LoRaWAN Temperature Sensors for Local Government Asset Management

ICP Innovation Internship

Jack Downes
Summer Intern
Innovation Central
Perth, Australia
jack.downes@student.curtin.edu.au

***Abstract*—The purpose of this project is to investigate the suitability of using LoRaWAN technology to conduct temperature studies on local government assets in Australian metropolitan and residential areas. Temperature sensing devices were integrated into the existing LoRaWAN infrastructure at Curtin University with data collected and stored on a remote server. Case studies were performed for the City of Melville to address the suitability for such a system to provide insights into heat islands and urban forests. Testing was completed on the Curtin University campus, replicating the climate conditions, asset types and, dense building and tree environment found in the City of Melville.**

Keywords—local government; asset management; lorawan; lora; innovation central; australia; internet of things; iot; temperature data; cisco; wireless network; heat islands; urban forests; wireless sensor network;

I. INTRODUCTION

A. Motivation

The City of Melville is a local government entity in Perth, Western Australia. Covering a metropolitan area to the South West of Perth's CBD the City of Melville is home to 98 083 residents [1]. The City of Melville has been working with the Curtin University Spatial Sciences Department to decipher spatial information related to heat dissipation in their suburbs.

Heat Islands are areas related spatially by their heat retention properties. This often occurs as a result of the use of man-made materials when constructing environments. The phenomena can be observed in Perth's hot months during the night, as the ambient heat in areas using dense materials is noticeably higher than surrounding areas. Materials responsible for creating heat islands include concrete, asphalt and paving. This is an important issue for residence due to the unpleasantness of the heat, and financial pressure caused by the cost of air conditioning.

Urban Forests are areas of vegetation found in metropolitan environments. Through shading of the ground and absorption of thermal energy these areas have a cooling effect on ambient air temperatures. Overnight in hot Perth months these areas will have a comparatively lower temperature than the Heat Islands mentioned above.

It is in the interest of the City of Melville to mitigate the effects of Heat Islands and fully utilise the cooling effect of Urban Forests. Getting this balance right poses a number of questions that can affect budget, maintenance requirements and the livability index of an area. Thermal imagery has been used to assess the effects of Heat Islands and Urban Forests. Thermal imagery can be captured aerially or terrestrially providing a radiation map that can be used to get an understanding of a surface's heat properties.

The City of Melville in conjunction with Curtin University's Department of Spatial Sciences conducted three studies in 2017 to address interests in Heat Islands and Urban Forests. Bradley Schupp's paper [2] analyses the City of Melville's aerial thermal imagery and LIDAR data investigating land use and Heat Island correlation. Stuart McEvoy's paper [3] looks into cooling effects by type of vegetation using terrestrial thermal imagery and on foot temperature sampling. Luke Ellison's paper [4] links the changing of land use types to the prominence of Heat Islands.

The nature of this research lends itself to bigger datasets and higher granularity of data.

B. Scope

To ensure the viability of this internship project, it is important to limit the project scope. This is defined as:

- Implement a LoRaWAN system to wirelessly record ambient temperature data
- Evaluate the effectiveness of this use case for the City of Melville.

C. Thesis Statement

Investigate the suitability of using LoRaWAN technology to conduct research on City of Melville assets in the form of a short term deployments of an array of temperature sensor devices.

II. BACKGROUND

LoRaWAN is a trending solution for implementation of the Internet of Things style wireless devices. The following helps

highlight its value, providing depth of knowledge as to why it may be of benefit to the City of Melville for temperature studies of in future.

A. The Internet of Things

The Internet of Things (IOT) is a concept born out of the evolution of the Internet. With the increasing reach of networks and the low cost of transmitting data it is possible to connect more devices to the internet. So much so that objects often thought of as mundane and serving a single purpose can be connected and addressable through internet enabled devices to open up a realm of informational possibility. This allows for network interaction with existing everyday items such as home appliances, building climate control, street lights, or car parks. Through this trend large datasets of information can be gathered and used to improve processes conventionally ignored. The concepts of IOT are largely accepted as the direction of the Internet of the future [5]. As enabling technologies reach a level of maturity and standards are adopted it is expected that IOT will be adopted as a mainstay of smart cities [6].

B. LoRaWAN

Low-Power, Wide-Area Networks (LPWAN) are networks with a priority on keeping power usage for transmitting data to a minimum. They are designed specifically for battery powered devices, having the potential to extend expected battery life cycles to years. This battery performance is not possible through common wireless networks like WI-FI. LPWANs will support a large portion of the billions of devices forecasted for the IOT [7].

LoRaWAN (LoRa Wide-Area Network) is a realisation of the LPWAN concept and is becoming a popular solution [8].

LoRaWAN is a standardised system architecture that implements LoRa, a low power long range communication protocol using technology not available for commercial use prior to its release in June 2016 [7] [9]. To achieve low power transmission the data throughput is restricted, meaning it is unsuitable for transmission of large data necessary to transmit video or audio. This system is designed to be a solution for small packets sent at large intervals, such as weather station data, non-frequent GPS locations or gas sensor readings.

Figure 1 shows the LoRaWAN structure. The system is composed of four parts; the end nodes, LoRaWAN gateways, a network server and application servers. Together these components act to read sensor data, transmit it wirelessly to a gateway, and repeat it through the internet onto an application server to be accessed by an end user.

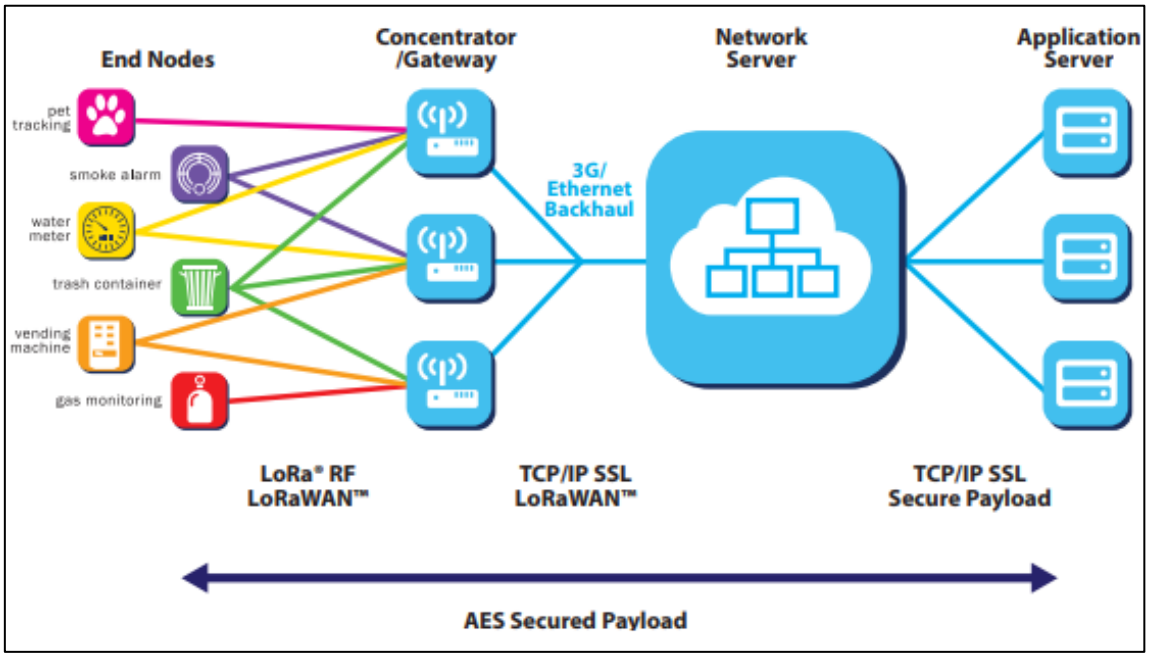

Figure 1 - LoRaWAN System Architecture [7]

III. IMPLEMENTATION

A LoRaWAN system for the City of Melville was built. It would allow insight into the practicality of the proposed use cases. The new method of data capture would then be available for trials replicating the findings of the previous studies. A break-down of the system design follows.

A. End Node Device

Figure 2 - Temperature Sensor Device

Three sensor devices were created to form the array of end nodes. The array allows the compilation of data from many devices in unison. Using the Cisco Wireless Gateway [10] the size of the array can be expanded far beyond these three devices. Having three sensors allows comparison of asset material properties that are exposed to simultaneous variables caused by weather.

Each temperature sensor device (shown in Figure 2) was built to collect ambient air temperature and light intensity, transmitting it through to the common parts of a LoRaWAN. The device contains an Adafruit Feather M0 RFM95 LoRa Radio microcontroller, rechargeable 3.7V 2500mAh battery, Waterproof DS12B20 digital temperature sensor (on a 900mm long cable), TSL2561 digital luminosity/LUX/light sensor, and a UFL antenna. It is contained within a sealed plastic enclosure, providing protection against dust and low pressure water jets [11].

The external waterproof temperature sensor and internal light sensor provide complementary data. The temperature sensor records ambient air readings above a surface of interest which in daylight hours is affected by direct sunlight. By recording light intensity at the same time, direct sunlight could be identified as a contributing factor to a high temperature reading. In turn this extra information helps to rule out outliers in the data sets.

B. Gateway and Network Server

The gateways and network server were implemented using existing infrastructure available to Innovation Central Perth. Two gateway antennae exist at Curtin University's Bentley Campus. One above building 216, and another atop building 105 (the Robertson Library). The network server used for the system was hosted on Cisco infrastructure. When receiving a packet of information from a sensor device the gateway forwards it onto the network server. A web hook hosted on the network server was programmed to interrogate packets passing through, identify those belonging to this project, and send the appropriate information to the application server through a MYSQL connection.

C. Application Server

The Database is hosted within Cisco's Virtual Private Network (VPN) on a virtual machine (VM) running Ubuntu Server 16.04. The server contains the structural ability to store information about each device unit, projects, user details, material properties, location properties as well as the readings themselves. It was important to structure the data in this way to retain its integrity and accessibility. Even at a low number of nodes data management is an important aspect to the success of a LoRaWAN system. To access data from the server the database must be queried using a frontend program, e.g. MYSQL Workbench, then the requested data is saved to a client PC to be filtered and analysed.

IV. RESULTS

Using a deployment plan to manage the various case studies in parallel the devices could be loaded with the properties required for each assignment, taken to the location of interest and deployed. Data was captured and stored for post analysis, as live feed data was not considered important for this use case.

The datasets produced rapidly exceeded readings in excess of ten thousand timestamped entries. As such, tools like Microsoft Excel were not suited to interrogating the size of the dataset. This led to the use of the data analytics tools available through Python packages. Python's Spyder IDE along with the pandas, numpy and matplotlib libraries allowed filtering of data for gross errors, periods of interest and trends, and the ability to plot the data in a digestible format. The plots following and found in the appendix were generated using this process.

The plots proved to be of good use for quantitative analysis. A case study into the different cooling effects of concrete and grass measured the dissipation of heat of the surface coverings in parallel over the same night. This study replicates the interests the City of Melville has in Heat Islands and Urban Forests. Figure 6 outlines the expected behaviour. After moving into the shade of the day, concrete begins to dissipate heat, but at a slow rate. The ambient temperature above the concrete by the time it begins to reheat from the sun sits just above 20°C. Conversely by 18:30 the grass rapidly drops to below 20°C within 4 hours. This evidence of materials dissipating heat at different rates can be easily replicated, showing the value in having a quick way of attaining large data sets.

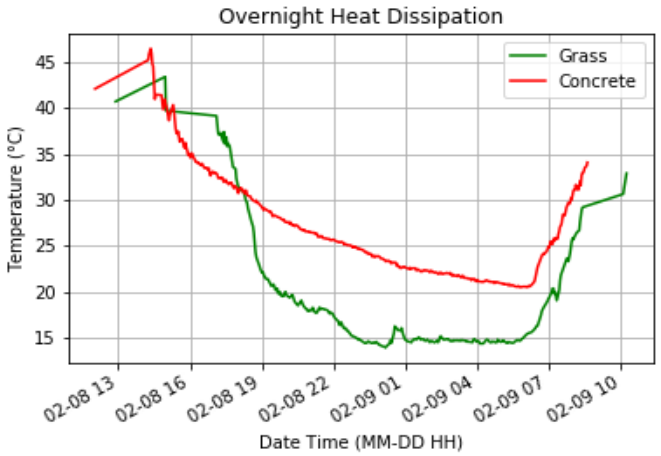

Figure 3 - Heat dissipation overnight with grass and concrete

Data from the system also lends itself to comparison with third party data sets. Figure 8 shows temperature fluctuation from a node's sensor that was implemented on a red brick wall in an area receiving sunlight over 4 days, plotted against dry bulb temperatures sourced from a weather station run by the Bureau of Meteorology at the Perth Airport [12] ten kilometres away from the Curtin University's Bentley Campus. It outlines the behaviour of material temperatures when the effects of sunlight are no longer present. Any timestamped data can be used in conjunction with data captured through the LoRaWAN system.

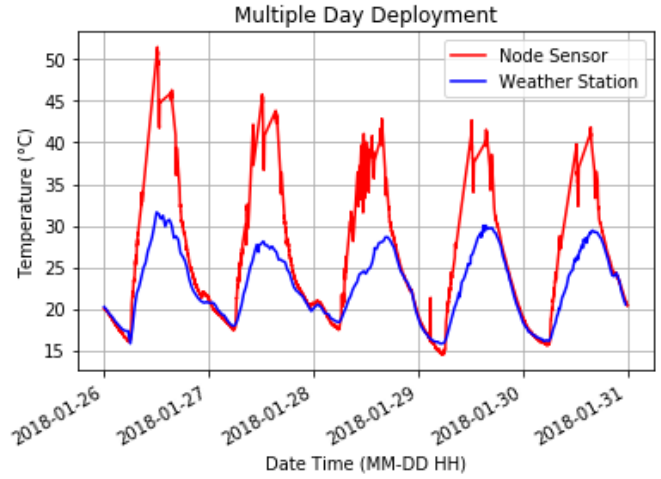

Figure 4 - Heat dissipation over multiple days with weather station data and a node setup on red brick

V. CHALLENGES

Building a LoRaWAN system with the help of Innovation Central Perth proved to be straight forward. The hardware and network components parts of the project came together without fault. This allowed the use case specific challenges to become abundantly clear. They are as follows:

A. Network Coverage

LoRaWAN can achieve transmission distances of five kilometres with line of sight between the gateway and end node antennae, however poor network reception was encountered. This was due to non-terrain obstructions like trees and buildings. These network issues became apparent five hundred and fifty metres away from the gateway, when the antenna was low to the ground. This can be mitigated by altering the end node design to allow for raising the antennae from the ground, installing more gateways in the area of interest to create an overlapping coverage, or utilising temporary gateways on location connected through a mobile network.

B. Overheating Nodes and Direct Sunlight

At surface temperatures of around 40°C the microprocessor within the end node device turns off, avoiding damage caused by the electronics operating at high temperatures. Figure 5 shows two full days of deployment for a device recording red brick, the time scale annotation shows the month (January), the day (the 26th to the 27th) and the hour in 24-hour time. It can be observed here how direct sun light affects devices readings in two ways. Firstly the deviation of the surface temperature from the dry bulb temperature readings. Secondly, periods of no readings, or gross errors experienced during the heat of the day between 10:00 and 16:00 hours on both the 26th and the 27th of January. To mitigate the second issue for the case studies the device body was kept out of direct sunlight by trailing the external sensor to the area of interest. Through installing a fan, a heat sink, using more appropriate enclosure materials, or using an opaque enclosure lid, the design can be modified to operate in higher temperatures.

In practice the light sensor provided little use, often recording gross errors as a result of the temperatures reached inside the box. It remains a contender as a solution to recording the effects of shade, cloud cover, and the passing of direct sunlight at sunrise and sunset, but its data was not as telling as that collected from the temperature sensor and contained far more noise that needed to be filtered out.

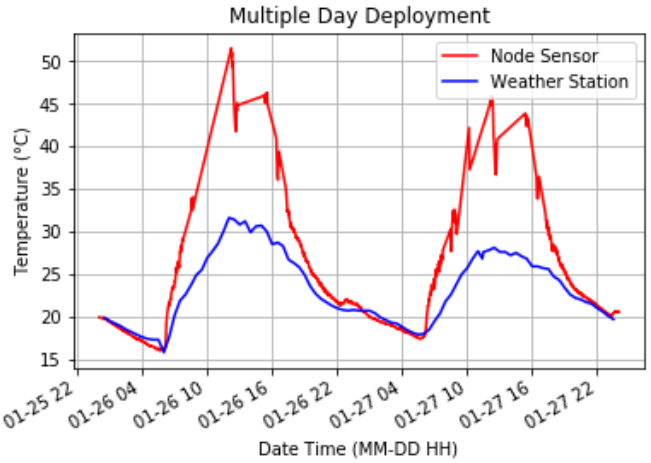

Figure 5 – End node deployed on red brick at Curtin University plotted against Perth Airport weather station data provided by the Bureau of Meteorology [12]

C. Theft

With dimensions similar to a modern smart phone, the unit's small size does lend it to the possibility of theft. For this project the devices were most often deployed in areas of low pedestrian traffic that were locked outside of business hours. For the use of local government this will be a rare luxury. In future iterations the design could be altered to mitigate the risk of theft by using a more robust enclosure with the capacity to be locked down, GPS tracking, camouflaging, larger more cumbersome boxes, smaller more discrete boxes or an opaque top.

In one case study a device was deployed in a public area outside the Curtin University Stadium. It was left on an open grass field with a medium level of foot traffic. During this deployment it became evident how discrete the unit is at its small size: it was difficult to see until standing within metres of the unit. As the unit cost for these devices can be quite low, and if theft only happens rarely, allowing for occasional theft may be within reason.

The system is designed to transmit information and store it remotely for every reading, not saving anything on the device. This abstraction means that the impact made by a device being stolen or broken on a project's data set, is far less than for a device storing data within its local storage.

D. Maintenance

The system requires maintenance after the initial setup. Some maintenance requirements that have been identified are:

- The battery for each unit needed to be recharged after around two hundred hours of use, or before each deployment (taking around four hours).
- Time needed to be taken to support unexpected software faults.
- The external temperature sensor may rust and batteries will degrade over time, needing to be replaced.

VI. DISCUSSION

The range of applications for network technology means the LoRaWAN system designed for this project can only address a small use case out of the vast possibilities. Following are points of discussion into how the system may be better utilised for the City of Melville in the future.

A. User Interface

Beside the initial setup the system requires an understanding that may take up to a week for the non-inducted user to become familiar with. This includes an understanding of **MYSQL** to pull information from the database, **low level coding** to alter the properties for each sensor node (such as transmission interval) and **data analysis** to interrogate the data. To manage this project the software packages Arduino IDE, MYSQL Workbench and Spyder were used. If all of this knowledge can be encapsulated to one user interface abstracting the low level information from the user, it would make the system much more user friendly. Options include a custom desktop client, a browser client or existing third party software like AdafruitIO [13].

B. Device Sensor Options

Once the supporting structure of a LoRaWAN is established, the device array it supports can be developed and improved to better suit the needs of local government. For example, an observation from the data in Figure 5 is the deviation between the dry bulb temperature reading of ambient air and that recorded on the ground in direct sunlight by the device. The effects from the direct sunlight on the external temperature probe could be mitigated using a temperature sensor that relies on radiation emitted from surfaces instead of the electronic resistance based sensor that was implemented, as this may be affected by its own material properties. A study is possible with the system as it stands to add a range of different types of temperature sensors to each node and to investigate how they behave in comparison. The end node devices can be further customised to monitor asset characteristics including soil moisture levels, path and road utilization, air quality, and weather data, along with array of Smart City [14] opportunities.

C. Set and Forget

A major benefit of LoRaWAN is its support for set and forget style devices. Set and forget devices are left in the field and do not require any servicing, either achieved by a solar power source or by a battery that services the life of the project, often expected to be years. LoRaWAN is built to enable devices to operate on a single battery charge for more than a year; use cases for local government may call for this capability. Longer deployment periods can allow deeper insights than those achieved in testing for this project. For example, temperature data tracking the growth of forest canopies into maturity, seasonal effects on material heat dissipation, temperature properties relating to long term material degradation, and more accessible real time and retrospective interrogation for events such as heat waves and freak storms.

This project focussed on shorter period deployments to suit the timeframe of the internship. It allowed for shorter transmission intervals of one to five minutes, chosen to achieve a higher temporal granularity for plotting temperature and to account for redundancy in the event of lost data. This higher transmission interval was achieved at the cost of battery life. With a transmission interval of two minutes the end nodes last up to two hundred hours. This can be extended by increasing the recording interval to thirty minutes. Furthermore by adding a low power timer component [15] (a quick task) the devices battery life can be greatly expanded. This component helps to regulate power draw by reconnecting the battery supply at a timed interval, allowing the microcontroller to boot up for only the reading and transmission cycle then turn itself off. This addition can increase the battery utilisation and operating time to better suit set and forget deployment.

VII. RECOMMENDATIONS

A major barrier to entry for end users of this system is the base knowledge required to navigate the data capture pipeline. To ensure ease of use for the system and a high level of uptake for projects it is recommended to develop a user friendly digital interface. Most important would be the abstraction of downloading data from the application server. Once in a familiar format on a client PC the data can be intuitively analysed. The data analysis process can be stream-lined to output comparative plots and trend information, which are easily digestible for local government use cases. Failing the above, it would be advised to package this solution as a service, to ensure customer expectations are met, and to remove any need for training that may be required.

Network coverage proved a hindrance to the effectiveness of available device locations. This meant some surfaces could not be tested as they were at the fringe of the network range on the Curtin University campus, or had poor line of sight to the gateway antennae. A portable LoRaWAN gateway may provide a solution to this issue by filling in shadows created by obstructive trees or buildings. The time spent trouble shooting packet drop and consequently reassessing the deployment plan in the case study phase of the project highlights network coverage as a key component to the success of using LoRaWAN devices for temperature sensing in the City of Melville. Investigating solutions to improve network coverage in obstructed terrain is recommended.

LoRaWAN has been developed to support devices deployed for long periods of time. It is recommended to investigate extending the battery life of device nodes and compare to that achieved for this project. This will allow use cases for set and forget devices for the City of Melville to be trialled.

VIII. CONCLUSION

As metropolitan living becomes denser the management of local government assets becomes more complex. As we move further into the information age, there is an opportunity to take advantage of big data solutions like the Internet of Things. The Australian Government's Smart Cities initiative [14] is one example of public effort to realise the transformative potential such technologies. LoRaWAN and sensors, such as those developed in this project, provide a standardised, widely adopted and supported way to capture data to inform Smart Cities now and in the future.

IX. ACKNOWEDGEMENT

I'd like to note my sincere gratitude to Innovation Central for the opportunity to work on this project, entrusting me with the room to manage it as I saw fit, to Valerie Maxville for the guidance she provided to myself and my peers, and to Jim Wyatt for encouraging us to take pride in our work. My thanks to Janine Ahola from the City of Melville for fostering excitement for any outcomes the project might produce. And my thanks to Dave Belton and Petra Helmholz from Curtin University's Department of Spatial Science for their professional advice and vision for this project.

X. REFERENCES

[1] Australian Bureau of Statistics, "2016 Census QuickStats," Australian Bureau of Statistics, 12 January 2017. [Online]. Available: http://www.censusdata.abs.gov.au/census_services/getproduct/census/2016/quickstat/LGA55320. [Accessed 19 February 2018].

[2] B. D. Schupp, "Urban Forest Canopy Coverage and its Effect on Heat Islands in the City of Melville," Curtin University, Department of Spacial Sciences, Perth, unpublished.

[3] S. McEvoy, "Effect on Ground Surface Temperature of different tree species within the urban forest canopy.," Curtin University, Department of Spacial Sciences, perth, unpublished.

[4] L. Ellison, "Urban Forest Mapping and Analysis: The Development of Urban Heat Islands in the City of Melville Due to Changing Land Use Types," Curtin University, Department of Spatial Sciences, Perth, unpublished.

[5] L. Atzori, A. lera and L. Moarabito, "The Internet of Things: A Survey," *ScienceDirect*, vol. 54, no. 15, pp. 2787-2805, 2010.

[6] A. Zanella, N. Bui, A. Castellani, L. Vangelista and M. Zorzi, "Internet of Things for Smart Cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22-32, 2014.

[7] Lora Alliance, "LoRaWAN, What is it? A Technical Overview of LoRa and LoRaWAN," November 2015. [Online]. Available: https://www.lora-alliance.org/what-is-lora. [Accessed 20 February 2018].

[8] M. S. Mahmoud and A. A. H. Mohamad, "A Study of Efficient Power Consumption Wireless Communication Techniques/ Modules for Internet of Things (IoT) Applications," *SciRes*, vol. 6, no. 2, pp. 19-29, 2016.

[9] LoRa Alliance, "LoRaWAN R1.0 Open Standard Released for the IoT," 16 June 2015. [Online]. Available: https://www.businesswire.com/news/home/20150616006550/en/LoRaWAN-R1.0-Open-Standard-Released-IoT. [Accessed 20 02 2018].

[10] Cisco Systems, Inc, "Cisco Wireless Gateway for LoRaWAN Data Sheet," 20 December 2017. [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/se/internet-of-things/datasheet-c78-737307.html. [Accessed 20 February 2018].

[11] MPL AG, "IP Ratings (Ingress Protection)," 2018. [Online]. Available: https://www.mpl.ch. [Accessed 20 February 2018].

[12] Commonwealth of Australia, Bureau of Meteorology, "Climate Data Online," Commonwealth of Australia, Bureau of Meteorology, 2018. [Online]. Available: http://www.bom.gov.au. [Accessed 31 January 2018].

[13] Adafuit Industries, "Adafruit IO," 2018. [Online]. Available: https://io.adafruit.com/. [Accessed 23 February 2018].

[14] © Commonwealth of Australia 2016, "Smart Cities Plan," 2016. [Online]. Available: https://cities.dpmc.gov.au. [Accessed 21 February 2018].

[15] Adafruit Industrues, "Adafruit TPL5110 Low Power Timer Breakout," 2018. [Online]. Available: https://www.adafruit.com/product/3435. [Accessed 21 February 2018].

[16] Adafruit Industries, "Adafruit M0 Radio With LoRa Radio Module," 2018. [Online]. Available: https://learn.adafruit.com/adafruit-feather-m0-radio-with-lora-radio-module. [Accessed 27 February 2018].

[17] © 2018 Arduino, "The Arduino Playground," 2018. [Online]. Available: https://playground.arduino.cc/. [Accessed 27 Febraury 2018].

[18] The Things Network, "The Things Network," 2017. [Online]. Available: https://www.thethingsnetwork.org/. [Accessed 20 February 2018].

[19] LoRa Alliance, "LoRa Alliance Technology," 2018. [Online]. Available: https://www.lora-alliance.org/what-is-lora. [Accessed 02 February 2018].

XI. APPENDIX

A. Case Studies

1) Concrete vs Grass

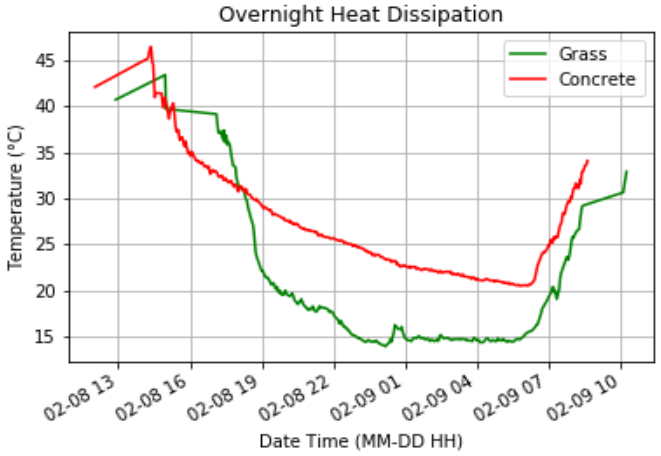

Figure 6 - Heat Dissipation Overnight Grass and Concrete

2) Playground Materials

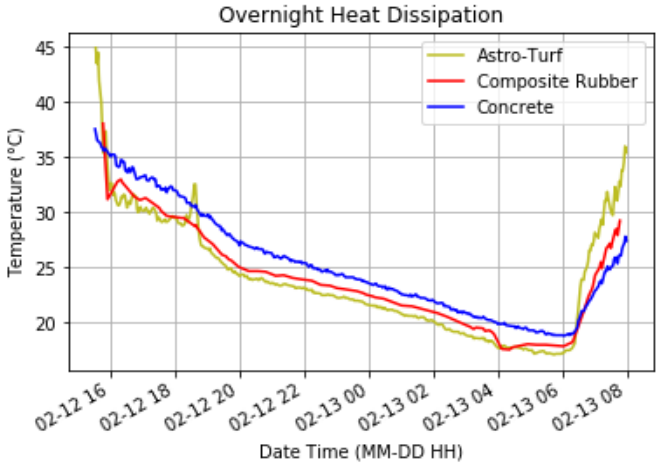

Figure 7 - Heat Dissipation Overnight Playground Materials

B. Implementation

1) Bill of Materials

The Adafruit Feather M0 FRM95 LoRa Radio (900MHz) micro controller was used for its built in 900Hz radio capability, as a LoRa Network must be deployed on the 915Hz band in Australia. The sensors listed in were either one-wire or I²C compatible ensuring they would work with the capabilities of the Feather M0 variant. Some sensors were not utilized in the final build but all are compatible with the microcontroller used.

3) Weather Station Integration over a Week

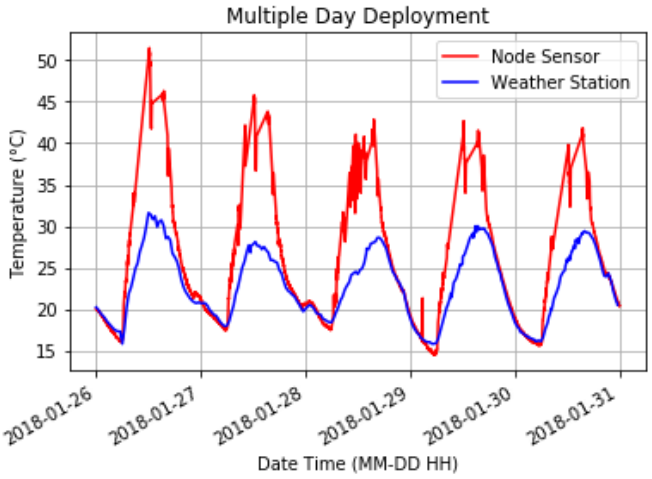

Figure 8 - - Heat dissipation over multiple days with weather station data and a node setup on red brick

4) Tin and Concrete

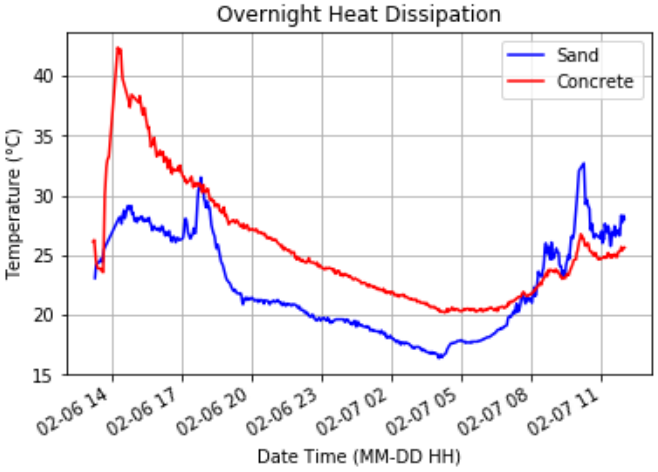

Figure 9 – Heat dissipation of tin and concrete

Part Number	Part Name	Qty	Unit Cost (USD)	Cost ($USD)	Source	Comment
3178	Adafruit Feather M0 RFM95 LoRa Radio (900MHz)	3	$34.95	$104.85	https://www.adafruit.com/product/3178	Micro Controller
2886	Feather Header Kit - 12-pin and 16-pin Female Header Set	3	$0.95	$2.85	https://www.adafruit.com/product/2886	
3435	Adafruit TPL5110 Low Power Timer Breakout	3	$4.95	$14.85	https://www.adafruit.com/product/3435	
2884	FeatherWing Proto - Prototyping Add-on For All Feather Boards	3	$4.95	$14.85	https://www.adafruit.com/product/2884	
328	Lithium Ion Polymer Battery - 3.7v 2500mAh	3	$14.95	$44.85	https://www.adafruit.com/product/328	
259	USB LiIon/LiPoly charger - v1.2	3	$12.50	$37.50	https://www.adafruit.com/product/259	
3251	Adafruit Si7021 Temperature & Humidity Sensor Breakout Board	3	$6.95	$20.85	https://www.adafruit.com/product/3251	Temp/Hum sensor
439	Adafruit TSL2561 Digital Luminosity/Lux/Light Sensor Breakout	3	$5.95	$17.85	https://www.adafruit.com/product/439	Light Sensor
1782	MCP9808 High Accuracy I2C Temperature Sensor Breakout Board	3	$4.95	$14.85	https://www.adafruit.com/product/1782	High Accuracy Temp Sensor
381	Waterproof DS18B20 Digital temperature sensor + extras	3	$9.95	$29.85	https://www.adafruit.com/product/381	Temp Sensor w/ Waterproof housing, needs code work to implement
165	TMP36 - Analog Temperature sensor - TMP36	3	$1.50	$4.50	https://www.adafruit.com/product/165	Temp sensor
1296	TMP36 - Analog Temperature sensor - TMP36	3	$9.95	$29.85	https://www.adafruit.com/product/1296	Contactless heat sensor, needs coding work to implement
1661	uFL SMT Antenna Connector	3	$0.75	$2.25	https://www.adafruit.com/product/1661	
3340	900Mhz Antenna Kit - For LoPy, LoRa, etc	3	$12.75	$38.25	https://www.adafruit.com/product/3340	
1951	Premium Female/Female Jumper Wires - 20 x 3" (75mm)	2	$1.95	$3.90	https://www.adafruit.com/product/1951	
1956	Premium Male/Male Jumper Wires - 20 x 3" (75mm)	2	$1.95	$3.90	https://www.adafruit.com/product/1956	
1953	Premium Female/Male 'Extension' Jumper Wires - 20 x 3"	2	$1.95	$3.90	https://www.adafruit.com/product/1953	
142	USB cable - 6" A/MiniB	3	$2.95	$8.85	https://www.adafruit.com/category/142	
898	USB cable - 6" A/MicroB	1	$2.95	$2.95	https://www.adafruit.com/product/898	
			Total ($USD)	$401.50		

Part Number	Part Name	Qty	Unit Cost ($AUD)	Cost ($AUD)	Source	Comment
MP3446	4 Port USB Mains Power Adaptor	1	$29.95	$29.95	https://www.jaycar.com.au/4-port-usb-mains-power-adaptor/p/MP3446	
H0323	105Lx75Wx40mmH IP65 sealed ABS enclosure	3	$14.95	$44.85	http://www.altronics.com.au/p/h0323-ritec-105lx75wx55hmm-ip65-sealed-abs-enclosure/	Water Resistant Enclosure
			Total ($AUS)	$74.80		

Table 1 – Bill of Materials

2) Device Build

Figure 10 and Figure 11 show the concept designs used for the building the device. The final design did not implement the following:

- T/H Sensor Breakout Board
- High Accuracy Temperature Sensor
- Contactless IR Temperature Sensor
- Analog Sensor
- Timer Breakout

Documentation for connecting the components, including sensors, the antenna and battery can be found on the Adafruit website by searching each component as listed in Table 1 [16].

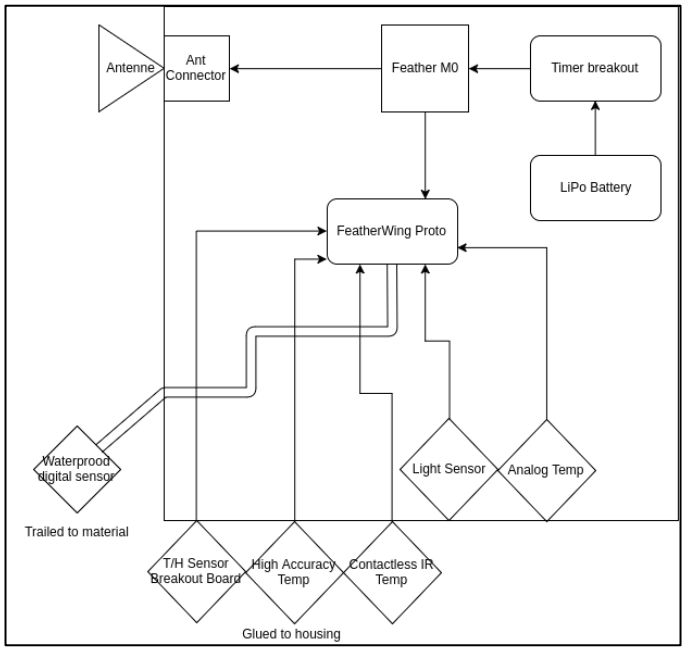

Figure 10 - Initial Design A

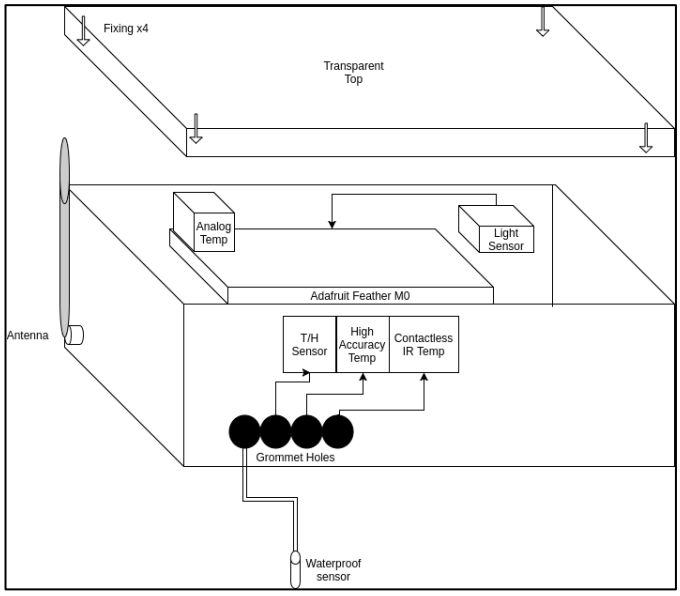

Figure 11 - Initial Design B

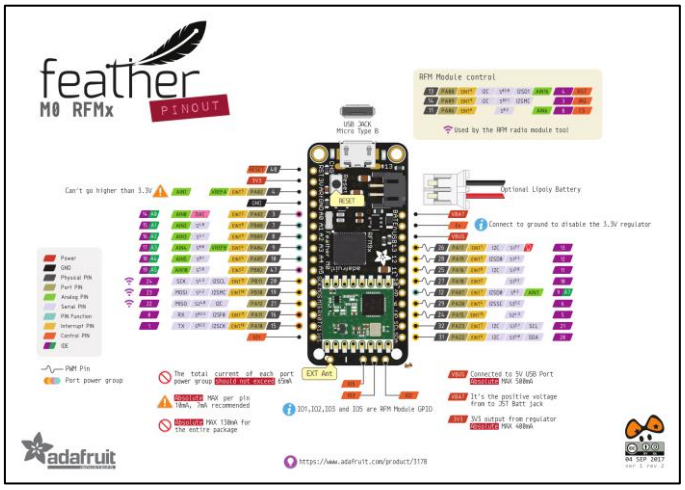

Figure 12 - Feather M0 Pinout [16]

3) Device Code

The code running on the devices are written as Arduino sketch files. The Arduino IDE program is used to compile and upload the sketch files via the Micro Type B USB jack on the microcontroller shown in Figure 12. This means each time the header file of the sketch is altered to change project ID, transmission interval at similar a device must be opened and physically connected to a PC to update the code.

Once the lipoly battery is connected to the microcontroller the software will begin attempting to send packets at the set interval. The red LED (output pin 13) flashes on and off once to indicate a packet has been sent.

The code shown in appendix XI.B.3)b) allows room for a packet with information from 6 sensors to be sent, they must be appended to the mydata array.

The following Arduino library files are needed to compile the code in Arduino IDE:

- lmic.h
- hal/hal.h
- Wire.h
- SI7021.h
- SPI.h
- OneWire.h
- DallasTemperature.h
- Adafruit_Sensor.h
- Adafruit_TSL2561_U.h

These libraries are available from the Adafruit or Arduino Playground websites then added to the Arduino IDE install directory [17] [16].

a) LoRa Cisco Header File

```
#include <hal/hal.h>
#include <lmic.h>

// LoRaWAN NwkSKey, network session key
// This is the default Semtech key, which is used by the early prototype TTN
// network.
static const PROGMEM u1_t NWKSKEY[16] = { 0x2B, 0x7E, 0x15, 0x16, 0x28, 0xAE, 0xD2, 0xA6, 0xAB, 0xF7, 0x15, 0x88,
0x09, 0xCF, 0x4F, 0x3C };

// LoRaWAN AppSKey, application session key
// This is the default Semtech key, which is used by the early prototype TTN
// network.
static const u1_t PROGMEM APPSKEY[16] = { 0x2B, 0x7E, 0x15, 0x16, 0x28, 0xAE, 0xD2, 0xA6, 0xAB, 0xF7, 0x15, 0x88,
0x09, 0xCF, 0x4F, 0x3C };

// LoRaWAN end-device address (DevAddr)
static const u4_t DEVADDR = 0x78978911 ; // <-- Change this address for every node!

// These callbacks are only used in over-the-air activation, so they are
// left empty here (we cannot leave them out completely unless
// DISABLE_JOIN is set in config.h, otherwise the linker will complain).
void os_getArtEui (u1_t* buf) { }
void os_getDevEui (u1_t* buf) { }
void os_getDevKey (u1_t* buf) { }

// Pin mapping
const lmic_pinmap lmic_pins = {
    .nss = 53,
    .rxtx = 46,
    .rst = A0,
    .dio = {2, 6, LMIC_UNUSED_PIN},
};
```

b) Arduino Header File – config.h

```
/*
*       Project ID, Location ID, and device address will have to be added to the database before
*               these values can be changed.
*       Device address will also have to be added to the network server forward list
*               Available Device addresses are 0x00540006, 0x00540008 and 0x00540008
*/

#define PROJID 8

#define LOCID 3

#ifndef DEVADDR
#define DEVADDR 0x00540006
#endif

#ifndef TX_INTERVAL
#define TX_INTERVAL 120
#endif
```

c) Arduino Sketch

```
#include <lmic.h>
#include <hal/hal.h>
#include <Wire.h>
#include <SI7021.h>
#include <SPI.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_TSL2561_U.h>
#include "config.h"

// LoRaWAN NwkSKey, network session key
// This is the default Semtech key, which is used by the prototype TTN
// network initially.
static const PROGMEM u1_t NWKSKEY[16] = { 0x2B, 0x7E, 0x15, 0x16, 0x28, 0xAE, 0xD2, 0xA6, 0xAB, 0xF7, 0x15, 0x88,
0x09, 0xCF, 0x4F, 0x3C };

// LoRaWAN AppSKey, application session key
// This is the default Semtech key, which is used by the prototype TTN
// network initially.
static const u1_t PROGMEM APPSKEY[16] = { 0x2B, 0x7E, 0x15, 0x16, 0x28, 0xAE, 0xD2, 0xA6, 0xAB, 0xF7, 0x15, 0x88,
0x09, 0xCF, 0x4F, 0x3C };

// LoRaWAN end-device address (DevAddr)
// See http://thethingsnetwork.org/wiki/AddressSpace https://www.thethingsnetwork.org/wiki/LoRaWAN/Address-Space
//static const u4_t DEVADDR = 0x00540006; Define in config file

// These callbacks are only used in over-the-air activation, so they are
// left empty here (we cannot leave them out completely unless
// DISABLE_JOIN is set in config.h, otherwise the linker will complain).

void os_getArtEui (u1_t* buf) { }
void os_getDevEui (u1_t* buf) { }
void os_getDevKey (u1_t* buf) { }

//static uint8_t mydata[] = "Hello, world!?";
//Standard format for server to recieve for up to 8 sensors
uint8_t mydata[14] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};



static osjob_t sendjob;

// Schedule TX every this many seconds (might become longer due to duty
// cycle limitations).
//const unsigned TX_INTERVAL = 60; Define in config.h

// Pin mapping
const lmic_pinmap lmic_pins = {
    .nss = 8,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 4,
    .dio = {3,5,6},
};
```

```
/*****************************************************************/
/*              Waterproof Sensor Setup                          */
/*****************************************************************/

// Data wire is plugged into pin 12 on the Arduino
#define ONE_WIRE_BUS 12
/*****************************************************************/
// Setup a oneWire instance to communicate with any OneWire devices
// (not just Maxim/Dallas temperature ICs)
OneWire oneWire(ONE_WIRE_BUS);
/*****************************************************************/
// Pass our oneWire reference to Dallas Temperature.
DallasTemperature sensors(&oneWire);
/*****************************************************************/


/*****************************************************************/
/*                  LUX Sensor Setup                             */
/*****************************************************************/
Adafruit_TSL2561_Unified tsl = Adafruit_TSL2561_Unified(TSL2561_ADDR_FLOAT, 12345);



void onEvent (ev_t ev) {
    //Serial.print(os_getTime());
    //Serial.print(": ");
    switch(ev) {
        case EV_SCAN_TIMEOUT:
            //Serial.println(F("EV_SCAN_TIMEOUT"));
            break;
        case EV_BEACON_FOUND:
            //Serial.println(F("EV_BEACON_FOUND"));
            break;
        case EV_BEACON_MISSED:
            //Serial.println(F("EV_BEACON_MISSED"));
            break;
        case EV_BEACON_TRACKED:
            //Serial.println(F("EV_BEACON_TRACKED"));
            break;
        case EV_JOINING:
            //Serial.println(F("EV_JOINING"));
            break;
        case EV_JOINED:
            //Serial.println(F("EV_JOINED"));
            break;
        case EV_RFU1:
            //Serial.println(F("EV_RFU1"));
            break;
        case EV_JOIN_FAILED:
            //Serial.println(F("EV_JOIN_FAILED"));
            break;
        case EV_REJOIN_FAILED:
            //Serial.println(F("EV_REJOIN_FAILED"));
            break;
            break;
        case EV_TXCOMPLETE:
            //Serial.println(F("EV_TXCOMPLETE (includes waiting for RX windows)"));
```

```
            if(LMIC.dataLen) {
                // data received in rx slot after tx
                //Serial.print(F("Data Received: "));
                //Serial.write(LMIC.frame+LMIC.dataBeg, LMIC.dataLen);
                //Serial.println();
            }
            // Schedule next transmission
            digitalWrite(13, LOW);
            os_setTimedCallback(&sendjob, os_getTime()+sec2osticks(TX_INTERVAL), do_send);
            break;
        case EV_LOST_TSYNC:
            //Serial.println(F("EV_LOST_TSYNC"));
            break;
        case EV_RESET:
            //Serial.println(F("EV_RESET"));
            break;
        case EV_RXCOMPLETE:
            // data received in ping slot
            //Serial.println(F("EV_RXCOMPLETE"));
            break;
        case EV_LINK_DEAD:
            //Serial.println(F("EV_LINK_DEAD"));
            break;
        case EV_LINK_ALIVE:
            //Serial.println(F("EV_LINK_ALIVE"));
            break;
         default:
            //Serial.println(F("Unknown event"));
            break;
    }
}

void do_send(osjob_t* j){
    // Check if there is not a current TX/RX job running
    if (LMIC.opmode & OP_TXRXPEND) {
        //Serial.println(F("OP_TXRXPEND, not sending"));
        //Serial.print("OP_TXRXPEND, not sending; at freq: ");
        //Serial.println(LMIC.freq);
    } else {
        uint16_t int_temp, int_lux;
        // Prepare upstream data transmission at the next possible time.
        //LMIC_setTxData2(1, mydata, sizeof(mydata)-1, 0);
        digitalWrite(13, HIGH);
        sensors.requestTemperatures(); // Send the command to get temperature readings

        /*****************************************************************************/
        /*                      WP Temp sensor                       */
        int_temp = (uint16_t)(sensors.getTempCByIndex(0)*100);
        //Serial.print("Temperature is: ");
        //Serial.println(int_temp/100);

        mydata[2] = (uint8_t)(int_temp >> 8);
        mydata[3] = (uint8_t)(int_temp & 0xFF);


        /*temp = (mydata[0] << 8 | mydata[1]);
        Serial.print("Temperature from uint8_t array: ");
```

```
      Serial.println(temp/100);
      */
      //Serial.println("");
      /*************************************************************************/
      /*                    LUX Sensor                         */
      sensors_event_t event;
      tsl.getEvent(&event);

      /* Display the results (light is measured in lux) */
      if (event.light)
      {
        /*
        Serial.print("Lux is :");
        Serial.println(event.light);
        Serial.println("");*/
        int_lux = (event.light);
        mydata[4] = (uint8_t)(int_lux >> 8);
        mydata[5] = (uint8_t)(int_lux & 0xFF);
      }
      else
      {
        /* If event.light = 0 lux the sensor is probably saturated
           and no reliable data could be generated! */
        //Serial.println("Sensor overload");
      }

      //mydata[0]=data.celsiusHundredths/100;
      //mydata[1]=data.humidityBasisPoints/100;

     LMIC_setTxData2(1, mydata, sizeof(mydata), 0);


      //Serial.println(mydata[0]);
      //Serial.print(F("Packet queued for freq: "));
      //Serial.println(LMIC.freq);
    }
    // Next TX is scheduled after TX_COMPLETE event.
}

void setup() {
    //Serial.begin(9600);
    //while (!Serial);
    //Serial.println(F("Starting"));
    pinMode(13, OUTPUT);          // set pin to input

    sensors.begin();

    //Set Project ID
    mydata[0] = PROJID;
    //Set Location ID
    mydata[1] = LOCID;

    // LMIC init
    os_init();
    // Reset the MAC state. Session and pending data transfers will be discarded.
    LMIC_reset();
```

```
    // Set static session parameters. Instead of dynamically establishing a session
    // by joining the network, precomputed session parameters are be provided.
    #ifdef PROGMEM
    // On AVR, these values are stored in flash and only copied to RAM
    // once. Copy them to a temporary buffer here, LMIC_setSession will
    // copy them into a buffer of its own again.
    uint8_t appskey[sizeof(APPSKEY)];
    uint8_t nwkskey[sizeof(NWKSKEY)];
    memcpy_P(appskey, APPSKEY, sizeof(APPSKEY));
    memcpy_P(nwkskey, NWKSKEY, sizeof(NWKSKEY));
    LMIC_setSession (0x1, DEVADDR, nwkskey, appskey);
    #else
    // If not running an AVR with PROGMEM, just use the arrays directly
    LMIC_setSession (0x1, DEVADDR, NWKSKEY, APPSKEY);
    #endif

    // THIS IS WHERE THE AUSTRALIA FREQUENCY MAGIC HAPPENS!
    // The frequency plan is hard-coded
    // But the band (or selected 8 channels) is configured here!
    // This is the same AU915 band as used by TTN

    // First, disable channels 8-16
    for (int channel=8; channel<16; ++channel) {
      LMIC_disableChannel(channel);
    }
    // Now, disable channels 16-72 (is there 72 ??)
    for (int channel=16; channel<72; ++channel) {
       LMIC_disableChannel(channel);
    }
    // This means only channels 8-15 are up

    // Disable link check validation
    LMIC_setLinkCheckMode(0);
    LMIC_setAdrMode(0);
    // Set data rate and transmit power (note: txpow seems to be ignored by the library)
    LMIC_setDrTxpow(DR_SF10,20);

    // Start job
    do_send(&sendjob);
}

void loop() {
    os_runloop_once();

}
```

4) LoRaWAN Gateway

The LoRaWAN Gateway setup is out of the scope of this project and was setup by Innovation Central Perth through Cisco [10].

5) Network Server Code

The network server receives packets from the LoRaWAN gateway and redirects them to the appropriate application server. It was calibrated to run a custom procedure within a Python webhook when a device address was recognised. The device addresses used for this project were “0x00540006”, “0x00540007” and “0x00540008”, and are defined in the Arduino Header File – config.h. Alternatively, Adafruit IO can be used as an application server providing a more intuitive user interface [13].

a) webhook.py Procedure - jdownes()

```
from mysql.connector import MySQLConnection
from time import strftime, sleep

@app.route('/jdownes', methods=['POST'])
def jdownes():

    packet = json.loads(request.data)

    conn = MySQLConnection( host = '10.0.10.220',
                        database = 'ICP',
                        user = 'tran',
                        password = 'GreenX0123')
    curs = conn.cursor()

    # unit ID
    uID = "0x"+packet["DevEUI_uplink"]["DevAddr"]
    # time
    time = strftime('%Y-%m-%d %H:%M:%S')

    pl =  packet["DevEUI_uplink"]["payload_hex"]

    #ProjectID
    pID = (int(pl[0:2], 16))

    #LocationID
    lID = (int(pl[2:4], 16))

    #temp
    tmp = (float(int(pl[4:8], 16)))/100

    #humidity
    lux = (float(int(pl[8:12], 16)))

    #ints
    three = (float(int(pl[12:16], 16)))

    four = (float(int(pl[16:20], 16)))

    #floats
    five = (float(int(pl[20:24], 16)))/100

    six = (float(int(pl[24:28], 16)))/100


    add_read = ("CALL addRead(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)")

    data = (uID, time, pID, lID, tmp, lux, three, four, five, six)
    curs.execute(add_read, data)

    conn.commit()

    curs.close()
    conn.close()
    return Response()
```

6) Application Server Code

This project utilised a Virtual Machine running Ubuntu Server 16.4.3 with MYSQL installed and the default MYSQL TCP/IP port 3306 opened.

The following SQL script create_tables.sql creates a MYSQL schema defined as 'ICP' to retain the data from the sensors. Figure 13 is the UML design for the database. It is designed to serve as the backend of a user-interface. Values01 was merged into the Readings table, so it included a total of six float fields for sensor values to reduce complexity at the cost of abstraction and space. In hindsight the schema could be reduced to include only the tables: Units, Readings and Projects.

The create_procs.sql script defines functions for interacting with the database.

A password protected user with read-write privileges 'tran' was created for transferring and entering data. It's details are used when connecting to the database within the jdownes() webhook procedure. The 'admin' account was used for creating the schema and adding tuples of structural importance into the tables UnitTypes and Locations. A 'query' account with read-only privileges is recommended to be used when accessing the database as a client to retain integrity.

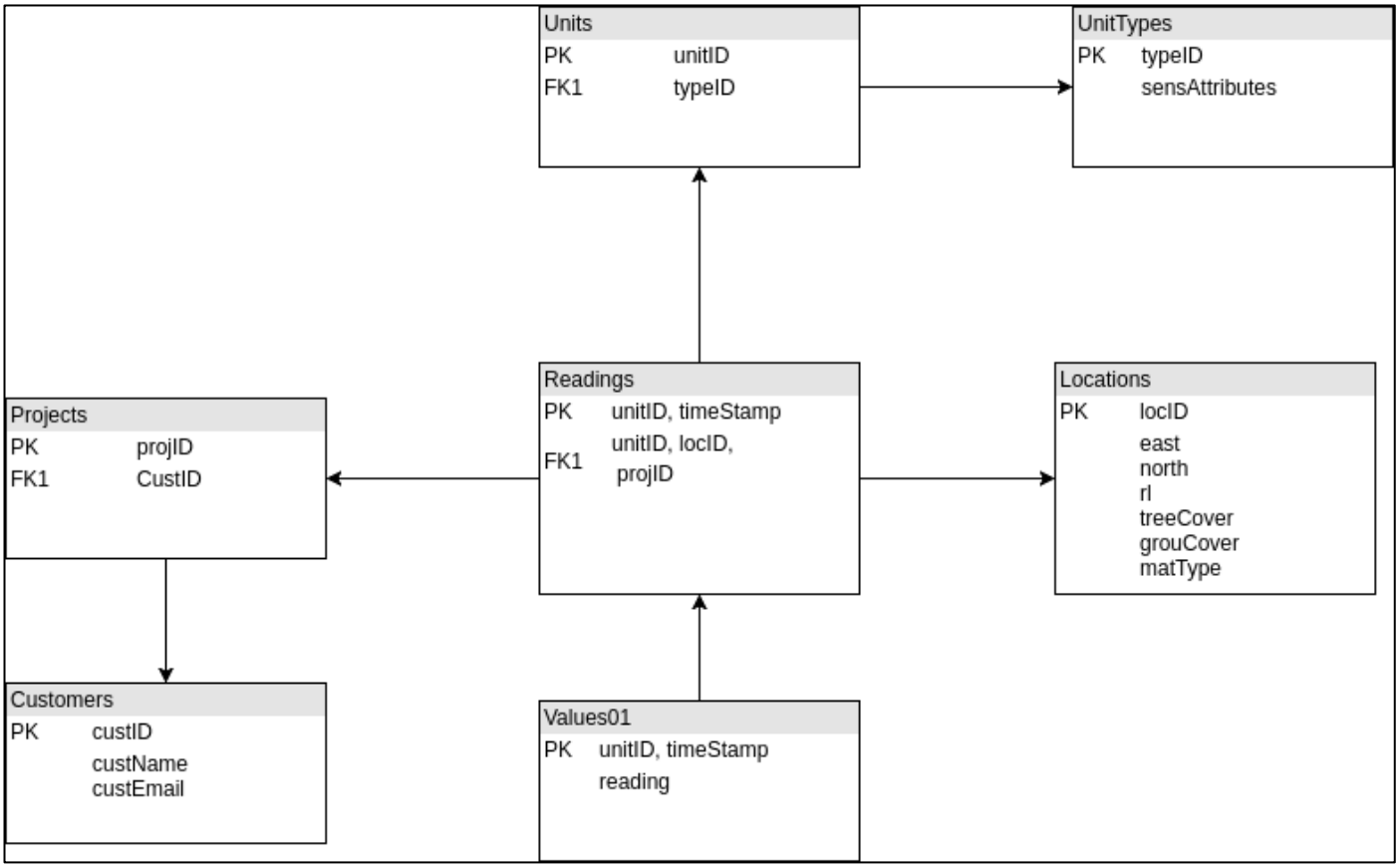

Figure 13 - UML application server database design

a) create_tables.sql

```
##LoRaWAN Application database structure

ROP TABLE IF EXISTS UnitType;
CREATE TABLE UnitType (
   typeID      INT(4),
   attributes   CHAR(200),
   PRIMARY KEY (typeID)
);

DROP TABLE IF EXISTS Unit;
CREATE TABLE Unit (
```

```
  unitID VARCHAR(10),
  typeID INT(4),
  PRIMARY KEY (unitID),
  FOREIGN KEY (typeID) REFERENCES UnitType(typeID)
);

DROP TABLE IF EXISTS Cust;
CREATE TABLE Cust (
   custID  INT(4),
   name    CHAR(50) NOT NULL,
   email   CHAR(50) NOT NULL,
   PRIMARY KEY (custID)
);

DROP TABLE IF EXISTS Proj;
CREATE TABLE Proj (
   custID INT(4),
   projID INT(4),
   projName CHAR(200) NOT NULL,
   PRIMARY KEY (projID),
   FOREIGN KEY (custID) REFERENCES Cust(custID)
);

DROP TABLE IF EXISTS Mat;
CREATE TABLE Mat (
   matID      INT(4),
   descr      CHAR(200),
   PRIMARY KEY (matID)
);

DROP TABLE IF EXISTS Loc;
CREATE TABLE Loc (
   locID      INT(4),
   matID      INT(4),
   east       DECIMAL(6,2),
   north      DECIMAL(6,2),
   rl         DECIMAL(4,2),
   treeCover  DECIMAL(2,2),
   comment    CHAR(200),
   PRIMARY KEY (locID),
   FOREIGN KEY (matID) REFERENCES Mat(matID)
);


DROP TABLE IF EXISTS Reading;
CREATE TABLE Reading(
   unitID     VARCHAR(100),
   dt         DATETIME,
   projID     INT(4),
   locID      INT(4),
   valOne     FLOAT,
   valTwo     FLOAT,
   valThree   FLOAT,
   valFour    FLOAT,
   valFive    FLOAT,
   valSix     FLOAT,
   PRIMARY KEY (unitID,dt),
```

```
    FOREIGN KEY (projID) REFERENCES Proj(projID),
    FOREIGN KEY (unitID) REFERENCES Unit(unitID),
    FOREIGN KEY (locID) REFERENCES Loc(locID)
);
```

b) create_procs.sql

```
##PROCEDURES TO HANDLE SAFE DATABASE MANIPULATION - to be called by front end

DROP PROCEDURE IF EXISTS addCust;
DELIMITER $$
CREATE PROCEDURE addCust(
   n CHAR(50),
   e CHAR(50)
   )
   COMMENT 'Get next Cust ID and add new customer'
   BEGIN
      DECLARE c INT(5);
      SET c = (SELECT COUNT(*) FROM Cust);
      IF NOT EXISTS (SELECT email FROM Cust WHERE email = e) THEN
         IF (c > 0 && c < 9999) THEN
            SET @id = (SELECT (MAX(custID)+1) FROM Cust);
            INSERT INTO Cust (custID, name, email)
            VALUES (@id, n, e);
         ELSEIF (c = 0) THEN
            INSERT INTO Cust (custID, name, email) VALUES (0001, n, e);
         ELSEIF (c > 9999) THEN
            SELECT 'customer table exceeds limit of 9999 entries';
         END IF;
      ELSE
         SELECT 'customer email already exists' AS Error, e AS Email;
      END IF;
   END $$
DELIMITER ;



DROP PROCEDURE IF EXISTS addProj;
DELIMITER $$
CREATE PROCEDURE addProj(
   n CHAR(50),
   i CHAR(50)
   )
   COMMENT 'Get next Proj ID and add new proj'
   BEGIN
      DECLARE c INT(5);
#TODO: if name exists deny
      IF EXISTS (SELECT custID FROM Cust WHERE custID = i) THEN
         SET c = (SELECT COUNT(*) FROM Proj);
         IF (c > 0 && c < 9999) THEN
            SET @id = (SELECT (MAX(projID)+1) FROM Proj);
            INSERT INTO Proj (custID, projID, projName)
            VALUES (i, @id, n);
         ELSEIF (c = 0) THEN
            INSERT INTO Proj (custID, projID, projName)
            VALUES (i, 0001, n);
         ELSE
            SELECT 'project table exceeds limit of 9999 entries';
         END IF;
      END IF;
   END $$
DELIMITER ;
```

```
DROP PROCEDURE IF EXISTS addUnitType;
DELIMITER $$
CREATE PROCEDURE addUnitType(
   a CHAR(200)
   )
   COMMENT 'Get next UnitTypeID and add new unitType'
   BEGIN
      DECLARE c INT(5);
      SET c = (SELECT COUNT(*) FROM UnitType);
      IF (c > 0 && c < 9999) THEN
         SET @id = (SELECT (MAX(typeID)+1) FROM UnitType);
         INSERT INTO UnitType(typeID, attributes)
         VALUES (@id, a);
      ELSEIF (c = 0) THEN
         INSERT INTO UnitType(typeID, attributes)
         VALUES (0001, a);
      ELSE
         SELECT 'UnitType table exceeds limit of 9999 entries';
      END IF;
   END $$
DELIMITER ;

DROP PROCEDURE IF EXISTS addUnit;
DELIMITER $$
CREATE PROCEDURE addUnit(
   t INT(4),
   i CHAR(10)
   )
   COMMENT 'Get next UnitID and add new unit'
   BEGIN
      DECLARE c INT(5);
      IF EXISTS (SELECT typeID FROM UnitType WHERE typeID = t) THEN
         INSERT INTO Unit(unitID, typeID)
         VALUES (i, t);
      ELSE
         SELECT 'unit type does not exist' AS Error, t AS typeID;
      END IF;
   END $$
DELIMITER ;

DROP PROCEDURE IF EXISTS addLoc;
DELIMITER $$
CREATE PROCEDURE addLoc(
   m INT(4)
   )
   COMMENT 'create new location ID, other vals are NULL'
   BEGIN
      DECLARE c INT(5);
      IF EXISTS (SELECT matID FROM Mat WHERE matID = m) THEN
         SET c = (SELECT COUNT(*) FROM Loc);
         IF (c > 0 && c < 9999) THEN
            SET @id = (SELECT (MAX(locID)+1) FROM Loc);
            INSERT INTO Loc (locID)
            VALUES (@id);
         ELSEIF (c = 0) THEN
            INSERT INTO Loc (locID)
            VALUES (0001);
```

```
        ELSE
          SELECT 'project table exceeds limit of 9999 entries';
        END IF;
      END IF;
    END $$
DELIMITER ;

DROP PROCEDURE IF EXISTS addMat;
DELIMITER $$
CREATE PROCEDURE addMat(
    d CHAR(200)
    )
    COMMENT 'create new Material ID, details recorded in comment'
    BEGIN
      DECLARE c INT(5);
      SET c = (SELECT COUNT(*) FROM Mat);
      IF (c > 0 && c < 9999) THEN
        SET @id = (SELECT (MAX(matID)+1) FROM Mat);
        INSERT INTO Mat(matID, descr)
        VALUES (@id, d);
      ELSEIF (c = 0) THEN
        INSERT INTO Mat(matID, descr)
        VALUES (0001, d);
      ELSE
        SELECT 'Unit table exceeds limit of 9999 entries';
      END IF;
    END $$
DELIMITER ;

DROP PROCEDURE IF EXISTS addRead;
DELIMITER $$
CREATE PROCEDURE addRead(
    u VARCHAR(100),       #unit ID
    dt DATETIME,    #timestamp of recording time
    p INT(4),       #project ID
    l INT(4),       #location ID
    a FLOAT, #sensor 1 value....
    b FLOAT,
    c FLOAT,
    d FLOAT,
    e FLOAT,
    f FLOAT
    ) #...sensor 6 value
    COMMENT 'add a new reading to the Reading table, comes from LoRa Server'
    BEGIN
      IF EXISTS (SELECT unitID FROM Unit WHERE unitID = u) &&
      EXISTS (SELECT projID FROM Proj WHERE projID = p) &&
      EXISTS (SELECT locID FROM Loc WHERE locID = l) THEN
        INSERT INTO Reading VALUES(u, dt, p, l, a, b, c, d, e, f);
      ELSE
        SELECT 'ID does not exist, review table entries and try again';
      END IF;
    END$$
DELIMITER ;
```

7) Client Code

The following are example MYSQL queries that can be used to interface with the database. Utilising procedures where available. The Reading Query was executed in MYSQL Workbench, then the export function was used to save the data as a comma separated values (csv) file for in

a) Add a new Sensor ID

```
CALL addUnit(1, ‘0x00540006’);
```

b) Reading Query

```
SELECT
        unitID 'Unit ID',
   DATE_SUB(dt, INTERVAL 3 HOUR) AS 'Date Time',
   ProjID AS 'Proj ID',
   valOne AS 'Temperature',
   valTwo AS 'Light'
FROM Reading
WHERE projID = 2
        AND unitID = "0x00540006"
ORDER BY dt;
```

8) Data Analysis

Analysis was completed using Python within the Spyder IDE. The Spyder IDE is a program aggregating Python data analysis libraries in a user friendly interface supporting file management, text editing, command line and file viewing functionality. The following python scripts will also operate in a terminal with access to the correct libraries and could be implemented into a user friendly platform. The data structure header names used for the following scripts are dependent on how the file is queried, they use the format outlines in section 7)b)XI.B.7)b).

a) Gross and Systematic Error Filtering

```
import pandas as pd #dataframe and plotting
import numpy as np #data clean


# ========================================================================
# clean to data by recognizing zeros and temp of 85 as nan
# axis = 0 for row, 1 for column
# ========================================================================
def clean(data):
    data = data.replace(0,np.nan)
    data = data.replace(85,np.nan)
    data = data.dropna(axis=0,  how = 'any')
    data['Date Time'] = pd.to_datetime(data['Date Time'])
    return data;

input = pd.read_csv('input.csv')
input = clean(data = input)
```

b) Time Filtering

```
import pandas as pd #dataframe and plotting

def timeframe(start, end, data):
    s = pd.to_datetime(start)
    e = pd.to_datetime(end)
    data = data[data['Date Time'] > s]
    data = data[data['Date Time'] < e]
    return data;

sensor = pd.read_csv('input.csv')
sensor = clean(data = sensor)


start = '2018-01-25 12:00:00'
end = '2018-01-26 12:00:00'
```

c) Plotting

```
import pandas as pd #dataframe and plotting
import numpy as np #data clean
import matplotlib.pyplot as plt


sensor = pd.read_csv('input.csv')
sensor = clean(data = sensor)


ax = sensor.plot(color = 'r', x='Date Time', y= 'Temperature')
```

```
sensor.plot(secondary_y=True, color = 'y', x='Date Time', y= 'Light', ax=ax)
ax.set(xlabel='Date Time (DD-MM HH)', ylabel='Temperature (°C)')
ax.legend(["Red Brick", "Light"])
plt.savefig(brick_sensors_light.pdf', bbox_inches='tight')
```

d) Example Filter and Plot from Multiple Sources

```
import pandas as pd #dataframe and plotting
import numpy as np #data clean
import matplotlib.pyplot as plt
#import matplotlib.dates
#import datetime as dt


# =============================================================================
# clean to data by recognizing zeros and temp of 85 as nan
# axis = 0 for row, 1 for column
# =============================================================================
def clean(data):
    data = data.replace(0,np.nan)
    data = data.replace(85,np.nan)
    data = data.dropna(axis=0,  how = 'any')
    data['Date Time'] = pd.to_datetime(data['Date Time'])
    return data;


def timeframe(start, end, data):
    s = pd.to_datetime(start)
    e = pd.to_datetime(end)
    data = data[data['Date Time'] > s]
    data = data[data['Date Time'] < e]
    return data;
# =============================================================================
# Import todays csv
# =============================================================================

sensor = pd.read_csv('7_Thursday.csv')
sensor = clean(data = sensor)

bom = pd.read_csv('bom.csv')
bom = clean (data = bom)

start = '2018-01-25 12:00:00'
end = '2018-01-26 12:00:00'

sensor = timeframe(start, end, sensor)
bom = timeframe(start, end, bom)

# =============================================================================
# Plot light and temp seperately
# =============================================================================
ax = sensor.plot(color = 'r', x='Date Time', y= 'Temperature')
#sensor.plot(secondary_y=True, color = 'y', x='Date Time', y= 'Light', ax=ax)
bom.plot(color = 'b', x='Date Time', y= 'Temperature', ax=ax)
ax.set(xlabel='Date Time (DD-MM HH)', ylabel='Temperature (°C)')
ax.legend(["Red Brick", "Weather Station"])
plt.savefig('bom_vs_brick_sensors_labels.pdf', bbox_inches='tight')
```