REMOTE MONITORING OF HEALTH PARAMETERS

USING SMART WALKERS

PATRICK LEWANDOWSKI, 201802946
HARSHIT MAHAPATRA, 201803187
TOMAS MANUEL REBELO MOTA, 201802941

PROJECT REPORT - BUILDING THE INTERNET OF THINGS WITH P2P AND CLOUD COMPUTING
December 2018

Advisor: Niels Olof Bouvin

AARHUS UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

REMOTE MONITORING OF HEALTH PARAMETERS

PATRICK LEWANDOWSKI
HARSHIT MAHAPATRA
TOMAS MANUEL REBELO MOTA

Using smart walkers

Project Report - Building the Internet of Things with P2P and Cloud Computing
Department of Computer Science
Science & Technology
Aarhus University

December 2018

Patrick Lewandowski, Harshit Mahapatra & Tomas Manuel Rebelo Mota: *Remote Monitoring of Health Parameters,* Project Report - Building the Internet of Things with P2P and Cloud Computing , © December 2018

We would first like to thank the Head of Health IT Lab at the Alexandra Institutet Morten Kyng and our Professor Niels Olof Bouvin who is a member of the Participatory Information Technology department at the Aarhus University. Both gentlemen made this project in several ways possible.

The door to Professor Olof's and Professor Doctor Morten Kyng's office was always open whenever we had problems or a question about our research or writing. They consistently gave us different perspectives and helped in creating this final report which states the final results of our research.

We would also like to thank the experts that we have consulted alont the way, namely Orbit Lab and ChomskyLab. Without their passionate participation and input, we would not have had the resources to make the project as successful.

ABSTRACT

Nowadays Healthcare is a field of growing interest for data collection, as the remote monitoring of patients not only allows us to reduce our response time for emergencies but also gives us valuable insights into their health, which in turn leads to better diagnosis regarding disabilities or diseases.

In this project we want to build a proof of concept that realizes the above ideas. We will introduce our approach and explain how we managed to get data out of a walker user, building the foundation for analyzing it in the future.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

Part I

THE LORAWAN WALKER

The following part reflects our thoughts and time we invested in this project, in both the theoretical and practical side.

FOREWORD

The percentage of danish citizens over 65 years old has risen from 15% to 20% in the last 10 years and the danish government is trying to find new ways of adapting to this aging population, one of them being investing in health/it related research.

We wanted to help and approached the Alexandra Instituttet, asking if they had any IoT projects that could also involve bioinformatics and machine learning. Morten Kyng, head of Health IT Lab, already had in mind the idea of building a walker which could track it's users heath parameters, so a doctor could inspect these. The projects objective is the long term monitoring of elderly people using walkers.

1 INTRODUCTION

Recent advances in sensor technology have made it possible to embed sensors in everyday objects to collect data related to various parameters. With the advent of low power network technologies like LoRaWAN, it is possible to create a sensor network which requires little power and low maintenance.

Furthermore, current developments in the field of machine learning and big data have made it possible to acquire valuable insights from the collected data and have thus made its acquisition even more lucrative.

Healthcare is a field of growing interest for this data collection, as the remote monitoring of patients not only allows us to reduce our response time for emergencies but also gives us valuable information regarding their health, which in turn leads to better diagnosis and insight about a disability or disease.

Our project consists of building a smart walker which leverages the low power networking capabilities of the LoRaWAN protocol, decentralized computing in the cloud and the power of data analysis to monitor and analyse the health parameters of its user.

More concretely, we aim to track usage statistics, the position and common usage area of walkers, and monitor some health parameters of the user.

The objective of our project is thus to test the following hypothesis: "Is it practical to monitor walkers and their users by sending data over LoRaWAN?". We can evaluate the practicality of the this system by analysing the following properties:

1. Reliability: We define a system to be reliable if it has a low probability of failure.
2. Accuracy: We defience a system to be accurate if its output has low deviation with respect to the true value of a given measurement.
3. Consistency: We define a system to be consistent if it behaves the same way over a period of time in response to a given action.
4. Durability: We define a system to be reliable if it has a long operational life.
5. Modularity: We define a system to be modular if it is relatively easy to add or remove components from it.

6. Scalability: We define a system to be scalable based on how well it performs in case of increase in the number of users.
7. Servicability: We define a system to be servicable if it relatively easy to diagnose and fix errors occuring in the system.
8. Latency: Latency is how long it takes to send a piece of information.
9. Throughput: Throughput is how much information can be sent at once.
10. Energy: How much energy the system consumes.

This project was authored by the group as a whole, and a short demonstration can be seen at https://vimeo.com/304645369

2 RELATED WORK

This section briefly covers the relevant related work. At first we have a look at comprehensive surveys about using Internet of Things for healthcare and mobile health monitoring systems. We then look at an example architecture which uses Arduino and Raspberry Pi for monitoring of environment using WPAN. We then move on to a system which uses the same components but uses LoRaWAN protocol instead of WPAN. We then briefly take a diversion to give a brief summary about LoRaWAN, its working and security. This is followed by a description of recent implementation of smart walker. Finally we have a look at studies describing some of the applications of collected data.

A comprehensive survey on the application of the Internet of Things for healthcare was done by Islam et al. [10]. The paper presented the current IoT healthcare networks. It discussed the topology, architecture, and platform of the said networks. It also presented the current security models used in healthcare networks and proposed a novel security model for the same. Finally, the paper briefly presented some of the IoT healthcare technologies such as grid computing, big data, ambient intelligence, etc.

Alahmadi et al. [1] in their paper discussed in detail about the architecture of mobile e-health monitoring systems. They proposed a decentralized architecture, where the sensors are connected to a mobile device via Bluetooth, which in turn is connected to a server over the internet. They also presented a Wide Area Network architecture in which the mobile device is connected to a base station, which in turn is connected to a server over the internet. This server can then be accessed by the healthcare professionals to provide proper advice in case of unusual biosignals.

Ferdoush et al. [7] have done interesting work on the application of Wireless Personal Area Network (WPAN) for environment monitoring. They use Arduino Uno as their sensor node and the Raspberry Pi as their gateway. The Raspberry Pi itself acts as a hub containing the gateway application, database server, PHP web application, web server, and HTML web interface.

Mdhaffar et al. [11] have used a hardware setup similar to Ferdoush but with LoRaWAN to monitor healthcare. They collected sensor data using Arduino and sent it to a Raspberry Pi acting as the gateway using LoRaWAN. The Pi then forwards the data to a network server which is connected to an application server. They collected data on blood pressure, glucose level and temperature and were able to successfully monitor the said data. They were also able to get a large coverage area with low power consumption.

A survey on the LoRaWAN technology was presented by Silva et al. [4]. The paper presents the advantages of LoRaWAN over GSM architecture. This includes low power consumption, low maintenance, cheap hardware, and well-defined security measures. It then also presents some of the disadvantages of LoRaWAN such as low bandwidth, slow transmission rate and lack of established infrastructure. It then presents the architecture of a typical LoRaWAN network consisting of sensor nodes, gateway, network server and web application. This is followed by a brief presentation of the two-layer security of LoRaWAN networks in the network and application layer respectively.

Aras et al. [2] in their paper present a summary of the security and vulnerabilities of the LoRa network. The paper analyses the security risks present in the physical layer, network layer, join procedure and end gateways of LoRaWAN. The physical layer of LoRaWAN uses chip spread spectrum in which longer than usual transmission can either be corrupted or intercepted. In the network layer the payloads before and after encryption have the same length and thus can be used together with overflowing counters to restore the keystream. LoRaWAN uses two joining procedures Over The Air Activation (OTAA) and Activation By Personalization (ABP). The paper claims that while OTAA is secure, ABP's key can be compromised.

The paper also presents various possible attacks such as compromising the device and network keys via physical access, jamming using dedicated hardware, replay and wormhole attacks.

Another interesting paper about the security of LoRaWAN was presented by Tomasin et al. [16]. It presents a detailed analysis of attacks on the join procedure of LoRaWAN.

Although several attempts have been made to create a smart walker, the most similar one to our project was done by Postolache et al. [14]. In their project they connected a MEMS accelerometer, Doppler radar and flexible force sensor to an Arduino fixed to a walker. The Arduino then sent the sensor data to an Android device via Bluetooth. The mobile device contained an application to monitor and analyse the data. This approach is significantly different from ours, as our project relies on the cloud for monitoring and analysis of collected data. Furthermore, in our project, the walker is connected to a LoRaWAN gateway directly as we do not expect the walker users to always carry a smartphone, and we aim to achieve remote monitoring with a range far greater than that of Bluetooth.

Banaee et al. [3] in their paper present the various use cases for data collected through health monitoring systems. They propose that the data can be used for anomaly detection, decision making and sending an alarm to authorities in case of emergencies. They further discuss the architecture for mining the said and the use of various popular machine learning and statistical algorithms for the same. They also present a brief summary of the various types of acquired data and their properties.

Gondalia et al.[8] recently presented an interesting paper on the remote health tracking of soldiers in battlefield using IoT. They propose an architecture in which each soldier has a sensor node which is connected to a base node belonging to squadron leader. The connection between the base node and sensor node is via Wireless Body Area Sensor Networks (WBASN). The squadron leader's node, in turn, is connected to a control room via LoRaWAN. The control room contains access to the cloud which is then used for storage and analysis. They further propose that the collected data can be fed to a K-Means clustering algorithm to classify sensor status at different events such as sitting, walking, sleeping, etc.

3 ANALYSIS

Based on our study, we found that the Internet of Things can be used for healthcare with mobile sensors and the LoRaWAN protocol, which is made viable due to features such as security, high range, low cost, low power consumption and compatibility with cloud architecture. We also found that work has been done on building smart walkers which use bluetooth technology.

In [7] the authors proposed an architecture with a sensor and server to monitor remote environment with an Arduino acting as a node and a Raspberry Pi acting as a server. This system is easy to set up and the wide availability of suggested hardware makes it even more attractive. However, the system fails to leverage the cloud architecture and is limited by the computing capability of Pi, hence we in our project decided to use Pi as a gateway to connect to a server instead of the Pi acting as a server itself.

In [14] the authors built a smart walker which uses bluetooth to connect to a nearby mobile phone and use it for analysis and computation of data. This approach has several demerits, such as the range of bluetooth, the inaccurate assumption that walker users will always have a smartphone with them and the system being dependent on the computational power of the smartphone. Moreover, we need to also take into consideration the energy consumption of both the sensors on the walker and that of the smartphone.

Given that the LoRaWAN protocol overcomes exactly these demerits, it makes sense to build a smart walker which uses LoRaWAN to send data to a server. This is exactly what we want to test with our hypothesis.

We also decided to collect the following information from the sensors based on previous work [14] and our requirements:

3.0.0.1 *Location tracking*

This allows us to analyse how sedentary this user is and take preventive action. The collected data can also be used for tracking what kind of areas people using walkers are more likely to frequent, getting insights into what areas are good for walker users and what areas might be worth investing in more infrastructure.

Location tracking will also allow us to know quite precisely how often the walker is being used, making it possible to, for example, analyse what kind of patients are more dependent on the walker and also allows the manufacturers to match usage statistics to how long the walker lasts.

3.0.0.2 *Heart rate*

The heart rate variability, maximum, minimum and average, can give us great insights into the user's health over time, while the heart rate allows us to compare how much strain different walkers put on the user.

3.0.0.3 *Pressure applied by the hands*

This allows us to track how much the user relies on the walker and how his/her grip strength changes over time.

3.0.0.4 *Movement tracking*

This allows us to measure the speed at which the user is walking. If we track both handles independently, together with the pressure measurements, we can in a crude way analyse gait of the user.

4 DESIGN AND METHODOLOGY

Based on our analysis and the information we want to collect, we decided on the following architecture for our system [1]:

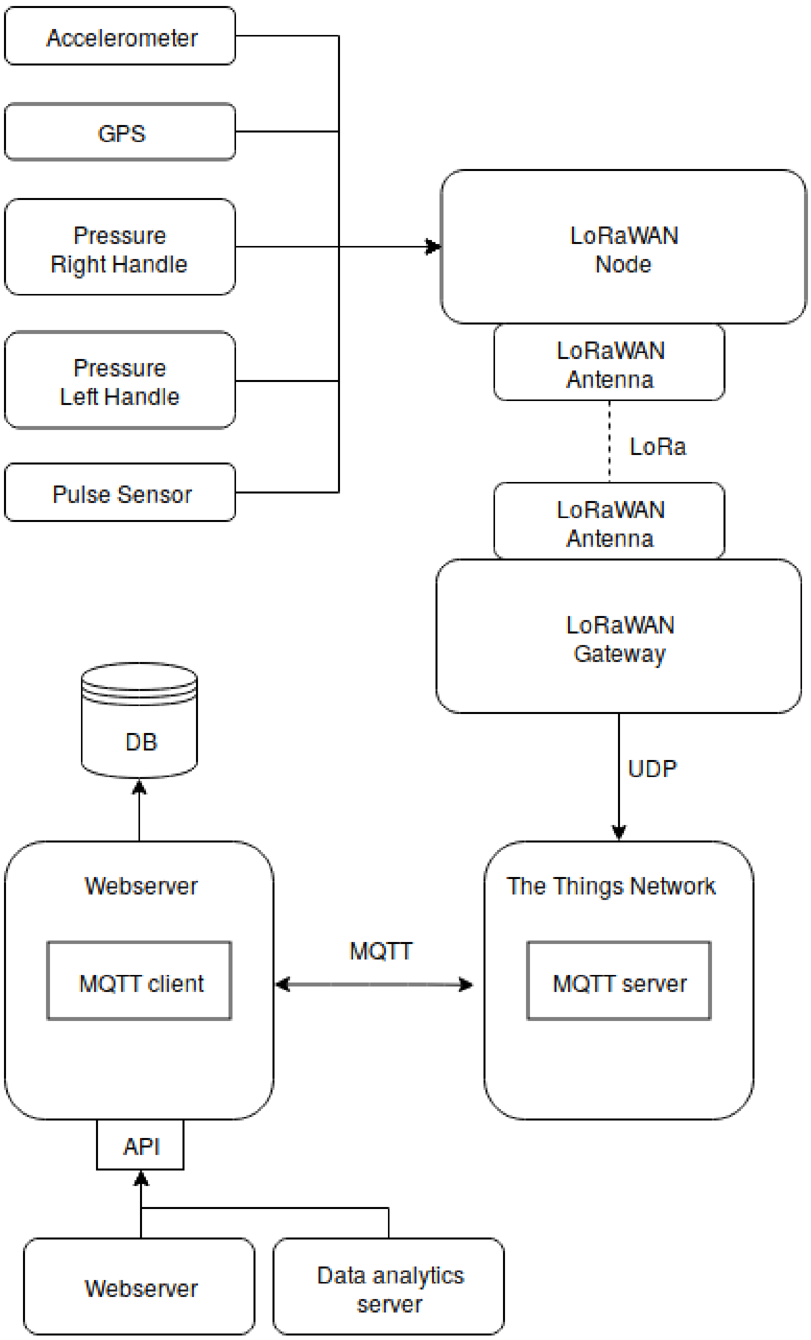

Figure 1: Proposed architecture

4.1 COMPONENTS AND THEIR INTERACTIONS

4.1.1 *LoRaWAN node*

Polls the sensors in predefined intervals and sends them through LoRaWAN every few minutes

4.1.2 *LoRaWAN gateway*

Receives LoRaWAN packets and relays them to "The Things Network" through the Semtech UDP protocol

4.1.3 *The Things Network (TTN)*

It is possible to have our gateway communicate directly with our server, but we decided to use TTN because it provides some very useful features out of the box, some of them being:

- Node authentication
- Packet deserialization
- Integration with MQTT, HTTP, Google Cloud, Amazon AWS and Azure
- Encryption
- Has thousands of registered gateways

TTN has become the standard in LoRaWAN applications, due to its unparalleled potencial for scalability, integration and security. In our case it deserializes the packets coming from the gateways and publishes them in an MQTT broker

4.1.4 *Server*

The server subscribes to the MQTT broker on TTN and stores the information received from there, while also exposing an API that can be used, for example, to have other services process the data and display it to the interested parties

The interaction of the mentioned components can be visualized in the following diagram [2]:

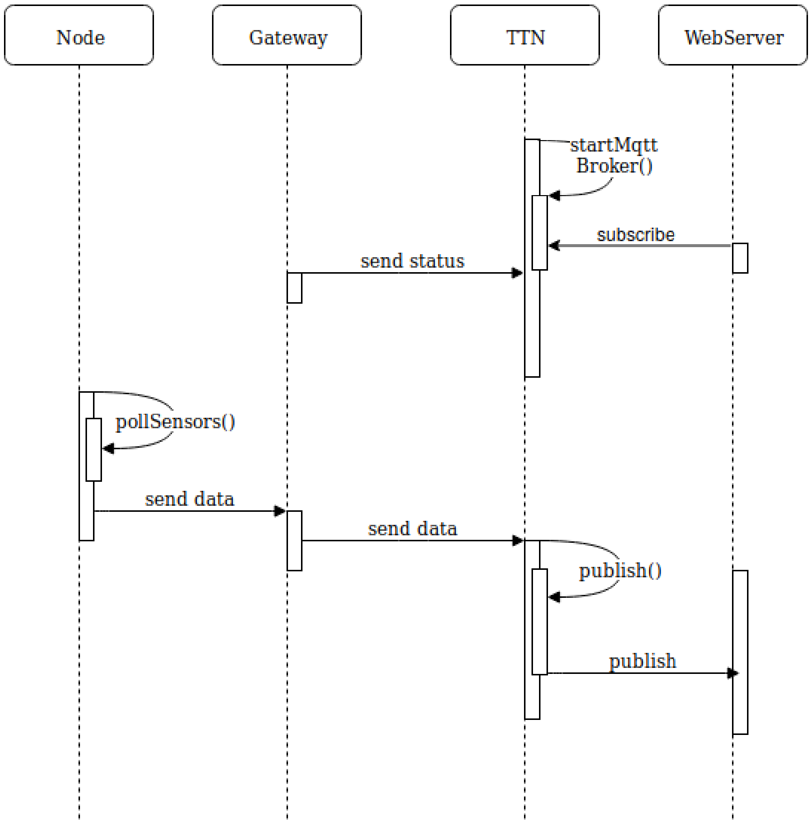

Figure 2: Proposed architecture

4.2 EXPERIMENTS

In order to test the previously mentioned quality parameters and thus our hypothesis, we designed the following experiments:

4.2.1 *Normal Usage*

In order to test the consistency and accuracy of our readings, the reliability of the whole system, its latency, and its energy consumption when in use, we have to use the walker and analyse what output we get. We decided to do two runs of 30 minutes, doing as much as possible to keep all the factors constant and save all the information we get. We can then use the sensor readings to find out our consistency and accuracy, the percentage of packets dropped for reliability, the delay between them for latency and the energy used for energy consumption.

4.2.2 *Idle Energy Consumption*

We will leave the walker idle for around 12 hours while measuring its energy comsumption. We can then calculate how much energy the node consumes per hour both idle and in use (above experiment) and conclude how much it will last on common 9V batteries.

4.2.3 *Integrating the GPS sensor*

In order to test the modularity of the system, we will deliberately only integrate the GPS sensor in the testing phase of the project, this way we can measure how much time it takes and analyse the complications brought by doing so.

4.2.4 *Testing the GPS sensor*

It would be hard for us to integrate the GPS data testing with the bigger experiment mentioned above, due to the fact that we would have to be outside for it to work and the sensors we are using break very easily, so we are divising an experiment just for it. We are going to get the location of the walker 40 times and do some statistical analysis on that data.

4.2.5 *Setting up the Arduino from scratch*

Due to the fact that we only have one LoRaWAN hat for the Arduino, we are not able to have two setups running simultaneously, so we will re-register the Arduino in "The Things Network" and re-upload the code to it, measuring how much time the process takes. We will do it once without practicing the process, and then one more time when we know exacly which steps to take. This experiment is the closest we can get to testing the scalability of our system.

4.2.6 *Disconnecting cables*

In order to test serviceability, we will have someone disconnect random cables connecting the sensors to the node and we will then have another member of the team get the system back up and running while we measure the time it takes.

4.2.7 *Finding maximimum payload*

We will increase the ammount of bytes being sent by the node until they no longer arrive, this will allow us to discover the maximum amount of information we are able to send in each packet and, together with the latency numbers, the maximum throughput.

5 IMPLEMENTATION

The final implementation of our system is as follows [3]:

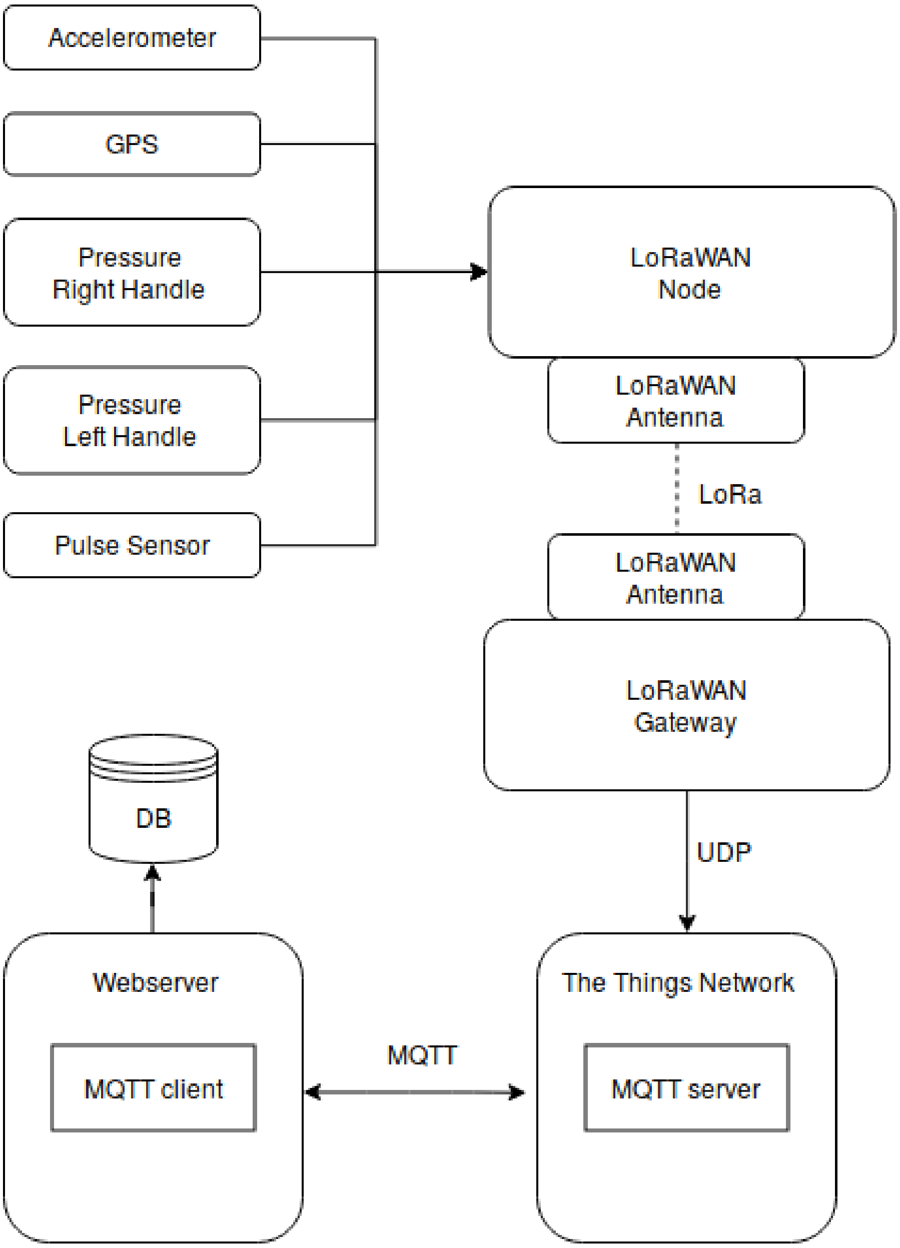

Figure 3: Implemented Architecture

It differed from the proposed design in the following ways:

We decided to use a webserver with an mqtt client and a database instead of a central server with API. We also decided to not implement the analytics server as part of the current project. We made these decisions prioritising the components that we needed to test our hypothesis. Since the presence or absence of a central server or analytics platform don't affect the parameters we need to test our hypothesis we decided to ignore them in the implementation part.

5.1 NODE AND GATEWAY HARDWARE

The presented architecture can be realized in a number of ways, meaning we had to make the following choices before going into details:

- Use sensors with inbuilt LoRa functionality or use a single-board computer connected to sensors and LoRa transmitter?
- What hardware to use to interface with the sensors and act as the node?
- What hardware to use as LoRaWAN gateway?

5.1.1 *Single-board Computer vs Inbuilt LoRaWAN*

We had the option of using a single-board computer (SBC) to interface with all of the sensors (Arduino or Rpi) or using sensors with inbuilt LoRaWAN capabilities.

We summarized the characteristics of sensors with inbuilt LoRaWAN as follows:

- They require little to no configuration and work out of the box. This means using them would grant us more time to we can focus on other parts of the project.
- Data can be sent at different rates for each sensor.
- They are much more expensive than standard sensors.
- They are bulkier than standard sensors, making them harder to fit into the walker.
- Their is low variety when it comes to such sensors in the market, limiting choice of vendor and sensors.
- Having several LoRaWAN transmitters instead of one would make the power consumption higher.
- Each sensor usually has its own battery or charging method, which makes recharging more complex.

We summarized the following characteristics of SBCs:

- Fine grained control of the sensors
- We might want to do some preprocessing of the sensor data before sending it, which is possible with SBCs.

- Good amount of variety in the market allowing us a lot of choice.

We decided to go with an SBC because we value the flexibility provided by the vast array of sensors to choose from and the possibility to manually program them.

5.1.2 *Comparing Raspberry Pi and Arduino*

We then had the option of either using a Raspberry Pi or an Arduino. We summarized their characteristics as follows:

Pi:

- Faster and more powerful processor
- Overhead of operating system, hdmi output, wifi/ethernet port, audio output, all of which take physical or disk space.
- More power consumption

Arduino:

- Easy to get up and running
- Easier to connect with analog sensors
- Lots of different models to choose from, meaning we can choose the one that best fits our goals
- Cheaper than the Raspberry Pi
- Smaller
- No operating system, meaning we are closer to the hardware and have more control over the sensors

Due to the points mentioned above, especially due to its flexibility, and low power consumption, we decided to go with the Arduino as the node.

Due to its superior computational power, inbuilt networking capabilities, we decided to go with the Raspberry Pi as the gateway.

Our reasoning is supported by the arguments put forward in [14].

The specifics of the implementation of each component of our system is described in detail in the following sections.

5.2 NODE

Hardware

As mentioned, the node consists of an Arduino Mega (R3 ATmega2560) with a Seeedstudio Dragino Lora Shield (868 M Frequency). It is powered by a standard 9V battery pack. The node also consists of a breadboard, which contains two Analog to Digital converter and a GPS Sensor. The entire node setup is mounted on the front face of the lower left leg of the walker. The accelerometer is directly connected to the Arduino and is mounted on the back face of the left leg of walker. Two pressure sensors are mounted on the handles of the walker and are connected to the A/D converters in the breadboard, which in turn are connected to the Arduino. The pulse sensor is mounted on the left handle and is directly connected to the Arduino.

The circuit diagram of the node is as follows [4]:

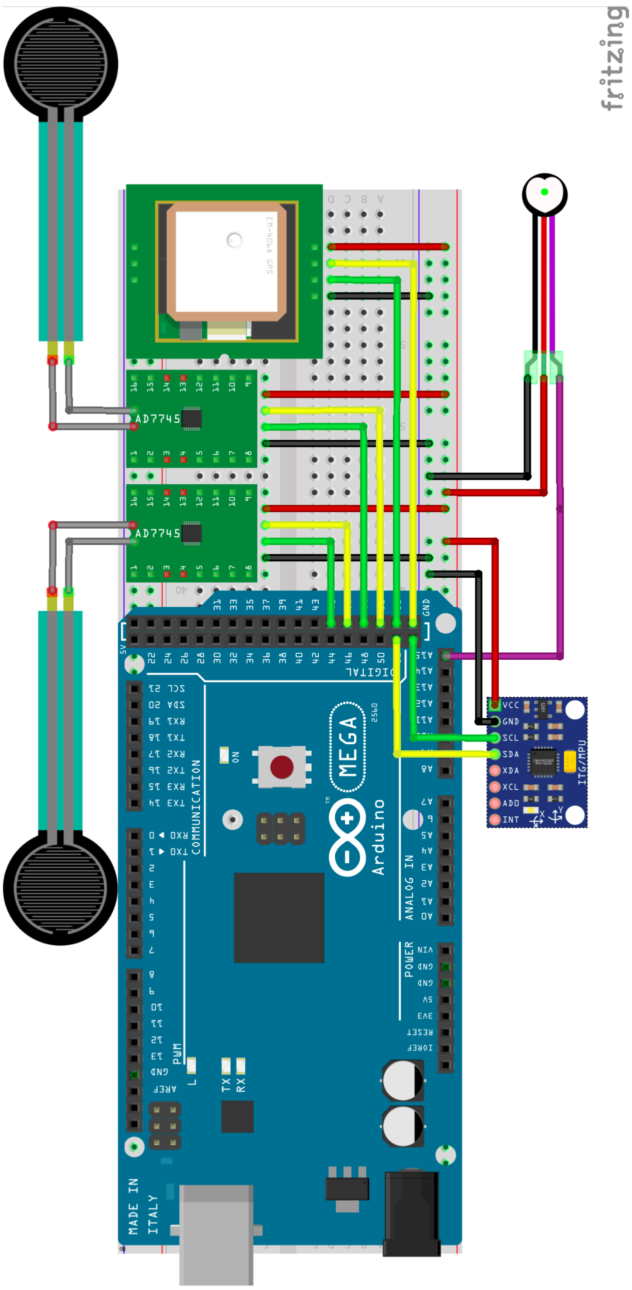

Figure 4: Implemented Architecture of the Node

The Blueprint table shows how we connected the sensors to the pins of our Arduino board [1]:

Sensor	Out_Sensor	In_Arduino
GPS	VCC	5V
	RX	D12
	TX	D13
	GND	GND
Pressure Right	VCC	5V
	SCK	D37
	DT	D36
	GND	GND
Pressure Left	VCC	5V
	SCK	D35
	DT	D34
	GND	GND
Movement	VCC	5V
	GND	GND
	SCL	SCL
	SDA	SDA
Pulse	VCC	5V
	GND	GND
	S	A15

Table 1: A: Analog, D: Digital

Software

The IDE used for writing and uploading the code was Visual Studio with Platform I/O extension. The Platform I/O simplified the process of uploading the code and monitoring the output of the Arduino mega.

The libraries used were lmic, hal, SPI, SoftwareSerial, MPU9250 and HX711 along with the standard Arduino,stdlib and string library.

The folder structure of the Arduino is defined in the appendix [**??**].

Every 15 seconds, the Arduino checks if the walker is moving through the accelerometer, and if so, polls all the other sensors by using their respective libraries, packages them into a buffer, and tries to send them over LoRaWAN.

Each LoRaWAN packet contains its application and network keys and the device's address, which are required by The Things Network.

5.3 PRESSURE

Hardware

For measuring pressure we used two Aluminum Alloy pressure sensors, which have a measurement range of -10 to 10 kg, connected to the Arduino via an HX711 AD [9] converter module.

Software

The sensor data was collected using the HX711 library to transform the analog signal along with the standard Arduino library.

5.4 PULSE

Hardware

For pulse, we used the pulse sensor from pulsesensor.com [15]. It contains an LED that shines light into the capillary tissue, and the sensor reads the light that bounces back.

Software

For reading the data from the sensor, we simply read the analog input. This data is then processed to get the average heart rate of the patient by finding its pattern in the raw data.

5.5 ACCELEROMETER

Hardware

We used an MPU9250 accelerometer [12] to measure whether the walker is moving or not.

Software

The library used to read the accelerometer's output was the sensor specific MPU9250 library. If the gyroscope reading from the accelerometer increased beyond a certain threshold, the walker was considered to be moving.

5.6 GPS

Hardware

For measuring GPS we used the GY-GPS6MV2 sensor [13].

Software

The library used to read GPS data was SoftwareSerial and it was then parsed to get the value of latitude and longitude.

5.7 GATEWAY

Hardware

The gateway consists of a Raspberry Pi 3+ with a Dragino LoRa hat for Pi [6]. The Pi was connected to a router which in turn was connected to the internet.

Software

The Raspberry Pi is running a single channel packet forwarder written in C++, which forwards the packets it receives from the node to The Things Network's european router.

5.8 THE THINGS NETWORK (TTN)

We describe our reasoning behind using TTN in the previous section [4.1.3].

The application in TTN receives packets from the gateway, decodes the byte array in the payload to strings and puts them into a JSON object, which is then published using the mqtt broker.

5.9 WEBSERVER

We created a webserver using the Django framework. It contains an mqtt client implemented using the paho-mqtt library which subscribes to our TTN application on startup. The client than waits for the TTN broker to send it the published information, which it then stores in an sqlite database and displays on the webpage using the django-tabular library.

We chose to use the Django framework with sqlite as the database for our server, because they allow for quick prototyping and testing.

6 EVALUATION

Having in mind our hypothesis: "Is it practical to monitor walkers and their users by sending data over LoRaWAN?", we devised in the "Design and Methodology" chapter a series of experiments [4.2] we intended to run in order to measure the parameters of our system which reflect its practicality [1].

What follows is an in-depth description of each experiment and its results.

6.1 NORMAL USAGE

Pourpose

We ran this big experiment because measuring accuracy, consistency, reliability, latency and energy consumption, all have in common the fact that the best way to get data on these parameters is simply using the walker, so we devised this test, which had the goal of helping us simultaneously assess all the aforementioned quality parameters.

Expectations

Regarding *Consistency*, we had different expectations, depending on the sensor:

- Heart Rate: The sensor we are using is very sensitive to noise and we have no way of keeping our heart rate constant when running the test, so we expected the readings to vary quite a lot.
- Movement: Since this is a boolean value and the accelerometer is very sensitive, we expected walking around to be enough to report movement consistently.
- Pressure: We expected that if we manage to put constant force on the pressure sensors, they will give consistent readings.

As with consistency, each sensor can be considered *Accurate* if it meets conditions specific to it:

- Heart Rate: The sensor we are using is very sensitive to the pressure put on it and the thickness of the person's skin. We are also calculating the user's heart rate by processing the raw pulse data, so we didn't expect much accuracy.

- Movement: Somewhat analogous to our consistency expectations, the movement someone makes while walking should be more than enough to trigger the accelerometer so we expected an accuracy of 100%, since we see no reason it would give an erroneous report.

- Pressure: The pressure sensor we use does not provide an accuracy threshold, but from our preliminary usage it seemed to be at least 95% accurate.

Regarding *Reliability*, the only failure we could see happening in this experiment would be LoRaWAN packets being sent from the node to the gateway being dropped, which doesn't seem to happen very often. We were thus expecting at least 90% reliability.

Regarding *Latency*, the air-time of LoRaWAN packets is dependent on the payload, the distance to the gateway, the spreading factor and much more, so it was hard for us to make an accurate prediction. We had nonetheless noticed, while building our system, that the time between packets was usually between 4 and 9 minutes, so we were expecting an average of around 6 minutes and a high standard deviation from the mean.

The *Energy consumption* of our components, detailed in the datasheets referenced in each sensor's implementation details, is as follows [2]:

Sensor	Max current draw (mA)
Pulse	4
Pressure (x2)	3
Accelerometer	0.5
GPS	57
LoRaWAN shield	10
Mega 2560	80
Total	159

Table 2: Energy consumption Arduino

Based on this, we expected the current draw to be around 160mA.

Parameters

We had one of our team's members use the walker normally for 30 minutes, while the information was coming in and stored in our server for analysis. There were only three details:

- In order to test the accuracy of the heart rate, we had the team member testing the walker use an Iwatch for comparison, which is also not the most accurate of devices, but we don't have access to a better one

- In order to test the consistency of the pressure sensor readings, we put an elastic band applying constant force on the right handle's sensor
- In order to measure the current draw we used a USB Current Meter during the experiment

Results

6.1.0.1 *Heart Rate*

The results of the heart rate measured in the first run are shown in plot [5]

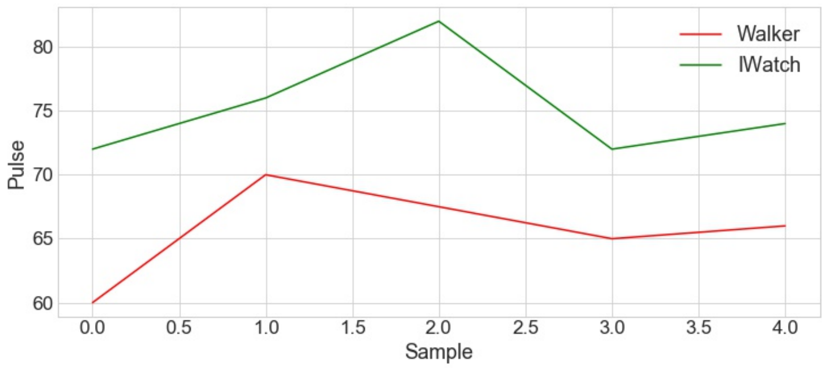

Figure 5: Heart rate measurement, first run

The results of the heart rate measured in the second run are shown in plot [6]

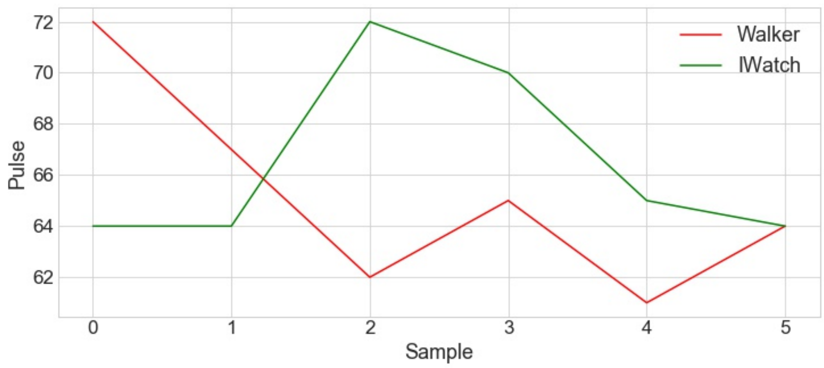

Figure 6: Heart rate measurement, second run

We can observed that our values are consistently lower than that of the IWatch, and seem to stabilize after a few measurements. In the first run we got a Mean Average Percentage Error (MAPE) of 11.27%, between this and the second run, we found an small improvement that could be done to the algorithm, so we changed it for the second run, where we got a MAPE of 7.94%.

6.1.0.2 *Movement*

The movement detection was 100% accurate.

6.1.0.3 *Pressure*

The results of the pressure measured in the first run and second run are shown in plot [7]

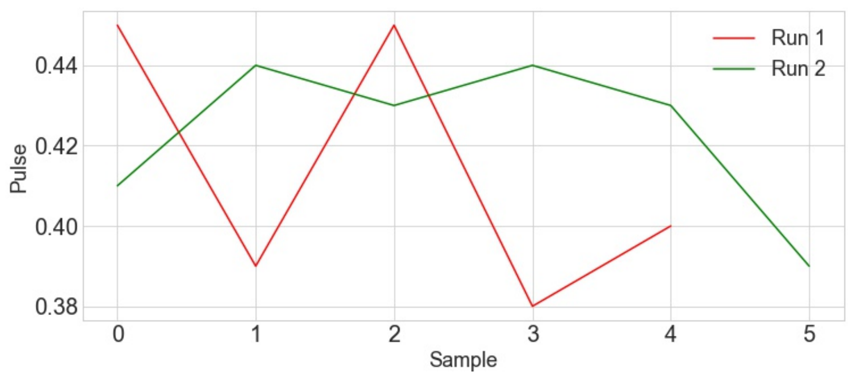

Figure 7: Pressure measurements

The graph demonstrates the variability that can often be observed from one measurement to another, having a relative standard deviation was 8.27% on the first run and 4.76% on the second run.

6.1.0.4 *Reliability*

All the packets arrived successfully, but in two of them the heart rate was 0.

6.1.0.5 *Latency*

Here are the times, in minutes and seconds, between packets [8]:

Gap	First Run	Second run
1st - 2nd	7:35	6:00
2nd - 3rd	8:06	4:43
3rd - 4th	6:47	5:05
4th - 5th	6:13	5:14
5th - 6th		4:24
Average	7:18	5:05
Standard deviation	50.13s	36.30s

Table 3: LoRaWAN latency

Plot [8] visualizes these results:

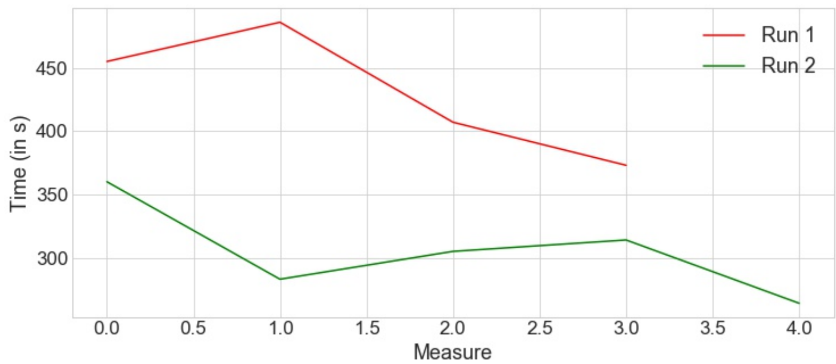

Figure 8: LoRaWAN latency

In the 1st run we got an average of 7:18, with a relative standard deviation of 11.7%, while in the second run we got an average of 5:05, with a relative standard deviation of 11.9%. The overall average was 6:04.

6.1.0.6 *Energy consumption*

The walker used 105 and 104 mAh during the first and second runs respectively. This means it was drawing a current of around 210mA.

Matching results and expectations

6.1.0.7 *Heart Rate*

In both runs we got a reading of 0 in one of the measurements, so we will take these as errors of the overall system and not take them into account in our statistical analysis.

Given the constraints mentioned in our expectations, these results are better than we expected, perhaps because worked a lot on improving our algorithm for transforming the raw pulse into heart rate.

We believe doing an analysis of the precision is not appropriate, since there was a gap of several minutes between measurements and the person was walking around, so there is no reason we should expect the heart rate to be constant.

6.1.0.8 *Pressure*

Regarding Accuracy, under the given conditions, we were not able to get the real value of the pressure being applied on the handle while the walker is in use, so it doesn't seem like a relevant analysis to do. We have nonetheless previously tested the pressure sensors with a 1 kg bag of rice, and both reported that exact value, so we can expect them to be similarly accurate while in use on the walker.

The relative standard deviation of 8.27% on the first run and 4.76% on the second one is whithin our expectations. The deviations we see could be explained by the quality of the sensor, the movement of the walker itself and/or changes in the position of the elastic band.

6.1.0.9 *Movement*

As expected, we got an accuracy of 100% on the movement detection, and, since it is a boolean value, also a standard deviation of 0. Getting the movement status of the walker is a very simple measurement, so any result other than this would have been surprising.

6.1.0.10 *Reliability*

We believe the two failed measurements are due to the method we have for calculating the heart rate from the pulse, which times out if it doesn't find a value within a certain time frame, so that is most likely what happened in both these cases. We can notheless say that the whole data pipeline was 100% reliable, which is the most relevant parameter for our hypothesis.

6.1.0.11 *Latency*

We got, on both runs, equaly spread out times, which is good evidence for the consistently high variablity of the latency, which is what we had expected. Our expectation of 6 minutes average time is right in the middle of the two runs (7:18 and 5:05), which leads us to believe that if we did more and longer runs, it would converge somewhere between 5:30 and 6:30.

6.1.0.12 *Energy consumption*

The walker consumed 30% more energy than we expected, we think this could be because we are using two of the 5V pins, but it is most likely a factor we are not aware of and decided investigating it would be out of the scope our knowledge and most importantly this project.

6.2 idle energy consumption

Pourpose

Most of the time, the walker will not be in use, so it is important to know how much energy it draws while idle

Expectations

When the walker is not moving, the only activity the Arduino does is checking the movement status every 15 seconds. We nonetheless had reason to think that just the fact that the sensors are connected has them draw current, so we expected the power consumption to be close but a bit less than to that of the walker in use, which we expected to be 160mA.

Parameters

We connected a USB Current Meter to the walker and left it idle for 12:30h the first time and the 18:00h the second time

Results

During the 12:30h window, the walker drew 2498mAh, which equates to an average current of 200mA, while in the 18 hour run it drew 3505mAh, equating around 195mAh.

Matching results and expectations

As in the normal usage experiment, the Arduino drew much more current than the expected, and the explanation we have is the same. It was, as we predicted, just a bit lower than that of the walker in use, meaning a future improvement over our system would be setting up the node such that the sensors are not drawing power while the walker is idle. With the current setup, we estimate the walker would last around 2h on a Zinc battery, 02:45h on an Alcaline battery and 06:00 on a common 1200mAh Lithium battery.

6.3 INTEGRATING THE GPS SENSOR

Pourpose

This experiment had the goal of figuring out how quickly a new sensor can be added to the walker. Our walker is a proof of concept for something bigger, so it is important to know what kind of problems can arise when attempting to extend it.

Expectations

Having in consideration how much time it took us to mount, get readings, and send the current sensor's information, we expected to take around 1 or 2 days to integrate the GPS sensor into the system.

Parameters

For this experiment, we had two of the team's members work exclusively on integrating the GPS sensor into the system. This consisted in mounting it on the walker, getting some readings, extracting the useful information, figuring out the best way to encode the data for sending over LoRaWAN, and then adapting the rest of the components to handle the latitude and longitude.

Results

Integrating GPS into the system ended up taking 2 whole days, not only was it hard to find a port layout which allowed all the sensors to work, but library support is also not very good, so after unsuccessfully trying to use external libraries, we ended up implementing it ourselves, which took some extra time.

Matching results and expectations

Integrating GPS into our system took as much time as we were roughly expecting, perhaps it is on the higher end of our expectation because of the usual delays that come with getting the right documentation for sensors and how they interplay with the other components. Two days doesn't seem like much having in consideration that repeating the process on new nodes would be much faster the following times.

While performing this experiment, coincidentally, we noticed that when the GPS wasn't working and we wanted to use the walker for other tests, we had to remove it, so we could also say the system is modular in the sense that it is easy to remove sensors.

6.4 testing the gps sensor

Pourpose

This experiment has the goal of letting us know how accurate and precise the GPS sensor is.

Expectations

According to the specification, the NEO 6M module should be accurate within 2 meters, so that was our expectation. Regarding precision, our module's datasheet does not provide a threshold, so we didn't know what to expect there.

Parameters

We took the walker outside and got its location through the GPS sensor 40 times.

Results

The GPS had an average error of 3.5 meters, and its error distribution is show in boxplot furthermore the boxplot shows that the median error is 3 meters and we can see that 75% of our data points is in between 0 and 6 meters of error to the goal [9].

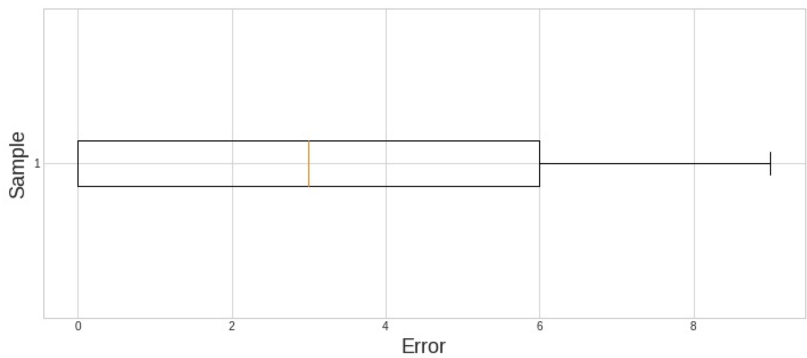

Figure 9: GPS error boxplot

Matching results and expectations

The GPS was less accurate then expected, perhaps because of being positioned in the side of the walker, not getting the perfect reception.

6.5 SETTING UP THE ARDUINO FROM SCRATCH

Pourpose

We planned this experiment so we could get an idea of how scalable our system is, which is one of its most important features.

Expectations

We expected the first attempt to take around 4 minutes because we hadn't registered a device in a while, but witha streamlined process we expected it to take around 1:30 minutes, since it just requires navigating menues and slighly changing the node.

Parameters

We have, on the same computer, registered a new device on The Things Network, changed the node files to include its new authentication codes and uploaded them to the Arduino.

Results

The first attempt took 4:50 and the second one 01:02.

Matching results and expectations

The first attempt took a bit longer than expected, perhaps because we didn't remember very well how to navigate the menus and options on The Things Network. On the second run we were even faster than predicted, maybe because what takes the most time is simply navigating the menus, and after knowing exacly what do it is very fast. This experiment is not a complete demonstration of the scalability of our system, but it is the best we could come up given the situation. It doesn't show how the system would handle several nodes in parallel but at least we know that the act of adding them would be quick.

6.6 disconnecting cables

Pourpose

We wanted to have a test for assessing the serviceability of the walker, this would allow us to have an idea of how long the walker takes to be fixed when hardware problems arise.

Expectations

We expected the node to be fixed in around 2:30 minutes, which included looking up the blueprint [1] and arranging the cables accordingly.

Parameters

One of the team members disconnected 7 random cables from the node and another one attempted to fix it.

Results

The node was back up and running in 1:22, having the first packet arriving on our server at 1:28.

Matching results and expectations

We overestimated the time it would take by a minute, we think this might be because the team member fixing the walker was the most familiar with the cabling and pin layout.

6.7 finding maximum payload

Pourpose

Our node doesn't need to send much information, but it is nonetheless usefull knowing how much could theoretically be sent.

Expectations

We have configured our system to be able to use a spreading factor between 7 and 12, which have a maximum payload of 222 and 51 bytes, respectively, so we were expecting any value in-between, but most likely one of these two, since they are the most commonly used.

Parameters

We increased the payload step by step until the packets wouldn't successfully arrive at our server anymore.

Results

We kept increasing, testing twice and decreasing the payload in the following fashion, until we found the maximum we could successfully send [8]:

Payload (bytes)	15	30	60	45	50	55	54	53	52	51
Success	yes	yes	no	yes	yes	no	no	no	no	yes

Table 4: Payload testing LoraWAN

For the payloads of 60, 55 and 54 bytes, the packets never arrived at ttn, for loads of 53 and 52 bytes, they arrived but the payload was not readable.

Matching results and expectations

The maximum payload we got was 51 bytes, which is one of the options we considered most likely. With this experiment we can be sure the system is using a spreading factor of 12, which provides a very long range, but a very small payload and thus small throughput. If we match the obtained payload with the observed average latency of 6:04, we get a maximum throughput of 8.4 bytes/minute, and our current throughput, with the 15 bytes we send per packet, of 2.5 bytes per minute.

7 CONCLUSION

This work presents an analysis of the practicality of monitoring health parameters of walker users by sending data over LoRWAN.

We attached an accelerometer, a pulse sensor, a GPS sensor and two pressure sensors to a walker, and had an arduino collecting and sending their data over LoRaWAN, so it could be stored on a remote server for future analysis.

Our experiments indicate that if the node is optimized for its energy consumption, which we didn't have time to do, the LoRaWAN protocol, especially when integrated with a stack like *The Things Network*, provides a very simple and streamlined way of connecting walkers that can send their information over very large distances, with high reliability and serviceability.

We belive that with higher quality sensors, well integrated into the walker, the overall fidelity of the system would be very high, making it a viable way to obtain health parametets. Our system can achieve a relatively small throughput of 8.4 bytes per minute, but since the goal is not real-time tracking, this is not a concern.

Having all this in consideration, we can conclude that with further improvements to overall quality and battery life, sending walker data over LoRaWAN could be a practical way to monitor long term health parameters.

As for future work, having a node capable of storing and preprocessing information would allow for a better adaption to the limitations of LoRaWAN, making the most of the small amount of information that can be sent. Another interesting addition would be the capability of using a GSM enabled module for alerting in emergency scenarios and the ability to send higher bandwidth data, for instance, a video feed.

Part II

THE MANUAL

It was the best of times, it was the worst of times,
it was the age of wisdom, it was the age of foolishness,
it was the epoch of belief, it was the epoch of incredulity,
it was the season of Light, it was the season of Darkness,
it was the spring of hope, it was the winter of despair

— *Charles Dickens - A Tale of Two Cities* [5]

ACKNOWLEDGMENTS

The above opening passage from 'A Tale of Two Cities ' by Charles Dickens would describe excellently the highs and lows in this project. In the following we want to acknowledge people who mattered during this time.

First and for most, we would first like to thank the Head of Health IT Lab at the Alexandra Instituttet Morten Kyng and our Professor Niels Olof Bouvin who is a member of the Participatory Information Technology department at the Aarhus University. Both gentlemen made this project in several ways possible.

The door to Professor Olof's and Professor Doctor Morten Kyng's office was always open whenever we had problems or a question about our research or writing. They consistently gave us different perspectives and helped in creating this final report which states the final results of our research.

We would also like to thank the experts that we have consulted alont the way, namely Orbit Lab and ChomskyLab. Without their passionate participation and input, we would not have had the resources to make the project as successful.

7.1 SETUP THE LORAWAN GATEWAY

At first we log into the Gateway, for this we type the following into the terminal [1]:

Listing 1: Logging into the gateway

```
YOU:~$ ssh pi@<ip_address>
sign_and_send_pubkey: signing failed: agent refused operation
pi@<ip_address>'s password:
Linux raspberrypi 4.14.69-v7+ #1141 SMP Mon Sep 10 15:26:29 BST
    2018 armv7l

The programs included with the Debian GNU/Linux system are free
    software; the exact distribution terms for each program are
    described in the individual files in /usr/share/doc/*/
    copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Dec  5 16:44:47 2018 from <ip_address>
```

Next we start it with the following command [2]:

Listing 2: Starting LoraWAN Gateway

```
pi@raspberrypi:~ $ LoraWanGateway/single_chan_pkt_fwd/
    single_chan_pkt_fwd
SX1276 detected, starting.
Gateway ID: b8:27:eb:ff:ff:d0:8e:22
Listening at SF7 on 868.100000 Mhz.
------------------
stat update: {"stat":{"time":"2018-12-06 18:02:38 GMT","lati"
    :0.00000,"long":0.00000,"alti":0,"rxnb":0,"rxok":0,"rxfw":0,"
    ackr":0.0,"dwnb":0,"txnb":0,"pfrm":"Single Channel Gateway","
    mail":"bla@gmail.com","desc":"smart walkers!!"}}
```

7.2 SETUP THE LORAWAN NODE

The node is a plug and play device, if its gets energy out of a 9V battery or through its USB Micro input, it will start to work immediately without any software upload.

7.3 RUN WEBPAGE

First navigate to the directory with the server files. Next open a terminal and type in [3]:

Listing 3: Starting Webserver

```
YOU:~/smartwalker$ python manage.py runserver --noreload
Performing system checks...

System check identified no issues (0 silenced).
December 06, 2018 - 18:11:22
Django version 2.1.2, using settings 'smartwalker.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
Connected with result code 0
```

Open http://127.0.0.1:8000/ in a webbrowser. The webside should be now visible.

Part III

THE SHOWCASE

7.4 WALKER

7.4.1 *GPS Module*

The GPS Module is used to analyse how sedentary the user is and what kinds of areas are most frequented by walker users and thus figure out what places might be worth investing in [10].

Figure 10: Used GPS Module

7.4.2 *Heartrate Sensor*

This pulse sensor allows us to measure the user's heart rate [11].

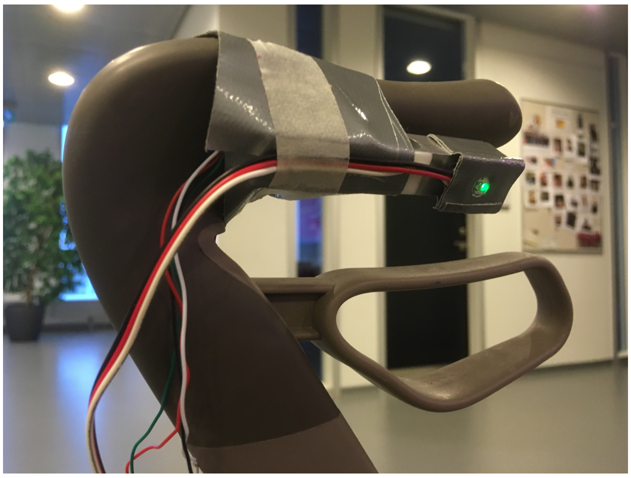

Figure 11: Used Heartrate Sensor

7.4.3 *Movement Sensor*

The Movement Sensor allows us to measure the speed at which the user is walking. If we track both handles independently, together with the pressure measures, we can in a crude way analyse gait of the user [12].

Figure 12: Used Movement Sensor

7.4.4 *Pressure Sensors*

The Pressure Sensors allow us to track how much the user relies on the walker and how his/her grip strength changes over time [13].

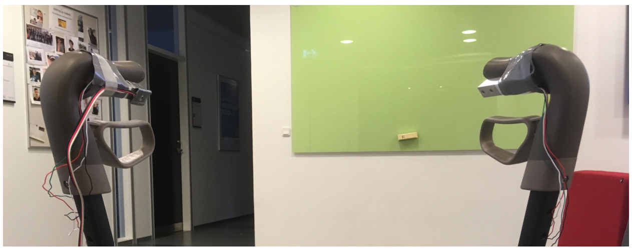

Figure 13: Used Pressure Sensors

7.4.5 *LoRaWAN node*

A Arduino Mega with a LoRaWAN hat and connected to a battery, retrieving and sending sensor data [14].

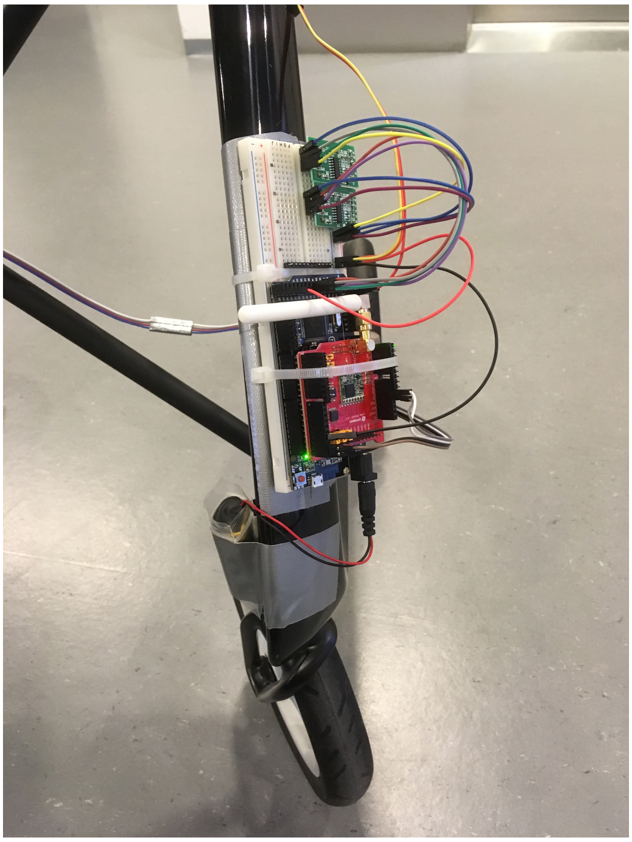

Figure 14: Used LoRaWAN node

7.5 LORAWAN GATEWAY

Raspberry Pi 3 with a LoRaWAN hat, forwarding LoRaWAN packets to The Things Network [15].

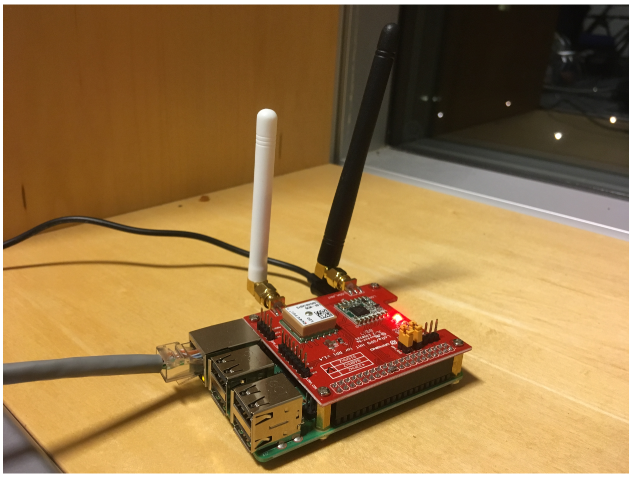

Figure 15: Used LoRaWAN gateway

7.6 PRODUCT VIDEO

In the following you can find a demonstration of the project:
https://vimeo.com/304645369

Part IV

APPENDIX

A

APPENDIX

A.1 APPENDIX DATA

In the following tables you can find all our test measurements:

Our Reading	IWatch Reading
60	72
70	76
00	82
65	72
66	74

Table 5: Measuring heart rate first run

Our Reading	IWatch Reading
72	64
00	64
62	72
65	70
61	65
64	64

Table 6: Measuring heart rate second run

Mesurements	1	2	3	4	5	6
Run 1	0.45	0.39	0.45	0.38	0.40	
Run 2	0.41	0.44	0.43	0.44	0.43	0.39

Table 7: Measuring pressure first and second run

Payload (bytes)	15	30	60	45	50	55	54	53	52	51
Success	yes	yes	no	yes	yes	no	no	no	no	yes

Table 8: Payload testing LoraWAN

BIBLIOGRAPHY

[1] Ahmed Alahmadi and Ben Soh. "A smart approach towards a mobile e-health monitoring system architecture." In: *Research and Innovation in Information Systems (ICRIIS), 2011 International Conference on*. IEEE. 2011, pp. 1–5.

[2] Emekcan Aras, Gowri Sankar Ramachandran, Piers Lawrence, and Danny Hughes. "Exploring the security vulnerabilities of lora." In: *Cybernetics (CYBCONF), 2017 3rd IEEE International Conference on*. IEEE. 2017, pp. 1–6.

[3] Hadi Banaee, Mobyen Ahmed, and Amy Loutfi. "Data mining for wearable sensors in health monitoring systems: a review of recent trends and challenges." In: *Sensors* 13.12 (2013), pp. 17472–17500.

[4] Jonathan de Carvalho Silva, Joel JPC Rodrigues, Antonio M Alberti, Petar Solic, and Andre LL Aquino. "LoRaWAN—A low power WAN protocol for Internet of Things: A review and opportunities." In: *Computer and Energy Science (SpliTech), 2017 2nd International Multidisciplinary Conference on*. IEEE. 2017, pp. 1–6.

[5] Charles Dickens, Hablot K. Browne, and Frederick Barnard. "A Tale of Two Cities." In: *Print* (1942).

[6] Dragino. *Lora Shield*. URL: `http://wiki.dragino.com/index.php?title=Lora_Shield` (visited on 12/06/2018).

[7] Sheikh Ferdoush and Xinrong Li. "Wireless sensor network system design using Raspberry Pi and Arduino for environmental monitoring applications." In: *Procedia Computer Science* 34 (2014), pp. 103–110.

[8] Aashay Gondalia, Dhruv Dixit, Shubham Parashar, Vijayanand Raghava, Animesh Sengupta, and Vergin Raja Sarobin. "IoT-based Healthcare Monitoring System for War Soldiers using Machine Learning." In: *Procedia computer science* 133 (2018), pp. 1005–1013.

[9] *HX711 - 24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales*. AVIA Semiconductor.

[10] SM Riazul Islam, Daehan Kwak, MD Humaun Kabir, Mahmud Hossain, and Kyung-Sup Kwak. "The internet of things for health care: a comprehensive survey." In: *IEEE Access* 3 (2015), pp. 678–708.

[11] Afef Mdhaffar, Tarak Chaari, Kaouthar Larbi, Mohamed Jmaiel, and Bernd Freisleben. "IoT-based health monitoring via LoRaWAN." In: *Smart Technologies, IEEE EUROCON 2017-17th International Conference on*. IEEE. 2017, pp. 519–524.

[12] *MPU-6500 Product Specification Revision 1.1*. InvenSense.

[13] *NEO-6 u-blox 6 GPS Modules*. ublox.

[14] O Postolache, P Girao, JM Dias Pereira, J Pincho, C Moura, and G Postolache. "Smart walker for pervasive healthcare." In: *Sensing Technology (ICST), 2011 Fifth International Conference on*. IEEE. 2011, pp. 482–487.

[15] *Pulse Sensor Datasheet*. World Famous Electronics LLC.

[16] Stefano Tomasin, Simone Zulian, and Lorenzo Vangelista. "Security analysis of lorawan join procedure for internet of things networks." In: *Wireless Communications and Networking Conference Workshops (WCNCW), 2017 IEEE*. IEEE. 2017, pp. 1–6.

DECLARATION

I declare that the report has been composed by myself and that the work has not be submitted for any other degree or professional qualification. I confirm that the submitted work is my own, that my contribution and those of the other authors to this work have been explicitly indicated. Additionally the experimental work is entirely the work of the other authors and me. I confirm that appropriate credit has been given within this report where reference has been made to the work of others.

Aarhus, December 2018

Patrick Lewandowski

Harshit Mahapatra

Tomas Manuel Rebelo Mota

COLOPHON

This document is a Project Report for the Building the Internet of Things with P2P and Cloud Computing subject by Niels Olof Bouvin from the University of Aarhus, Aarhus, in Denmark.

Title	**Remote Monitoring of Health Parameters Using smart walkers**
Version:	*Final Version* as of December 7, 2018 (1.0).
Date:	07.12.2018
Authors:	Patrick Lewandowski, 201802946 Harshit Mahapatra, 201803187 Tomas Manuel Rebelo Mota, 201802941
University:	University of Aarhus Åbogade 34 8200 Aarhus Faculty of Computer Science Master of Science
First Supervisor:	Niels Olof Bouvin
Second Supervisor:	Michal Ratajský