IOT USING LORAWAN

TEAM F

MICHAEL OWCZAREK

JOEL MORRAN  
LUIS SHIGETOMI

REMI PETIT

JAMES CURNOW

2019

Table of Contents

Title Page

Copyright Information

Warranties and Contractual Obligations

Table of Changes

Table of Contents

Table of Figures

1.0 - Introduction

1.1 - System Overview

1.2 - Audience Description

1.3 - Applicability Statement

1.4 - Purpose Statement

1.5 - Document Usage Description

1.6 - Conventions

1.7 - Change Log

2.0 - Software Design Scope

2.1 - Major Software Functions

2.2 - Major Design Constraints and other Requirements

3.0 - Reference Documents

3.1 - Existing Software Documentation

3.2 - System Documentation

3.3 - Vendor Documentation

3.4 - Other Documentation

4.0 - User Stories

4.1 - User Story Dictionary

4.2 - Iterative User Story Documents

4.2.1 - User Story Definition

4.2.2 - Flow of Interaction Diagram

4.2.3 - User Story Testing

4.2.4 - Wireframes

5.0 - Object-Oriented Design

5.1 - High Level System Architecture

5.3 - Use Case Analysis

5.4 - Domain Model and Class Diagram

5.5 - Establishment of the Database Objects and Data Access Strategy

5.7 - Object Dictionary

6.0 - Software release report

6.1 - Usability Test Report

6.2 - System (User Story) Test Report

8.0 - Setup/Installation Guide

8.1 - Installation of the Raspbian operating system for Pi

8.2 - Gateway (Raspberry Pi) Setup

8.3 - Arduino Setup

8.4 - Azure Setup

8.5 - Power BI

9.0 - Special Notes

9.1 - Appendices

9.2 - Bibliography

9.3 - Glossary of Terms

9.4 - Reference Materials

9.5 - Index

9.6 - Statement of effort

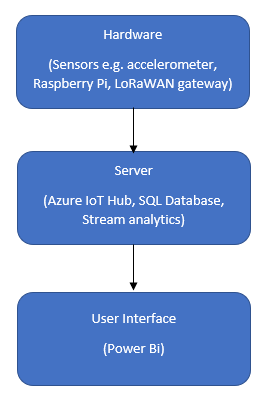
9.7 - Inspect and Verified by (Quality Assurance)

1.0 - Introduction

1.1 System Overview

The integration of IoT using LoRaWAN is a network that has been developed to help visualize sensor data being generated by an entity. The data can be represented graphically in multiple statistical reporting formats using tools on Power BI.

The block diagram below portrays the system.



**Hardware:** Consists of the physical components of the software responsible for sensor data transmission to the server side of the project

**Server:** Home to the IoT hub responsible for storing and managing data

**User Interface:** The viewable interface by users containing graphical data

1.2 Audience Description

The IoT using LoRaWAN is intended to appeal towards a demographic above the age of 18. Users of the system should be technologically aware and be able to understand statistics represented in a graphical manner.

The users of the system include;

* + CTI Staff on the LaTrobe University campus

Users are expected to have access to a phone or computer that has internet access so that they can connect Power Bi.

1.3 Applicability Statement

The software runs on any web browser capable of supporting Power Bi. Power Bi is a service provided by Microsoft. Access to Power Bi requires an internet connection.

1.4 Purpose Statement

The purpose of the IoT using LoRaWAN project is to:

* + Create a visualization environment for sensor activity
  + Provide an environment capable of handling additional sensors
  + Provide separate restricted views of data for users based on their role in the system
  + Display sensor specific data in an understandable format

1.5 Document Usage Description

The following provides a brief summary of the 8 main sections of the document:

**1. Introduction:**

Consists of a short description outlining an overview of the system, it also discusses the audience and provides purpose of the usage of the document. The conventions used on the document and changelog are also included.

**2. Software Design Scope:**

Outlines a couple of the software components that are used in a predominately hardware-based project. Any design constraints and further requirements are also included.

**3. Referencing Documents:**

Provides a detailed description of any documentation used in relation to the system.

**4. User Stories:**

Consists of user stories that are taken into the consideration of the system. A user story dictionary as well as testing and wireframes are also included.

**5. Object-Oriented Design:**

Any diagrams and descriptions relating to the architecture of the system are included in this part of the document.

**6. Software Release Report:**

System usability and any testing performed are included in this section of the documentation.

**7. Additional User Interface Design**

Any user interface design diagrams are included in this domain of the document.

**8. Special Notes**

Additional information is included in this section of the document such as a glossary or any further comments that are not included elsewhere.

1.6 Conventions

The project being documented is strongly hardware based resulting in any documentation regarding the software less heavily documented.

1.7 Change Log

|  |  |  |
| --- | --- | --- |
| Date | Change Made | Changed By |
| 9 June 2019 | Revised formatting | All |
| 7 June 2019 | Revised document created | All |
| 7 June 2019 | 1.7 Changelog | James Curnow |
| 3 June 2019 | 4.2 Iterative user story documents | Michael Owczarek |
| 29 May 2019 | Table of contents | Michael Owczarek |
| 29 May 2019 | 8.1 Gateway setup | Joel Morran |
| 26 May 2019 | 8.4 Power Bi | Luis Shigetomi |
| 26 May 2019 | 4.0 User stories | Joel Morran |
| 26 May 2019 | 3.3 Vendor Documentation | Remi Petit |
| 25 May 2019 | 1.5 Document usage description | James Curnow |

2.0 - Software(u mean hardware?) Design Scope

2.1 - Major Software Functions

Showing a report

2.2 - Major Design Constraints and other Requirements

Teams will deliver the project with the following outcomes:

A group of sensors connected through LoRaWAN to a gateway

A connection through the gateway directly to the IoT platform for sending and receiving the collected data

An IoT platform set up for managing the data to and from the sensors

A visualisation environment to illustrate the sensors activities

A data processing algorithm to extract features and detect certain patterns in the logged data

3.0 - Reference Documents

3.1 - Existing Software Documentation

Any existing documents related to the software being designed are listed here:

Project Description: [2019 Project Summary](https://lms.latrobe.edu.au/pluginfile.php/4632170/mod_resource/content/6/2019%20Industry%20Project%20Summary.pdf) (Project 1)

LoRaWan CTI setup.pdf

3.2 - System Documentation

The software is not being developed from an embedded existing system.

3.3 - Vendor Documentation

**Microsoft Azure:** Microsoft Azure is a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through Microsoft-managed data centers.

<https://azure.microsoft.com/en-au/>

<https://docs.microsoft.com/en-us/azure/iot-edge/how-to-install-iot-edge-linux-arm>

<https://azure.microsoft.com/en-au/resources/samples/custom-vision-service-iot-edge-raspberry-pi/>

**Raspberry Pi:** The Raspberry Pi is a series of small single-board computers used to promote teaching of basic computer science in schools and in developing countries.

<https://www.raspberrypi.org/documentation/>

**Arduino:** Arduino is an open-source hardware and software company, project and user community that designs and manufactures single-board microcontrollers and microcontroller kits.

<https://www.arduino.cc/en/Main/Docs>

**Seeeduino:** Seeeduino V4.2 is an Arduino-compatible board, which is based on ATmga328P MCU.

<http://wiki.seeedstudio.com/LoRa_LoRaWan_Gateway_Kit/>

<http://wiki.seeedstudio.com/Seeeduino_LoRAWAN/>

<http://wiki.seeedstudio.com/Grove-IMU_9DOF_v2.0/>

**Power BI:** Power BI is a business analytics service by Microsoft. It aims to provide interactive visualizations and business intelligence capabilities with an interface simple enough for end users to create their own reports and dashboards.

<https://powerbi.microsoft.com/en-us/learning/>

<https://docs.microsoft.com/en-us/power-bi/desktop-getting-started>

Python: Python is an interpreted, high-level, general-purpose programming language.

<https://docs.python.org/3/>

3.4 - Other Documentation

References used for hardware setup mostly for the PI

<https://onedrive.live.com/view.aspx?cid=d9f0c8185603d174&page=view&resid=D9F0C8185603D174!20181&parId=D9F0C8185603D174!20132&authkey=!AFmCQcn4oTFxpbg&app=Word>

4.0 - User Stories

4.1 - User Story Dictionary

“As a user, I want a clean UI which is easy to understand”

“As a user, I want an easy setup for the hardware & software”

“As a user, I want the solution to be cost effective”

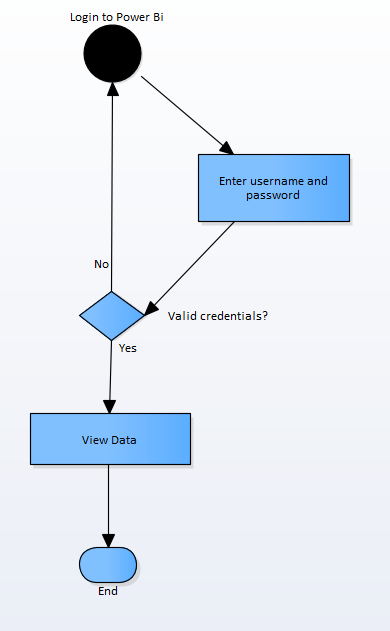
“As a user, I want the data to be easy to export”

4.2 - Iterative User Story Documents

4.2.1 - User Story Definition

|  |  |
| --- | --- |
| **US-1**    As a user, I want a clean UI which is easy to understand | **Acceptance Criteria**  1. Easy to understand and Clean UI (pass)  2. Hard to understand and Clunky UI (fail) |
| **US-2**    As a user, I want an easy setup for the hardware & software | **Acceptance Criteria**  1. Easy setup(pass)  2. Hard/Complex setup(fail)  3. Setup requires and expert(fail) |
| **US-3**    As a user, I want the solution to be cost effective | **Acceptance Criteria**  1. Cost effective(pass)  2. Not cost effective(fail) |
| **US-4**    As a user, I want the data to be easy to export | **Acceptance Criteria**    1. Data easy to export(pass)  3. 2. Hard to export(fail)  3. 3. No export supported(fail) |

4.2.2 - Flow of Interaction Diagram

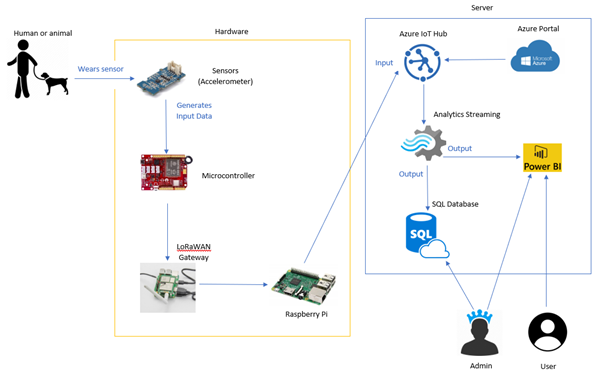


4.2.3 - User Story Testing

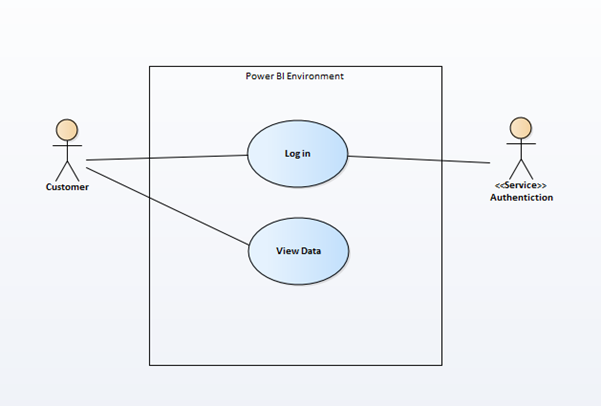
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User Story** | **Test ID** | **Acceptance Criteria** | **Description** | **Status** |
| US-1 | US1T-1 | Easy to understand and Clean UI (pass) | Users do not have difficulty understanding and navigating the UI | PASS |
|  | US1T-2 | Hard to understand and Clunky UI (fail) | Users have difficulty understanding and navigating the UI and find it confusing and clunky | FAIL |
| US-2 | US2T-1 | Easy setup(pass) | Users find the setup process to be easy based on the documentation provided | PASS |
|  | US2T-2 | Hard/Complex setup(fail) | Users find the setup process to be hard and complex and to technical for them based on their skill level and poor documentation/no documentation provided | FAIL |
|  | US2T-3 | Setup requires and expert(fail) | The setup requires and expert to complete the process | FAIL |
| US-3 | US3T-1 | Cost effective(pass) | Cost are within/ below extreme low-cost mass production targets | PASS |
|  | US3T-2 | Not cost effective(fail) | Cost exceed extreme low-cost mass production targets | FAIL |
| US-4 | US4T-1 | Data easy to export(pass) | Data is easy to export to desired data format should support wide range of formats | PASS |
|  | US4T-2 | Hard to export(fail) | Data is not easy to export and only supports limited formats | FAIL |
|  | US4T-3 | No export supported(fail) | Data cannot be exported | FAIL |

5.0 - Object-Oriented Design

5.1 - High Level System Architecture



5.3 - Use Case Analysis



USE CASE 1: User Login

TRIGGER/GOAL: A user wants to login to Power Bi

ACTOR: Admin/User

MAIN FLOW

1. Actor enters username into the username field and password into the password field

2. User is directed to Power Bi dashboard

EXTENSIONS

1a - User enters the wrong credentials. Step 1 in main flow is repeated

5.4 - Domain Model and Class Diagram

5.5 - Establishment of the Database Objects and Data Access Strategy

5.7 - Object Dictionary

6.0 - Software release report

6.1 - Usability Test Report

For usability, we will test our application with different people from different age groups. We will ask them to look at our live reports and give their review and rating of how the complex the data seems to them and if it is easy for them to understand or not.

Since we have not implemented the system fully yet and don’t have much working sample data we have not performed many tests until now.

The test we perform is simple as we are only showing a single page that shows the report on it and example is listed below.

**Name:** xxx

**Age:** xxx

**Gender:** xxx

**Computer Skills:** xxx

**Overall Rating:** xxx

|  |  |  |
| --- | --- | --- |
| **FUNCTIONALITY** | **COMMENTS** | **RATING (OUT OF 10)** |
| Login | xxx | 7 |
| Data easy to interpret | xxx | 3 |
| Data easy to export | xxx | 9 |
| Live data updates at a relevant pace to the user | xxx | 5 |
| Overall report design | xxx | 4 |
|  |  |  |
|  |  |  |

6.2 - System Test Report

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Id | Test Purpose | Steps | Input Parameters | Actual Data Input | Expected Output | Test Status |
| 1.0 | User Login | 1.Enter login details  2.Click login | “Password” | “Password” | The user should be logged into Power Bi | Success |
| 2.0 | View Data | 1.View the dashboard | N/A | N/A | The user should be able to view the graphical data on the dashboard | Success |

8.0 - Setup/Installation Guide

8.1 Installation of the Raspbian operating system for Pi

1. Download Raspbian.

This image is tested and working for a Raspberry pi 3 B+ <https://downloads.raspberrypi.org/raspbian/images/raspbian-2018-11-15/>

2018-11-13-raspbian-stretch.zip, 2018-11-13 14:13, 1.0G

1. Install Raspbian to the microSD card.

Download and install the Etcher SD card burner utility.

<https://www.balena.io/etcher/>

1. Run Etcher and select the Raspbian image that you extracted in step 1.

Select the microSD card drive. Etcher may have already selected the correct drive.

Click Flash to install Raspbian to the microSD card.

Remove the microSD card from your computer when installation is complete. It's safe to remove the microSD card directly because Etcher automatically ejects or unmounts the microSD card upon completion.

1. Insert the microSD card into Pi.

8.2 – Gateway (Raspberry Pi) Setup

8.2.1. Install Azure IoT Edge runtime on Linux (ARM32v7/armhf)

8.2.2. Install the container runtime

The following commands install both the Moby-based engine and command-line interface (CLI). The CLI is useful for development but optional for production deployments.

bash

# You can copy the entire text from this code block and

# paste in terminal. The comment lines will be ignored.

# Download and install the moby-engine

curl -L <https://aka.ms/moby-engine-armhf-latest> -o moby\_engine.deb && sudo dpkg -i ./moby\_engine.deb

# Download and install the moby-cli

curl -L <https://aka.ms/moby-cli-armhf-latest> -o moby\_cli.deb && sudo dpkg -i ./moby\_cli.deb

# Run apt-get fix

sudo apt-get install –f

8.2.3. Install the IoT Edge Security Daemon

# You can copy the entire text from this code block and

# paste in terminal. The comment lines will be ignored.

# Download and install the standard libiothsm implementation

curl -L <https://aka.ms/libiothsm-std-linux-armhf-latest> -o libiothsm-std.deb && sudo dpkg -i ./libiothsm-std.deb

# Download and install the IoT Edge Security Daemon

curl -L <https://aka.ms/iotedged-linux-armhf-latest> -o iotedge.deb && sudo dpkg -i ./iotedge.deb

# Run apt-get fix

sudo apt-get install -f

8.2.4 Connect your device to an IoT hub

8.2.4.1. Manual provisioning

To manually provision a device, you need to provide it with a device connection string that you can create by registering a new IoT Edge device in your IoT hub.

Open the configuration file.

bash

sudo nano /etc/iotedge/config.yaml

yaml

provisioning:

source: "manual"

device\_connection\_string: "<ADD DEVICE CONNECTION STRING HERE>"

# provisioning:

# source: "dps"

# global\_endpoint: "<https://global.azure-devices-provisioning.net>"

# scope\_id: "{scope\_id}"

# registration\_id: "{registration\_id}"

Save and close the file.

CTRL + X, Y, Enter

After entering the provisioning information in the configuration file, restart the daemon:

sudo systemctl restart iotedge

8.2.5. Verify successful installation

You can check the status of the IoT Edge Daemon using:

Bash

systemctl status iotedge

Bash

journalctl -u iotedge --no-pager --no-full

And, list running modules with:

Bash

sudo iotedge list

8.2.6. If your gateway is a Raspberry Pi, don't forget to enable SPI

Enable VNC, SPI, Serial, SSH and I2C

Click the Raspberry icon > Preferences > Raspberry Pi Configuration.

On the Interfaces tab, set I2C and SSH to Enable, and then click OK. If you don't have physical sensors and want to use simulated sensor data, this step is optional.

Reboot

8.3 – Arduino Setup

* 1. Download & install the latest version of Arduino IDE
  2. Open Arduino IDE, select **‘File’ > ‘Preferences’**
  3. Paste **‘**[**https://raw.githubusercontent.com/Seeed-Studio/Seeed\_Platform/master/package\_seeeduino\_boards\_index.json**](https://raw.githubusercontent.com/Seeed-Studio/Seeed_Platform/master/package_seeeduino_boards_index.json)**’** into **‘Additional Boards Manger URLs’**
  4. Select **‘Tools’ > ‘Board’ > ‘Board Manager’**
  5. Search for and install **‘Seeed SAMD Boards’**
  6. Select **‘Tools’ > ‘Board’ > ‘Seeeduino LoRaWAN’**
  7. If not done already, connect the Arduino board to the PC.

Note: The driver for the Seeeduino should automatically install. If it doesn’t use the driver provided here: [**https://github.com/SeeedDocument/Seeeduino\_LoRa/raw/master/res/driver.zip**](https://github.com/SeeedDocument/Seeeduino_LoRa/raw/master/res/driver.zip)

* 1. Select **‘Tools’ > ‘Port’** and select the Seeeduino COM port.
  2. Select **‘Open’** and select **‘LoRaWAN.ino’**
  3. Select **‘Upload’** and wait for the code to be uploaded onto the Seeeduino.

Note: Uploading to the Seeeduino is known to be temperamental, so if it doesn’t work on the first try, just try again.

* 1. The Seeeduino is now setup and will function whenever powered via battery or USB

8.4 – Azure Setup

8.4.1 – Resource Group Setup

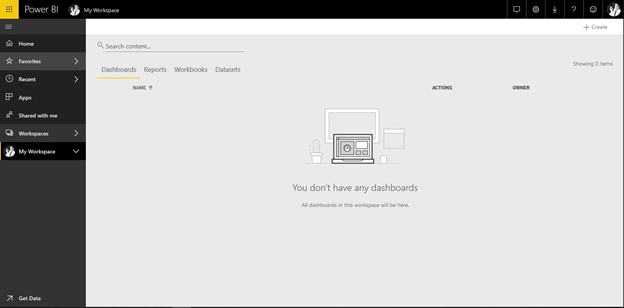
* 1. Navigate to <https://portal.azure.com/#create/Microsoft.ResourceGroup>
  2. Enter in any name for **‘Resource group’**
  3. Set **‘Region’** to a region of your choice (**‘(Asia Pacific) Australia East’** is recommended)
  4. Select **‘Review + Create’**
  5. After verifying the details, select **‘Create’** and wait for Azure to confirm setup of the resource group

8.4.2 – Template Deployment

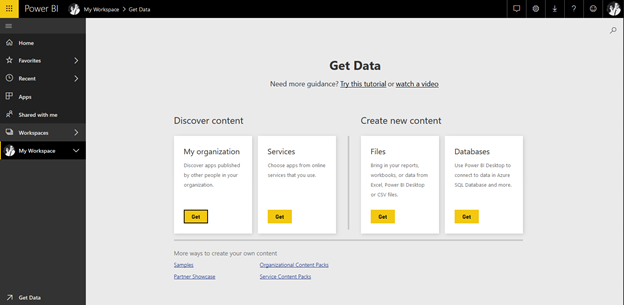
* 1. Navigate to <https://portal.azure.com/#create/Microsoft.Template>
  2. Select **‘Edit template’**
  3. Select **‘Load file’** and then select **‘template.json’**
  4. Select **‘Save’**
  5. Set **‘Resource group’** to the same resource group that was created in
  6. Select **‘Purchase’** and wait for Azure to finish the deployment (which should take around 10 mins to complete)

8.5 – Power BI

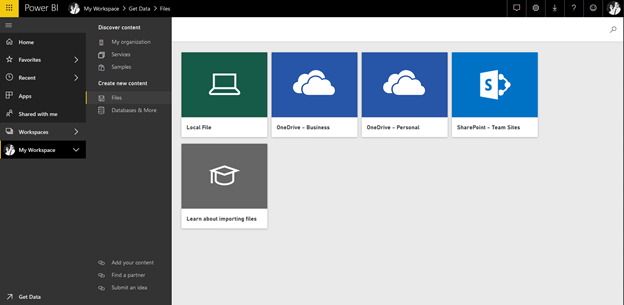
1. Navigate to <https://app.powerbi.com/home> and login with your account.
2. On the bottom left, select ‘Get Data’



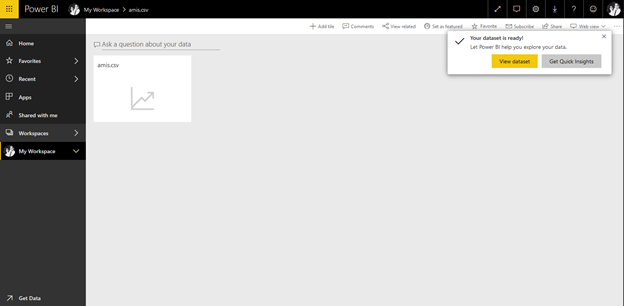
1. In Create new content, click on ‘Get’ under Files



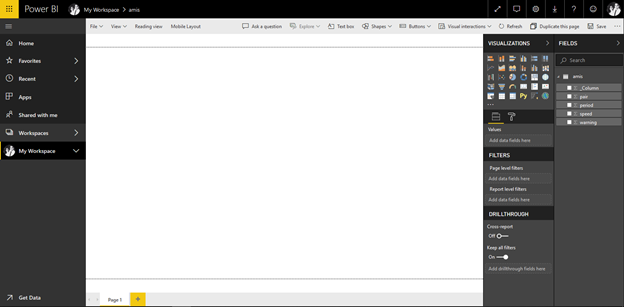
1. This will take you to a screen where you can pick the file to upload, whether it is from the cloud or a local file.



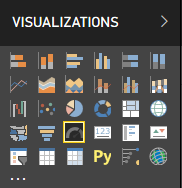
1. Click on Local File, and select your .csv or excel file to load the dataset to power bi

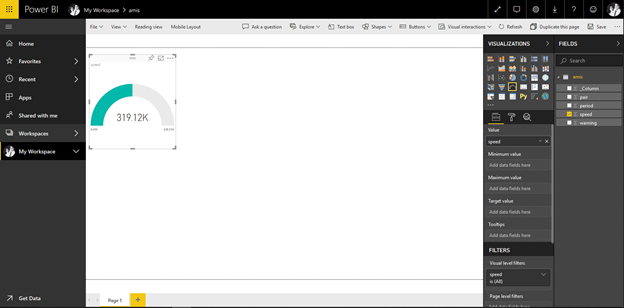


1. If loaded correctly, you should see on the right side of the screen under Fields section, the tables from the database.



1. Under Visualizations, select desired graph, and click on it once, it will transfer the graph to the working area. Once the graph is selected you can select the field or fields desired from your tables, and it will fill the graph with colors and numeric values.

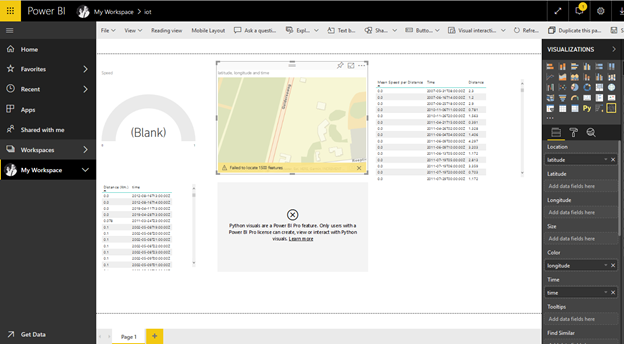




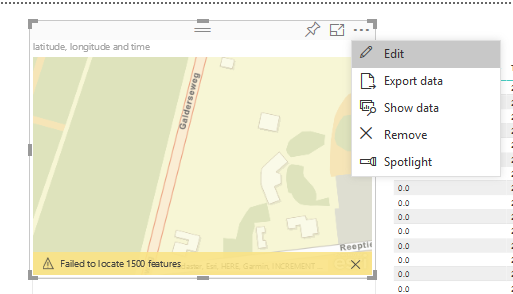
1. In the value section, you can modify or pick how you want the data to behave. Eg. Average, sum, etc. The same applies with the rest of the graphs.



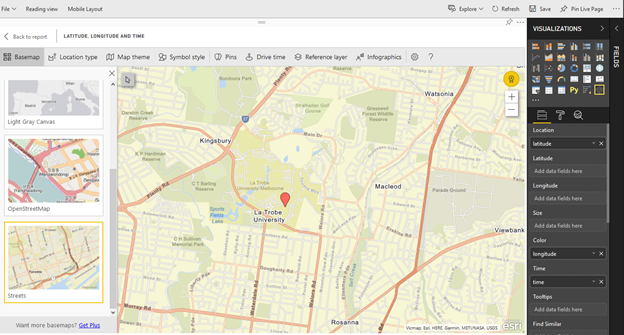
1. To add a map click on any globe icon in Visualizations and select latitude and longitude values

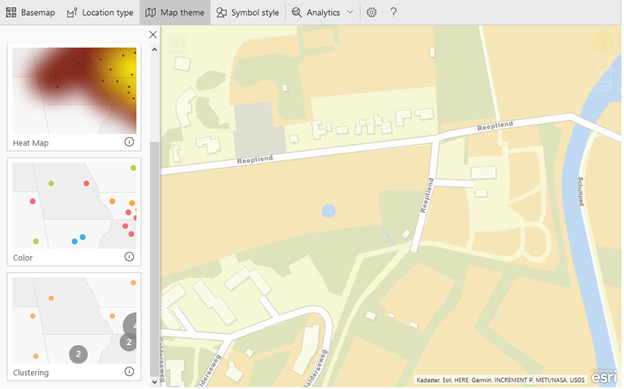


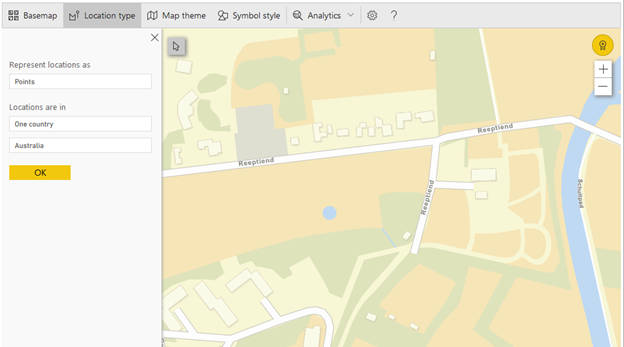
1. Maps are also editable, such as colors and themes. To do this, click on the … on the upper right side of the map, and select edit



Feel free to edit as suited.







1. Once you have finished your work, save as report

