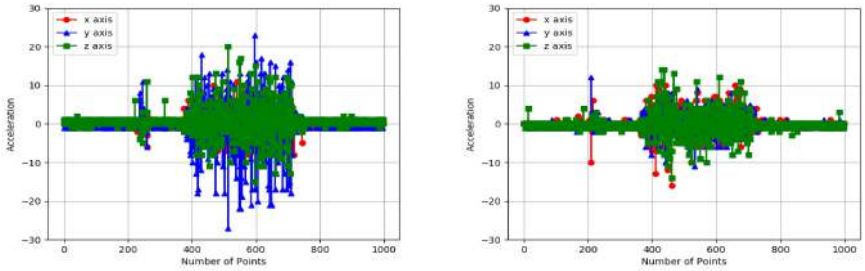
(a) (b) (c) (d)

Figure 5.1: Different installations of the accelerometer

conducted, served to measure if the sensor could easily distinguish if someone was using the bike or if it was in rest. This test was made by making various measurements of one hundred seconds each. During that time a test subject would wait for about twenty seconds and then climb onto the bike, next the subject would wait for another fifteen seconds, about thirty five seconds had passed since the beginning of the test, and only then would the subject start pedaling. This would go on for about another forty seconds and then the subject would quickly get out of the bike, about seventy five seconds after the beginning of the test. Graphs that show some of the results of this experiment are presented below.



(a) Test on the handlebar (b) Test on the seat

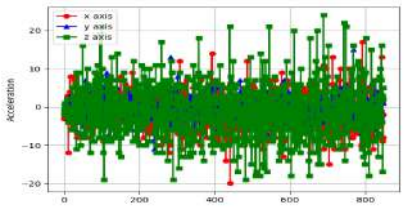


(c) Test on the front of the bike (d) Test on the back of the bike

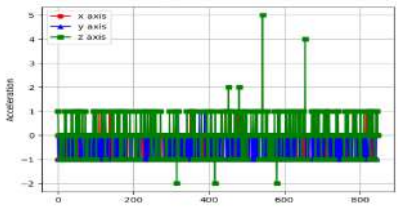
Figure 5.2: Results of the first test in different installations

As can be observed in figure 5.2, all installations present an increase in amplitude of signal when the test subject either gets on the bike, or when is pedaling. This proves that all installations seem reliable for movement classification, although the results seem to better confirm the existence of movement when the sensor is attached to the handlebar.

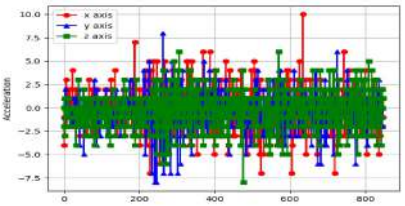
The tests that were conducted next, only measured when the bike was either being used or when it was in rest. These tests had the purpose of calculating the standard deviation of both states, in all implementations and compare the two.



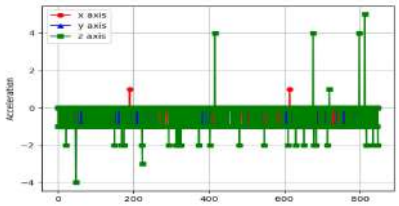
(a) Movement on the handlebar



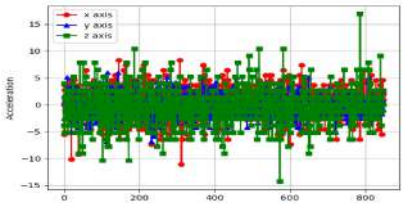
(b) Noise on the handlebar



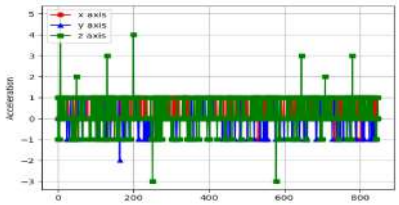
(c) Movement on the seat



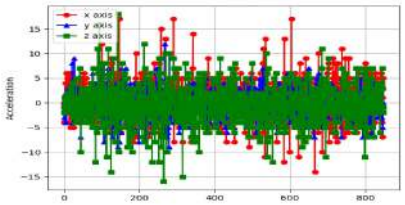
(d) Noise on the seat



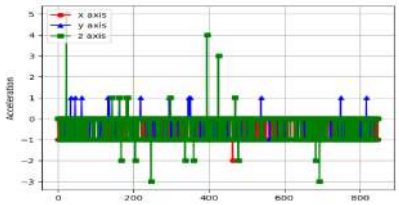
(e) Movement on the front of the bike



(f) Noise on the front of the bike



(g) Movement on the back of the bike



(h) Noise on the back of the bike

Figure 5.3: Movement vs noise for all implementations

The graphs that are shown above depict some of the collected results and are presented in different scales to facilitate observation. As can be viewed in figure 5.3, the graphs that were collected when the bike was in use [(a),(c),(e),(g)] display erratic results with a high amplitude of signal, while the ones that show the results of when the bike was in rest [(b),(d),(f),(h)], display a much more steady set of results. This is reflected on the standard deviation of each group of results.

The gathered values were studied, by axis, and a standard deviation was calculated for each test, the mean values of these results are shown in table 5.1. To compare the sensibility between installations, the mean value of the difference between the mean standard values of the bike in use and in rest, were calculated using all axis' values. This value, that comes from the calculated standard deviation, is going to be referred to as sensitivity gauge, and is going to be used from here on to compare the sensibility between installations.

As can be concluded from table 5.1 the most sensitive installation is the handlebar.

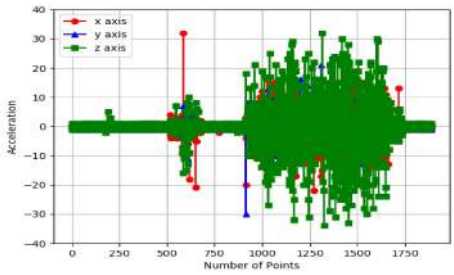
			Axis		
			X	Y	Z
Mean values of standard deviation	Handlebar	Stationary bike in use	3.925	2.652	6.428
		Stationary bike in rest	0.440	0.470	0.573
		Difference between bike in use and rest	3.485	2.182	5.855
		Sensitivity gauge		3.841	
	Seat	Stationary bike in use	2.719	2.370	2.090
		Stationary bike in rest	0.441	0.364	0.640
		Difference between bike in use and rest	2.278	2.007	1.450
		Sensitivity gauge		1.912	
	Front	Stationary bike in use	2.930	2.039	3.906
		Stationary bike in rest	0.481	0.403	0.603
		Difference between bike in use and rest	2.449	1.636	3.302
		Sensitivity gauge		2.462	
	Behind	Stationary bike in use	3.168	2.399	3.004
		Stationary bike in rest	0.484	0.391	0.559
		Difference between bike in use and rest	2.684	2.008	2.446
		Sensitivity gauge		2.379	

Table 5.1: Table with results on the sensibility of the different installations

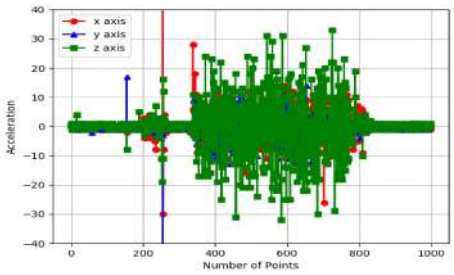
5.2 Sampling Rates

As not much information regarding the most commonly adopted sampling rate for applications that work with wireless IoT devices was found in the literature, a small analyses of this problem had to be made.

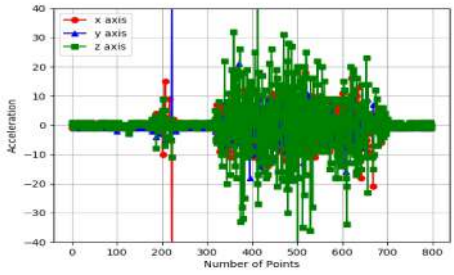
When battery life and data storage are limited resources, the choice of which sampling rate to use becomes much more important. If a high sampling rate is chosen then, a high battery potency and a considerable data storage capability will also be needed. If the chosen sampling rate is to small, then the collected values might be missing important pieces of information.



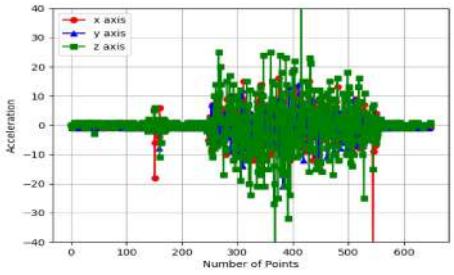
(a) Time between each Point is 40ms



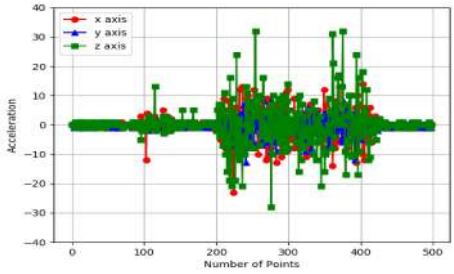
(b) Time between each Point is 80ms



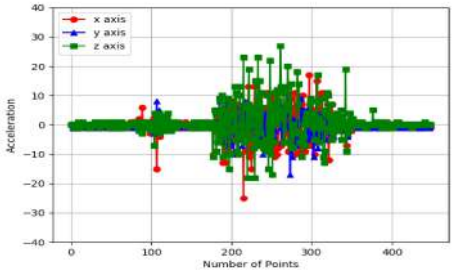
(c) Time between each Point is 100ms



(d) Time between each Point is 120ms



(e) Time between each Point is 160ms



(f) Time between each Point is 200ms

Figure 5.4: Different tests to study sampling rates in this application

In order to study which sampling rate was best suited for this particular application, various tests were made with the accelerometer installed in the handlebar. These tests were conducted similarly to the first tests of the previous chapter. Initially a test subject would wait for about twenty seconds and then he/she would climb into the bike. After approximately thirty seconds had passed since the beginning of the test, the subject would start pedaling, and after about seventy seconds had passed since the beginning of the test, the test subject would quickly get out of the bike.

As can be seen in figure 5.4 these tests were repeated with sampling periods of 40, 80, 100, 120, 160 and 200 milliseconds. These sampling periods are all lower than what most modern human activity recognition applications use, and were chosen due to the low level of activity recognition needed for the application. As this application only needs to recognise if a stationary bike, or other gym equipment, is being used at any time, a higher sampling rate would only indulge some of the previously discussed detriments to the system.

The sampling period that was chosen as the best out of these options was the 100 millisecond. This choice was made because, the lower sampling periods showed a smaller standard deviation when the bike was in use, and also showed a smaller reaction, in amplitude of signal, to when the test subject climbed into the bike.

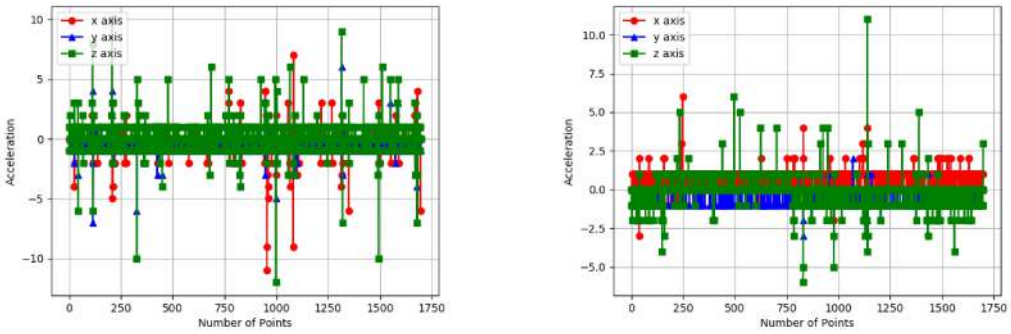
From here on all presented graphs will be presented with a sampling period of 100 milliseconds.

5.3 Tests

In this section, various tests were made to examine the reliability and versatility of the application.

The first tests served to check if the system was susceptive to false positives when someone leaned on the bike, two different tests were made to study this case. In the first, a test subject would lean on the handlebar while the sensor was installed there, and in the second, the test subject would lean on the seat of the

bike while the sensor was installed on the seat. The results of these tests can be seen in figure 5.5 where its possible to see that the system appears to be in rest but suffers from a lot of noise, particularly in the X and Z axis.



(a) The handlebar (b) The seat

Figure 5.5: Noise when someone is leaning on:

After some calculations of the results mentioned above, table 5.2 was made. From this table it's possible to conclude that the effect of leaning against the machine won't affect the usage classification.

			X	Y	Z
Mean Values	Handlebar	Leaning on stationary bike	0.922	0.6877	1.083
		Stationary bike in rest	0.440	0.470	0.573
		Difference between leaning on bike and rest	0.482	0.217	0.510
		Sensitivity gauge		0.403	
	Seat	Leaning on stationary bike	0.626	0.412	0.782
		Stationary bike in rest	0.441	0.364	0.640
		Difference between leaning on bike and rest	0.185	0.048	0.142
		Sensitivity gauge		0.125	

Table 5.2: Table with results on the sensibility of when someone leans on the bike

The next set of tests studied the effect of one person just sitting on the bike, without actually using it, as before these tests were made for both handlebar and seat installations and the results are shown in table 5.3. These results show that neither of these installations are capable of detecting seat hogging.

The subsequent tests were made to study if the effect of using the bike while not touching the handlebar had any effect on the usage classification. By analysing table 5.4 it's possible to conclude that the application has no problems in detecting

			X	Y	Z
Mean Values	Handlebar	Sitting on stationary bike	0.439	0.481	0.654
		Stationary bike in rest	0.440	0.470	0.573
		Difference between sitting and bike in rest	-0.001	0.012	0.081
		Sensitivity gauge	0.031		
	Seat	Sitting on stationary bike	0.513	0.388	0.796
		Stationary bike in rest	0.441	0.364	0.640
		Difference between seating and bike in rest	0.072	0.024	0.157
		Sensitivity gauge	0.084		

Table 5.3: Table with results on the sensibility of seat hogging

if the bike is being used, even if the user is not touching the handlebar. This application can also detect if someone is using the bike even if they are pedaling without touching the seat, the test that was conducted to study this case was made when the sensor was installed in the seat. The sensitivity gauge in this particular case was 2.653, which is about the same as the values measured with regular bike usage, which proves that the application also has no problems in identifying movement when the user is not touching the seat.

			X	Y	Z
Mean Values	Handlebar	Using the bike with no hands	6.342	4.267	9.400
		Stationary bike in rest	0.440	0.470	0.573
		Difference between bike in use and rest	5.902	3.797	8.827
		Sensitivity gauge	6.175		
	Seat	Using the bike with no hands	2.487	2.212	2.083
		Stationary bike in rest	0.441	0.364	0.640
		Difference between bike in use and rest	2.046	1.848	1.443
		Sensitivity gauge	1.779		

Table 5.4: Table with results on the sensibility of when the bike is misused

Tests were also made to ensure that this application was viable with other exercise machines. Various tests were made with an elliptical bike and the results are summarized in table 5.5. With the elliptical bike, only two different installation methods were studied, pictures of these installations can be seen in figure 5.6.

From table 5.5 it's possible to conclude that this application can also work well on an elliptical bike, but only if the sensor is installed on the screen of the bike.

(a) Sensor Installed on Bike Screen

(b) Sensor Installed on Back of the Bike

Figure 5.6: Different installations of the accelerometer on the elliptical bike

			X	Y	Z
Mean Values	Screen	Elliptical bike in use	3.763	2.707	3.307
		Elliptical bike in rest	0.478	0.480	0.621
		Difference between bike in use and bike in rest	3.285	2.227	2.686
		Sensitivity gauge		2.733	
	Back	Elliptical bike in use	1.938	1.702	1.661
		Elliptical bike in rest	0.404	0.503	0.583
		Difference between bike in use and rest	1.534	1.199	1.078
		Sensitivity gauge		1.271	

Table 5.5: Table with results on the sensibility of an elliptical bike

5.4 Over Time Analysis

The last set of tests made in this work were conducted in a stationary bike, over several hours during three different days. Graphs showing bike utilization during these days are presented in figure 5.7.

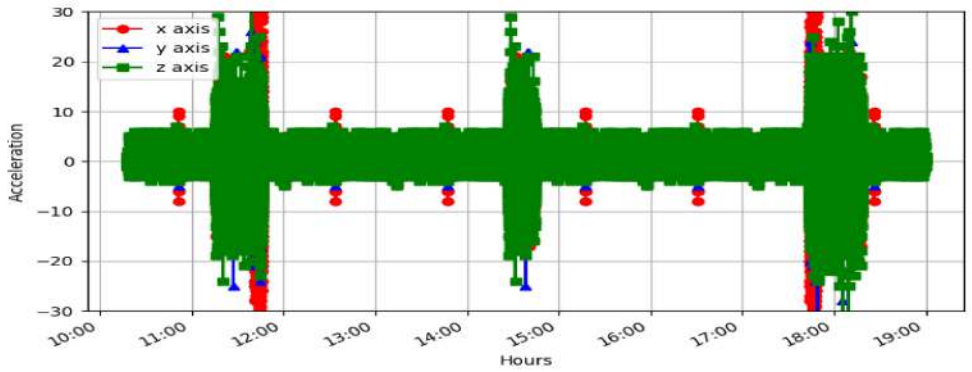
In the first day the bike was barely used (only three times). A total of eight hours and forty one minutes (8h:41m) was recorded during this day and only one hour and twenty minutes (1h:20m) of this time were spent pedaling.

In the second day the bike was used only four times, but the overall length of time spent pedaling was much longer, a total of one hour and fifty seven minutes (1h:57m) were spent pedaling, against a total of 5 hours and 37 minutes (5h:37m) spent recording.

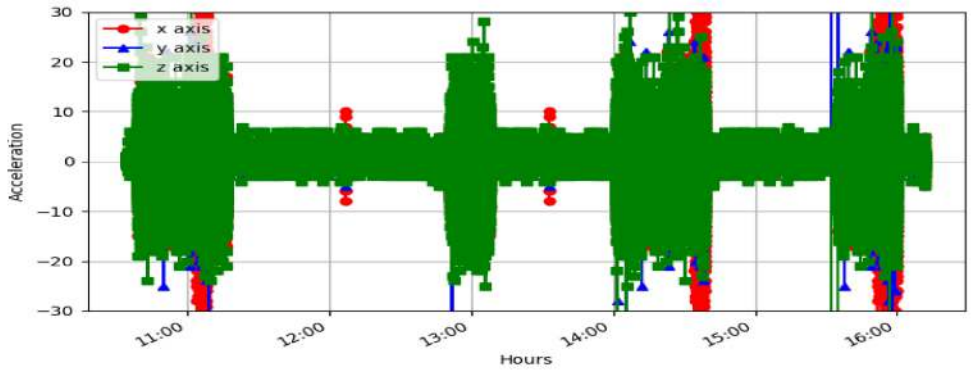
In the final day the bike was used much more frequently but for shorter amounts of time. In total, the bike was used for one hour and 53 minutes (1h:53m)

of the seven hours and thirty five minutes (7h:53m) recorded.

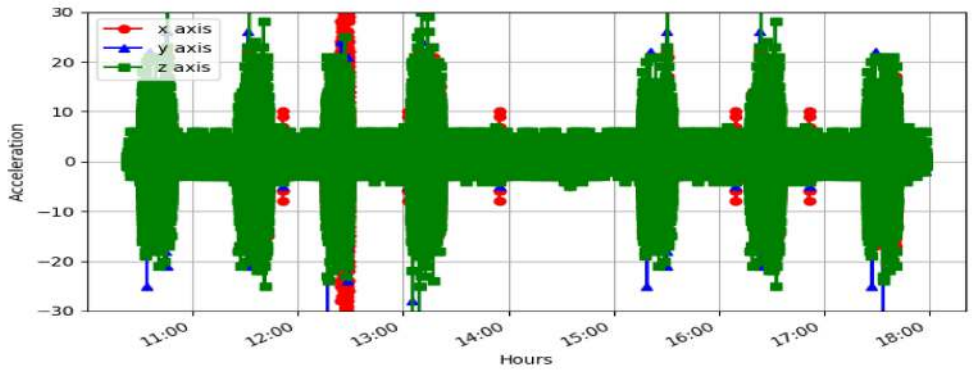
Summing up the three days recorded, the bike was in use for a total of five hours and ten minutes of the total twenty one hours and fifty three minutes (21h:53m) recorded.



(a) Data collected on 26/2/2020



(b) Data collected on 27/2/2020



(c) Data collected on 28/2/2020

Figure 5.7: Full day data collection

Chapter

6 Conclusion

The present work managed to create a system that is able to monitor occupancy in gym machines, and store all collected data on a remote CLOUD-based database. During this work various tests were made to test different implementations of the system on a stationary and on an elliptical bike. From these tests, it was concluded that the best place for the sensor to be installed is on the handlebar and that the best sampling period for collecting data was one hundred milliseconds (100ms) for this application in specific. Some other tests were also conducted in order to ascertain the sensibility and the accuracy of the application. From these it was concluded that the application is successfully able to block the influence of noise and is able to detect movement even if the bike is being used incorrectly. However it cannot distinguish between a person just sitting still on the bike and an empty seat. A small analysis of the overall bike utilization during several hours was also successfully conducted and the results show that the application suffers no apparent detriments even if used for prolonged periods of time.

6.1 Future Work

The natural evolution of this work is to leave the system in an autonomous state, which can be achieved with the integration of a battery. As previously mentioned, the main problem with this,From table 5.5 it's possible to conclude that this application can also work well on an elliptical bike, but only if the sensor is installed on the screen of the bike.

is the use of WiFi and the high power demanded by this protocol. As future work, the implementation of a low power communication protocol is proposed, such as Zigbee.

A change to the algorithm, implemented in the microprocessor that works with the sensor, is also proposed. This algorithm can save even more battery power if it is able to store some of the gathered data, analyse it and only send information to the database periodically and when movement is detected.

Bibliography

[1] https://www.trossenrobotics.com/c/robot-force-sensor-fsr.aspx. (Accessed in August 2019).

[2] https://www.excelr.comskewness-and-kurtosis. (Accessed in February 2020).

[3] http://www.ieee802.org/15/pub/TG4.html/. (Accessed in July 2019).

[4] https://www.olimex.com/Products/IoT/ESP8266/MOD-WIFI-ESP8266-DEV/open-source-hardware/. (Accessed in July 2019).

[5] https://www.amazon.co.uk/DFRobot-Triple-Axis-Accelerometer-ADXL345/dp/B00G3IJVXU/. (Accessed in July 2019).

[6] M. Abomhara et al. "Cyber security and the internet of things: vulnerabilities, threats, intruders and attacks." In: *Journal of Cyber Security and Mobility* 4.1 (2015), pp. 65–88.

[7] A. Avci, S. Bosch, M. Marin-Perianu, R. Marin-Perianu, and P. Havinga. "Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey." In: *23th International conference on architecture of computing systems 2010*. VDE. 2010, pp. 1–10.

[8] J. Baek, G. Lee, W. Park, and B.-J. Yun. "Accelerometer signal processing for user activity detection." In: *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer. 2004, pp. 610–617.

[9] A. Braun, M. Majewski, R. Wichert, and A. Kuijper. "Investigating low-cost wireless occupancy sensors for beds." In: *International Conference on Distributed, Ambient, and Pervasive Interactions*. Springer. 2016, pp. 26–34.

[10] T. DiCola. *MPR121 Capacitive Touch Sensor on Raspberry Pi & BeagleBone Black*. https://learn.adafruit.com/mpr121-capacitive-touch-sensor-on-raspberry-pi-and-beaglebone-black/hardware. (Accessed in August 2019).

[11] E. Fadel, V. C. Gungor, L. Nassef, N. Akkari, M. A. Malik, S. Almasri, and I. F. Akyildiz. "A survey on wireless sensor networks for smart grid." In: *Computer Communications* 71 (2015), pp. 22–33.

[12] V. C. Gungor and G. P. Hancke. "Industrial wireless sensor networks: Challenges, design principles, and technical approaches." In: *IEEE Transactions on industrial electronics* 56.10 (2009), pp. 4258–4265.

[13] N. Y. Hammerla, J. Fisher, P. Andras, L. Rochester, R. Walker, and T. Plötz. "PD disease state assessment in naturalistic environments using deep learning." In: *Twenty-Ninth AAAI conference on artificial intelligence*. 2015.

[14] J. Hoey, T. Plötz, D. Jackson, A. Monk, C. Pham, and P. Olivier. "Rapid specification and automated generation of prompting systems to assist people with dementia." In: *Pervasive and Mobile Computing* 7.3 (2011), pp. 299–318.

[15] A. J. Jerri. "The Shannon sampling theorem—Its various extensions and applications: A tutorial review." In: *Proceedings of the IEEE* 65.11 (1977), pp. 1565–1596.

[16] Q. Jing, A. V. Vasilakos, J. Wan, J. Lu, and D. Qiu. "Security of the Internet of Things: perspectives and challenges." In: *Wireless Networks* 20.8 (2014), pp. 2481–2501.

[17] A. Khan, N. Hammerla, S. Mellor, and T. Plötz. "Optimising sampling rates for accelerometer-based human activity recognition." In: *Pattern Recognition Letters* 73 (2016), pp. 33–40.

[18] D. Kim, O. Hilliges, S. Izadi, A. D. Butler, J. Chen, I. Oikonomidis, and P. Olivier. "Digits: freehand 3D interactions anywhere using a wrist-worn gloveless sensor." In: *Proceedings of the 25th annual ACM symposium on User interface software and technology*. 2012, pp. 167–176.

[19] T. Labeodan, K. Aduda, W. Zeiler, and F. Hoving. "Experimental evaluation of the performance of chair sensors in an office space for occupancy detection and occupancy-driven control." In: *Energy and Buildings* 111 (2016), pp. 195–206.

[20] T. Labeodan, C. De Bakker, A. Rosemann, and W. Zeiler. "On the application of wireless sensors and actuators network in existing buildings for occupancy detection and occupancy-driven lighting control." In: *Energy and Buildings* 127 (2016), pp. 75–83.

[21] C. Marcroft, A. Khan, N. D. Embleton, M. Trenell, and T. Plötz. "Movement recognition technology as a method of assessing spontaneous general movements in high risk infants." In: *Frontiers in neurology* 5 (2015), p. 284.

[22] H. H. Nguyen, N. Gulati, Y. Lee, and R. K. Balan. "Real-time detection of seat occupancy & hogging." In: *Proceedings of the 2015 International Workshop on Internet of Things towards Applications*. ACM. 2015, pp. 29–34.

[23] T. Plötz, N. Y. Hammerla, A. Rozga, A. Reavis, N. Call, and G. D. Abowd. "Automatic assessment of problem behavior in individuals with developmental disabilities." In: *Proceedings of the 2012 ACM conference on ubiquitous computing*. 2012, pp. 391–400.

[24] A. Sehgal, V. Perelman, S. Kuryla, and J. Schonwalder. "Management of resource constrained devices in the internet of things." In: *IEEE Communications Magazine* 50.12 (2012), pp. 144–149.

[25] *Siemens's MindSphere is and IIoT application*. https://siemens.mindsphere.io/en. (Accessed in February 2020).

[26] V. S. Soto, I. Muller, J. M. Winter, C. E. Pereira, and J. C. Netto. "Control over wirelesshart network through a host application: A wirelesshart network control proposal." In: *2014 Brazilian Symposium on Computing Systems Engineering*. IEEE. 2014, pp. 91–96.

[27] J. M. C. Tavares. "Internet of Things: security and organization." Doctoral dissertation. 2015.

[28] P. Through. *The LightBlue Bean*. https://punchthrough.com/bean/. (Accessed in June 2019).

[29] J. Titus. https://www.edn.com/electronics-blogs/dev-monkey-blog/4408668/What-the-Heck-is-Wavenis-/. (Accessed in July 2019).

[30] J Tripathi, J. De Oliveira, and J. Vasseur. "Applicability study of RPL with local repair in smart grid substation networks." In: *2010 First IEEE International Conference on Smart Grid Communications*. IEEE. 2010, pp. 262–267.

[31] G. Tuna, V. C. Gungor, and K. Gulez. "Wireless sensor networks for smart grid applications: a case study on link reliability and node lifetime evaluations in power distribution systems." In: *International Journal of Distributed Sensor Networks* 9.2 (2013), p. 796248.

[32] K. Van Laerhoven and A. K. Aronsen. "Memorizing what you did last week: Towards detailed actigraphy with a wearable sensor." In: *27th International Conference on Distributed Computing Systems Workshops (ICDCSW'07)*. IEEE. 2007, pp. 47–47.

[33] Z. Yang, Y. Yue, Y. Yang, Y. Peng, X. Wang, and W. Liu. "Study and application on the architecture and key technologies for IOT." In: *2011 International Conference on Multimedia Technology*. IEEE. 2011, pp. 747–751.

[34] N. Ye, Y. Zhu, R.-c. Wang, R. Malekian, and Q.-m. Lin. "An efficient authentication and access control scheme for perception layer of internet of things." In: (2014).

[35] K. Zhao and L. Ge. "A Survey on the Internet of Things Security." In: *2013 Ninth International Conference on Computational Intelligence and Security*. 2013, pp. 663–667. DOI: 10.1109/CIS.2013.145.