



UNIVERSITY of LIMERICK

OLLSCOIL LUIMNIGH

Department of Electronic & Computer Engineering

Low-Power Wireless Inclinator

By:

Piotr Kurzynoga

Student ID - 14143097

Supervisor – Dr Richard Conway

Course Title – Electronic and Computer Engineering

Course Code – LM118

Graduating Year – 2018

Abstract:

This project features a low-power wireless inclinometer that provides accurate measurements of roll and pitch using an accelerometer and transmits them securely over the SigFox network, those measurements then can be accessed online through the hosted website where they are organized and visualized to provide an effortless way to read back the data.

Throughout this report the following research and procedures can be observed, the accelerometer calculations of the tilt of an object which are processed using the SmartEverything Fox onboard Atmel microcontroller, the Telit SigFox module and server-side operation, the Amazon Web Services applications usage and deployment where the website is hosted and most importantly the low-power optimizations that help achieve long operation of the device while powered by a battery.

Acknowledgements

I would like to sincerely thank my supervisor Dr Richard Conway for all his help and support throughout this project. I would also like to thank all the lecturing staff of the University of Limerick, particularly the Electronic and Computer Engineering Department for their guidance over the past four years. I would mostly like to thank my parents and family for their encouragement and patience especially during these last stressful months. Finally, I would like to thank all my friends for their backing and friendship over the years.

Table of Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 2 | Analytical Background..... | 3 |
| 2.1 | SmartEverything Fox board | 3 |
| 2.2 | Tilt Angle Measurement | 4 |
| 2.2.1 | Measurement Issues..... | 5 |
| 2.3 | SigFox | 5 |
| 2.3.1 | Security | 6 |
| 2.3.2 | REST Interface..... | 7 |
| 2.4 | Amazon Web Services..... | 7 |
| 2.4.1 | IoT | 8 |
| 2.4.2 | Lambda | 8 |
| 2.4.3 | Elastic Beanstalk | 9 |
| 2.4.4 | Relational Database Service..... | 9 |
| 2.5 | Power Consumption..... | 10 |
| 2.6 | Applications: | 11 |
| 3 | Specification and Design | 12 |
| 3.1 | Preparing Local Environment | 13 |
| 3.2 | Laravel Installation | 13 |
| 3.3 | SigFox Module registration | 14 |
| 3.4 | Roll and Pitch Readings..... | 14 |
| 3.5 | SigFox Transmission | 16 |
| 3.6 | Low-Power Software Design | 18 |
| 3.7 | RDS payload insertion | 19 |
| 3.8 | Website Design..... | 20 |
| 4 | Implementation&Integration | 22 |
| 4.1 | Uploading code to SmartEverything Fox board..... | 22 |
| 4.2 | Initialization of the board | 22 |
| 4.3 | Sigfox Cloud - AWS Connector | 24 |
| 4.4 | Website | 26 |
| 5 | Results..... | 29 |
| 5.1 | Battery & Power Results | 29 |

| | | |
|-----|--|----|
| 5.2 | Data Upload Results..... | 31 |
| 5.3 | Software Errors | 33 |
| 5.4 | Possible Improvements | 33 |
| 6 | Conclusion | 34 |
| 7 | References and sources of information..... | 35 |
| | Appendix A | 37 |
| | Poster..... | 41 |

Table of Figures

| | |
|--|----|
| Figure 1 SmartEverything Fox board | 2 |
| Figure 2 Layout of the SME-F Board | 3 |
| Figure 3 SigFox Coverage | 6 |
| Figure 4 Overview of the IoT connector action..... | 8 |
| Figure 5 Lambda configuration tree | 9 |
| Figure 6 Elastic Beanstalk Server Configuration View..... | 9 |
| Figure 7 RDS view from AWS Panel | 10 |
| Figure 8 Hardware representation of the set-up..... | 12 |
| Figure 9 Representation of the axis in respect to the SME-F board | 15 |
| Figure 10 Accelerometer Test Results from Console..... | 15 |
| Figure 11 Hex and Signal Message Representation..... | 16 |
| Figure 12 Back-End view of the SigFox cloud web service | 16 |
| Figure 13 SigFox location approximation..... | 17 |
| Figure 14 SigFox Uplink Payload..... | 17 |
| Figure 15 SigFox Uplink Frame | 18 |
| Figure 16 Low-Power Code Structure | 19 |
| Figure 17 MySQL DB Table data types..... | 19 |
| Figure 18 MySQL table creation code | 20 |
| Figure 19 SME-F, SigFox and AWS Platform interaction | 21 |
| Figure 20 SME-F board initialization snippet..... | 22 |
| Figure 21 Accelerometer Activation Snippet | 23 |
| Figure 22 Accelerometer X, Y, Z Read snippet..... | 23 |
| Figure 23 Roll and Pitch Calculation Snippet..... | 23 |
| Figure 24 Low-Power snippet..... | 24 |
| Figure 25 Callback Configuration..... | 25 |
| Figure 26 Uplink POST Configuration..... | 25 |
| Figure 27 Snippet of the PHP Chart function in the Controller file | 26 |
| Figure 28 Graphs data point insertion..... | 27 |
| Figure 29 Redirecting variables in Laravel | 27 |
| Figure 30 Routing configuration in Laravel..... | 28 |
| Figure 31 Inserting Battery Dates into DB | 28 |
| Figure 34 Power Consumption readout set-up | 30 |
| Figure 32 Results taken from MySQL DB | 31 |
| Figure 33 Website charts created by LavaCharts | 32 |

Table of Acronyms

AWS – Amazon Web Services

IoT – Internet of Things

RDS – Relational Database Service

SME-F – SmartEverything Fox

IDE – Integrated development environment

COM – Communication Port

MCU – Microcontroller Unit

UART – Universal asynchronous receiver-transmitter

LPWAN – Low Power Wireless Area Network

VM – Virtual Machine

SNR – Signal to Noise Ratio

JSON – JavaScript Object Notation

VPN – Virtual Private Network

1 Introduction

This project features the SmartEverything Fox development kit [5] visible in Figure 1 which is the heart of the low-power wireless inclinometer whose goal is to provide accurate measurements of roll and pitch using an accelerometer and transmits them securely over the SigFox network where it is further transferred to a hosting service which allows for the data to be accessed online through the website where they are organized and visualized to provide an effortless way to read back the data.

For the SME-F board to provide lowest possible current consumption various libraries that provide low power functions will have to be researched additionally the datasheets of all the used modules will have to be analyzed carefully to check if additional power saving options are available. The device will be energy efficient enough to guarantee months of uninterrupted data collection, the software that will be designed for this application will provide flawless operation and ensure that no physical interaction is needed while it functions

The 3D accelerometer calculations of the tilt of an object are processed using the SmartEverything Fox onboard Atmel microcontroller and then fed to the Telit SigFox module, finally the transmission occurs, and the payload is sent to the SigFox cloud where upon receiving data a process chain is activated. The callback is the first event to be triggered once the data arrives in the cloud and is responsible for pushing the payload onto the Amazon Web Services platform where through utilizing their services the data is finally displayed on the front-end.

There is two distinct goals in this project, one focusing on using the Arduino IDE to create the code and adopt the appropriate software libraries used for development of the SME-F board functionality which involves the readback from the accelerometer sensor, SigFox module operation and MCU processing while maintain low power usage, the secondary goal in this project is to become familiar with the SigFox cloud service and its data interpretation and researching AWS cloud services that would be appropriate to provide storage, hosting and connectivity between the two cloud services.

In relation to AWS the storage element will focus on adapting the right database for this project and familiarizing with the operations involved in successfully setting it up and using it, hosting translates to deploying the website which will be developed using the Laravel environment as part of this project, finally connectivity will result in the payload transfer from the SigFox platform to the AWS storage.

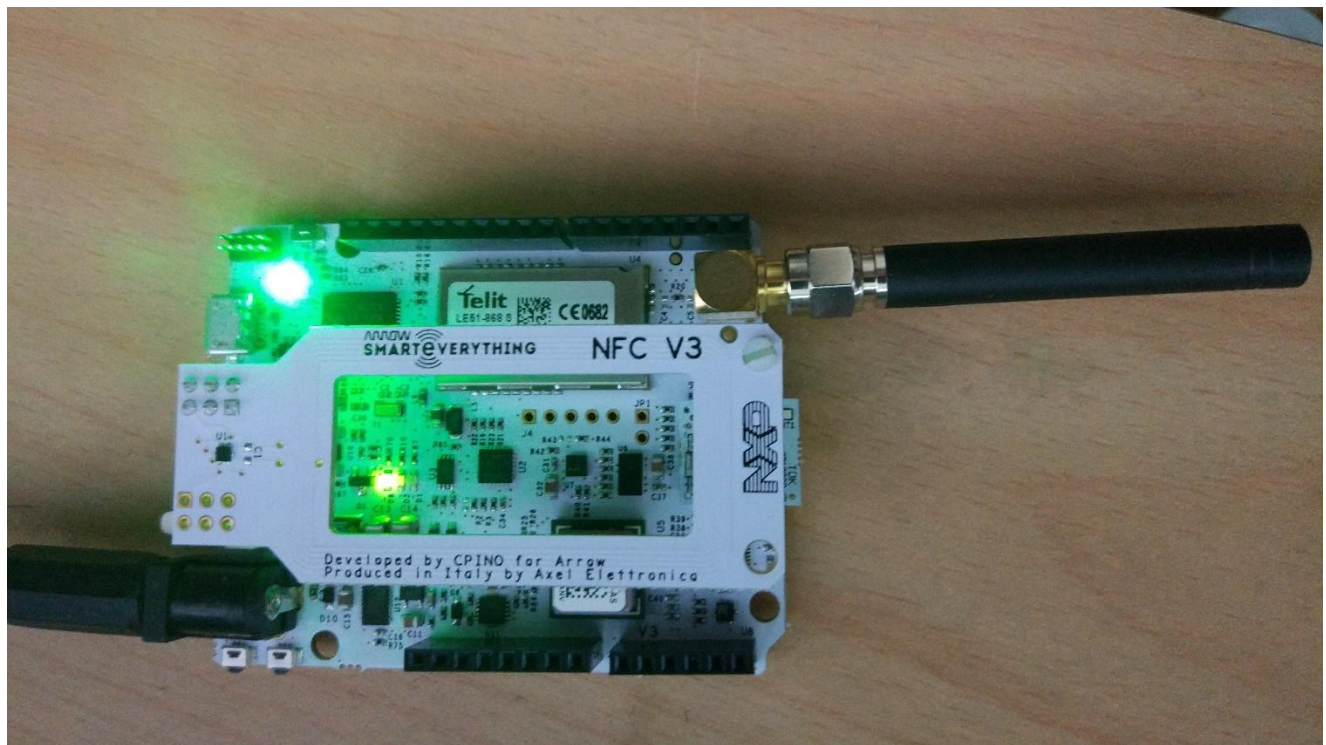


Figure 1 SmartEverything Fox board

After deploying the functional device, the end user will be able to view the tilt sensor data online and be able to compare it to previous results which will be collected over a longer period of time without the need to maintain it. This data can be used to assess how an object behaves(tilts) over a period of days, months and years, which is beneficial for remote applications.

To finalize the goal of this report is to compare and contrast between what the projects expectations are, and the actual results which can be achieved, observing those results gives perspective on how assumptions and theory translates to reality.

2 Analytical Background

2.1 SmartEverything Fox board

When planning the project there were dilemmas whether to design a board and connect the necessary modules to it or use an off the market board that contains the necessary features including the SigFox module and accelerometer, it quickly became apparent that the SmartEverything Fox board was the perfect candidate for this project as not only did it carry the SigFox module and accelerometer on board, the whole device is created with a low-power oriented design which plays a huge role in controlling power management and maintaining the current consumption down to a minimum.

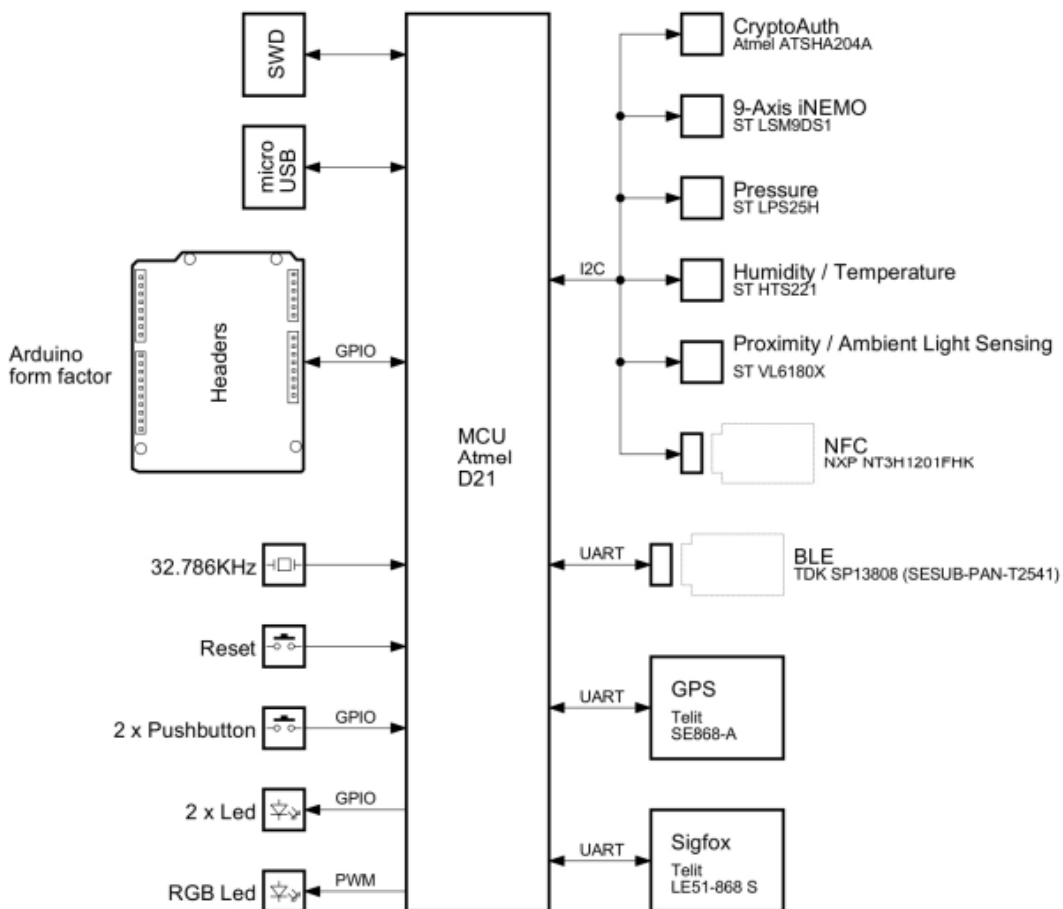


Figure 2 Layout of the SME-F Board [22]

The block diagram in Figure 2 shows all of the modules and sensors embedded in the SME-F board, the ones used in this project are listed below.

- Atmel SAMD21J18A MCU
- 9-axis iNEMO – ST LSM9DS1
- SigFox module – LE51-868 S

2.2 Tilt Angle Measurement

The accelerometer provides raw information regarding the difference between the linear acceleration of the sensor and forces the local gravitational field [11] [2], therefore to obtain the slope of the current position of the board, the roll, pitch and yaw readings need to be estimated from the three-axis accelerometer.

- Roll – is the rotation around the x-axis
- Pitch – is the rotation around the y-axis
- Yaw – is the rotation around the z-axis

The Pitch Roll estimation is done by calculating the roll, pitch and yaw rotation matrices that transform a vector using the co-ordinate system around the x, y and z axis.

Using these rotation calculations in the “xyz” sequence to solve for the roll and pitch angles the following formulas are obtained [13].

Gx, Gy, Gz represent the acceleration in three-axis system (x, y and z)

$$\text{Roll formula: } \tan^{-1} \frac{G_y}{G_z}$$

$$\text{Pitch formula: } \tan^{-1} - \frac{G_x}{\sqrt{(G_y^2 + G_z^2)}}$$

The results returned by the above formulas are in Radians therefore they need to be converted to degrees for ease of readability this is done by multiplying the result by $\frac{180}{\pi}$ which gives a result in the desired format of degrees.

Yaw calculation is not considered as it would require the use of a magnetometer to obtain precise values due to the fact that there is no change in the force of gravity in the z-plane since the SME-F board with the accelerometer sensor is insensitive to rotations around the gravitational field. This does not make the system flawed in a case where the object falls as the z-plane force is still taken into account in the roll and pitch calculations.

2.2.1 Measurement Issues

There are angles at which the calculations of roll and pitch become inaccurate or more precisely unstable, the region where G_y and G_z is equal to zero, this occurs when the accelerometer is oriented with its x-axis pointing vertically upwards or downwards.

Therefore, when the x-axis of the accelerometer is oriented vertically the roll calculation is inaccurate as the rotation about the gravitational field vector cannot be detected, an accurate reading is obtainable up to the point where the axis is vertically oriented.

2.3 SigFox

SigFox is a LPWA (Low-Power-Wide-area) network that was designed to be used with IoT (Internet of Things) devices, the idea behind SigFox is to securely transmit data to and from the cloud whilst reducing the energy consumption and cost of transmitting the data.

SigFox offers national coverage which means we do not have to worry about accessing the network, the data transfer is cost efficient and most importantly the solution features ultra-low power consumption, to top things off SigFox uses a triple diversity scheme which results in secure transmissions that cannot be intercepted by a third party.

The only considerable downside to SigFox is that it is not suitable for high-bandwidth usage communications, but this is not a problem if it is transferring small amounts of data from a sensor.

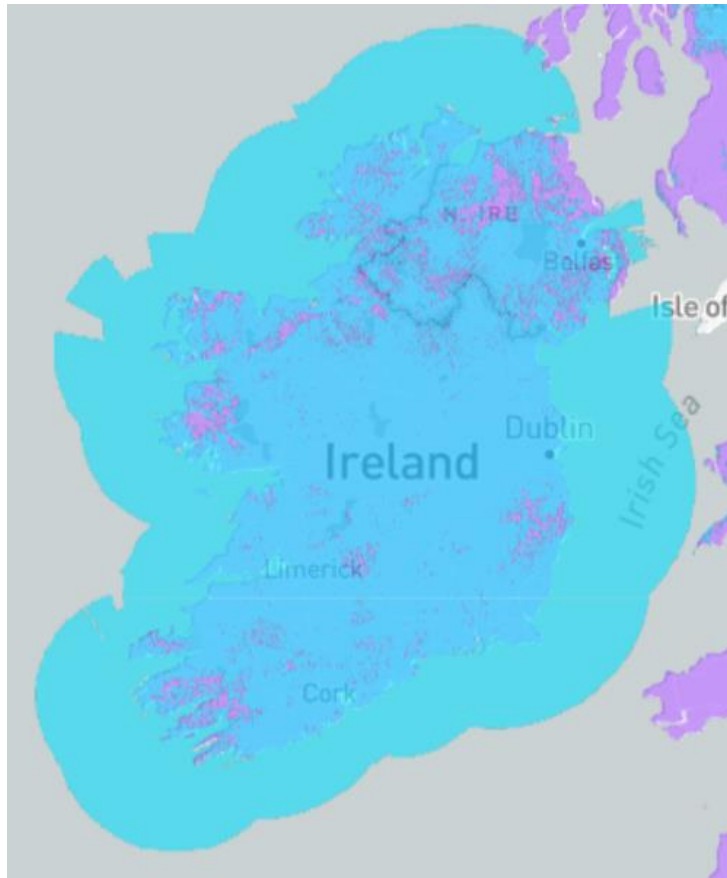


Figure 3 SigFox Coverage

In Figure 3 is the live SigFox coverage of Ireland the blue area is the area covered by the network whilst the purple areas do not offer coverage these are usually mountainous or depression areas.

SigFox provides callbacks that connect to a server in this case the native AWS IoT connector that forwards the data using Lambda functions to a (RDS) Relational Database Service where its stored.

2.3.1 Security

In terms of Security, both the SigFox network and the SigFox cloud offer a secure and robust environment that is protected from attackers. To protect the transmission from the SigFox module to the SigFox cloud end-end authentication is used which is based on a secret key that is stores inside a non-accessible memory that is linked to the read-only memory with a visible unique ID. When sending a message, the secret key creates a signature unique for each message that authenticates the sender of the message inside the signature a sequence number is inserted which

is added to the transmission frames to avoid replay attacks. Furthermore, each message is sent three times, each time selecting a random frequency which protects the packets from being sniffed because it makes it impossible to guess where within the operation bandwidth the transmission will occur.

Signal Jamming is prevented by space-reception diversity which means that the message is sent to different available base-stations in the region.

The communication between the base stations and the SigFox cloud happens over a point to point SSL Encrypted VPN which introduces another layer of security. The SigFox cloud is virtualized on servers hosted in secured data centers distributed in many different physical locations.

The data retrieval using Callbacks which uses the REST Interface from the SigFox cloud is itself HTTPS encrypted which means that there is no weak links in the whole platform starting with the physical module and finishing on the communication link between different platforms.

2.3.2 REST Interface

Callbacks are activated when the SigFox cloud receives data transmitted from the device, the Callback then issues a POST operation which pushes the data to the AWS IoT connector.

The POST operation is part of the RESTful API which is based on Representational State Transfer (REST) architecture, the interface is specifically oriented around the HTTP protocol which makes it the obvious choice when interacting with cloud services, REST also consumes less bandwidth comparing to traditional services such as the Simple Object Access Protocol (SOAP).

2.4 Amazon Web Services

AWS Provides the means to deploy various applications and services in the cloud. In this project the services are used to process the data received from the SigFox network, store them in a database and deploy the website that presents the project related data. The AWS clouds infrastructure is oriented around security and the clouds physical servers are stored in highly secure AWS data centers [1].

2.4.1 IoT

The IoT service provides modules that allow to monitor, secure and act upon events when connected to the IoT device, “act” is the module of interest as it allows for sending E-mails, Texts and executing commands when communication is received from the device.

In this project the IoT service is used to directly connect with the SigFox cloud interface, upon receiving communication from the SigFox cloud a query that selects all the data is ran on the received payload and the Lambda service responsible for further manipulation of the payload is invoked as seen in Figure 4.

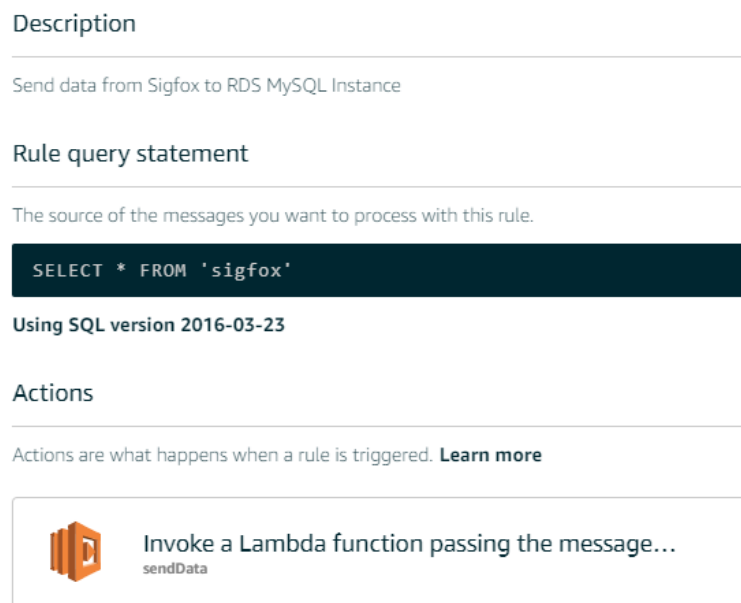


Figure 4 Overview of the IoT connector action

2.4.2 Lambda

Lambda can be interpreted as a server that does not have to be deployed, it allows the user to quickly process code without the need to set up a server.

As it can be seen in Figure 5 the Lambda script “Send Data” is triggered upon an AWS IoT event and the computation is handled by “Amazon EC2” which is the virtual server instance, in addition

logs are submitted to the “Amazon CloudWatch” service where any issues can be viewed for simple troubleshooting.

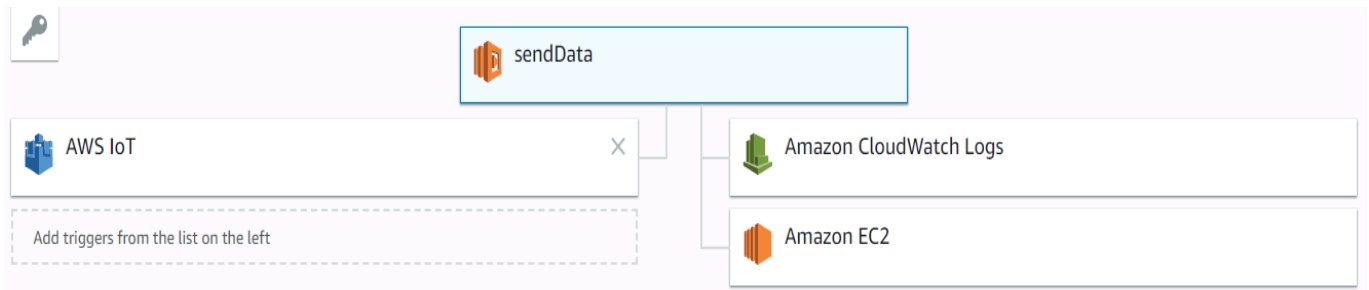


Figure 5 Lambda configuration tree

2.4.3 Elastic Beanstalk

With Elastic Beanstalk a user can directly upload an application to the service and use it from the go, one click upload and deploy. In this project the service demonstrated in Figure.6 will be used to deploy the Laravel application which will be the front-end of the project.



Figure 6 Elastic Beanstalk Server Configuration View

Alongside the ease of deployment Elastic Beanstalk handles Scaling, Load Balancing and Health Monitoring on the go.

2.4.4 Relational Database Service

The RDS allows for prompt set-up of a database instance, in this project a MySQL database is used to store the data that is relayed from the SigFox cloud to the IoT connector.

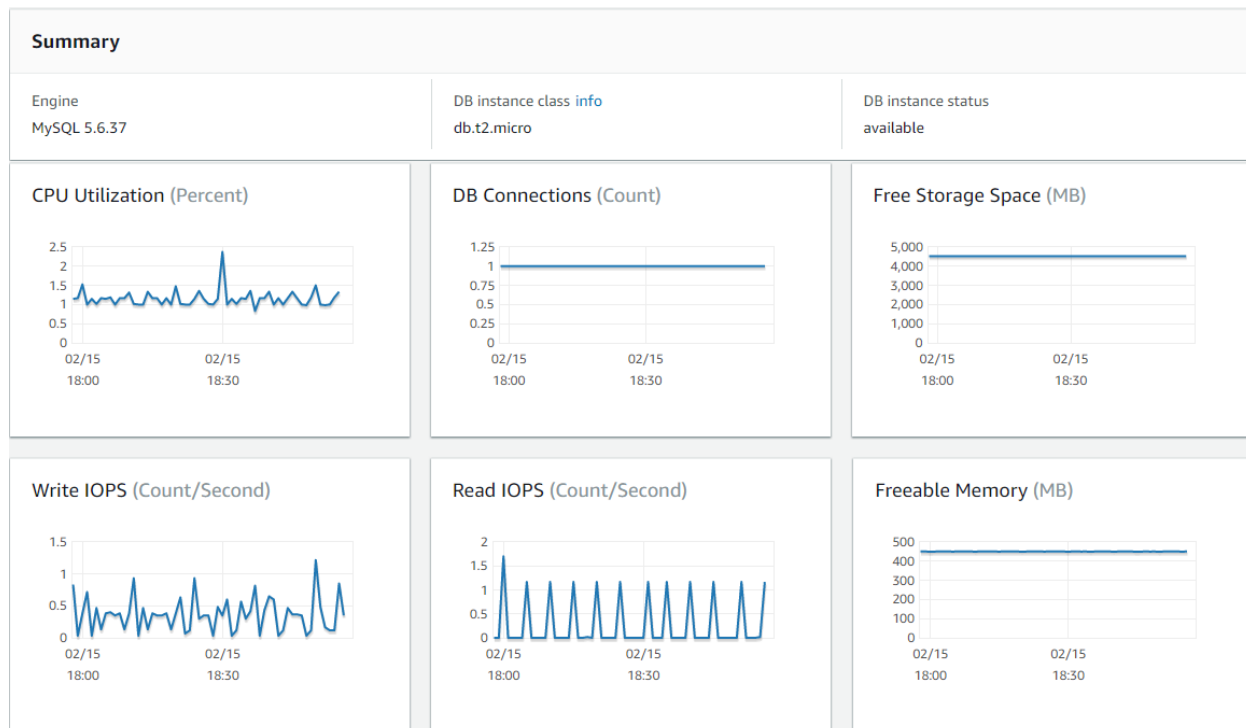


Figure 7 RDS view from AWS Panel

2.5 Power Consumption

Because Low-Power is a focus in this project it is necessary to revise all the possible sources of current consumption to achieve the lowest possible usage and see where most of the consumed power will be directed.

Table 1 Power Consumption according to data sheet specification

| | | | |
|--------------------------|---------------------------------|--|---------------------------------|
| Current Consumption | Atmel SAM D21 (ARM MCU) [22] | Telit LE51-868 S (SigFox module) [17] | LSM9DS1 (Accelerometer) [11] |
| Sleep Mode (Standby) | 23uA | 2uA | 0A |
| Running (Low Power Mode) | 50uA | Rx:32mA Tx:55mA | 1.9mA |

Looking at Table 1 we obtain a value of 25uA while the device is in sleep mode and averaging around 50mA while transmitting and acquiring the sensor data.

Using the above specifications and the following assumptions:

- The time frame to acquire and send the data is 1 minute.
- The battery used has a capacity of 500mAh
- The device goes to deep sleep immediately after the data is sent.
- External factors are not taken into regard.
- A month taking an average of 730 hours.

We come to these hypothetical conclusions:

Sleep Mode – 27 months of operation ($\frac{500mAh}{0.025mA} = 20000$ hours).

Running Mode – 10 hours of operation ($\frac{500mAh}{50mA} = 10$ hours).

- Continuing we get 600 minutes from the 10 hours, at 3 data acquisitions per day we conclude hypothetically the device could last for 200 days without changing batteries.

2.6 Applications:

In this project there is a broad selection of applications. The most interesting of such applications that makes full use of the devices functionality is placing the SME-F at a high-altitude structure for example a power plant chimney where the device is mounted at the top of the chimney and can be monitored over long period of times to check if the structural integrity is intact after suffering blows from strong winds or in the unfortunate case of a small tremor. Since the tilt measurement is to be very accurate down to 0.1 of a degree therefore even the smallest shift in the chimneys structure could be detected and a catastrophe where the chimney collapses could be prevented.

3 Specification and Design

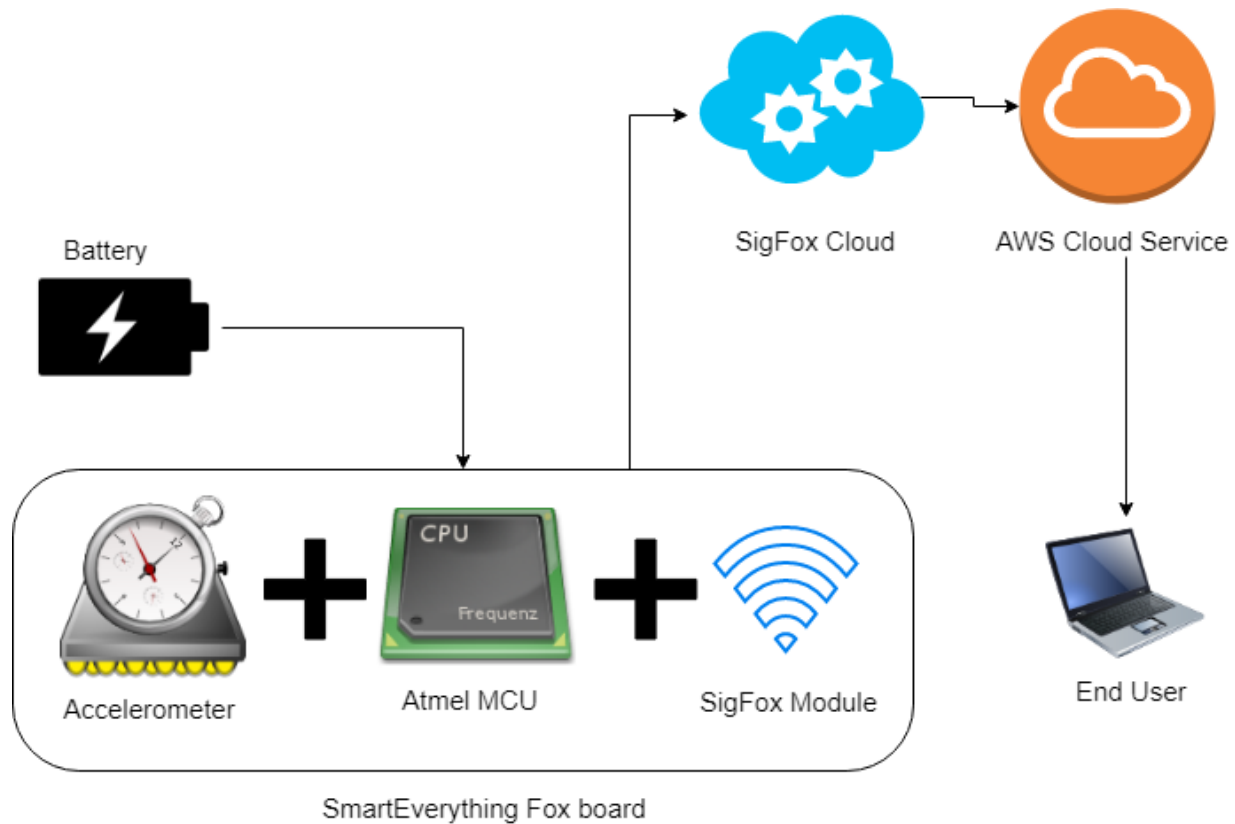


Figure 8 Hardware representation of the set-up [8]

From the hardware aspect the project comprises of the SME-F board with the with the ARM Processor, Accelerometer and SigFox Modules powered by a battery. Then the cross-over to the software part occurs were the projects execution takes place in the cloud as in Figure 8, once the SigFox module transmits the data the SME-F board does not have to be operational until the next reading.

3.1 Preparing Local Environment

Requirements:

Arduino IDE [4] and SME-F Libraries Installed.

Oracle VM VirtualBox [19] Installed.

Vagrant [18] and Laravel Homestead [12] Installed and Configured.

The initial set-up consisted of downloading and installing the Arduino IDE software, to avail of the SME-F board functionality with the Arduino IDE, the necessary libraries that were used in this project are listed below:

- Arduino SAMD Package – Used to communicate with the ARM MCU.
- Arduino Low Power Library – Used to manage the interrupts on the board.
- SmartEverything LSM9DS1 Library [9] – Used by the accelerometer.
- SmartEverything SIGFOX LE51-868 Library [9] – Used by the SigFox module.

3.2 Laravel Installation

Vagrant aids in quick creation and configuration of a virtual machine development environment, while Laravel provides a pre-configured environment called “Homestead”.

Using the Laravel provided guide to install and configure the Homestead package which is the pre-configured environment for Laravel application development, once the installation process for the local virtual machine environment is complete, the virtual machine is started by issuing a “vagrant up” command, when the VM initialization process is complete the development can begin. The first step is to install an open source bootstrap template “New Age” [15] which aids in speeding up the development process of the website.

3.3 SigFox Module registration

To register the SigFox module the ID and a Hex key provided with the SME-F development kit is required.

After registering for the SigFox network which is hosted by VT Networks in Ireland we can access the back-end user panel.

3.4 Roll and Pitch Readings

The first revision of code printed out the acceleration on each of the accelerometer axis to the console which can be seen in Figure 10, the goal of this was to check if the accelerometer was fully functional and most importantly to learn how the sensor was oriented on the SME-F board. Based on the readings it was possible to establish where the x, y and z co-ordinates were in respect to the board as demonstrated in Figure 9.

The LSM9DS \pm 1 sensor supports various levels of linear acceleration in the range of $\pm 2g$, $\pm 4g$, $\pm 8g$ and $\pm 16g$ meaning the accelerometers intensity can be adjusted to be sensitive to readings. This gives a possibility of deploying the sensor for different uses, in environments where a lot of everyday vibration is present such as passing trucks, the highest $\pm 16g$ setting should be used to minimize the amount of false readings caused by the vehicle vibrations. In sensitive environments where the small variation in the objects position is necessary the $\pm 2g$ setting should be used to sharpen the sensors detection.

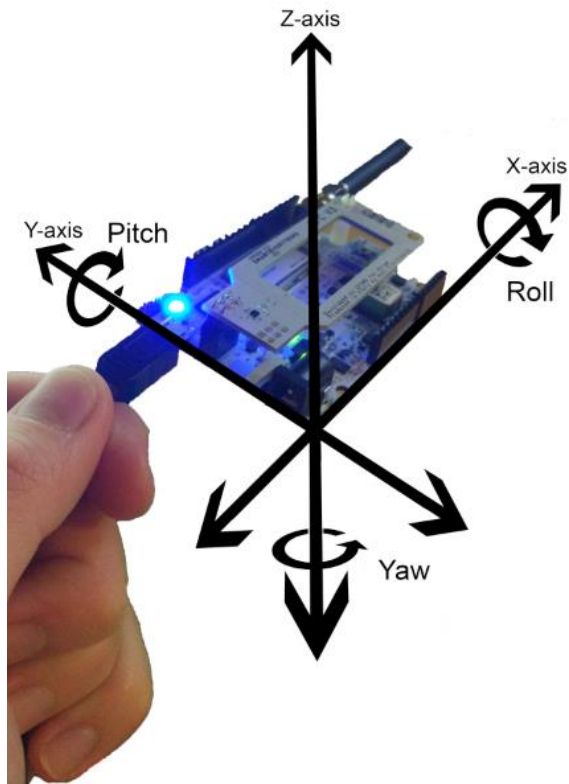


Figure 9 Representation of the axis in respect to the SME-F board



Figure 10 Accelerometer Test Results from Console

When passing the data to the SigFox module the value is multiplied by 10 to get rid of the “.” delimiter by converting the value from a decimal to an integer which introduces an additional byte in the payload that is rather limited, this way two bytes are saved.

3.5 SigFox Transmission

To test the SigFox module a code sample included in the SigFox library was used to transmit the message “Hello” to the SigFox cloud, when the message was registered it showed up in the back-end as seen Figure 12, one other essential element to observe is that the original message is in hex format and is only converted to ASCII in the back-end. Apart from the message sent we obtain the signal strength statistics visible in Figure 11 and a very rough location approximation seen in Figure 13. The rough location is calculated by the SigFox backend by taking the signal strength from the base stations and approximating the location using a probability model, because by design there will be three or more base stations available.



Figure 11 Hex and Signal Message Representation

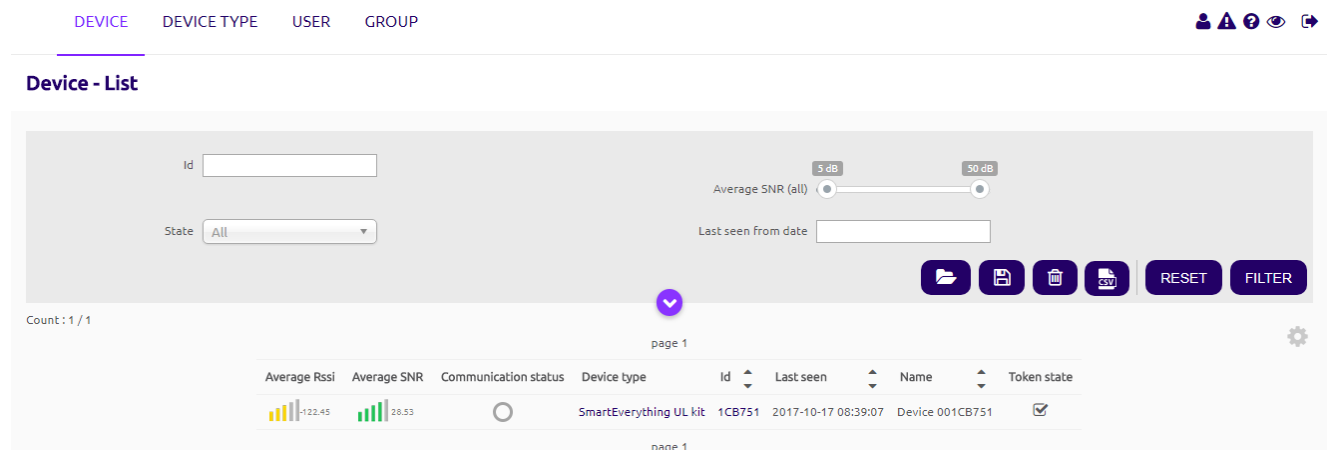


Figure 12 Back-End view of the SigFox cloud web service

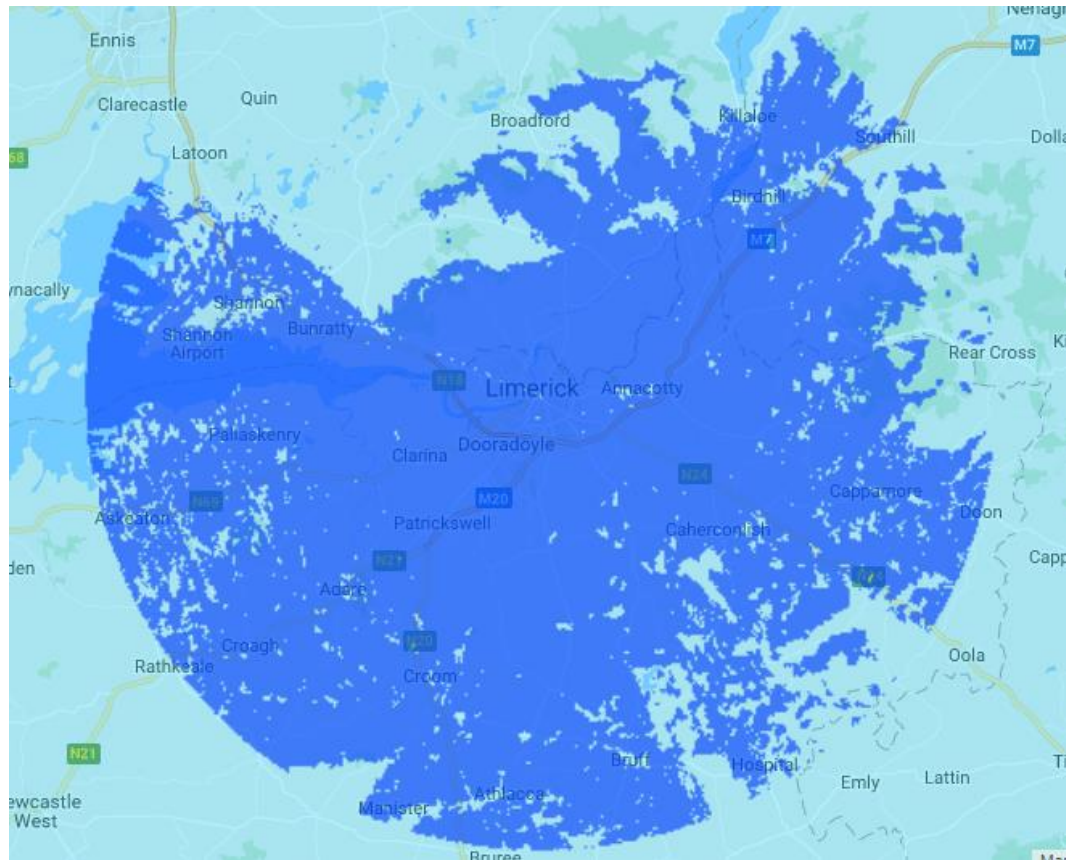
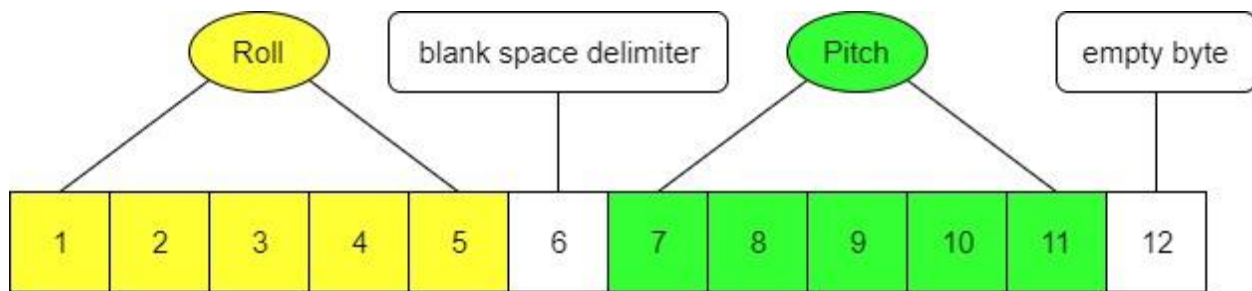


Figure 13 SigFox location approximation



12-byte SigFox payload.

Figure 14 SigFox Uplink Payload [8]

The SigFox network offers 12 bytes of payload [3] which will be used to transfer the roll and the pitch of the object, the blank space delimiter will be used to make the separation of the two measurements easier when inserting the information into a database as seen in Figure 14.

The Telit LE51-868S chip embedded on the SME-F board provides a Radio Data rate of 100bps this means that theoretically the payload should transmit in under 1 second $8\text{bits} * 12 = 96\text{bits}$, however there is more to the frame being transmitted than just the payload. The whole uplink frame generated by the SigFox module takes 26 bytes and contains the Authentication which allows the device to transmit data, the Device ID identifying the SigFox module, the Frame Check Sequence (FCS) which is used as a cyclic redundancy check which is an error-detection mechanism and finally the Payload itself as visible in Figure 15, thus the transmission time is increased to $8\text{bits} * 26 = 208\text{bits}$, $\frac{208\text{bits}}{100\text{bps}} = 2.08\text{ seconds}$. Although this transmission time might seem long for traditional applications, the slow transmission time and small frame size contribute to small power consumption by the module by not bursting huge amounts of data in a small time frame.

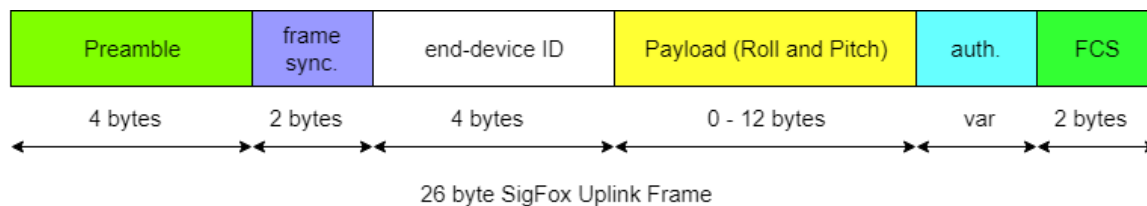


Figure 15 SigFox Uplink Frame [8]

3.6 Low-Power Software Design

Focusing on the low-power aspect application of the project, the SmartEverything Fox board must deliver functionality while consuming negligible power. To achieve that the on-board devices and sensors need to be disabled while not in use.

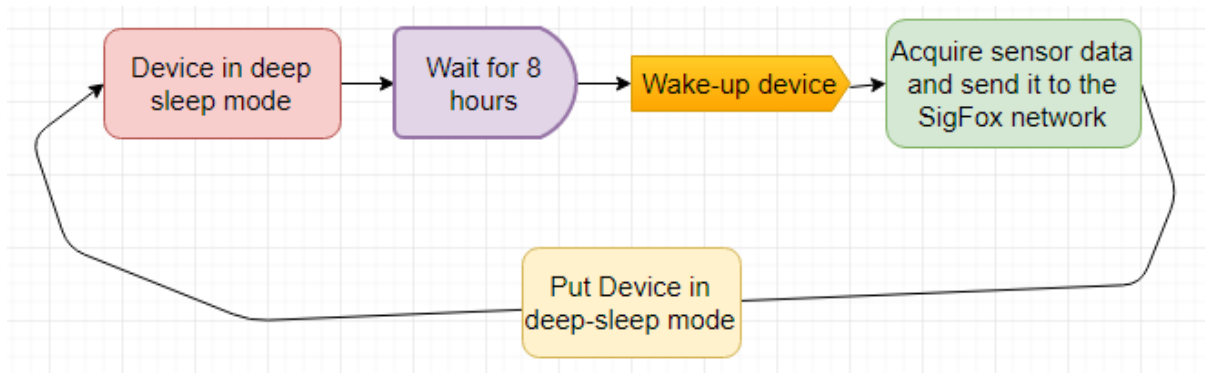


Figure 16 Low-Power Code Structure [8]

To optimize the battery life, the sensor data will be collected and sent to the SigFox network 3 times per day as visible above in Figure 16. Due to the optimization a much longer operating time will be achieved which means the unit will be maintenance free for a longer time.

The software libraries for the modules provide low-power modes which further reduces the amount of current used while acquiring or transmitting data.

3.7 RDS payload insertion

To insert the SigFox payload into the MySQL database a piece of python code which can be found in Appendix A is executed by the “Amazon Elastic Compute Cloud”. This is a very important operation as it reverts the original roll and pitch reading and splits the payload into appropriate columns as visible in Figure 17.

| # | Name | Datatype |
|---|-----------|----------|
| 1 | timestamp | DATETIME |
| 2 | deviceid | TEXT |
| 3 | roll | FLOAT |
| 4 | pitch | FLOAT |
| 5 | snr | TEXT |
| 6 | avgsnr | TEXT |

Figure 17 MySQL DB Table data types

To create the columns of the table “tilt_data” visible in Figure 17 MySQL code from Figure 18 is executed on the RDS database and the table is created. Now the database is prepared, and the Lambda function can successfully execute its code.

```
CREATE TABLE `tilt_data` (  
  `timestamp` DATETIME NOT NULL,  
  `deviceid` TEXT NULL,  
  `roll` FLOAT NULL DEFAULT NULL,  
  `pitch` FLOAT NULL DEFAULT NULL,  
  `snr` TEXT NULL,  
  `avgsnr` TEXT NULL,  
  PRIMARY KEY (`timestamp`),  
  INDEX `Index 1` (`timestamp`)  
)
```

Figure 18 MySQL table creation code

3.8 Website Design

The website is developed using a PHP open-source web framework called Laravel, Laravel provides an effortless way to deploy web application, the front-end represents the data collected and stored in the MySQL Database, a Bootstrap template “New-age” is used in combination with the Laravel PHP framework and is coded using mostly PHP, HTML, CSS and JavaScript. The role of the Bootstrap template is to provide a modern, responsive and cross-platform website that is user-friendly.

When implementing new features to the website the Laravel shipped online documentation proved itself helpful especially when the concept of Controllers and Routes was introduced, the pop-out calendar which allows setting the date of the last battery change was definitely the biggest challenge where the documentation and the use of various online resources has helped in overcoming the problem and ultimately achieving the finished website.

The overview of the entire system can be seen in Figure 19 which shows how the platforms come together and utilize all the services to create one seamless network that interchanges information. Starting from the bottom of the ladder where the actual hardware is implemented all the way up to the end-user level where the website can be viewed, and the data can be analyzed.

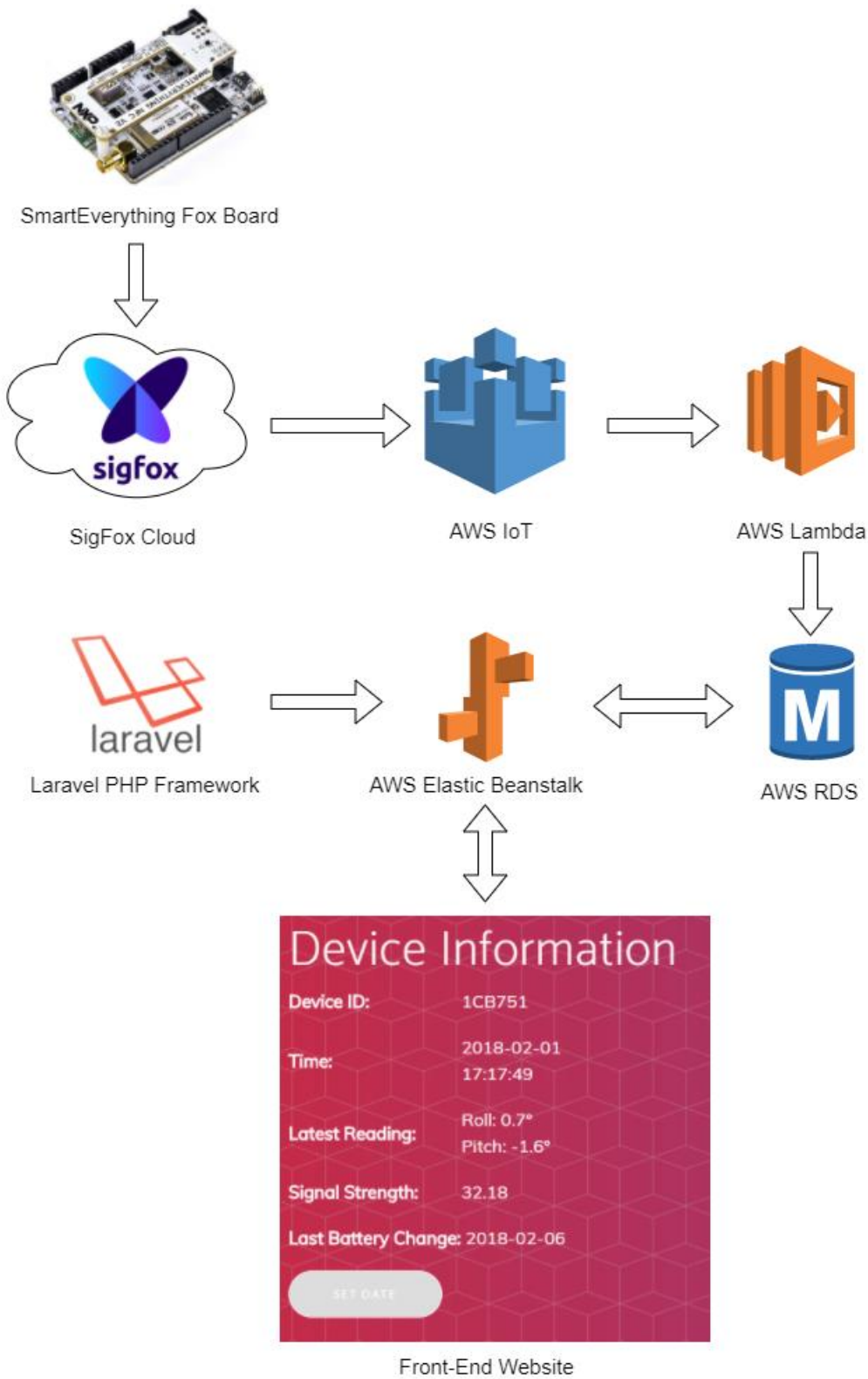


Figure 19 SME-F, SigFox and AWS Platform interaction [8]

4 Implementation & Integration

The final phase of the project is the integration of the AWS platform with the SigFox platform which consists of implementing the connectors that allow the communication of both platforms and effectively the exchange of data.

4.1 Uploading code to SmartEverything Fox board

To program the board, it is connected to the computer running the Arduino IDE and the upload of the code from Appendix A “Tilt.ino” to the board can be started on the selected COM port, once the upload is finished the board is disconnected from the computer.

4.2 Initialization of the board

Attach it or place it on the object whose tilt will be analyzed and connect it to the battery. The initialization process will go through the steps visible below in Figure 20 the “void setup()” function which initialize all the modules that will be used by the board. The 115200 in the “sfxAntenna.begin()” indicates the baud rate the SigFox module uses to communicate with the MCU over the serial UART in the SME-F board.

```
void setup() {  
  Wire.begin();  
  sfxAntenna.begin(115200); //initialize the SigFox module  
  smeAccelerometer.begin(); //initialize the accelerometer  
  SerialUSB.begin(115200); //initilize USB debugging console  
  sfxWakeup(); //in case the module is in sleep mode  
  delay(10);  
  setPowerSaveMode(); //put the module in power save mode  
  delay(1000*60); //Allow for reprogramming of the board  
}
```

Figure 20 SME-F board initialization snippet

After the above 1-minute initialization the first data point will be transmitted to the SigFox cloud by following a series of function calls that are designed to provide maximum power efficiency. Firstly the code enters in the “void loop()” function which is executed continuously as long as the device is running, in the loop the following steps are taken, at first the accelerometer is activated by calling the appropriate registers and given a small delay to prevent readback errors as per Figure 21.

```
smeAccelerometer.activate(); //Activating the sensor from Power Saving mode.
delay(70);
```

Figure 21 Accelerometer Activation Snippet

Once the sensor is activated, Figure 22 shows how the readback of the forces acting on the accelerometers x, y and z co-ordinates takes place and how the sensor is deactivated.

```
x = smeAccelerometer.readX(); //Reading the x-axis acceleration
y = smeAccelerometer.readY(); //Reading the y-axis acceleration
z = smeAccelerometer.readZ(); //Reading the z-axis acceleration
smeAccelerometer.deactivate(); //Accelerometer going into Power Saving mode.
```

Figure 22 Accelerometer X, Y, Z Read snippet

Then the SigFox module is woken up from the power saving mode and the calculation of the roll and pitch takes place, the “atan2(x, y)” function returns the results in radians therefore it must be converted to degrees by multiplying the radians result by 57.3 as pictured in Figure 23.

```
sfxWakeup(); //Wakeup the SigFox module
roll = atan2(y_Buff , z_Buff) * 57.3; // 57.3 = 180/PI (conversion to degrees)
pitch = atan2((- x_Buff) , sqrt(y_Buff * y_Buff + z_Buff * z_Buff)) * 57.3;
```

Figure 23 Roll and Pitch Calculation Snippet

Then the SigFox modules attempts to send the roll and pitch information by firstly communicating with the base stations the “SFX_DATA_ACK_START” identifier tells us the SigFox module is waiting for an answer, upon successful communication the “SFX_DATA_ACK_PROCESSING” identifier informs that the data processing has started and finally the “SFX_DATA_ACK_OK” identifier confirms successful message transmission. Once the last step is completed the SigFox module is put to sleep and the deep sleep mode is engaged for the set duration which in this case is 8 hours. In deep sleep mode all modules and internal timers are disabled except the one clock

that provides the interrupt to wake up the device, the power consumption during those 8 hours is reduced to 25 μA thanks to the Arduino Low Power library [20] [6] [7] that provides the “LowPower.deepSleep” function visible in Figure 24.

```
sfxSleep(); //Put The SigFoxmodule to sleep  
LowPower.deepSleep(8*60*60*1000);
```


Figure 24 Low-Power snippet

In the meantime, the callback event will be triggered in the SigFox cloud and after 3-5 seconds the data point will appear on the website.

4.3 Sigfox Cloud - AWS Connector

As previously mentioned in 2.4.1 the AWS IoT service is responsible for performing operations on the MySQL database when a callback is issued. Before IoT service can be configured, the creation of the Callback and the Stack in the AWS cloud must be performed simultaneously as authentication information is necessary from both parties. Firstly, a new Callback is created and a unique “External ID” is generated, this is the ID that will be used in AWS to inform the stack that it is to expect communication from this entity in this project being the SigFox platform. Then the AWS Stack is created and the “External ID” is pasted alongside some other details, once that is done the Stack generates the Amazon Resource Name (ARN) which is responsible for addressing the Stack, the ARN is pasted into the Callback creation screen visible in Figure 25. Finally, the payload which includes the Device ID, Data (Roll and Pitch), Time, Signal Strength and Average Signal Strength is formatted by addressing the appropriate variables in the JSON body, JSON is a lightweight data format which is easily interpreted by both humans and machines which makes the interchanging of data simple. The IoT service can now be configured to act upon receiving the transmission by the AWS Stack.


Device type SmartEverything UL kit - Callback edition

You can find complete documentation about AWS IoT following this [link](#). Click on  buttons to display help relative to a particular field.

Callbacks


Config method

CROSS_ACCOUNT

Launch Stack 


External Id

5a6b549e




ARN Role

arn:aws:iam:::role/SigfoxIoTConnector-crossAccountRoleForAWSIoT-




Topic

sigfox




Region

EU (Ireland)



Custom payload config



Json Body

```
{
  "device" : "{device}",
  "data" : "{data}",
  "time" : "{time}",
  "snr" : "{snr}",
  "avgSnr" : "{avgSnr}"
}
```

Figure 25 Callback Configuration

Issuing the callback triggers the HTTP POST request to create a new object which is later processed by calling the JSON variables and inserted into the MySQL database. The POST request operation carries the contents of the JSON body seen in Figure 25 and is directed at the AWS URI as seen in Figure 26. This process is part of the REST interface.



| Downlink | Enable | Channel | Subtype | Duplicate | Batch | Information |
|---|---|---------|--------------------------|--------------------------|---|-------------|
|  |  | UPLINK | <input type="checkbox"/> | <input type="checkbox"/> | [POST] https://[REDACTED].amazonaws.com/topics/sigfox?qos=1 | |

Figure 26 Uplink POST Configuration

4.4 Website

The function in Figure 27 shows how the charts and tables are created in the controller file. The charts are visualized using LavaCharts [10] which is an extension for PHP powered by the Google Chart API which produces interactive charts for the web. The data is fetched from the MySQL database using a previously defined connection to the database using a special “.env” file which contains all the necessary fields that allow connecting to the RDS database. This is an excellent show of Laravel’s simplicity when it comes to development as once the database connection information is defined the database can be referenced using the “DB” keyword. The “->get()” function retrieves the table from the database, the “->first()” keyword is used to retrieve the newest element of the table.

```
//Creates a chart of the signal strength gets data from DB
public function chart() {
    $lava=new Lavacharts;
    $signal=$lava->DataTable(); //SNR chart
    $roll=$lava->DataTable();
    $pitch=$lava->DataTable();
    $data=DB::table('tilt_data')->get();
    $device=DB::table('tilt_data')->latest('timestamp')->first();
    $bdate=DB::table('batteryDates')->latest('date')->first();

    $signal->addDateTimeColumn('Date')
        ->addNumberColumn('SNR(dB)')
        ->addNumberColumn('Average SNR(dB)');
```

Figure 27 Snippet of the PHP Chart function in the Controller file

Figure 28 shows a snippet of code responsible for the while loop that is used to insert all the data points into the named charts representing the Average Signal Strength, Roll and Pitch.

```
foreach ($data as $results) {  
    $signal->addRow([$results->timestamp,$results->snr,$results->avgsnr]);  
    $roll->addRow([$results->timestamp,$results->roll]);  
    $pitch->addRow([$results->timestamp,$results->pitch]);  
}
```

Figure 28 Graphs data point insertion

The charts data is then returned to the charts view page where it can be viewed using the return statement in Figure 29.

The created Charts are demonstrated in Figure 33

```
return View::make('chart', compact('lava','data','device','bdate'))  
);
```

Figure 29 Redirecting variables in Laravel

Laravel uses the concept of routes to navigate between pages, in Figure 30 the “Route:get” is responsible for retrieving data in form of variables from the controller file and pushing them to the chart sub-page, which in Laravel structure is denoted as “chart.blade.php”, the “blade” part comes from Laravels own unique powerful and simple templating engine which allows for easier integration of PHP language and HTML code. Similarly the “Route:post” is responsible for retrieving data from the subpage, when the user performs a post operation the controller takes in that data and processes it. For the purpose of this project the button “Set Date->Save Changes” triggers the POST operation and the controller file function as seen in Figure 31 inserts a new row into a table called “batteryDates” which stores the dates on which a battery change occurred to help the user keep track of how long a battery lasted and whether the power consumption and system behavior is as expected. Once the operation of inserting the data is complete the site view is returned to the “chart” subpage.

```
Route::get('/', function () {  
    return view('welcome');  
})->name('home');  
  
Auth::routes();  
  
Route::get('/chart', 'TiltController@chart')->name('chart');  
  
Route::post('/chart', 'TiltController@setDate')->name('chart');
```

Figure 30 Routing configuration in Laravel

```
public function setDate() {  
    $bdate = Input::post('bchange');  
    DB::table('batteryDates')->insert(['date' => $bdate]);  
  
    return redirect('chart');  
}
```

Figure 31 Inserting Battery Dates into DB

5 Results

The results of the accuracy and precision of the tilt measurements as well as the key Low-Power aspect where current consumption is analyzed, and the overall software functionality is discussed in this section and compared to the expectations of this project.

5.1 Battery & Power Results

For the power consumption testing phase, the code uploaded to the SmartEverything Fox board has been modified to decrease the testing time. The “cycle” was reduced to 12 minutes to maximize the amount of readings sent during a 24-hour frame while obeying the limits of the SigFox network which recommends to not send more than 140 messages in a single day. The 12 minutes time frame comes from simple math:

$$12 \text{ minutes} * 5 \text{ messages} = 60 \text{ minutes}$$

$$5 \text{ messages} * 24 \text{ hours} = 120 \text{ messages}$$

As it can be seen in the calculations above sending a message every 12 minutes gives us 120 messages per day which fits the 140-message limit recommended by the SigFox network.

The current consumed by the board during its stages of operation was measured used a multimeter which can be seen in Figure 34, three most important stages were taken into consideration: the initialization, the accelerometer x, y z force readout combined with the roll and pitch calculation executed using the MCU and the transmission to the SigFox network and finally the low power mode during which the device is in hibernation.

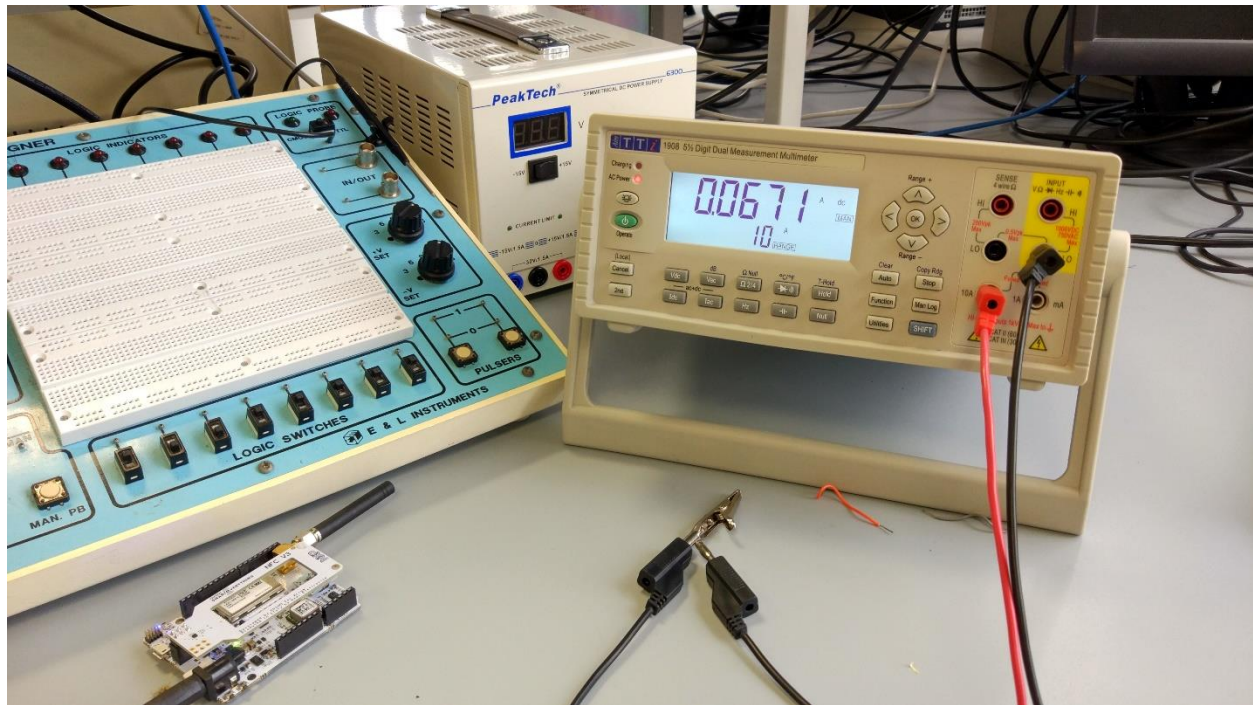


Figure 32 Power Consumption readout set-up

Table 2 Actual versus Theory Current Consumption of the SME-F board

| Power Source | 9V 500mAH | 600mAH 3.7V | Theory |
|--|--------------|-------------|------------|
| Initialization | 36mA | 60mA | 59.3mA |
| Accelerometer Read + SigFox Transmission | 67mA | 100mA | 57mA |
| Low Power Sleep Mode | 23mA | 37mA | 25 μ A |

During the test the board was able to send just slightly above 100 messages which proves the previous theory calculations wrong and significantly shortens the functioning time of the device.

In theory the battery efficiency came up to 6 months given 3 transmissions per day in other words the device would trigger every 8 hours send the tilt measurements and return to sleep mode.

The actual battery consumption provides $\frac{100}{3} = 33.3$ days of usage which roughly translates to a month, this can be lengthened by using a battery with more ampere hours.

When it comes to storage each of the payloads takes very little space the RDS instance is running a MySQL 5.6.37 database, the data stored in the database needs very little storage space therefore a 5GB database can store millions of events. A table with 100 rows takes 0.03MB of storage thus 1MB provides 3333 rows, $1000 * 3333 = 3.33$ million rows. Further demonstrating the projects scalability and the possibility to handle hundreds of incoming payloads over prolonged periods of time by the single database.

The website <http://tiltapp.eu-west-1.elasticbeanstalk.com> features a flawless design and provides a user-friendly interface that is simple to navigate as intended, it is also fully functional on mobile platforms which means the end-user can monitor their device on the go. The charts feature all the necessary details from the payload and additionally the results can be viewed in a simple table where they can be organized by their timestamp, tilt, pitch or snr. Additionally, in case of large result tables, the data can be narrowed down using the search function.

5.2 Data Upload Results

| ▲ timestamp | ▼ deviceid | roll | pitch | snr | avgsnr |
|---------------------|------------|------|-------|-------|--------|
| 2018-02-15 14:42:42 | 1CB751 | 0.1 | 0.1 | 15.50 | 26.18 |
| 2018-02-15 14:55:43 | 1CB751 | 0 | 0.2 | 19.86 | 26.11 |
| 2018-02-15 15:08:42 | 1CB751 | 0 | 0.1 | 13.78 | 26.09 |
| 2018-02-15 15:21:43 | 1CB751 | 0.1 | 0.1 | 23.62 | 26.08 |
| 2018-02-15 15:34:42 | 1CB751 | 0 | 0.1 | 19.07 | 26.14 |
| 2018-02-15 15:47:42 | 1CB751 | 0 | 0.2 | 11.86 | 26.07 |
| 2018-02-15 16:00:43 | 1CB751 | 0.1 | 0.2 | 11.63 | 26.01 |

Figure 33 Results taken from MySQL DB

In Figure 32 which shows an extract of the MySQL database we can observe the accuracy of the sensors readings, the SME-F board was placed on a flat surface and left stationary while transmitting data every 12 minutes as indicated by the timestamps. The accelerometers readings converted to Roll and Pitch using the formulas presented in this paper prove their correctness as the results are in the 0.0-0.2 range which shows the accuracy of the measurement on the flat

surface, the flatness of the surface was verified using a digital spirit level and a mobile phones accelerometer both of which showed results in a similar range of 0 – 1 degree. What is particularly interesting is the variation in the signal to noise ratio, in this case the difference between the highest and lowest value of snr is $23.62 - 11.63 = 11.99dB$ which is a clear indication to how various signals around the device can impact and interfere with the devices snr even though its stationary.

Charts

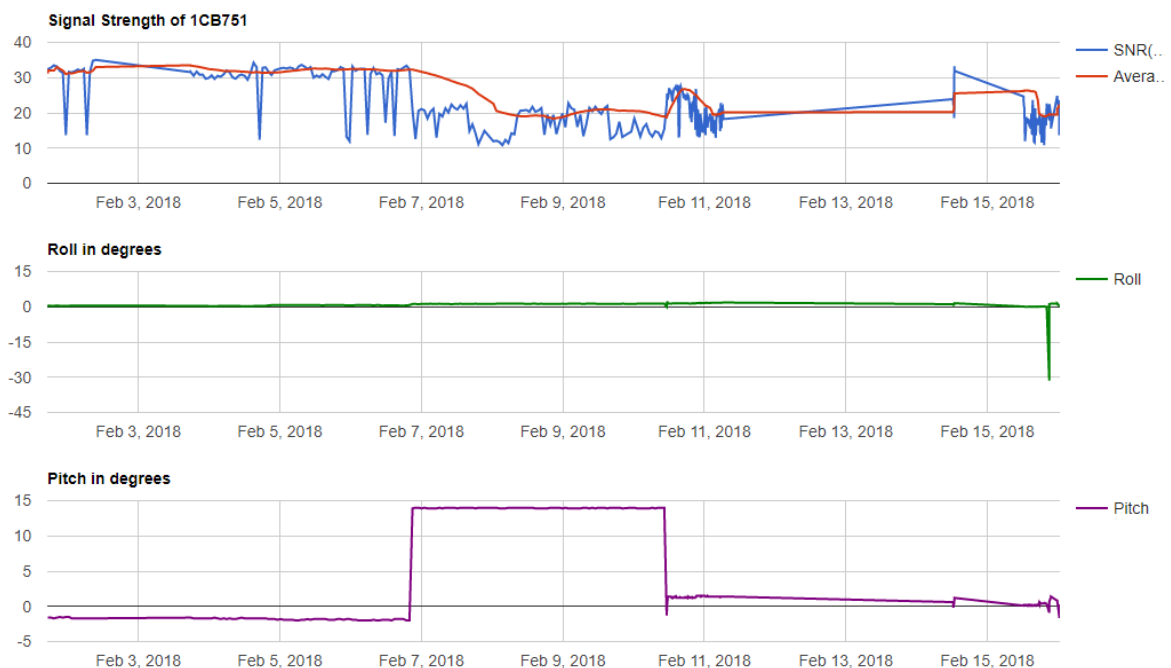


Figure 34 Website charts created by LavaCharts

5.3 Software Errors

After the real test was concluded and the results were analyzed it was discovered that, while the reading and transmission current of 60mA is within the theory calculations of 50mA, the deep sleep current seemed extremely high and floats around 20mA comparing to the theoretical 25uA which would greatly improve the longevity of the battery powering the SME-F board. The most probable case is that some current is leaking in the board due to a possible defect, after browsing through Arduino forums no other users of the SAMD21 experienced such high currents in sleep mode. To confirm this theory a second board would have to be bought unfortunately due to the excessive cost of the board this is not possible.

5.4 Possible Improvements

In terms of deploying the device in open environments, the possibility to add a solar panel exists to significantly extend the maintenance free period by providing uninterrupted power supply throughout the day from the solar panel and power from the battery during the night, if a Lithium-Ion battery is used the possibility to recharge the battery with the excess current during the day also exists to theoretically infinitely extend the run time of the device.

If more precise location of the sensor would be necessary there is a possibility to enable the GPS module that is already on the SME-F board, it is worth to note that this would significantly increase the current usage at transmission time as per the modules datasheet [21].

6 Conclusion

First, it's worth mentioning that this project was a major success as it was completed to its formerly set out specifications and works without fail, the SmartEverything Fox board processes and transmits the data to the SigFox server as often as it is necessary basing on the values set out by the user while the SigFox and AWS clouds process the information that is displayed on the website.

In regard to the communication network used to connect the SME-F board to the world the SigFox LPWAN does exactly as designed to and it is the perfect choice for this project as the accelerometer data payload does not exceed the networks uplink limitations, the power consumption of the SigFox module was designed to be as low as possible and naturally SigFox was created with the IoT as the main driving force behind the invention.

The complexity and at the same simplicity of the Amazon Web Services platform greatly helped to finalize this project and assemble it together by supplying all the necessary services that helped both the front-end as well as the back-end of the project. At the beginning it was difficult to understand which services are going to be of most use as AWS offers an abundance of applications which can be configured and used in many ways although after reading multiple guides and experimenting with various applications AWS truly appeals to the user and it is worth to familiarize oneself with the platform and the services it offers.

One particularly fascinating and the most compelling factor of the project is the possibility to scale it and deploy hundreds of SME-F devices across the country without any changes to the code and have them provide accurate tilt measurements from multiple locations at the same time.

The complete system functions end to end meaning that it is configured properly and can be deployed into the field with full functionality which is the standard set-out by international companies when delivering a product to their customer. This directly influences one of the secondary goals of this project to minimize the user input to an absolute minimum and this project achieves that by having virtually no user input at all.

If this project would be reproduced to cut down costs the SME-F would be replaced by a custom-built board with only the three hardware modules the SigFox, the Accelerometer and the MCU.

7 References and sources of information

- [1] Amazon Web Services, Inc. (2018). Amazon Web Services (AWS) - Cloud Computing Services. [online] Available at: <https://aws.amazon.com>.
- [2] Anon, (n.d.). DFRobot - Quality Arduino Robot IOT DIY Electronic Kit.
- [3] Anon, (n.d.). Sigfox - The Global Communications Service Provider for [online] Available at: <https://www.sigfox.com/en>.
- [4] Arduino.cc. (2018). Arduino - Home. [online] Available at: <https://www.arduino.cc/>.
- [5] Arrow Electronics SmartEverything IoT Bluetooth Smart (BLE), S. (2018). MCS7561 | Arrow Electronics SmartEverything IoT Bluetooth Smart (BLE), GLONASS (GNSS), GPS, Near Field Communication (NFC), SigFox | Arrow Electronics. [online] ie.rs-online.com. Available at: <http://ie.rs-online.com/web/p/radio-frequency-development-kits/9015121/>.
- [6] AT06549: Ultra Low Power Techniques. (2015). [PDF] Atmel, pp.3-11. Available at: http://ww1.microchip.com/downloads/en/AppNotes/Atmel-42411-Ultra-Low-Power-Techniques-AT06549_Application-Note.pdf.
- [7] AT11491: Peripheral Power Consumption in Standby Mode for SAM D Devices. (2015). [PDF] Atmel, pp.3-6. Available at: http://ww1.microchip.com/downloads/en/AppNotes/Atmel-42472-Peripheral-Power-Consumption-in-Standby-Mode-for-SAM-D-Devices_ApplicationNote_AT6490.pdf.
- [8] Draw.io. (2018). Flowchart Maker & Online Diagram Software. [online] Available at: <https://www.draw.io>.
- [9] GitHub. (2018). SmartEverything. [online] Available at: <https://github.com/ameltech>.
- [10] Hill, K. (2018). Lavacharts. [online] [Lavacharts.com](http://lavacharts.com/). Available at: <http://lavacharts.com/>.
- [11] LSM9DS1 - iNEMO inertial module: 3D accelerometer, 3D gyroscope, 3D magnetometer Datasheet. (2015). [PDF] ST. Available at: <http://www.st.com/content/ccc/resource/technical/document/datasheet/1e/3f/2a/d6/25/eb/48/46/DM00103319.pdf/files/DM00103319.pdf/jcr:content/translations/en.DM00103319.pdf>.
- [12] Otwell, T. (2018). Laravel - The PHP Framework For Web Artisans. [online] [Laravel.com](https://laravel.com). Available at: <https://laravel.com>.
- [13] Pedley, M. (2018). Tilt Sensing Using a Three-Axis Accelerometer. [PDF] Freescale/NXP, pp.4-19. Available at: <https://www.nxp.com/docs/en/application-note/AN3461.pdf>.
- [14] St.com. (2018). Home - STMicroelectronics. [online] Available at: http://www.st.com/content/st_com/en.html.
- [15] Start Bootstrap. (2018). New Age - One Page Bootstrap Theme. [online] Available at: <https://startbootstrap.com/template-overviews/new-age/>.

- [16] Stmicroelectronics.com.cn. (2018). LSM9DS1 - iNEMO inertial module, 3D magnetometer, 3D accelerometer, 3D gyroscope, I2C, SPI - STMicroelectronics. [online] Available at: <http://www.stmicroelectronics.com.cn/en/mems-and-sensors/lsm9ds1.html>.
- [17] Telit_LE51-868_S_Datasheet. (2015). [PDF] Telit. Available at: https://www.telit.com/wp-content/uploads/2017/09/Telit_LE51-868_S_Datasheet.pdf.
- [18] Vagrant by HashiCorp. (2018). Vagrant by HashiCorp. [online] Available at: <https://www.vagrantup.com>.
- [19] Virtualbox.org. (2018). Oracle VM VirtualBox. [online] Available at: <https://www.virtualbox.org>.
- [20] GitHub. (2018). ArduinoLowPower. [online] Available at: <https://github.com/arduino-libraries/ArduinoLowPower>.
- [21] ASME Fox 3 User Guide. (2016). [PDF] Axel Elettronica. Available at: <https://static4.arrow.com/-/media/files/pdf/asme-fox-3-user-guide-v1-3.pdf>.
- [22] SAMD21J18 Datasheet. (2017). [PDF] Microchip Technology. Available at: <http://ww1.microchip.com/downloads/en/DeviceDoc/40001882A.pdf>.

Appendix A

Code uploaded to the ARM Processor in the SmartEverything Fox board by Arduino IDE.

```
/*
    Low-Power Wireless Inclinator

    By: Piotr Kurzynoga ID: 14143097

    This code focuses on the low power aspect and delivery of the coordinates to
    the SigFox framework.
*/

#include <Arduino.h>
#include <Wire.h>
#include <ArduinoLowPower.h>
#include <SmeSFX.h>
#include <LSM9DS1.h>

//Counter variables.
unsigned long timepassed=0;
long last_time=0;
int interval=60-1; //desired time -1 to make up for the > sign.

// the setup function runs once when you press reset or power the board
void setup() {
    delay(1000);
    Wire.begin();
    sfxAntenna.begin(115200);
    smeAccelerometer.begin();
    SerialUSB.begin(115200);
    sfxWakeup();
    delay(10);
    setPowerSaveMode();
}

//function puts the SigFox chip into PowerSave mode
static void setPowerSaveMode(void){
    int initFinish = 1;

    SerialUSB.println("SFX in Command mode");
    sfxAntenna.setSfxConfigurationMode(); // enter in configuration Mode

    do {
        uint8_t answerReady = sfxAntenna.hasSfxAnswer();
        if (answerReady) {
            switch (initFinish) {
                case 1:
                    // set the PowerSave mode
                    sfxAntenna.setSfxSleepMode(SFX_HW_WAKE);
                    sfxAntenna.setSfxDataMode();
                    initFinish++;
                    break;
            }
        }
    } while (1);
}
```

```

    }
    }
    } while (initFinish != 2);
}

void RP_calculate(int x,int y, int z, double roll, double pitch){
    sfxWakeup();
    delay(10);
    char messageBuffer[12];
    memset(messageBuffer, 0, sizeof(messageBuffer));
    double x_Buff = float(x);
    double y_Buff = float(y);
    double z_Buff = float(z);
    roll = atan2(y_Buff , z_Buff) * 57.3; // 57.3 = 180/PI (conversion to degrees)
    pitch = atan2((- x_Buff) , sqrt(y_Buff * y_Buff + z_Buff * z_Buff)) * 57.3;
    int rollint=roll*10; //multiplying by 10 and converting to an integer
    int pitchint=pitch*10; //multiplying by 10 and converting to an integer
    char message[11]; //only 11 bytes need to be prepared for the payload
    sprintf(message, "%d %d", rollint, pitchint);
    tilt.toCharArray(messageBuffer,12);
    SerialUSB.print("Roll: ");
    SerialUSB.print(roll);
    SerialUSB.print("Pitch: ");
    SerialUSB.println(pitch);
    SerialUSB.println(message);
    sfxAntenna.sfxSendData(message, strlen((char*)message));
}

// the loop function runs over and over again forever
void loop() {

    timepassed=(millis()/1000);
    if(timepassed - last_time > interval) // greater or equal in case sync
issues occur.
    {
mode.
        smeAccelerometer.activate(); //Activating the sensor from Power Saving
mode.
        delay(70);
        ledGreenLight(LOW);
        ledBlueLight(HIGH);
        int x = 0;
        int y = 0;
        int z = 0;
        double roll=0.00,pitch=0.00;
        x = smeAccelerometer.readX(); //Reading the x-axis acceleration
        y = smeAccelerometer.readY(); //Reading the x-axis acceleration
        z = smeAccelerometer.readZ(); //Reading the x-axis acceleration
        last_time=timepassed;
        smeAccelerometer.deactivate(); //Accelerometer going into Power Saving
mode.
        RP_calculate(x, y, z, roll, pitch);
    }

    //Checking if there is contact with the SigFox base station.
    bool answerReady = sfxAntenna.hasSfxAnswer();

    if (answerReady) {

```

```

    if (sfxAntenna.getSfxMode() == sfxDataMode) {

        switch (sfxAntenna.sfxDataAcknowledge()) {
        case SFX_DATA_ACK_START:
            SerialUSB.println("Waiting Answer");
            break;

        case SFX_DATA_ACK_PROCESSING:
            SerialUSB.print('.');
            break;

        case SFX_DATA_ACK_OK:
            #ifndef ASME3_REVISION
                ledBlueLight (LOW);
                ledGreenLight (HIGH);
            #endif
            SerialUSB.println(' ');
            SerialUSB.println("Answer OK :) :) :) :)");
            sfxSleep(); //Put The SigFox module to sleep
            LowPower.sleep(60*60*1000); //1000ms*60=1minute*60=1 hour
            break;

        case SFX_DATA_ACK_KO:
            #ifndef ASME3_REVISION
                ledRedLight (HIGH);
            #endif
            SerialUSB.println(' ');
            SerialUSB.println("Answer KO :( :( :( :(");
            break;
        }
    }
}

```

Python Code used in the Lambda function to forward the payload from AWS IoT to AWS RDS.

```

import sys
import logging
import rds_config
import pymysql
import datetime
#rds settings
rds_host = 'aahme8ax00zfyta.cbrrgt5cpry1.eu-west-1.rds.amazonaws.com'
name = rds_config.db_username
password = rds_config.db_password
db_name = rds_config.db_name

logger = logging.getLogger()
logger.setLevel(logging.INFO)

#connecting to the database
try:
    conn = pymysql.connect(rds_host, user=name, passwd=password, db=db_name,
connect_timeout=5)

```

```

except:
    logger.error("ERROR: Unexpected error: :( Could not connect to MySQL
instance.")
    sys.exit()

logger.info("SUCCESS: Connection to RDS mysql instance succeeded")
def handler(event, context):
    """
    This function fetches content from mysql RDS instance
    """

    item_count = 0

    with conn.cursor() as cur:
        #converting the hex payload to String
        data=bytearray.fromhex(event['data']).decode()
        #seperating the data into Roll and Pitch
        datarray=data.split()
        #converting back to the original decimal format
        roll=float(datarray[0])/10
        pitch=float(datarray[1])/10
        #converting the timestamp to a date time format

    date=datetime.datetime.fromtimestamp(int(event['time'])).strftime('%Y-%m-%d
%H:%M:%S')

    #inserting the data into the db
    message='insert into tilt_data (timestamp, deviceid, roll, pitch, snr,
avgsnr)
            values("{}","{}","{}","{}","{}",'
"{}")'.format(date,event['device'],roll,pitch,event['snr'],event['avgSnr'])
    cur.execute(message)
    conn.commit()

    return "Added %s items from RDS MySQL table" %(event['data'])

```

Poster

Low-Power Wireless Inclinometer



Piotr Kurzynoga

BEng in Electronic and Computer Engineering

Introduction

The focus of this project is a battery powered inclinometer using the SmartEverything Fox board in Figure 1. The board uses an ARM Cortex M0+ processor to drive the onboard SigFox module and the Accelerometer sensor.

This projects aim is to detect variations in a buildings tilt to prevent permanent damage to the structure.



Figure 1 SmartEverything Fox board

The encrypted SigFox network transmits data securely to the cloud where it is pushed to (AWS) Amazon Web Services for processing and protected storage.

Method

The accelerometer takes readings of the forces acting on it and the ARM MCU calculates the roll and pitch based on the X, Y and Z components seen in Figure 2 and uses the SigFox module to transmit the payload with the calculations.

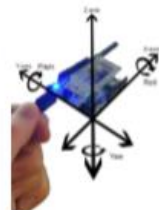


Figure 2 Roll and Pitch representation in respect to the board

This payload is logged in the SigFox cloud and sent to the AWS platform where it is processed and inserted into a database which the website uses to present the data visually.

Inside the AWS platform as presented in Figure 3, the SigFox data is stored securely in a MySQL DB, the website is based on the Laravel PHP framework which allows for unique communication with the DB using the Eloquent model and quick Elastic Beanstalk deployment of the application.

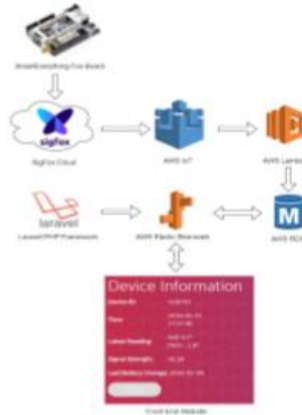


Figure 3 SmartEverything Fox, SigFox and AWS interaction

Results & Conclusion

To fulfil the Low-Power aspect of the project the software is designed to operate in energy saving modes, this allowed to reach low current consumption and long operating times that can span up to 6 months.

Tests visible in Figure 4 were carried out to measure the current consumption when the device is sleeping and transmitting messages to calculate the operational time.



Figure 4 Current Consumption Measurement

The website contains all the data from the MySQL DB and displays the Signal, Roll and Pitch information so the user can spot any outliers immediately in case of a rapid tilt change as seen in Figure 5.



Figure 5 Visual representation of the data

This project can be concluded with its main advantages which are:

- Low-Power Consumption of 25µA during hibernation
- Precise tilt measurement up to 0.1 degree
- Data security & confidentiality in the cloud
- No user input throughout the devices operation

Personal reflection

Having successfully finished the project and fulfilled all the requirements given, during the 7 months the project helped me to gain insight and knowledge on new technologies mainly on using SigFox, AWS and Laravel which will without a doubt benefit me in my future career.

Acknowledgements

I would like to thank my supervisor Dr Richard Conway for his help and my family for supporting me throughout this project.

E&CE Department of Electronic & Computer Engineering