



THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

FarmBot Wireless Sensors: Super Project

by

Daniel Ju

Submitted for the degree of Bachelor Engineering (Honours)
in the division of Electrical and Computer Engineering.

School of Information Technology and Electrical Engineering,
University of Queensland.

June 2019

Daniel Ju

daniel.ju@uqconnect.edu.au

10th June 2019

Prof Michael Brünig
Head of School
School of Information Technology and Electrical Engineering
The University of Queensland
St Lucia QLD 4072

Dear Professor Brünig,

In accordance with the requirements of the degree of Bachelor of Engineering (Honours) in the division of Electrical and Computer Engineering, I present the following thesis entitled

"FarmBot Wireless Sensors: Super Project"

The thesis was performed under the supervision of Mr Matthew DSouza I declare that the work submitted in the thesis is my own, except as acknowledged in the text and footnotes, and that it has not previously been submitted for a degree at the University of Queensland or any other institution.

Yours sincerely,

A handwritten signature in black ink, appearing to be 'Daniel Ju', with a stylized, cursive script.

Daniel Ju

Acknowledgements

This project was made possible from the work of open source projects and the incredible community behind them. Without the generosity of awesome people who make their work available for the public, certain parts of this project just wouldn't have been possible.

The project would like to further acknowledge the contributions and guidance provided by Mr Matthew DSouza, the academic who supervised the project, whose input helped guide a lot of the early design decisions.

Abstract

The goal of this project was to design and develop a minimal viable product which would be able to support the existing FarmBot super project (a UQ thesis project) by providing real-time data on plants surrounding the FarmBot. To achieve this the project uses a combination of Bluetooth enabled sensor probes and Bluetooth/LoRa enabled relays to deliver information across both short and long range scopes. Data delivered is handled by a central "LoRa server" which then forwards the received data to a separate web application where it is then stored using a database and presented from an online dashboard.

In assessment of the current conditions surrounding similar projects used in industry, a key focus has been placed on designing the project to be capable of working in isolated conditions where access to key power and internet infrastructure is lacking. The capable range and ability to track key environmental factors including temperature, lux, moisture and soil fertility were also considered. Ultimately the project is aimed to serve farmers in industry, by being an autonomous solution which gather and present key data on the current conditions of their agricultural holdings.

In its current the state the project operates as an independent module for the FarmBot project and functions sufficiently, currently collecting data on the plants used in testing the FarmBot project. Whilst only capable of handling small deployments, there is potential for further improvements which can scale the project and allow it to tackle greater deployment challenges.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 4 |
| 1.1 | Motivation | 4 |
| 1.2 | Case Studies | 5 |
| 1.2.1 | Product Modelling - MVP requirements of precision agriculture | 5 |
| 1.2.2 | User Modelling - Perceptions and attitudes toward precision agriculture technologies | 7 |
| 1.2.3 | Environment Modelling - Agriculture in rural Australia | 9 |
| 2 | Problem Synopsis | 11 |
| 2.1 | Problem Definition | 11 |
| 2.2 | Project Scope | 12 |
| 3 | Solution Synopsis | 13 |
| 3.1 | Problem Definition Mapping | 13 |
| 3.2 | Solution Overview | 15 |
| 3.2.1 | Sensor Nodes | 15 |
| 3.2.2 | Communication Client | 15 |
| 3.2.3 | Communication Gateway | 15 |
| 3.2.4 | Web Application | 16 |
| 3.3 | Comparable Solutions | 17 |
| 3.3.1 | Zigbee/IEEE802.15.4 | 17 |
| 3.3.2 | White-Fi/IEEE 802.11 ah | 17 |
| 3.3.3 | Sigfox | 17 |
| 3.3.4 | DASH7 | 18 |
| 4 | Technology Review | 19 |
| 4.1 | Firmware Review | 19 |
| 4.1.1 | Cost | 19 |
| 4.1.2 | Power Consumption | 20 |
| 4.1.3 | BLE4.0 Functionality | 21 |
| 4.2 | FreeRTOS | 25 |
| 4.2.1 | Task Management | 25 |
| 4.2.2 | Communication and Synchronisation | 25 |
| 4.3 | ESP32 LMIC | 26 |
| 4.3.1 | LoRa Protocol Stack | 26 |
| 4.3.2 | LoRa Frame Format | 27 |
| 4.3.3 | LoRa Network Architecture | 27 |
| 4.3.4 | LoRaWAN Protocol | 28 |
| 4.4 | LoRa Server | 30 |
| 4.5 | Web Application | 31 |
| 4.5.1 | MEAN | 31 |
| 4.5.2 | Service Workers | 31 |
| 4.5.3 | Database Solution | 31 |
| 4.5.4 | Modelling and Classification using TensorFlow | 32 |
| 5 | Current Implementation | 33 |
| 5.1 | Deployment Process | 34 |
| 5.1.1 | WSN | 34 |
| 5.1.2 | Web application | 35 |
| 5.2 | Component Breakdown | 36 |
| 5.2.1 | Sensor node | 36 |
| 5.2.2 | Communication client | 37 |
| 5.2.3 | LoRa Server | 38 |
| 5.2.4 | Web Application | 39 |

| | | |
|----------|---|-----------|
| 5.3 | Communication Scheme | 42 |
| 5.3.1 | BLE4.0/GATT | 42 |
| 5.3.2 | LoRaWAN Payload | 44 |
| 5.3.3 | HTTP Payload | 44 |
| 6 | Project Limitations | 46 |
| 6.1 | Limitations in sending downlik frames | 46 |
| 6.2 | Lack of Multi-hop, mesh, or P2P | 47 |
| 6.3 | Weak localisation features | 47 |
| 7 | Concluding Remarks | 48 |

List of Figures

| | | |
|----|---|----|
| 1 | Project model case study network overview (1) | 6 |
| 2 | Project model case study deployment in the field (1) | 6 |
| 3 | Proposed user case study model and path analysis (2) | 8 |
| 4 | Forecast percentage declines of wheat yields in Australia (3) | 9 |
| 5 | Internet coverage in Australia mid 2017 provided by Ookla (4) | 10 |
| 6 | Network overview | 16 |
| 7 | SigFox coverage in Australia | 18 |
| 8 | Architecture of the Classic BT and BLE controller (5) | 22 |
| 9 | BLE states on the ESP32 (5) | 23 |
| 10 | Example GATT server/client interaction (5) | 23 |
| 11 | Frequency channels used by BLE (6) | 24 |
| 12 | Example LoRa transmission as frequency variations over time (7) | 26 |
| 13 | Example LoRa frame (7) | 27 |
| 14 | Example LoRa network architecture | 28 |
| 15 | Example LoRaWAN frame (7) | 29 |
| 16 | (a) sample set of communication client and sensor nodes, (b) current deployment setting | 33 |
| 17 | Stripped Xiaomi MiFlora | 36 |
| 18 | Algorithm of the communication client | 37 |
| 19 | Snapshot of the device registration function on the LoRa server | 38 |
| 20 | Snapshot of the HTTP integration function on the LoRa server | 38 |
| 21 | Snapshot of the login page from the web application | 39 |
| 22 | Snapshot of the landing page for the account dashboard | 39 |
| 23 | Snapshot of the individual sensor node page from 22nd-23rd May, key observations include the event of rain during the period which had an impact on the sensor data collected | 40 |
| 24 | Current resource use by project service worker | 41 |
| 25 | DASH7 Alliance Protocol Communication Model (8) | 46 |

List of Tables

| | | |
|----|--|----|
| 1 | User case study highlights (2) | 8 |
| 2 | Core activities | 11 |
| 3 | External challenges | 11 |
| 4 | Operational requirements | 12 |
| 5 | Problem definition mapping | 14 |
| 6 | Price overview of considered micro controllers | 20 |
| 7 | ESP32 operating mode functionality (9) | 20 |
| 8 | ESP32 power consumption at each operating mode (9) | 21 |
| 9 | Example GATT server (5) | 24 |
| 10 | Available browser resources for common browsers (10) | 41 |
| 11 | Service and Characteristic UUIDs on the Xiaomi MiFlora | 42 |
| 12 | Xiaomi MiFlora device name | 42 |
| 13 | Xiaomi MiFlora device information | 42 |
| 14 | Xiaomi MiFlora device time | 43 |
| 15 | Xiaomi MiFlora real-time data | 43 |
| 16 | Xiaomi MiFlora historical data reference | 43 |
| 17 | Xiaomi MiFlora historical data value | 44 |
| 18 | LoRaWAN payload structure | 44 |
| 19 | HTTP payload structure | 45 |

1 Introduction

1.1 Motivation

Agriculture in Australia makes up approximately 12% of the nations GDP (11) with the industry covering close to 61% of Australia's landmass (12). As demands on the industry continue to expand with growing populations and increased export opportunities, modern agriculture is faced with a new set of challenges in attempting to serve larger production requirements. Alongside the increased production demand, modern agriculture also has to consider the growing effects of climate change (3) and rural water scarcity (1), the effects of which are heavily prominent in the Australian landscape.

With the industry continuing its rapid expansion, existing agriculture practices are beginning to be supplemented by new technologies. These technologies aim to improve general productivity through the collection and application of environmental and operational data (13). The effort to improve agriculture through the specific observing, measuring and responding to variability has been labelled "precision agriculture". Simply, precision agriculture is a management methodology in which data collection is used to greater inform and enhance decision making. Results from performing precision agriculture can include being able to identify and adjust sections of a field which are not receiving enough water or having early detection to decreasing soil fertility.

Precision agriculture is usually implemented over two levels (14). The first level, typically involves some form of physical interface; usually in the form of sensors and other measuring devices. Data collected from the first level is then fed to the second level which is then responsible for presenting the data in a way which it can then be interpreted by an end user.

Assumed benefits of this kind of data driven approach include (15):

- increased availability of informative data to farmers
- enhanced decision making by farmers' with a greater understanding of their paddock and plants,
- automation of tasks,
- increase certainty in operation outcomes,
- a general increase in potential yields,
- minimise wastage,
- save time,
- reduce costs.

A number of technologies have been proposed which all aim to assist farmers with their task of optimising their current practices. Some notable technologies include: wireless sensor networks (WSN), cloud computing and Internet of Things (IoT) (16). What these technologies offer is a solution to ways in which data can be delivered, stored and presented to farmers. As a basis, this thesis, aims to implement its own variation of precision agriculture which combines key concepts from the technologies mentioned previously to enable suitable data collection and presentation to assist farmers.

1.2 Case Studies

1.2.1 Product Modelling - MVP requirements of precision agriculture

Precision agriculture offers a technological solution to the traditionally labor intensive agriculture industry. The practice of precision agriculture can be seen as the amalgamation of many different methods and tools to facilitate accurate monitoring and automation in agriculture (17). Many precision agriculture solutions begin with a wireless sensor networks (WSN) which effectively connect agriculture fields to greater information technology infrastructure which can then be used to analyse and provide valuable insights to the current condition of the field that is being monitored (18). Ultimately it is the combined utility of sensing, processing, communication and actuation that makes precision agriculture extremely powerful in industry.

Whilst many implementations for precision agriculture exist, a fairly common example is autonomous irrigation. A basic implementation of autonomous irrigation uses a series of soil moisture probes which relay to a central service which coordinates through either a manual operator or autonomous system that certain areas of a field need watering (19). At higher levels this solution can be elevated to include temperature distribution, available light and soil fertility as factors for what areas require watering and in what volume.

In a previously implemented project titled *Automated Irrigation System Using a Wireless Sensor Network and GPRS Module* (1) the project utilised a WSN network which contained a combination of soil-moisture and temperature sensors to facilitate autonomous irrigation. The project was deployed with three primary components which included:

- **Wireless Sensor Unit:** The wireless sensor units (WSU) serve as the primary data collection device. Each WSU included a RF transceiver (XBeePro S2), soil moisture sensor (VH400), temperature sensor (DS1822), a microcontroller (PIC24FJ64GB004), and power sources (AA 2000-mAh Ni-MH CycleEnergy batteries).
- **Wireless information unit:** The wireless information unit (WIU) collects soil moisture and temperature data from each WSU. Data collected on the WIU is used to determine whether or not irrigation is required before transmitting the data over an internet connection to a MySQL database where it is then stored. Individual modules on the WIU include: a master microcontroller (PIC24FJ64GB004), XBee radio modem (XBeePro S2), GPRS module (MTSMC-G2-SP), an RS-232 inter-face (MAX3235E), two electronic relays, two 12 V dc 1100 GPH Livewell pumps, and a deep cycle 12 V at 100-Ah rechargeable battery (L-24M/DC-140).
- **Web application:** A simple graphical user interface is provided as the final component and is used to visually represent the data collected. The web application is a hosted solution and can therefore be accessed from any device with an internet connection.

Figure 1 shows the current network topology used by the project and figure 2 is a snapshot of the project in deployment. The primary modules of the network each facilitate a specific function namely: data collection (WSU), data delivery (WIU), data storage (web server/database) and data presentation (web server).

The project utilised Zigbee as the communication scheme between WSU and WIU devices. Justifications for this choice include lower cost, lower power consumption, and greater useful range in comparison with other wireless technologies (1). The implementation of Zigbee also allowed for mesh networking which enabled the dynamic expansion of the implemented network by simply introducing more devices to the network. Components of the Zigbee network architecture include: a coordinator, routers and end devices. The project configuration used WSUs as end devices to deploy a point-to-point networking topology based on the WIU which served as the coordinator for the network (1).

At the end, the project was seen as a success as it was able to viably, at cost, optimise water usage during its deployment. The project was able to action two specific activities which enabled it to accomplish its original objective: 1) data could be reliably delivered between relevant devices; 2) the data delivered could be interpreted into actionable events which could be carried out autonomously.

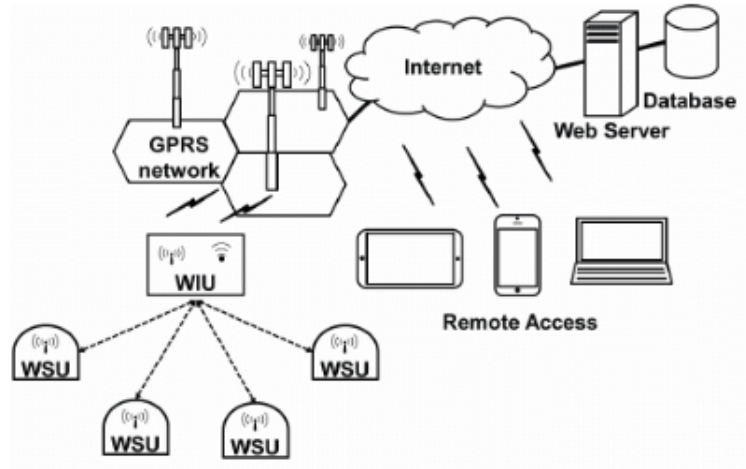


Figure 1: Project model case study network overview (1)

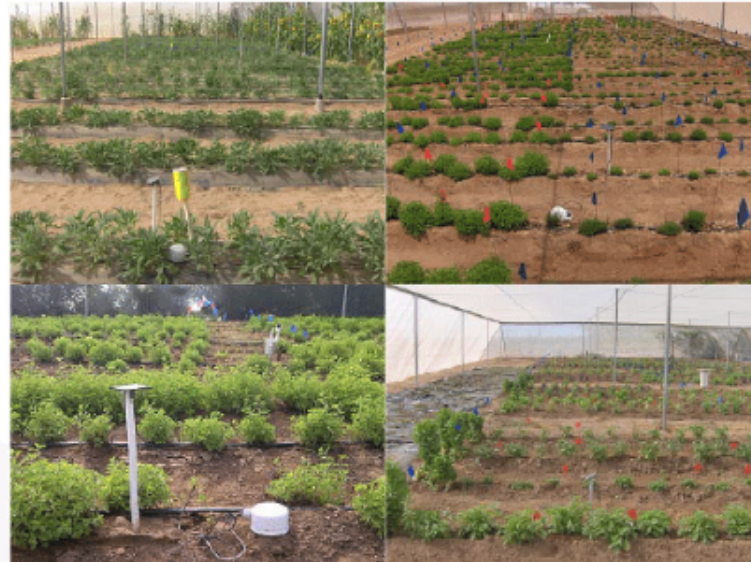


Figure 2: Project model case study deployment in the field (1)

1.2.2 User Modelling - Perceptions and attitudes toward precision agriculture technologies

In preparation of the projects design and development, a review of the current perceptions and attitudes of precision agriculture has been performed build an understanding of the environment and risks related. In six case studies from *Precision farming adoption and use in Ohio: case studies of six leading-edge adopters* (19) revealed that increased profitability alongside information for environmental compliance were the greatest motivator behind adoption of precision agriculture. Participants in the case studies also quoted the potential for on-farm experimentation, improved information to support decisions and risk reduction potentials as reasons for their adoption of precision agriculture. Alternatively, negative influences on the adoption of precision agriculture saw both financial and educational investments required to use the technology as primary factors, with secondary factors including the current compatibility and complexity of combining new practices with current operations and equipment.

In a later study *Producers' perceptions and attitudes toward precision agriculture technologies* (2), the study proposes a structural model and path analysis (figure 3) which may be used as a measure for current likelihood of adoption for precision agriculture. Captured in the model are the factors which contribute towards the end intention to either adopt or reject precision agriculture. Individually these factors are:

- **Perceived usefulness:** The perceived usefulness can be seen as the users belief that the use of the provided technology will enhance the current state of user/user's task. Possible points for perceived usefulness include: quality, control, productivity, effectiveness and performance.
- **Perceived ease of use:** The perceived ease of use is typically related to the complexity of the technology, with the objective being to reduce the physical and mental effort for the user.
- **Confidence:** Confidence, in context is a measure of how much trust an individual user has on his/her ability to be able to pick up and use the technology.
- **Perceived net benefit:** The net benefit defines the overall gain of adopting the technology. In the case of precision agriculture the net benefit can include: reductions in production costs, improving yields, better serving the environment and better logistical/management support.
- **Farm size:** Certain demographic factors may also influence the decision to adopt new technologies. For precision agriculture, farm size is one such consideration. Given the initial investment costs and augmentation to existing practises, small farms where there may be little variability or need for active monitoring may consider the adoption of precision agriculture unnecessary.
- **Education level:** The final consideration is the individual users level of education which may affect their ability and willingness to adopt new technologies.

At the conclusion of the study, results revealed that a steep learning curve and initial investment costs led to the majority rejection of precision agriculture by farmers who participated in the study. More so, the various combinations of tools and current complexity of solutions further complicated farmers' decisions. Highlights of the study are presented in table 1.

Key takeaways from the case study include the need to be able to reduce the current cost of precision agriculture solutions as well as an alternative method to reduce the current complexity so that any new solution is easier to simply deploy. Future solutions should also consider the demographic factors influencing decisions to adopt/reject precision agriculture. With farmers holding large farm sizes being the primary demographic, considerations should be made to accommodate for deployment fields of a larger size.

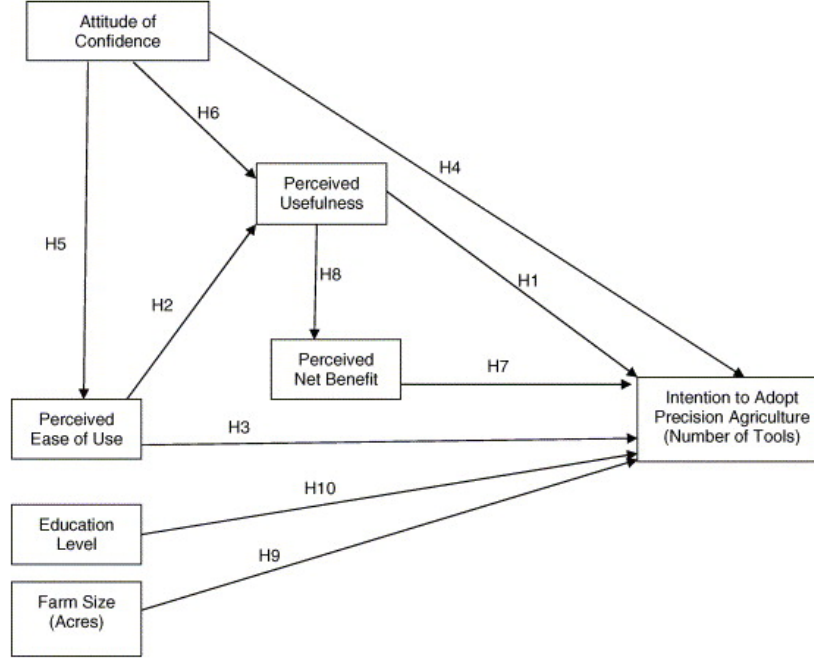


Figure 3: Proposed user case study model and path analysis (2)

| Item | Perceived usefulness (.9303) | Perceived ease of use (.8816) | Attitude of confidence (.8742) | Perceived net benefit (.8562) |
|---|---------------------------------|----------------------------------|-----------------------------------|----------------------------------|
| Using precision agriculture tools can improve the quality of my farm operation and management | .863 | .120 | .143 | .300 |
| I believe precision agriculture tools are cost effective | .046 | .354 | .061 | .737 |
| I believe precision agriculture tools can provide information for better decision making | .210 | .014 | .013 | .762 |
| Precision agriculture tools will be easy to use | .087 | .855 | .139 | .238 |
| I believe precision agriculture tools can increase profits | .336 | .149 | .137 | .806 |

Table 1: User case study highlights (2)

1.2.3 Environment Modelling - Agriculture in rural Australia

Agriculture in Australia faces many challenges, chief among them are drought and other extreme climate events which threaten daily operations (3) . Further effects from global impacts such as climate change has also seen the shift of climatic zones, with some areas reporting a 3°C change in mean annual temperature (3). The impact of such changes in combination with with altered rainfall, has diminished the agricultural capabilities of some areas. Figure 4 presents the future forecast for wheat yields in Australia as an affect of climate change.

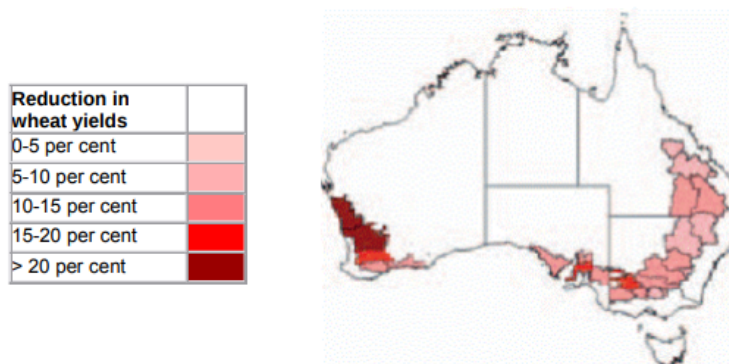


Figure 4: Forecast percentage declines of wheat yields in Australia (3)

In a paper by Western Australia’s department of agriculture, the effects of climate change in rural Australia in particular presents a unique challenge as it is likely to “*produce a diverse set of spatial impacts*” (3). The report further suggests that emergence of drier and warmer conditions are likely to lead to more extreme events such as drought, fire, excessive summer heat and severe storms. Also with fluctuating growing conditions farmers are most likely to suffer production hurdles being later reflected in significant business challenges; which come from an increased downward pressure for greater production; as growing demands for both domestic and foreign markets begin to rise.

Despite an obvious need for development an technological support in rural agriculture, many farmers of the region point out that their available access to the internet was simply not good enough to enable them to utilise new technological resources (20). In an interview with Jacki Schirmer an associate professor from the University of Canberra reported by the Australian Broadcasting Corporation (ABC) (21); Dr Schirmer states “*I’ve had farmers ringing me up saying ‘I’ve bought this piece of equipment for \$500,000 and to get the best out of it I need decent internet and I don’t have that, so I’ve wasted a bunch of money’*”. Dr Schirmer goes on to further highlight that whilst many farmers are being pushed to adopt new internet-based technology (precision agriculture included) many are simply unable to because of their current internet connection.

According to Ookla Australia’s internet rating is ranked 50th globally (22), beaten out by similar countries such as New Zeland and the UK who rank 27th and 15th respectively. The effects of Australia’s poor internet coverage is felt hardest in rural central regions of the country which sit outside the reach of major cities (4). Figure 5 provides a snapshot of Australia’s current coverage as of mid 2017. Regions which have the least coverage include: central Queensland, northern Western Australia and the the Northern Territory.

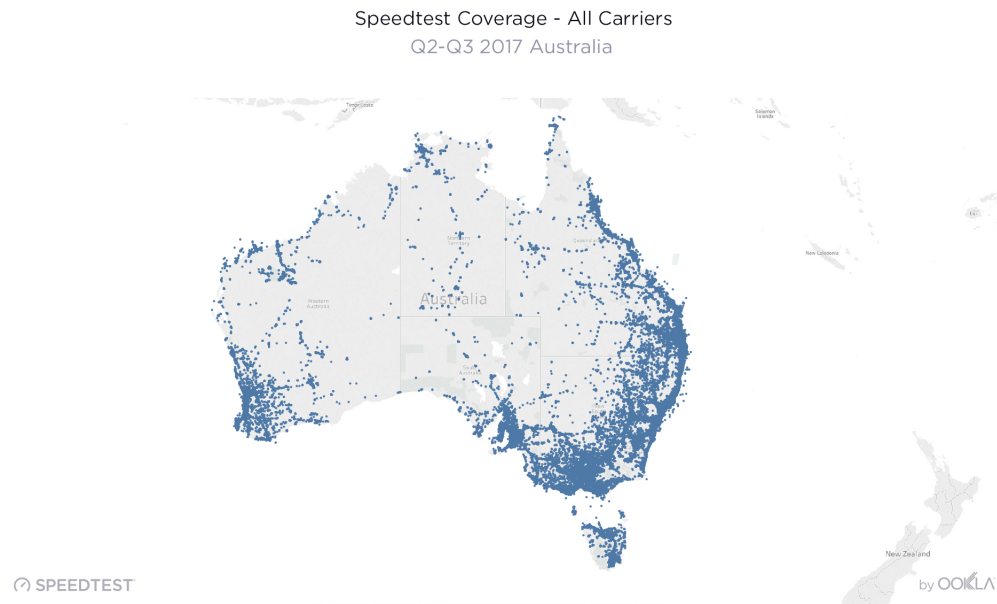


Figure 5: Internet coverage in Australia mid 2017 provided by Ookla (4)

From the environment model, key points raised include the need for robustness and durability of any intended solution to be capable of both monitoring and withstanding climate and weather effects in the field. A point could also be made about the available portability of a solution; with shifting "*spatial impacts*" there may be the need to have the solution picked up and redeployed. Operational constraints such as availability and limited access to internet services also need to be considered given the current situation.

2 Problem Synopsis

2.1 Problem Definition

Using the information gathered in the case studies, a problem definition was formed using the basis of core activities, which represent the critical requirements in order to deliver a minimum viable product (MVP) and external challenges which further enhance or improve the viability of the base product. Covered briefly in section 1.1, precision agriculture is typically carried out through both a physical component in the form of sensors and relays and a virtual component in the form of a web/mobile platform. A typical solution will utilise physical sensors to collect and a web platform to present the data. As an extension of the original two components, in developing the current implementation of the project four major activities were outlined which underline the minimum functionality required in both physical and virtual component. The four activities which are derived from the observations in section 1.2.1 form the core requirements of this project and are presented in table 2. Alongside the core activities, are the external challenges which are constructed

| Activity | Description |
|-------------------|--|
| Data collection | What mechanisms can be employed to facilitate the collection of relevant information. |
| Data delivery | How can data be transported between stages of the designed solution. |
| Data storage | Where will the data be stored and in what ways can it be managed/accessed. |
| Data presentation | How will the data be presented to the end user and what opportunity actions does it deliver. |

Table 2: Core activities

from both the user (section 1.2.2) and environment (section 1.2.3) models. These external challenges aim to relate the current project to the intended field of deployment and include considerations relevant to an agricultural setting. Specific considerations were made on the effects of working rurally in an outdoor environmental with little access to technological infrastructure. An outline of the specific external challenges considered are presented in table 3.

| Challenge | Description |
|---|---|
| Weather conditions | Because a part of the project must be deployed externally in the field. Project risks are introduced as a consequence of operating outdoors in exposure to the elements. Potential areas of risks include operations in high temperatures, rain or high winds |
| Availability of existing infrastructure | Given the remote setting rural Australia represents. The assumption is made that there will be little access to established power/communication infrastructure; this includes a lack of WiFi and cellular coverage and minimum access to socket power supply. |
| Length of deployment | The project should be able to independently sustain prolonged periods of deployments without the need of external attention/intervention from an operator. |
| Physical impact | In an agricultural plot the physical impact of the intended solution would aim to be minimised as a way to reduce the potential negative impact to field of deployment. |

Table 3: External challenges

2.2 Project Scope

With limited access to "realistic" agriculture fields this project uses a scaled version of the intended project scope to be implemented in conjunction with the current Farmbot super thesis project (23) ongoing at the University of Queensland. Operation with the Farmbot introduces a further set of requirements which have been included in (table 4). Due to the highly specific nature of operational requirements (table 4), these requirements are what will be used as a measure of success for the project. The requirements offer a qualitative measure of the current effectiveness of project functionality.

| Requirement | Description |
|--------------------------------------|--|
| Minimum sensor coverage | The sensor network to cover a plot of approximately 5m ² with reasonable precision and capability of determining basic operations i.e. watering (temperature/moisture sensors). |
| Minimum communication reach | The sensor network is capable of feeding data back to a central device spaced at least 20m away. |
| Operation in a "complex" environment | When deployed the implemented sensor network will be capable of communicating even in complex environments including both indoor and outdoor settings; which include external radio noise/signals. |
| Project costs | The final solution will be given a soft budget of approximately \$100 AUD, which will be used as a control to measure project costs. |
| Development requirements | The final solution will be modular, such that individual components of the project may be substituted/replaced in future iterations of the project. |
| Integration requirements | The final solution may be integrated with external projects through definable endpoints. |

Table 4: Operational requirements

3 Solution Synopsis

3.1 Problem Definition Mapping

In response to the outlined problem definition a constructed "problem-solution" mapping was performed (table 5) as a basis for determining the required feature-set for the project.

| Index | Problem Definition | Proposed Solution |
|-----------------|--------------------|---|
| Core Activities | | |
| 1.1 | Data collection | Data is to be collected via a series of stand alone probes (Xiaomi MiFlora). Sensors available on the probe include temperature, lux, moisture (as a percentage) and conductivity (used as an indicator of soil fertility). |
| 1.2 | Data delivery | Data is delivered across a hybrid model which includes both BLE4.0 and LoRa/LoRaWAN. |
| 1.3 | Data storage | Data will be stored through an online platform using MongoDB as the chosen database solution. |
| 1.4 | Data presentation | The project will include a web application, which will present data to end-users (farmers) through an online portal/dashboard. Specific elements on the web application will include graphing and classification tools. |

| | | |
|---------------------|---|--|
| External Challenges | | |
| 2.1 | Weather conditions | Network components will be housed in specific enclosures which will shield components from negative weather effects. |
| 2.2 | Availability of existing infrastructure | BLE4.0 and LoRa to be used as a replacement for WiFi/Cellular services. |
| 2.3 | Length of deployment | A duty cycle will be implemented which optimises power usage by powering down services when not in use. |
| 2.4 | Physical impact | A series of smaller sensors will be used in place of single larger device. This reduces the single point of impact and distributes the effect across a wider area. |

| Operational Requirements | | |
|--------------------------|--------------------------------------|--|
| 3.1 | Minimum sensor coverage | Though individual sensor capabilities are minimum, they can be used in a set to greater effect. |
| 3.2 | Minimum communication reach | LoRaWAN and BLE4.0 as an alternative to WIFI/Cellular |
| 3.3 | Operation in a "complex" environment | <p>BLE4.0 in short range and LoRa over long range allows for adequate cover in both scopes. Having sensors connected as individual probes over BLE4.0 also allows for the network to account for physical divisions (fenced plant beds) in the environment.</p> <p>This method also reduces the strain on LoRa channel usage by reducing the number of LoRa clients on the network. This allows the gateway to be more readily available and reduces the likelihood of transmission collision between communication clients.</p> |
| 3.4 | Project costs | Costs are minimised by using standalone modules. |
| 3.5 | Development requirements | Functionality to be implemented as separate drivers. |
| 3.6 | Integration requirements | An API layer to provide access to stored data. |

Table 5: Problem definition mapping

3.2 Solution Overview

3.2.1 Sensor Nodes

There are six typical abiotic factors which affect plant growth: air, water, space, temperature, light and soil (nutrients) (24). With the exception of air and space, each factor can be reliably tracked using a variety of different small scale sensors many of which come pre-bundled for the purpose of monitoring plants and agriculture. In the majority of cases the application of multiple sensors usually through sensor fusion (software which intelligently combines data from several sensors) has a significant impact in increasing application performance (1). The combination of data is aimed at correcting potential deficiencies of an individual sensor. By combining data from multiple sources, a more decisive conclusion can be reached.

In the case study presented in section 1.2.1 the project used a combination of both soil temperature and moisture sensors to determine irrigation needs of crops. As an extension, the current project will utilise a combined moisture, temperature, light, conductivity sensor in the Xiaomi MiFLora (25) to facilitate the data collection. By using this form of sensor, data can be reliably collected on temperature, light, water (moisture), and soil fertility (conductivity) from specific regions around the plot. Because the sensors are also BLE4.0 enabled, they can be polled remotely and can therefore be widely distributed. By capturing localised variables of four of the six factors which affect plant growth (24) and substituting the remaining air factors with general readings from a third party web API (26) and pre-defining the space factor manually. The overall data collected, will aim to give accurate predictions on the current state of plants growing within the Farmbot project.

3.2.2 Communication Client

As a way to optimise for the varying operating environments the current solution implements two communication protocols BLE4.0 and LoRa/LoRaWAN. Both the LoRaWAN and BLE4.0 functionality are handled by the communication client, with the client acting as a relay for the sensor nodes over long distances. The LoRaWAN client running on the gateway delivers an operational range of approx. 1.8KM allowing the client to reach ranges otherwise not possible by BLE4.0 alone. Whilst the LoRaWAN has a significantly longer range, it performs poorly in situations where multiple signals and large network traffic may be present.

To offset these effects BLE4.0 has implemented as an intermediary between the sensor nodes as opposed to having the sensor just be LoRa enabled. BLE4.0 has been seen to perform well in handling larger/more frequent network traffic (27)(28) and operates at a frequency away from the current LoRa implementation, 2400MHz range, to 923MHz respectively, avoiding cross communication interference. The application of BLE4.0 as an intermediary aims to streamline and minimise the number of end points (reducing the complexity) of the LoRaWAN component of the project. The result allows the LoRa gateway to be more readily available for uplink frames and reduces the likelihood of transmission collision between communication clients as there are less clients. Hence the combination of both technologies provide an optimised solution as the strengths of an individual communication protocol covers the weaknesses of the other.

3.2.3 Communication Gateway

The communication gateway is a LoRa Gateway Bridge which is used to forward received packets over MQTT to a local LoRaWAN network-server which handles the received uplink frames from gateway(s) and schedules downlink frames towards the clients. As an extension, the LoRaWAN network-server operating in conjunction with LoRa Gateway Bridge also provides its own set of application level functionality. Moreover, it includes an API layer which can be used to integrate with other available web applications.

3.2.4 Web Application

Outside of the implemented WSN is an external web application which operates as both a persistence layer and user-facing interface. The web application also includes a set of modeling (Chart.JS) and analytic type (TensorFlow) functionality which are aimed to enhance the data presented. The web application is built on top of a MEAN stack which includes technologies such as MongoDB (database solution), Express.JS (server framework), Angular (web framework) and Node.JS (development language). The overall implementation is a full end-end solution which delivers real time sensor data through a custom wireless sensor network to a user facing web interface (figure 6).

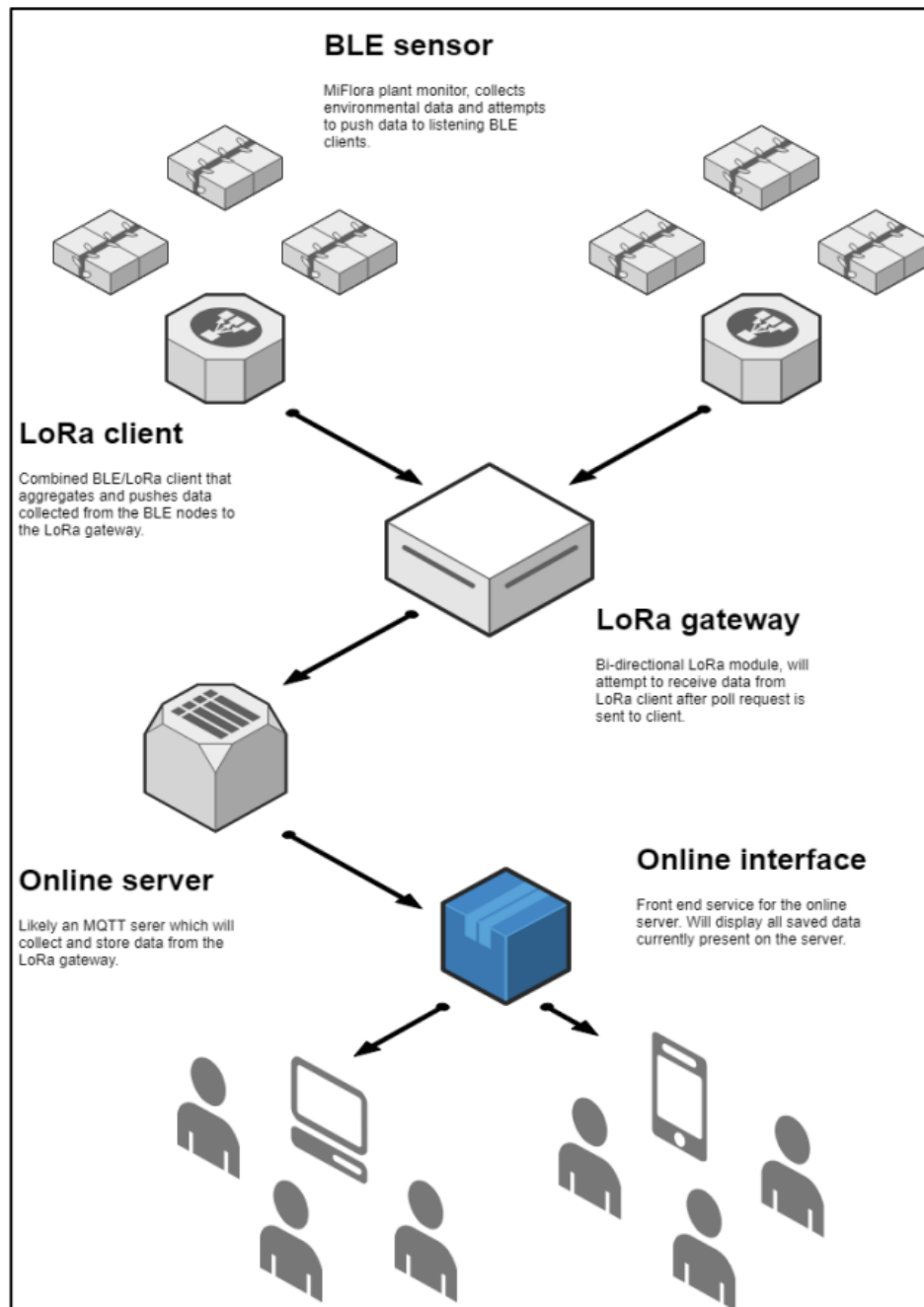


Figure 6: Network overview

3.3 Comparable Solutions

By leveraging both short range (BLE4.0) and a long range (LoRaWAN) technologies, the implemented WSN has the capacity to function well in both complex short range environments as well as in long range scenarios. Though the project has made the decision to move forward with this type of implementation there do exist alternatives which could be used in its place. However, for reasons provided, they have since been removed from consideration due to their respective cases.

3.3.1 Zigbee/IEEE802.15.4

Zigbee is a network layer extension built on top of the standards used in IEEE 802.15.4, a physical and data link layer for Low-Rate Wireless Personal Area Networks (29). Communication over Zigbee operates on three unlicensed frequency bands (868 MHz, Europe; 928 MHz, North America; 2.4 GHz, worldwide) and offers data rates up to 250 kbit/s (30). The range of Zigbee is relative to network size, as the communication protocol allows for meshing and ad-hoc connections between individual nodes, though the max range of an individual connection can range between 0-100m depending on obstructions between nodes (31).

Considerations which have removed Zigbee as an option for this project include the current limited support for Zigbee enabled devices and low availability of Zigbee modules. In comparison BLE4.0 is the most widely used communication protocol for small scale wireless technologies (32) and can also be meshed (28) if necessary to extend the short-medium range reach of each localised networks (not using LoRa) in the implemented WSN. Given the required range of this project it is also not feasible to use Zigbee as the possibility of meshing is not possible as there are no positions to intermediary nodes between the test field and the gateway. Cost is also a consideration when using Zigbee at scale, as multiple Zigbee nodes are required to cover the same distance as a single LoRa connection.

3.3.2 White-Fi/IEEE 802.11 ah

White-Fi is a variation of the traditional WiFi communication protocol which targets low power high range projects (33). IEEE 802.11 ah operates between 6-8MHz bandwidths and can support 26.7-35.6 Mbps at a ranges up to 1Km. A strong feature for White-Fi is support for up to four spatial data streams which can greatly increase the capable data rate (7). Major limitations of White-Fi is the current level of support and availability of the technology which has yet to see commercial use at the time of writing.

3.3.3 Sigfox

Sigfox is a popular variation of the traditional cellular system, targeted for low power/low cost projects (34). Sigfox is a proprietary technology developed by French company Sigfox and as such no public specification exists. However, from what information about the technology has been advertised: Sigfox operates on the 868-MHz frequency band using 400 channels of 100Hz (7). Standard end devices connected to the Sigfox network are given 140 messages per day, each of which are limited to a payload size of 12 Bytes which are sent at a data rate up to 100 bps (34). Poor coverage in Australia (figure 7) and the limitations of proprietary technology have removed Sigfox from consideration in this project.

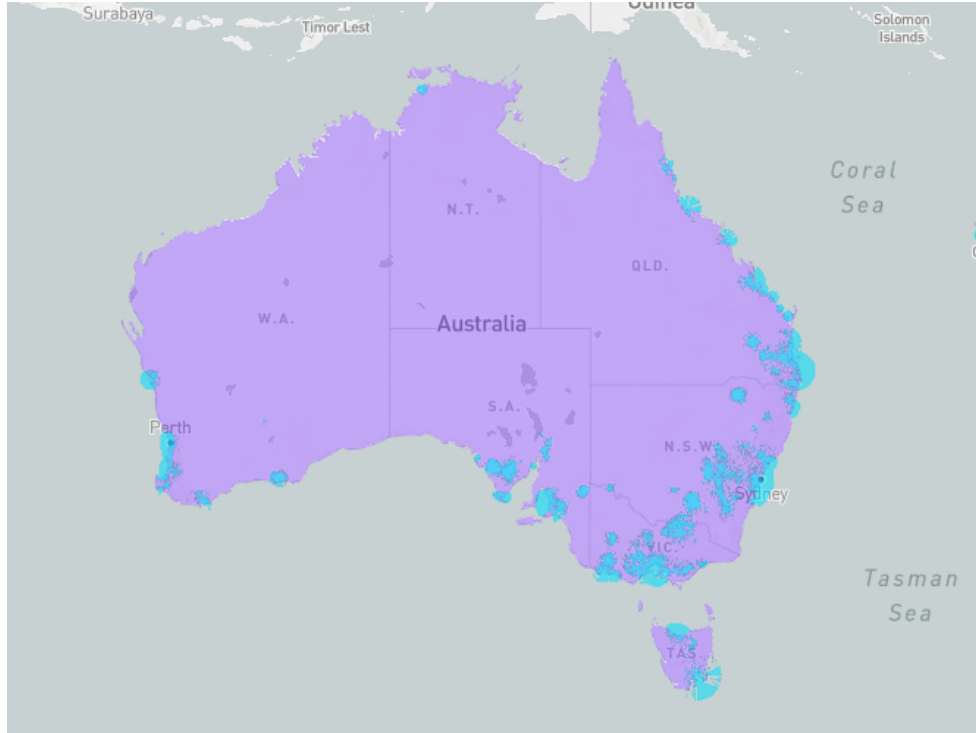


Figure 7: SigFox coverage in Australia

3.3.4 DASH7

Dash7 was the final alternative option considered. Advertised as a "instant on" wireless solution suitable for long range, low power communications (35). Dash7 operates as a full Open Systems Interconnection (OSI) stack protocol based off of the ISO 18000-7 standard for active RFID. With a functional range of up to 2 km, DASH7 uses frequency bands close to 433 MHz range (36). Dash7 also allows for functional network meshing between nodes. Haystack, the company behind the Dash7 alliance also suggests that Dash7 has a level of precision above other communication alternatives which allow it to support precise indoor location using RSSI-based localisation accurate to 1m (35). Due to the limited availability of Dash7 compatible MCU's and supported libraries, Dash7 has not been used in this project (8). Of all the alternatives, Dash7 showed to be the most promising and in future iterations of the projects, given more time to develop greater supports for Dash7, may be chosen over LoRaWAN. Section 6 further explores the possibilities of the potential use of Dash7.

4 Technology Review

As a review of the key mechanisms and project functionalists this chapter provides an overview of the critical infrastructure and underlying technologies used. Specifically covered are the individual modules used to generate project critical components in both the WSN and web application sections of the project. Namely these include:

- BLE client
- LoRaWAN client
- LoRaWAN gateway
- BLE4.0 client
- Duty Cyclor (Sleep functionality)
- Packet forwarder
- Database framework
- Web application framework

4.1 Firmware Review

The Firmware component of this project covers the WSN implementation, included are the modules required to facilitate communication over both LoRa/LoRaWAN and BLE4.0 as well as the internal battery monitoring and duty cycling. Because sensor nodes are based off a commercial product (Xiaomi MiFlora) they are not covered in this technology review more information regarding the specific implementation of the sensor nodes can be found in section 5.2.1.

Communication clients in the WSN are controlled by an ESP32 micro-controller (9), a lightweight development module by Espressif targeted for IOT projects. Included with each ESP32 is the FreeRTOS operating system as well as standard Bluetooth/WIFI capabilities. Specific features which led to its selection in this project include: low power features, support for a real-time operating system, and the ease of integration. Communication clients use the RFM95W LoRa modem (37) as a separate module to the micro-controller to enable LoRa type communication. The communication gateway was provided as a pre-existing component of the Farmbot project and uses an unspecified micro-controller and LoRa modem to feed uplink frames to a LoRa server running on the University of Queensland cloud infrastructure.

4.1.1 Cost

Large emphasis was placed on managing costs for the project. The effects of which guided the choice for the ESP32 as it provided one of most functional options at its given price point. As a comparison the Raspberry Pi Zero W is priced at \$44.30 (AUD) (38) as opposed to the esp32 at \$8.71 (AUD) (39). The difference in price is further realised when considering options to scale the network and increasing the number nodes; as options to bulk purchase ESP32 are more readily available and competitively priced (39).

Another consideration in regards to cost is the included functionality which comes included with the ESP32. Specifically the ESP32 module used in this project (ESP32/RFM95W) provides both Bluetooth and LoRa hardware modules as part of a single standalone solution. Whilst this package increases the individual cost of each unit (\$50.11 (AUD) (40)) the savings overall are validated when comparing the single cost for a standalone LoRa module (RFM95W \$28.60 (AUD) (41)). Table 6 provides an overview of the boards considered, the provided functionality and their relative price point. Note, that the prices included in this section were taken at the time of writing (May, 2019) and future price fluctuations may cause price changes which no longer reflect the values presented.

| Development Board | Included Functionality | Price (AUD) |
|---------------------------------|---|-------------|
| ESP32/RFM95W (40) | Tensilica Xtensa 32-bit LX6 microprocessor Memory: 448 KB WIFI: 802.11 Bluetooth: v4.2 LoRa Modem: RFM95W | \$50.11 |
| Raspberry Pi Zero W (38) | ARM11 micro controller Memory: 1 GB WIFI: 802.11n Bluetooth: v4.0 | \$44.30 |
| Arduino Uno (42) | ATmega328 micro controller Memory: 32KB | \$13.07 |
| B-L072Z-LRWAN1 LoRa/Sigfox (43) | STM32L072CZ micro controller Memory: 192 KB LoRa Modem: SX1276 | \$66.75 |

Table 6: Price overview of considered micro controllers

4.1.2 Power Consumption

Another feature of the ESP32 which made it an attractive option for this project is the included operating modes which scale power consumption. By dynamically disabling non-critical modules for different operating modes the ESP32 aims to optimise the available resources in the most power efficient way. Table 7 provides an overview of the available functionality in each of mode of operation.

| Power mode | Active | Modem-Sleep | Light-Sleep | Deep-Sleep | Hibernation |
|--------------------------------|--------|-------------|-------------|------------|-------------|
| CPU | On | On | Pause | Off | Off |
| Wi-Fi/BT baseband and radio | On | Off | Off | Off | Off |
| RTC memory and RTC peripherals | On | On | On | On | Off |
| ULP-coprocessor | On | On | On | On/Off | Off |

Table 7: ESP32 operating mode functionality (9)

Table 8 further provides the manufacture quote for the power consumption at each operating mode.

| Power mode | Description | Power Consumption |
|---------------------|---|--|
| Active (RF working) | WIFI Tx/Rx | 95mA-240mA |
| Modem-sleep | CPU on | Max speed 240MHz: 30mA-50mA Normal speed 80MHz: 20mA-25mA Slow speed 2MHz: 2mA-4mA |
| Light-sleep | - | 0.8mA |
| Deep-sleep | The ULP-coprocessor is powered on ULP sensor-monitored pattern RTC timer including RTC memory | 0.15mA 0.1mA 0.01mA |
| Hibernation | RTC timer only | 0.005mA |

Table 8: ESP32 power consumption at each operating mode (9)

By cycling into low power operating modes in idle scenarios, average energy consumption is greatly reduced as idle energy waste is minimised. As a trade off, the longer nodes spend in low power operating modes the less available they are to serve in network activities. The result is an increase in transmission latency and decrease in overall data throughput from the network.

To balance this the project prioritises critical activities (if data is available to transmit, transmit) before cycling into a low power modes and uses an established uplink schedule which favors majority deep-sleep (60min deep sleep and 5min active) to deliver non-time-urgent data. Uplink dependency is also limited to a single communication client and the communication gateway. With the communication gateway connected to mains power and always in an active state, there is no need to synchronise active states across multiple nodes, reducing unnecessary overhead. The two operating modes used throughout the project are Active (RF working) and Deep-sleep (RTC timer only). With an average duty cycle of 5 minutes active and 55 minutes in deep sleep the average assumed weekly power draw was estimated to be 3360mAh (from table 8). Given the operating voltage of the ESP32 to be between 2.3V-3.6V (9), the ESP32 (i.e. the communication client) can then be powered effectively from a light (4400mAh) mobile power bank (44) without the need to be constantly recharged.

4.1.3 BLE4.0 Functionality

The ESP32 supports both Classic Bluetooth and Bluetooth Low Energy (BLE) derivatives (5). The Bluetooth implementation on the ESP32 is separated in to two separate stacks; a controller stack, which handles the hardware interface and link management; and a host stack, used by higher level applications. Currently the ESP32 uses Bluedroid (an open source google project) as the host stack, and VHCI (software-implemented virtual HCI interface) as the interface for the controller stack (5).

The controller stack (figure 8) used on the ESP32 is an pre-integrated solution which gives access to functions to control specific BLE services such as: H4 protocol, HCI, Link Manager, Link Controller, Device Manager, and HW Interface (5). Functions are provided through an API library provided in the ESP32's development environment or via the `sdkconfig` command when compiling projects.

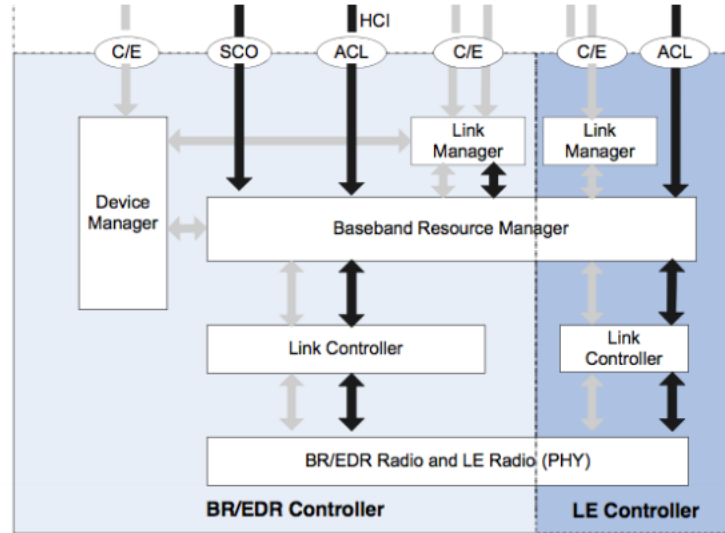


Figure 8: Architecture of the Classic BT and BLE controller (5)

The current host stack on the ESP32 is a modified derivative of the original Bluedroid implementation by Google (5). The host offers two main layers the BTU layer and the BTC layer. The BTU layer handles lower level protocol elements of the stack including processing L2CAP, GATT/ATT, SMP and GAP type requests. The BTC layer is an API tool used to manage GATT-based events and other higher level tasks.

Using the BTC layer an application can manage the GAP (the Generic Access Profile) settings of the ESP32; controlling critical functionality related to device discovery, device management and the establishment of device connection between BLE devices (5). The ESP32 includes four predefined GAP roles:

- **Broadcaster:** A broadcaster uses advertising packets to announce its presence to other devices in range. Though the device is discoverable, broadcasters cannot be connected to.
- **Observer:** An observer scans for advertising packets and is capable of reporting broadcasters seen whilst scanning. However in the same way as broadcasters, observers may only scan for advertising packets and cannot be connected.
- **Peripheral:** Peripheral device use connectable advertising packets which not only announce its presence to nearby devices but allow capable devices to connect, becoming a slave device once connected.
- **Central:** A central device is a device which receives and initiates connections using connectable advertising packets. Once connected central devices assume the role of master during the connection.

GAP profiles determine what states (figure 9) are reachable by the ESP32. As an example, a Broadcaster may be advertising or on standby but can never be connected.

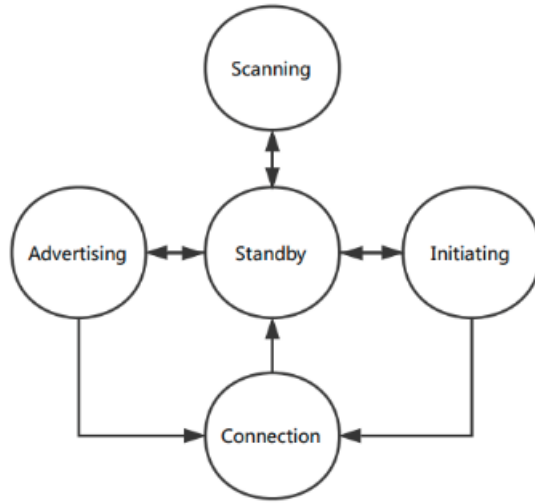


Figure 9: BLE states on the ESP32 (5)

When accessing data over a BLE connection, data is stored and accessed following the GATT architecture (27) which includes four basic elements:

- **Attribute Handle:** Used as an identifier to locate specific data. Attribute handles can be thought of as an address to locate data in the memory.
- **Attribute Type:** The attribute type is the information exposed which describes the data behind the current attribute handle being queried.
- **Attribute value:** The attribute value is the actual data being stored behind each attribute handle.
- **Attribute Permission:** The attribute permissions sets the current functionality of each attribute handle. They typically determine whether the current attribute handle can be read or written to.

Using the GATT architecture devices are further broken down into client/server roles, where servers hold data which can be queried and written using clients (27). A typical interaction between a GATT server and client is presented in figure 10.

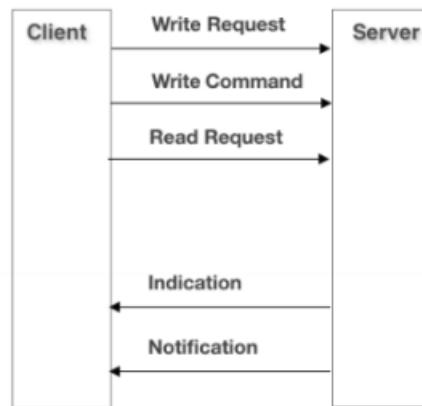


Figure 10: Example GATT server/client interaction (5)

The GATT architecture can be used to further describe attribute handles in terms of characteristics and services. Characteristics are used as specific values which describe the type of information being stored over multiple attribute handles (27). Characteristics usually include a declaration which inform a remote device that this point marks the beginning of a new characteristic; the characteristic value which represents the actual information being stored; and additional descriptors which describe the information being stored (i.e. providing configuration information). Services are used to group similar characteristics (27). Remote devices can select specific services based on the requirements of the particular device and how they align with the specific service. An example layout of a GATT server using characteristics and services is presented in table 9.

| Attribute Handle | Attribute Type |
|------------------|------------------------------|
| 0x0001 | Service 1 |
| 0x0002 | Characteristic Declaration 1 |
| 0x0003 | Characteristic Value 1 |
| 0x0004 | Descriptor 1 |
| 0x0005 | Characteristic Declaration 2 |
| 0x0006 | Characteristic Value 2 |
| 0x0007 | Descriptor 1 |
| 0x0008 | Descriptor 2 |
| 0x0009 | Service 2 |
| ... | ... |

Table 9: Example GATT server (5)

The physical layer of the BLE module operates over a 2.4GHz spectrum in a range between 2402MHz and 2480MHz and includes 40 1MHz wide channels (6). Channels are ordered between 0 and 39 with 2MHz spacings separating each channel. In a typical BLE implementation channels 37, 38 and 39 are reserved for advertising packets with the remaining channels to be used for general data exchange during a connection. BLE uses a form of channel hopping to seek available channels for transmission with independent queues on each channel to manage received packets (6). Figure 11 shows the channels used by BLE and their position between 2402MHz and 2480MHz limits.

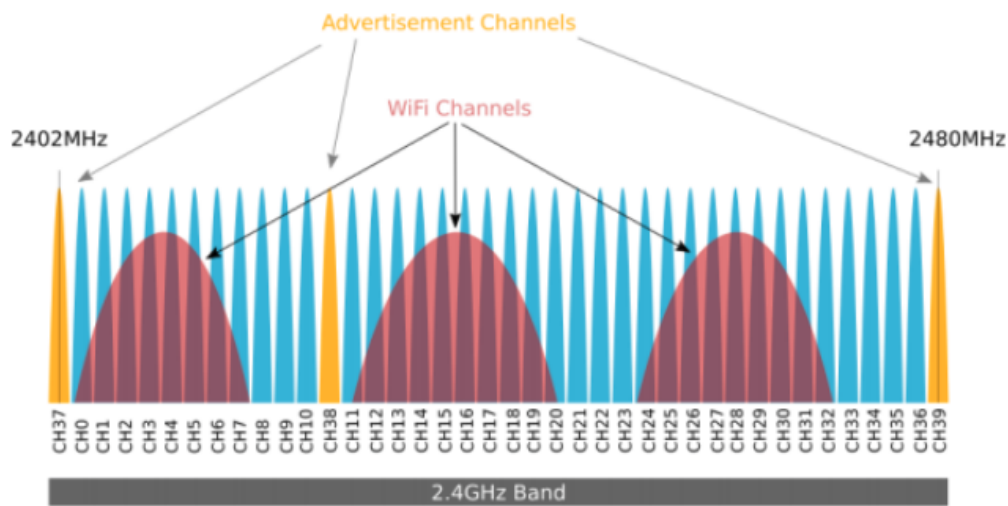


Figure 11: Frequency channels used by BLE (6)

4.2 FreeRTOS

Coding for the ESP32 was performed in C and uses the ESP-IDF toolchain and build system to compile and load code onto the ESP32. The ESP-IDF includes with it an implementation of the FreeRTOS kernel (an Amazon Web Services managed product). FreeRTOS provides options for: task management, mutual exclusion, inter-task communication and synchronization, memory management, real-time events, and control of input and output devices (45). FreeRTOS is implemented on the ESP32 through included libraries which give access to types and functions which are compiled alongside the application. While running an application using FreeRTOS, code is structured in two parts, the initialising code and task code (46). Code run on FreeRTOS cycles through a “boot phase” which constructs/registers tasks with the scheduler followed by an application execution phase which manages task scheduling, resourcing and operations.

4.2.1 Task Management

In FreeRTOS tasks form the primary computation block which facilitates application functionality (45). Tasks operate in one of a set of predefined states including: running, ready, suspended or blocked. The state of the task is determined by the scheduler which allocates CPU resources to tasks based on the tasks priority. The task priority is an integer value which describes the tasks position to be executed relative to other available tasks. Included with each task is an associated “execution context” which stores the call stack and register values when the task is not executing (45).

The scheduler operates as its own high priority task which directly enables the execution of another task in the “ready” state. As the “running” task, the task is granted CPU resources which enables it to perform application specific tasks. In the event that two or more tasks have the highest priority CPU resources are distributed equally over both tasks in a round-robin type scheduling. In managing tasks, high priority tasks will continue to consume CPU resources unless otherwise specified to relinquish the held resources. As a default option higher priority tasks also have the ability to preempt lower priority tasks through the scheduler. Also included is the addition of an idle task set by the scheduler which acts as a buffer to ensure that some task is always running, as well as executing administration tasks such as memory management.

4.2.2 Communication and Synchronisation

To coordinate communication across different tasks, FreeRTOS implements a shared message queue system which acts as a message-passing service between tasks (45). With this system tasks may actively post and receive messages from a queue. The use of the queue is however a “blocking” action, in the event that a task attempts to either read from an empty queue or write to full queue.

Single length queues are referred to as a semaphore and can be used as a synchronisation tool between different tasks. When the semaphore is present the queue is full and can be read/taken by a another task. In the event that the queue is empty an attempt to read/take semaphore results in a block action until the queue is filled (semaphore is given).

4.3 ESP32 LMIC

The ESP32 LMIC (47) is an external library module used in conjunction with the ESP-IDF to support LoRaWAN functionality on ESP32 alongside the RFM95W LoRa modem. Originally a port of the original IBM LMIC (48) for arduino, the ESP32 LMIC library provides the same functionality for the ESP-IDF with support for FreeRTOS. The library includes support for: transmitting uplink frames, scheduling downlink frames, encryption and message integrity, and over-the-air activation (OTAA).

4.3.1 LoRa Protocol Stack

LoRa (“Long Range”) is a long range wireless communication system targeted for low power systems. Projects using LoRa typically include two distinct layers: a physical layer which implements a radio modulation technique called Chirp Spread Spectrum (CSS), and a MAC layer protocol (LoRaWAN) which handles frame management (49).

The physical layer is a proprietary product developed by Semtech (29) which uses frequency chirps spread over a linear variation to encode information (CSS). The included benefits of this type of modulation, include an elimination of frequency offsets effects between the receiver and the transmitter which are handled as timing offsets. In the same way, impacts from the doppler effect are also removed as timing offsets. Both effects are mitigated due to the way in which chirps are linearly sent from the transmitter. The hand-able frequency offset can stretch to 20% of the original bandwidth (sensitivity of 130 dBm) without impacting decoding performance (7). LoRa communication can therefore operate on a looser timing requirement allowing LoRa to operate at lower frequencies with lower transmitting power.

LoRa communication can be configured in a variety of different ways including: Bandwidth (BW), Spreading Factor (SF) and Code Rate (CR). The BW, SF and CR determine the the effective bit rate of the modulation, its resistance to interference and its ease of decoding (7). The BW is the most critical factor to the LoRa implementation as it determines the available range at which LoRa chirps can be constructed. In practice, a LoRa transmission begins with a series of upward chirps which continues until the maximum frequency of the band is reached. At this point the frequency wraps and completes the process again from the lowest frequency. The position of discontinuities in this pattern is what represents the data during transmission. Figure 12 is a representation of what a possible LoRa transmission would appear as frequency variations over time.

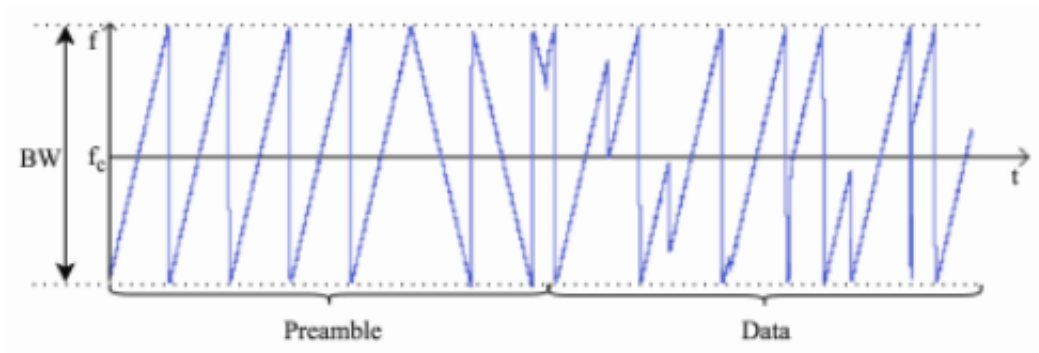


Figure 12: Example LoRa transmission as frequency variations over time (7)

The SF is a reference to the number of chirps per symbol as a base 2 logarithm. Variations to the spreading factor are inversely reflected in the CR by a factor of two, as 2^{SF} chirps cover the whole bandwidth. The CR is an indicator of how frequently chirps are sent from a transmitter and is directly dependent on the BW. This relationship is presented in equation 1, which highlights the effect of BW and SF on the duration of a symbol (T_s).

$$T_s = \frac{2^{SF}}{BW} \quad (1)$$

LoRa also includes a forward error correction code. Given a code rate (CR) equal to $4/(4 + n)$, with $n \in \{1, 2, 3, 4\}$ and assuming SF bits of information are transmitted per symbol. Equation 1 can be expanded to give the useful bit rate (equation 2).

$$R_B = SF \times \frac{BW}{2^{SF}} \times CR \quad (2)$$

4.3.2 LoRa Frame Format

Though not strictly enforced the LoRa stack includes a physical frame format that can be implemented when sending frames between transmitters and receivers (7). Frames assume a constant bandwidth and spreading factor and includes a preamble followed by frame header (optional), frame payload and an optional payload CRC. Figure 13 outlines the typical LoRa frame structure.

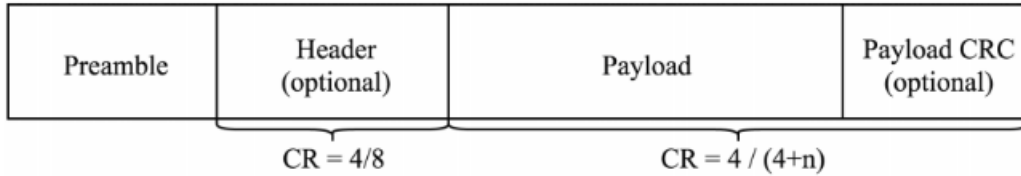


Figure 13: Example LoRa frame (7)

Each LoRa frame is required to include a preamble containing a series of upchirps used to define the timing requirement (transmission frequency) with the last two upchirps used to encode the sync word (a byte value used to differentiate LoRa networks on the same frequency) (7). The preamble is concluded by a two and a quarter “downchirps” which cover 2.25 symbols. An optional header can also be included after the preamble. The role of the header is to outline the payload length, coding rate and CRC of the frame (7).

A LoRa frame payload is an amalgamation of the required symbols needed to represent the information to be transmitted. The required number of symbols required to send a payload is given by the length of the payload (L_P), the length of the CRC (L_{CRC}), the header length (L_H) and the data rate (D_R). Equation 3 is used to determine the required number of symbols to transmit a single frame.

$$n_S = 8 + \max \left(\frac{8L_P - 4SF + L_{CRC} + L_H}{4 \times (SF - D_R)} \times \frac{4}{CR}, 0 \right) \quad (3)$$

4.3.3 LoRa Network Architecture

The typical LoRa network is star-of-stars topology, which uses an assortment of different devices at each level of the network. Figure 14 provides an overview for a common LoRa network deployment.

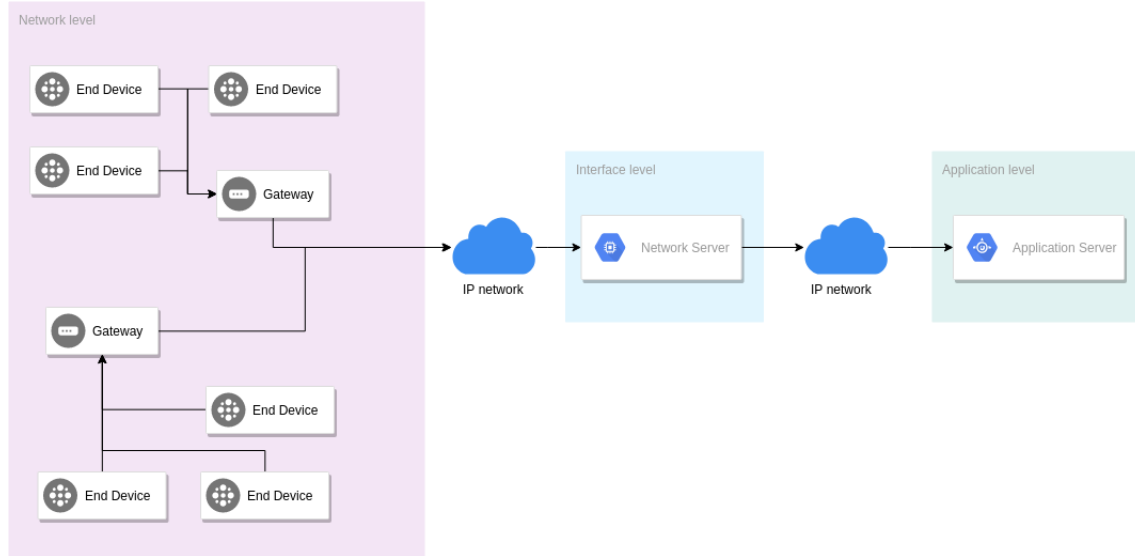


Figure 14: Example LoRa network architecture

The four major devices included in the network topology are (50):

- **End devices:** End devices typically include sensory nodes which collect and deliver data to gateways.
- **Gateways:** Gateways are relay devices which transform and forward received LoRa messages to a network server.
- **Network Server:** Network servers act as an intermediary layer between application servers and physical network devices.
- **Application Server:** Application servers present received data in a console or can be used to forward received data via http.

4.3.4 LoRaWAN Protocol

LoRaWAN is a MAC level protocol which interfaces with the physical layer of LoRa devices (49). It is targeted at WSNs where the expected data rate is relatively low and power consumption is a primary constraint. Using LoRaWAN, LoRa end devices can use any LoRa gateway to access the greater LoRa network. LoRa gateways connect end devices to a network server which assumes responsibility over handling network traffic.

As the principle network device, end devices can assume one of three different classes which assume different levels of functionality (8). These classes include:

- **Class A:** Devices can schedule uplink messages as well as receive downlink message between two receive windows which occur after the initial uplink message has been sent. As a requirement uplink messages also include a small self-introduced jitter which randomly staggers the moment when uplink messages are sent. class A devices offer the most balanced option for both power consumption and flexibility on downlink messages.
- **Class B:** Class B devices are the same as class A devices with the addition of being able to open extra receive windows by signaling the network receiver that the end device is ready to receive a response.
- **Class C:** Class C devices assume maximal receive slots, where they have an almost continuous receive window (receive windows are close when transmitting). As a consequence class C devices carry the highest power consumption amongst the three different classes.

Because end devices are linked to the network server they cannot explicitly communicate with one another without first passing through the network server (8). As LoRa communication uses ISM frequency bands, LoRa is limited by the respective regulations which control the maximum transmission power and the duty cycle. These controls translate over into LoRaWAN as an introduced delay between scheduled frames from the end device.

LoRaWAN expands the requirements of raw LoRa frames by having uplink frames include the optional header and CRC components. Downlink frames are also required to include the optional header however the CRC component still remains optional. An example LoRaWAN frame is presented in Figure 15. LoRaWAN

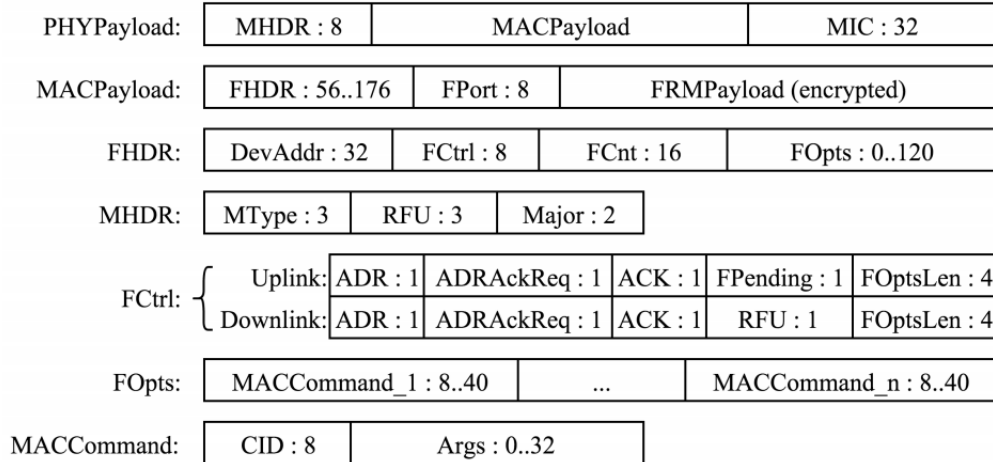


Figure 15: Example LoRaWAN frame (7)

further includes a procedural “device activation” which is used add/allow end devices to participate in the network. Devices in LoRaWAN can be activated in one of two ways: Over-The-Air Activation (OTAA) and Activation By Personalisation (ABP) (7). As part of either process devices are assumed to possess a copy of the following:

- End-device address (DevAddr)
- Application identifier (AppEUI)
- Network session key (NwkSKey)
- Application session key (AppSKey)

Applications using OTAA use join-request and a join-accept message between the end device and network server at the start of each new session to register the device. A positive response from the join-accept message will include new session keys (NwkSKey and AppSKey) to be consumed by the end device. Applications using ABP are assumed to already possess the two session keys which they can then use to authorise future uplink/downlink messages.

4.4 LoRa Server

The LoRa server is a combined module which includes both the network and application server components of the LoRa network. The current LoRa server is a derivative of the original project by lorasever.io (51) and is an artifact of previous projects from the FarmBot project (23) at the University of Queensland. Internally the LoRa server includes:

- **MQTT broker:** MQTT is used as part of the packet-forwarder UDP protocol implemented by Semtech. The MQTT broker provides a method to communicate information to and from the LoRa gateway as JSON with higher level services. Each LoRa gateway is permitted to subscribe to at least one MQTT topic which represents an end device. MQTT topics may be used for end device routing, allowing gateways to push and pull data from the topic for each device. Because MQTT can be run over TLS, the transport between gateway and network is secure.
- **Redis:** Redis is used as an in-memory database which can be used as a temporary cache and message broker.
- **PostgreSQL:** PostgreSQL acts as the long-term storage solution, which allows for both query and sort functionality.

The LoRa server also aims to deliver data to external applications as a push API or webhook. The webhook will deliver data to applications in real-time; in the same way that the MQTT data arrives to the LoRa server. Using an existing method on the LoRa server to deploy http integrations, a supplied http endpoint can be generated on the application side to consume that data received by the network server.

4.5 Web Application

The web application portion of the project is built as a Progressive Web App (PWA), which exists as a standalone module which interacts with the WSN through an API layer hosted by the LoRa server. The core function of the web application is to provide a way to graphically represent the data collected as well as improve the quality of the data to something which can be acted on by a future autonomous system.

4.5.1 MEAN

The project dashboard is the central platform which internally hosts all subsequent models of the web application. The dashboard includes both a server side and client side component which are structured using a MEAN stack. The specific features of the MEAN stack include:

- **MongoDB:** Database solution (52).
- **Express.JS:** Server side framework (53).
- **Angular:** Client side framework (54).
- **Node.JS:** Development language (55).

Ultimately the MEAN stack provides a relatively simple roadmap for deploying cloud native fullstack javascript applications.

4.5.2 Service Workers

In facilitating an environment where internet connectivity may not always be present, this project employs the use of service workers which aims to provide a means of continuous serviceability even when an internet connection is not present. The role of service workers is to manage content delivery to users by using cached versions of previously seen content in place of calling a back-end server (56). The service worker exists as an attached application on the browser which is automatically installed when the dashboard is first visited.

The service worker can also be used to store data retrieved from API calls in the browser using the IndexedDB module. Beyond offline functionality the service worker can also be used to support site reliability in instances where internet connectivity is slow and/or unreliable. In situations where there are intermittent drops in connectivity the service worker maybe used to initially serve a previous cached version of the site which may be updated later when new content becomes available.

4.5.3 Database Solution

In place of traditional relational databases this project has made the decision to pursue the use of MongoDB (52), a object-oriented database. Being object-oriented, data is represented as JSON objects with specific fields which can be made to either follow a strict schema or may be schemer-less. However, when stored JSON documents are serialised as BSON (Binary JSON) which allows non-JSON data types such as arrays to also be included. Additionally, JSON objects may also further be grouped into separate “collections” which form the table equivalent found in traditional relational databases like MySQL.

The benefits of using MongoDB as opposed to traditional relational databases include better performance and scalability for simple data structures (52). Performance is improved as a result of managing “workable” data in RAM (52). Generally, all data is held on the hard disk however during a query data can be called from RAM. Calling data from RAM requires the stored data to indexed and the resulting indexes to be stored in RAM. Improved scalability comes from the use of shards and database sharding which allows for horizontal scaling (52). Data can also be replicated asynchronously through replica sets, which improves general reliability as the data is available across more nodes.

Potential shortcomings include the lack of referential integrity (RI) as a mechanism to enforce relations between different pieces of data across the database. Moreover the inability to perform grouping/joins in

the database also limit the ability of MongoDB to serve more complex data sets.

In managing potential shortcomings, the project considered the use of an ORM (Object Relational Mapping) library (Mongoose (57)) to manage tasks such as automatic validation when writing data to the database. Mongoose enforces an established schema for data in the database making queries more reliable. In considering the simple data structure used in project as well as the minimal need for referential integrity and grouping/join operations, MongoDB provides a flexible solution which integrates well with the current web application stack through the use of JSON objects and the majority Javascript implementation. Moreover, MongoDB's ability to be horizontally scalable makes it a suitable choice for a project requiring continuous data integration.

4.5.4 Modelling and Classification using TensorFlow

As part of the included functionality to improve the quality of the data received by the web application, a modelling and classification tool has been added which uses TensorFlow (58) CNNs (Convolutional Neural Networks) to sort incoming data into categories of: poor, fruitless and fruit bearing. Typical CNNs are constructed using layers of filters aim to extract and learn higher-level features which are then used for classification. The typical layers of a CNN include (58):

- **Convolutional Layers:** Convolutional layers apply convolutional filters to obtain a single value output to be included as part of the overall feature map.
- **Pooling Layers:** Pooling layers work to down-sample the current dimensionality of the overall feature map as a way to decrease processing time.
- **Dense (fully connected) layers:** Dense layers classify the features extracted from the convolutional layers.

The current model used in project is provided data sorted as: temperature, lux, conductivity, and moisture. Without the need to down-sample or convolve data, the current CNN doesn't include convolution or pooling layers. The CNN simply uses two dense layers which act as the classifier. The first fully connected layer applies a ReLU activation function which delinearises the model so that the model does not over-fit to training data. With the second layer applying a soft-max activation function which associates a generated value behind each node in the layer (one for each potential class). The value associated represents the models certainty that the given data entry can be classified under a specific class, with the highest value being the class that the model believes that given data entry belongs to.

5 Current Implementation

Final project deliverables include both WSN and web applications components. As outlined in the solution overview, the WSN (figure 16) consists of two communication clients (although only one is required) which include both LoRa and BLE4.0 modems and can be paired between the current four sensor nodes. Communication clients are currently configured to run with the LoRa Gateway and Server currently running under the Farmbot project. Current notable capabilities of the WSN include:

- A coverable range greater than 80m (measured range 1.2km).
- Operation within a complex environment (functional between the farm site - roof of building 50 UQ and development lab - lab 401 building 49 UQ).
- WSN budget held under \$100 (AUD) limit (current cost \$84.30 (AUD)).
- Network components can sufficiently be powered from a 3V supply as either a coin cell battery or mobile power bank.

Development of the web application has also been able to achieve the intended functionality put forward in the initial solution overview. Notable features include:

- A user interface which delivers data represented as a graphical model.
- A classification tool which can be used to tag and label data based on previous observations.
- A scalable storage solution which can store and allows for suitable querying of the data received by the WSN.
- An API interface which allows the data recorded by the web application to be integrated with other projects.

The current web application exists as a packaged project currently hosted on a development machine, with external interactions made possible through a third party software (ngrok (59)) which enables reachable endpoints to services outside of the local machine (i.e. LoRa server).



(a)



(b)

Figure 16: (a) sample set of communication client and sensor nodes, (b) current deployment setting

5.1 Deployment Process

The following section is a review of the deployment process for the WSN and web application components of the project and includes the specific steps taken to activate individual network elements and integration guidelines.

5.1.1 WSN

The existing WSN can be deployed by distributing communication clients modules in range of the LoRa gateway and then introducing sensor nodes within range of the deployed LoRa clients. Communication clients are pre-configured to recognise and connect with MiFlora sensors over Bluetooth and will automatically begin forwarding data after a brief setup time.

New communication clients can be built by loading the communication client project onto available ESP32/RFM95W chips via the ESP-IDF build system. New clients must be configured with the correct Device EUI and Application EUI/Keys when first compiled. These EUI/keys are required to perform OTAA with the LoRa server. All required EUIs and keys can be generated from the LoRa server by creating a new device under an existing application. EUI/Keys only needed to be compiled with the device once, as there are options to have the device store the provided EUIs and keys in non-volatile memory after the first initial run.

When building the LoRa client project for the ESP32, the ESP-IDF will offer the option to build for specific frequency regions and spreading factors. In the current implementation the WSN uses the frequency range for AS923 (Asia-pacific) which includes the following frequency channels:

Uplink (MHz):

1. **923.2** - SF7BW125 to SF12BW125
2. **922.2** - SF7BW125 to SF12BW125
3. **922.4** - SF7BW125 to SF12BW125
4. **922.6** - SF7BW125 to SF12BW125
5. **922.8** - SF7BW125 to SF12BW125
6. **923.0** - SF7BW125 to SF12BW125
7. **922.0** - SF7BW125 to SF12BW125
8. **922.1** - SF7BW250
9. **921.8** - FSK

Downlink (MHz):

1. Uplink channels 1-10 (RX1)
2. **923.2** - SF10BW125 (RX2)

The common spreading factor used amongst the sample communication clients is 10, as a balanced option for the final symbol length (equation 1) and bit rate (equation 2). However any spreading factor can be used. Once the LoRa client is successfully activated via the LoRa server, the LoRa server can be used to track recent activity and monitor incoming frames from each attached LoRa client on the network.

5.1.2 Web application

Both the web application and the WSN exist as stand alone modules. Due to this modularity the web application can be deployed to any external service (i.e. AWS, UQ cloud, etc.), only requiring a public endpoint so that the web application is reachable from the LoRa server.

The web application assumes a deployment environment which includes a Linux based operating system as well as the following dependencies:

- **NPM (v6.4.1):** Project packet manager.
- **Node.JS (v10.15.3):** Server side implementation language.
- **Angular CLI (v7.3.8):** Client side framework.
- **MongoDB (v4.0.8):** Database system.
- **Ngrok (Optional):** Forwarding service to expose local projects.

Assuming all the required dependencies exist on the deployment environment the project can then be unpacked and run using NPM. Once running the the web application can then be linked to the WSN by supplying a public endpoint to the LoRa server as a HTTP integration. HTTP integrations may be used to post live events occurring throughout WSN as seen by the LoRa server. The currently available HTTP integrations include:

- **Uplink data:** The event which is fired upon the LoRa server receiving a new uplink frame from the WSN.
- **Join notification:** Upon establishing a new link with a LoRa client, the server will fire a join notification event signalling that the server has recognised the device and is ready to receive uplink frames.
- **ACK notification:** Signals that the server has received an ACK frame from one of its clients.
- **Error notification:** An error has occurred on the LoRa server and is marked with this event being fired.

If the project is running on a local machine, Ngrok maybe used as a separate service to generate a public endpoint from which the web application can be accessed from. Ngrok works by linking an existing process running on the local machine to a public address. As an example, if the web application was running on a local machine from `http://localhost:3000` it is possible through Ngrok to access this endpoint through a linked public url: `http://518d6d61.ngrok.io`.

5.2 Component Breakdown

The following section is an in depth look at each individual component of the current implementation and its contributions to the project as a whole.

5.2.1 Sensor node

Currently the role of the sensor node is filled by the Xiaomi MiFlora (25). The device which is commercially sold as a small scale plant monitoring tool includes an array of environmental sensors which are aimed to track the current conditions of the plant it is assigned to. The current outputs of the MiFlora units are temperature readings derived from an thermoresistor located on the body of the unit; lux values which are produced using a photodiode located at the high end of the unit and measures for both electrical conductance/soil moisture which are the product of two probes found at the base end of the unit (60). The MiFlora units also have an internal mechanism which records the current capacity of the battery powering device as well as the devices unique identifier (UUID) (60). Further selling points for Xiaomi MiFlora unit include the wireless capabilities

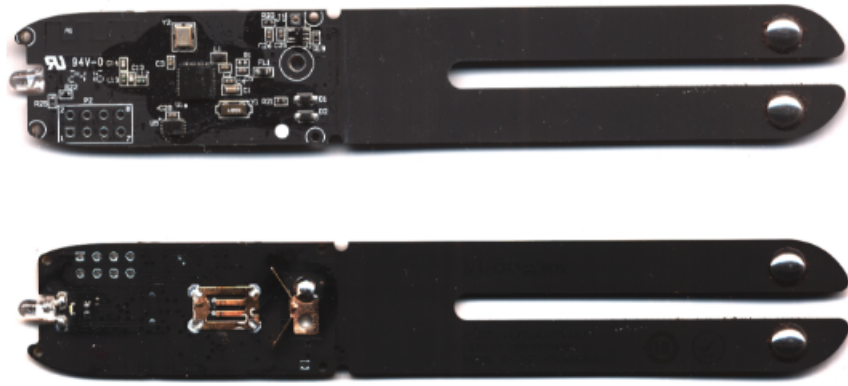


Figure 17: Stripped Xiaomi MiFlora

of the unit, capable of delivering data over BLE (v4.0); units are powered using a 3V coin battery, with an advertised life span of 1-2 years (25) and the overall cost of the sensors which when sourced from commercial sites such as Bangood, usually offered the most cost effective (\$24.70 (AUD) (61)) solution. Ultimately the usability and affordability of the MiFlora are what contributed the most to this sensors inclusion in project.

5.2.2 Communication client

The communication client component operates two tasks using FreeRTOS from the ESP32 which control both the BLE4.0 and LoRaWAN functionality. BLE4.0 is handled directly on the ESP32 and uses the onboard Bluetooth module to serve communication between the communication client and the sensor nodes. Alternatively LoRa is handled externally via the RFM95W using the ESP32 LMIC library to facilitate communication between the communication client and the LoRa server.

The communication client will actively seek to individually visit and poll data from available sensor nodes via Bluetooth. Once the data from a sensor node has been successfully polled the communication client will forward the data to the gateway via LoRa where it will then be handled by the LoRa server. Because both the BLE4.0 and LoRa functionality run on separate tasks, activities related to polling data from sensor nodes does not impact the LoRa task. Only when the LoRa task is notified of a new ready to send packet via an inter-task queue (a feature of FreeRTOS) does the LoRa task take notice of BLE4.0. In the event that the communication client has exhausted all available sensor nodes (all nodes visited at least once) the communication client will transition into a low power mode where it will stay for a configurable period of time before transitioning out and repeating the process. The general behavioural flow for the communication client is captured in figure 18.

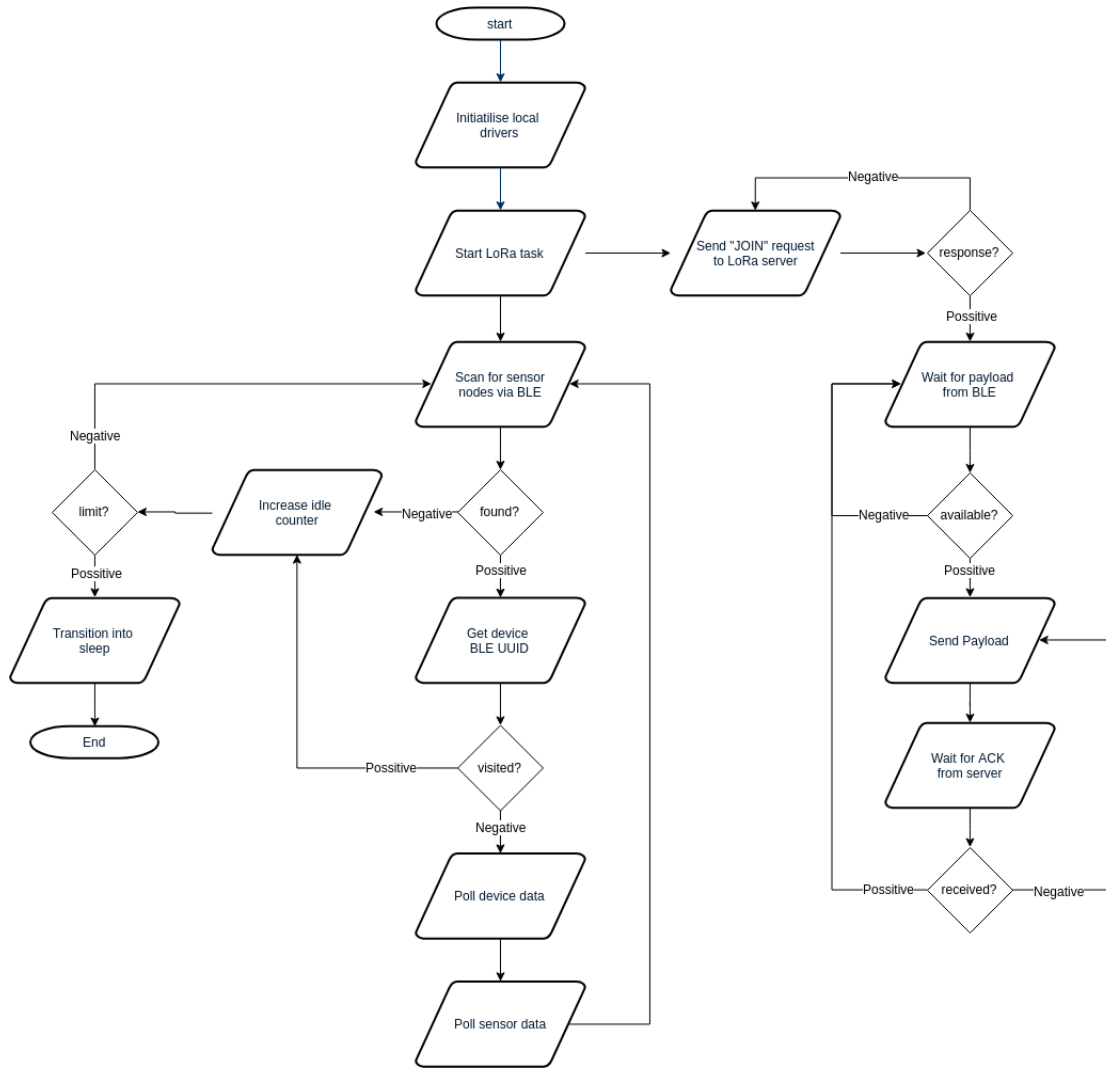
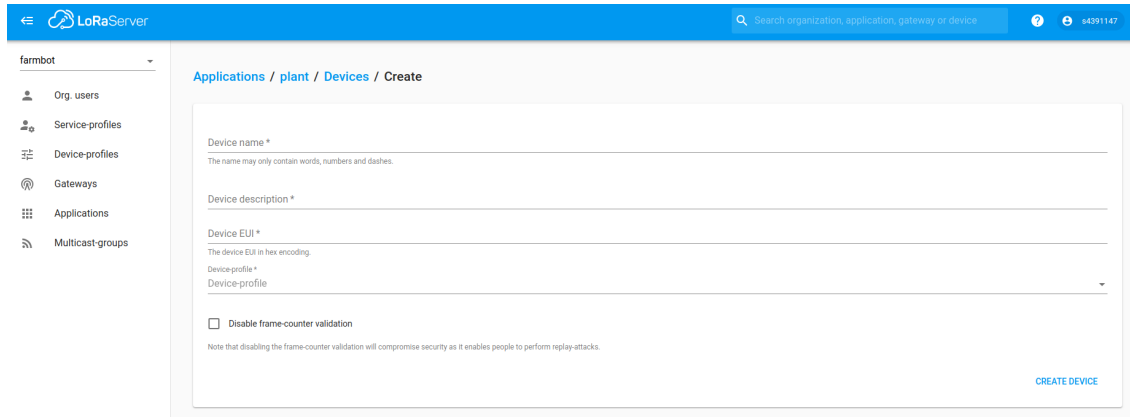


Figure 18: Algorithm of the communication client

5.2.3 LoRa Server

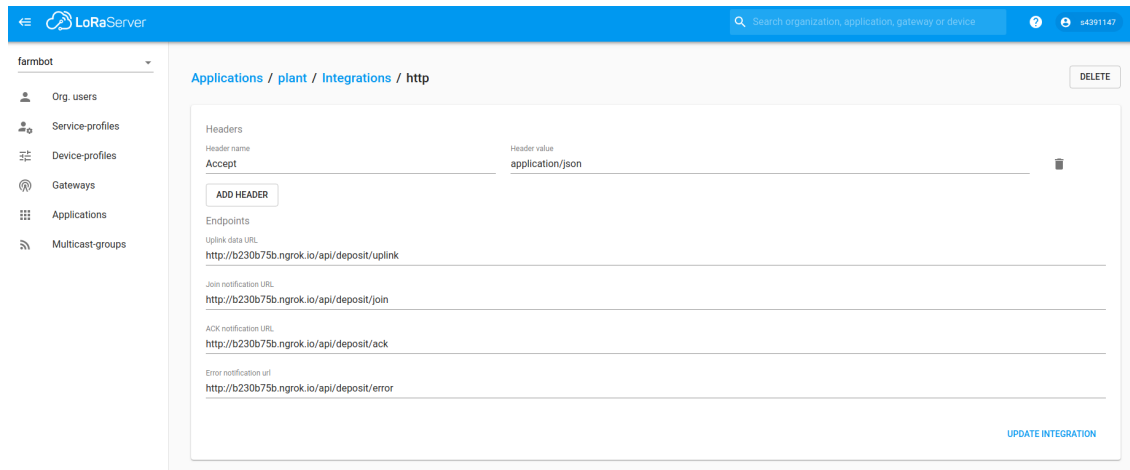
The LoRa server operates as a remote service which is used to receive and forward received LoRa frames via HTTP. Notable actions on the LoRa server include the registration/activation of new LoRa clients figure 19 as well as the construction of HTTP integrations figure 20. Whilst other actions exist on the LoRa server, within the scope of this project they do not directly add any further utility to the project beyond the configuration which currently exists. The LoRa server may also be used as a tool to troubleshoot individual LoRa clients as it actively records the timestamp of when each client was last seen (last received a frame).



The screenshot shows the 'Create Device' form in the LoRa Server interface. The breadcrumb navigation is 'Applications / plant / Devices / Create'. The form includes the following fields and options:

- Device name ***: A text input field with a note: 'The name may only contain words, numbers and dashes.'
- Device description ***: A text input field.
- Device EUI ***: A text input field with a note: 'The device EUI in hex encoding.'
- Device-profile ***: A dropdown menu with 'Device-profile' selected.
- Disable frame-counter validation**: A checkbox that is currently unchecked. A note below it states: 'Note that disabling the frame-counter validation will compromise security as it enables people to perform replay-attacks.'
- CREATE DEVICE**: A blue button at the bottom right.

Figure 19: Snapshot of the device registration function on the LoRa server



The screenshot shows the 'HTTP Integration' form in the LoRa Server interface. The breadcrumb navigation is 'Applications / plant / Integrations / http'. There is a 'DELETE' button in the top right corner. The form includes the following sections and fields:

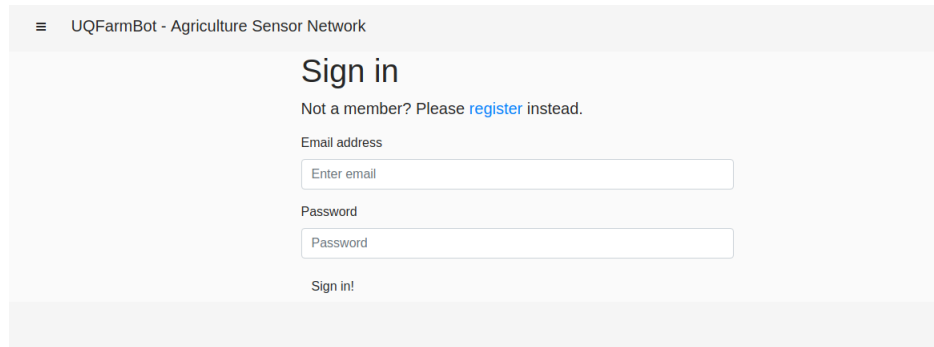
- Headers**: A table with two columns: 'Header name' and 'Header value'. It contains one entry: 'Accept' with the value 'application/json'. There is an 'ADD HEADER' button below the table.
- Endpoints**: Four text input fields for different endpoints:
 - Uplink data URL**: `http://b230b75b.ngrok.io/api/deposit/uplink`
 - Join notification URL**: `http://b230b75b.ngrok.io/api/deposit/join`
 - ACK notification URL**: `http://b230b75b.ngrok.io/api/deposit/ack`
 - Error notification url**: `http://b230b75b.ngrok.io/api/deposit/error`
- UPDATE INTEGRATION**: A blue button at the bottom right.

Figure 20: Snapshot of the HTTP integration function on the LoRa server

5.2.4 Web Application

The web application is responsible for data presentation and managing collected data. Users can access data via a dashboard which includes a basic account system, allowing users to register/login (figure 21). Once in their account, users can register specific sensor nodes on the WSN to the account and the web application will forward the received data to the account.

Accounts are constructed with the aid of an external node.JS library, Passport.JS. Details of the account are stored in the web application database (MongoDB) and include the users' email, name, the hash and salt generated by passport.JS and the sensor nodes registered to the account via the dashboard. Sensor nodes are registered to an account by specifying the unique BDA belonging to the device. Due to the assumption that only the owner of the sensor node will know the BDA of the device, sensor nodes are secured via a "security through obscurity" approach, which authenticates ownership of sensor nodes. Each account has



UQFarmBot - Agriculture Sensor Network

Sign in

Not a member? Please [register](#) instead.

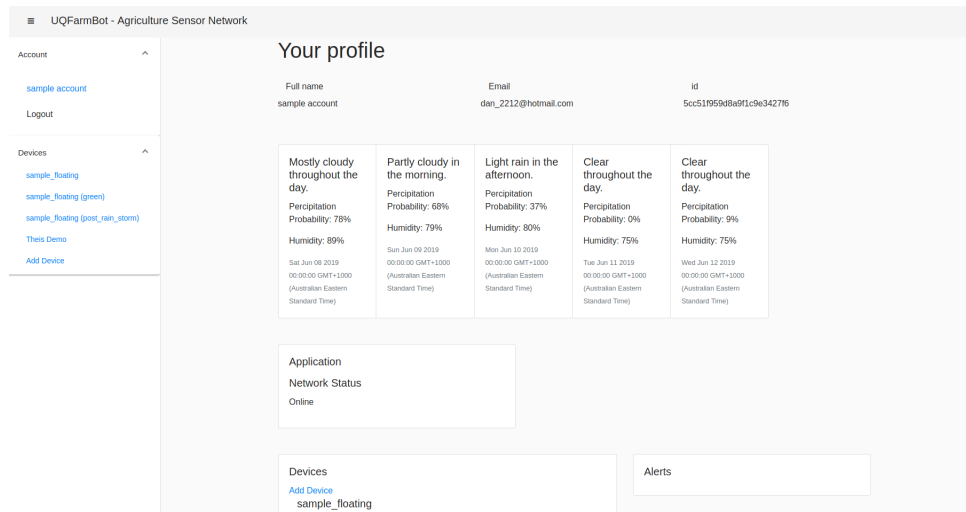
Email address

Password

Sign in!

Figure 21: Snapshot of the login page from the web application

access to the current weather forecast which is updated hourly via an internal service run from the back-end server. The server polls a provided endpoint from a public weather API (DarkSky.net), which returns the next five day forecast including forecast, precipitation probability and humidity. The current limit on the API restricts use of the current endpoint to 1000 calls per month, with the current polling method the web application averages 750 calls monthly varying on how often the back-end server is restarted. The forecast held on the server is the value which is distributed to clients as they make requests for the current weather forecast. Originally intended as a component for the classification tool, the use of the weather forecast API has since been degraded to a separate feature which exists purely as additional information (figure 22).



UQFarmBot - Agriculture Sensor Network

Your profile

Full name: sample account | Email: dar_2212@hotmail.com | ID: 5cc51959d8a9f1c9e3427f6

| Mostly cloudy throughout the day. | Partly cloudy in the morning. | Light rain in the afternoon. | Clear throughout the day. | Clear throughout the day. |
|--|--|--|--|--|
| Precipitation Probability: 78% | Precipitation Probability: 68% | Precipitation Probability: 37% | Precipitation Probability: 0% | Precipitation Probability: 9% |
| Humidity: 89% | Humidity: 79% | Humidity: 80% | Humidity: 75% | Humidity: 75% |
| Sat Jun 09 2019 00:00:00 GMT+1000 (Australian Eastern Standard Time) | Sun Jun 09 2019 00:00:00 GMT+1000 (Australian Eastern Standard Time) | Mon Jun 10 2019 00:00:00 GMT+1000 (Australian Eastern Standard Time) | Tue Jun 11 2019 00:00:00 GMT+1000 (Australian Eastern Standard Time) | Wed Jun 12 2019 00:00:00 GMT+1000 (Australian Eastern Standard Time) |

Application
Network Status
Online

Devices
[Add Device](#)
sample_floating

Alerts

Figure 22: Snapshot of the landing page for the account dashboard

Data from each sensor node is tracked from its own designated page which includes the data from each field (temperature, lux, moisture and conductivity) presented graphically as functions over time. The graphs generated on the web application are built using the external node.JS library, Chart.JS (figure ??). Also included is each devices BDA, current battery level, location when placed (manual input), and its description. The page also lists the last communication client used to forward the data from the WSN, its battery level and name as specified on the LoRa server.

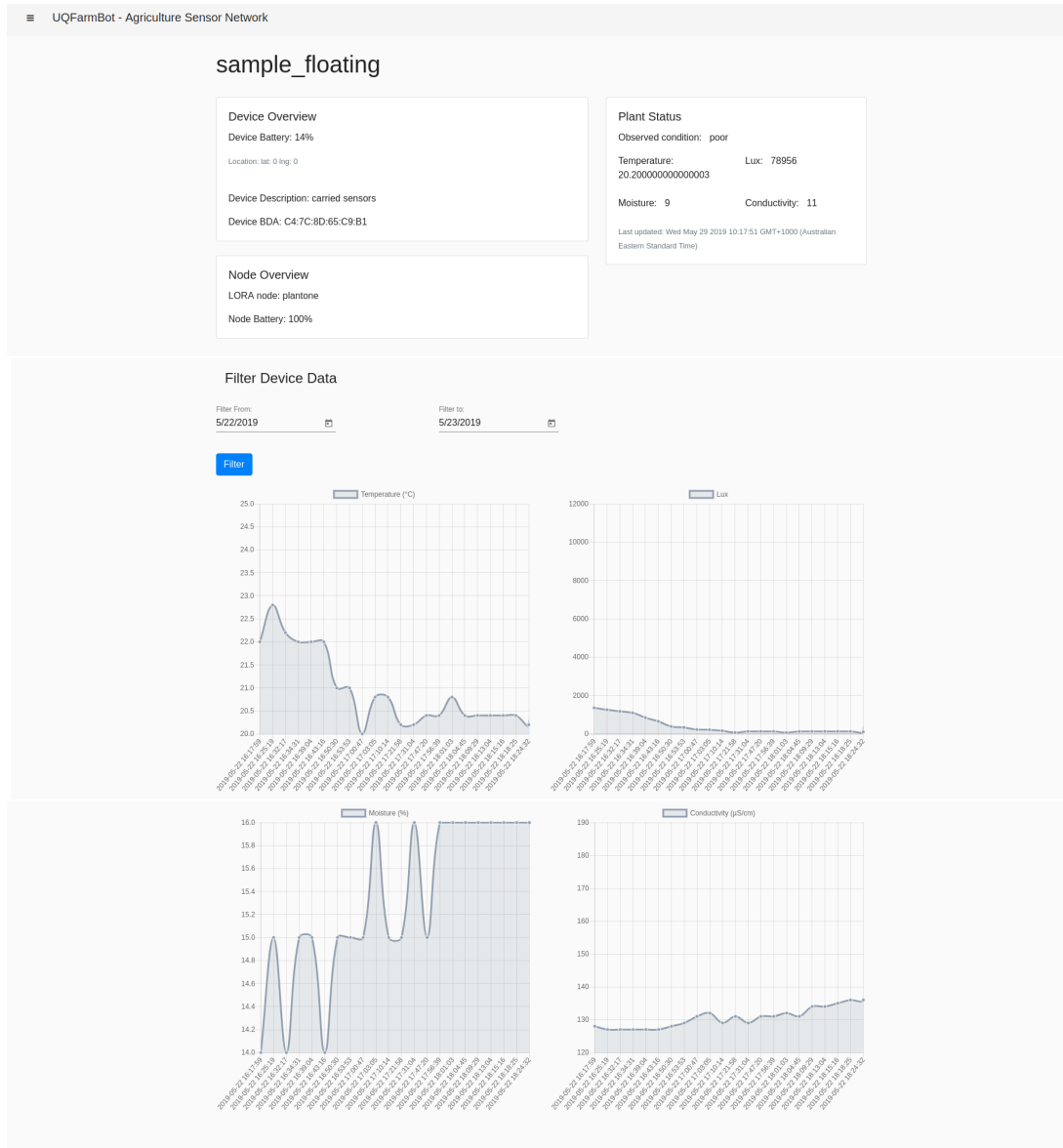


Figure 23: Snapshot of the individual sensor node page from 22nd-23rd May, key observations include the event of rain during the period which had an impact on the sensor data collected

The results of the classification tool are also presented for each sensor node page. The classification tool returns the assumed current condition of the plant the respective sensor node is currently monitoring. The generated classification is the result of passing the most recent data entries through the current TensorFlow CNN model which defines entries as either poor, fruitless, or fruit bearing. The CNN is trained on a novel training set each time the back-end sever is started. Because of the limited scope of the training data and the simplicity of the current CNN, the CNN in its current state simply serves as a proof of concept for the possible end applications that the gathered data can be used for.

The implementation of service workers allows the web application to operate even without a stable internet connection, provided that there is a cached version of the site currently stored on the browser. When running, the service workers actively cache page resources and previous data from past API calls. The drawbacks from using service workers is the increased overhead when the page is initially loaded or when the service worker is updated. The overhead comes from time required to install the service worker on the browser and store/retrieved resources. Browser resources are also limited in their ability to store received data, table 10 shows a breakdown of the availability of browser resources.

| Browser | Limit |
|---------|--------------------------|
| Chrome | < 6% of free space |
| Firefox | <10% of free space |
| Safari | <50MB |
| IE10 | <250MB |
| Edge | Dependent on volume size |

Table 10: Available browser resources for common browsers (10)

Because the current scope of the project is quite small (figure) the impact of the service worker is quite small. However, as the project grows in size these drawback may become more prominent as the volume of available data increases and the service work grows in complexity.

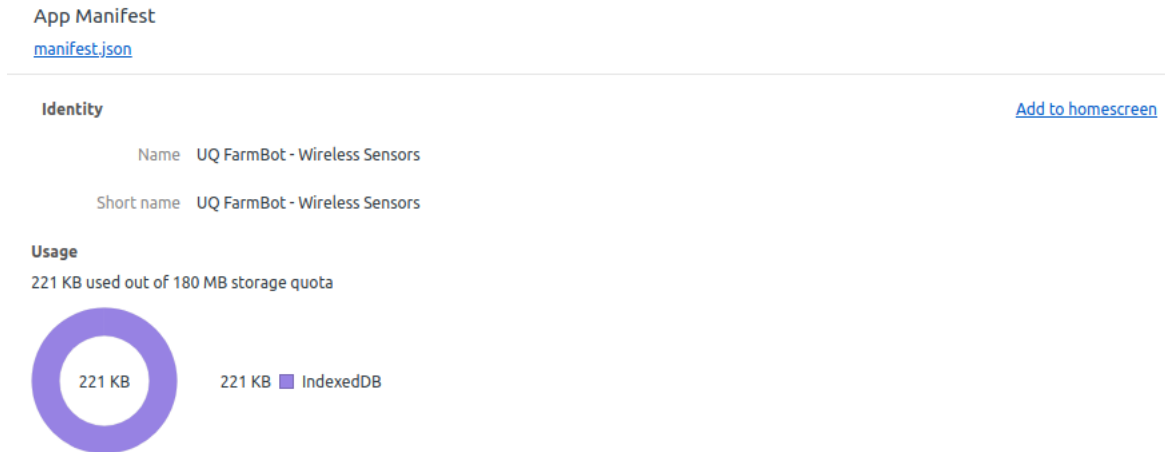


Figure 24: Current resource use by project service worker

5.3 Communication Scheme

This section covers the specific communication scheme used throughout the various stages of the project.

5.3.1 BLE4.0/GATT

Table 11 shows the available service and characteristic UUIDs on the Xiaomi MiFlora and their respective functions.

| Service UUID | Characteristic UUID | Handle | Read/ Write | Description |
|--------------------------------------|--------------------------------------|--------|----------------|------------------|
| 0000fe95-0000-1000-8000-00805f9b34fb | - | - | - | device discovery |
| 00001800-0000-1000-8000-00805f9b34fb | 00002800-0000-1000-8000-00805f9b34fb | 0x03 | read | device name |
| 00001204-0000-1000-8000-00805f9b34fb | 00001a00-0000-1000-8000-00805f9b34fb | 0x33 | write | mode change |
| 00001204-0000-1000-8000-00805f9b34fb | 00001a01-0000-1000-8000-00805f9b34fb | 0x35 | read | real-time read |
| 00001204-0000-1000-8000-00805f9b34fb | 00001a02-0000-1000-8000-00805f9b34fb | 0x38 | read | device info |
| 00001204-0000-1000-8000-00805f9b34fb | 00001a11-0000-1000-8000-00805f9b34fb | 0x3C | read | history read |
| 00001204-0000-1000-8000-00805f9b34fb | 00001a10-0000-1000-8000-00805f9b34fb | 0x3E | write | history control |
| 00001204-0000-1000-8000-00805f9b34fb | 00001a12-0000-1000-8000-00805f9b34fb | 0x41 | read | device time |

Table 11: Service and Characteristic UUIDs on the Xiaomi MiFlora

Device Name

The device name is held under the attribute handle 0x03. The generic device name for the Xiaomi MiFlora is Flower Care and is the typical identifier for sensor nodes in this project.

| Bytes | Type | Value | Description |
|-------|------------|-------------|-------------|
| All | ASCII text | Flower Care | device name |

Table 12: Xiaomi MiFlora device name

Device Information

Device information includes the current firmware version and battery level running on the device and can be accessed under the attribute handle 0x35.

| Bytes | Type | Description |
|-------|------------|--------------------|
| 00 | uint8 | battery level in % |
| 02-06 | ASCII text | firmware version |

Table 13: Xiaomi MiFlora device informaion

Device Time

The device time refers to the current time in seconds since the sensor node was powered on. The device time can be accessed from handle 0x41.

| Bytes | Type | Description |
|-------|--------|-------------------------------|
| 00-03 | uint32 | seconds since device power on |

Table 14: Xiaomi MiFlora device time

Real-time Data

Real-time data is retrieved in two parts. Initially the device mode is updated to prepare real-time data. The process to update the device mode is done through writing 2 bytes (0xa01f) to the mode change handle (0x33). After writing, the real-time data will then be available for collection from the handle 0x35.

| Bytes | Type | Description |
|-------|--------|---|
| 00-01 | uint16 | temperature in 0.1 °C |
| 03-06 | uint32 | brightness in lux |
| 07 | uint8 | moisture in % |
| 08-09 | uint16 | conductivity in $\mu\text{S}/\text{cm}$ |

Table 15: Xiaomi MiFlora real-time data

Historical Data

In a similar way to real-time data, historical data can be retrieved by updating the device mode to return historical data. The action requires 3 bytes (0xa00000) to be written to the history control handle (0x3E). An index of the historical data will then be available from history data handle (0x3C) where 16 bytes represents number of stored historical records.

| Bytes | Type | Description |
|-------|--------|-------------------------------------|
| 00-01 | uint16 | number of stored historical records |

Table 16: Xiaomi MiFlora historical data reference

Using the index, the address for individual entries can be computed as an additional offset of two bytes (0x100) for each consecutive entry from the base address 0xA10000. In the example case where there are 16 entries, entry 0 will be positioned at 0xA10000, entry 1 will be at 0xA10100 and so on till entry 16 at 0xA11000. Specific entries can be retrieved by writing the respective address to the history control handle (0x3E) and then reading the result from the history read handle (0x3c).

| Bytes | Type | Description |
|-------|--------|--|
| 00-03 | uint32 | timestamp, seconds since device power on |
| 04-05 | uint16 | temperature in 0.1 °C |
| 07-09 | uint32 | brightness in lux |
| 11 | uint8 | moisture in % |
| 12-13 | uint16 | conductivity in $\mu\text{S}/\text{cm}$ |

Table 17: Xiaomi MiFlora historical data value

5.3.2 LoRaWAN Payload

LoRaWAN payloads are the product of the BLE conversation between a sensor node and the communication client. Each conversation includes a combination of calls which retrieve the device BDA (from scan), device battery level - two calls, write/read to device information - and the current real-time data - two calls, write/read to real-time data. At the end of the conversation (all values retrieved) payloads are then constructed as strings using two length hex values separated by colons, which are then broken down into their byte values. Currently, each LoRaWAN frame captures a single BLE conversation and there is no overlap between frames. Table 18 is a representation of the implemented LoRaWAN payload.

| Bytes | Type | Description |
|-------|--------|---|
| 00-05 | uint16 | device BDA |
| 06-07 | uint16 | temperature in 0.1 °C |
| 08-11 | uint16 | brightness in lux |
| 12 | uint16 | moisture in % |
| 13-14 | uint16 | conductivity in $\mu\text{S}/\text{cm}$ |
| 15 | uint16 | battery level in % |

Table 18: LoRaWAN payload structure

5.3.3 HTTP Payload

The HTTP integration available on the LoRa server posts received data via HTTP as JSON. The fields posted include the payload of the received LoRa frame as well as status information regarding the source LoRa client. An outline of the posted JSON can be seen in table 19.

| Field | Type | Description |
|-----------------|---------|-------------------------------------|
| applicationID | string | associated application id |
| applicationName | string | associated application name |
| deviceName | string | associated device name |
| devEUI | string | massociated device EUI |
| rxInfo | Object | status of the receiving device |
| txInfo | Object | status of the transmitting device |
| adr | boolean | adaptive data rate on/off |
| fCnt | int | frame counter |
| fPort | int | end application/service identifier |
| data | string | payload received by the LoRa server |

Table 19: HTTP payload structure

6 Project Limitations

Major limitations of the project, surround the use of LoRaWAN as the MAC layer protocol for forwarding data past the communication client. Although sufficient in the current deployment which included two communication clients and 4 sensor nodes, between the testing site and the lab. The use of LoRaWAN has significant limitations when scaling the network for larger deployment ranges or more complex networks (more clients/sensor nodes). The specific limitations identified at the conclusion of this project include:

- Limitations in sending downlink frames.
- Lack of Multi-hop, mesh, or P2P.
- Weak localisation features.

As a potential solution to the current limitations imposed by the use of LoRaWAN, the project has briefly explored Dash 7 (section 3.3.4) as a potential alternative. Dash7 is a LoRa compatible protocol which covers presentation, session, transport, network and data link OSI layers. Dash7 has also introduced support for application layer protocols including IPv6. The notable functionality which Dash7 introduces are support for mesh networking and fully bi-directional communication.

6.1 Limitations in sending downlink frames

Communication clients in the current WSN operate as Class A devices for LoRaWAN and offer a single receive window for downlink frames after an uplink frame. Using this model, the only way for the communication gateway to push messages to the communication client is after first receiving an uplink frame from the client. Assuming the default configuration, where the uplink frames are sent between hour long intervals, this can introduce significant latency when attempting between when the downlink frame is scheduled and when it is physically sent. Though there is the option to implement the communication client as a LoRaWAN Class C type device and have the device constantly keep its receive window open. The lost power savings and ability to effectively sleep the communication client make it an impractical solution for the use case.

Using Dash7 as an alternative, will enable bi-directional communication between the gateway and the communication client. Dash7 implements both push and pull models (figure 25), the pull model allows data to be queried using a request/response mechanism (D7A query protocol data transfer (8)). This mechanism will allow the gateway to send requests to communication clients freely and without the need for receive windows. Whilst alternatively the push model can still be used to send uplink frames from communication client to the gateway.

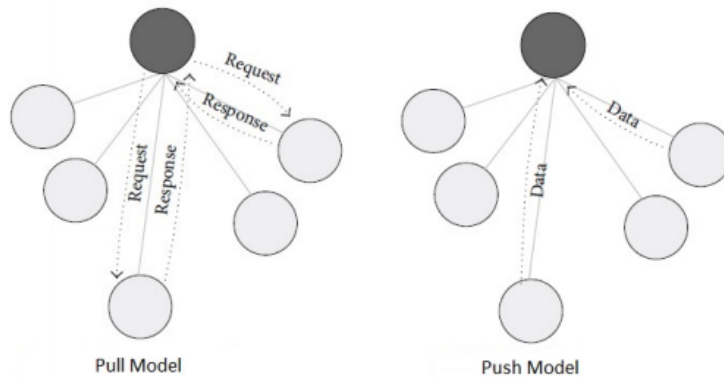


Figure 25: DASH7 Alliance Protocol Communication Model (8)

6.2 Lack of Multi-hop, mesh, or P2P

As a result of the projects current use of the LoRa server to manage MAC layer protocols on the gateway. The current implementation of the WSN using LoRaWAN is unable to support multi-hop or P2P functionality as the protocol is unable to support communication without the LoRa server (35). Dash7 can again be used as a solution in this case, as Dash7 currently supports P2P communication (35). The requirements to manage communication over Dash7 are handled in the data link layer using access profiles and access classes without the need for a central server. The result allows Dash7 to perform P2P and multi-hop operations but only for two hops (8).

6.3 Weak localisation features

Though not covered in the original scope, a potential requirement for future projects may include the need to be able to localise nodes deployed in the field. In the current scope of the FarmBot project this feature is redundant given the limited size of the deployment field but for "realistic" fields which are much larger the need to be able to localise and track the position of communication clients for retrieval may become necessary. This feature could also be a prominent addition to mapping variations in crop/pant conditions and could be extremely beneficial to farmers who using the current project would have to manually plot the position of sensors and communication clients. A potential future works for this project could be to apply localisation techniques such as triangular estimation or RSSI similarity matching methods (28), to resolve the current lack of localisation capabilities.

7 Concluding Remarks

The implemented WSN and web application proved to be a feasible and cost effective solution for wireless monitoring of the FarmBot project. The WSN allows key data to be precisely collected and delivered via both BLE4.0 and LoRa channels, where the web application is then able to relay and present the information in comprehensive way which can be used to trigger actionable events. Though successful in the current test environment considerable foresight and future design improvements are required before the system can be moved on to something more closely related to full scale agriculture.

In the current setting, the project is able to comfortably handle the 80m range requirement with a maximum measured range of 1.2Km whilst still connected. The possibility of sending multi-hop (section 6.2) has the potential to further expand this range. Operation in a complex environment is achieved through the use of separate BLE sensor nodes which can be positioned in a variety of different ways to meet the situation and then achieve the required range and operation through both indoor and outdoor environments using LoRa. The overall cost of the project was kept under a set \$100 (AUD) budget at \$84.30 (AUD) and all project components can be powered effectively from a mobile 3V power bank.

The web application also performs well, capable of managing and delivering the stored data from the WSN, with a database solution which can be horizontally scaled to accommodate for continuous data integration. The sample TensorFlow CNN also successfully demonstrates a possible use and integration case for the data collected and how it can be applied (with more development) using more complex logic to drive decisions on the FarmBot.

In retrospect of the final project outcomes, perhaps the greatest challenge throughout this project was in developing ways to accurately and consistently log performance and errors, especially during long deployments. Given the minimal resources available on embedded system projects, difficulty came in attempting to monitor the project when deployed in the field where no form of logging existed except for the messages which were received by the LoRa server. If reattempting the project, proper design and development time should be given to building a proper framework from which to test and monitor the firmware components of the project.

References

- [1] J. Gutiérrez, J. F. Villa-Medina, A. Nieto-Garibay, and M. Ángel Porta-Gándara, “Automated irrigation system using a wireless sensor network and gprs module,” 2014.
- [2] A. M. Adrian, P. L.Mask, and S. H. Norwood, “Producers’ perceptions and attitudes toward precision agriculture technologies,” 2005.
- [3] R. Kingwell, “Climate change in australia: agricultural impacts and adaptation,” 2006.
- [4] Ookla, “Australia mobile coverage, based on q2-q3 2017 data,” 2017. Last accessed 16 May 2019.
- [5] Espressif Systems, “Esp32: Ble api,” 2019. Last accessed 18 May 2019.
- [6] Argenox Technologies LLC, “Ble advertising primer,” 2017. Last accessed 10 May 2019.
- [7] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley, “A study of lora: Long range low power networks for the internet of things,” 2016.
- [8] W. Ayoub, A. Samhat, F. Nouvel, M. Mroue, and J.-C. Prévotet, “Internet of mobile things: Overview of lorawan, dash7, and nb-iot in lpwans standards and supported mobility,” 2018.
- [9] Espressif Systems, “Esp32 series datasheet,” 2019. Last accessed 18 May 2019.
- [10] M. Cohen, “Web storage overview,” 2019. Last accessed 3 June 2019.
- [11] Australian Bureau of Statistics, “7501.0 - value of principal agricultural commodities produced, australia, preliminary, 2016-17,” 2018. Last accessed 2 May 2019.
- [12] Australian Bureau of Statistics, “4627.0 - land management and farming in australia, 2016-17,” 2019. Last accessed 2 May 2019.
- [13] CSIRO, “Csiro: Precision agriculture,” 2019. Last accessed 18 May 2019.
- [14] B. Whelan and J. Taylor, *Precision Agriculture For Grain Production Systems*. CSIRO Publishing, 2013.
- [15] precisionagriculture.com.au, “Precision agriculture: Website,” 2019. Last accessed 2 June 2019.
- [16] CSIRO, “Csiro: Digital agriculture,” 2019. Last accessed 18 May 2019.
- [17] R. Bongiovanni and J. Lowenberg-Deboer, “Precision agriculture and sustainability,” 2004.
- [18] S. Patil, A. Kokate, and D. Kadam, “Precision agriculture: A survey,” 2016.
- [19] M. T. Batte and M. W. Arnholt, “Precision farming adoption and use in ohio: case studies of six leading-edge adopters,” 2003.
- [20] J. Curtin, “A digital divide in rural and regional australia?,” 2001.
- [21] A. Vidot, “Almost half of regional australians report internet is ‘very poor’, ‘inadequate’: University of canberra survey,” 2016.
- [22] Ookla, “Australia fixed coverage, based on q2-q3 2017 data,” 2017. Last accessed 16 May 2019.
- [23] M. DSouza, “Farmbot wireless sensors: Super project,” 2017. Last accessed 8 June 2019.
- [24] M. T. French, “Abiotic factors plants: A local pollution study with global implications,” 2017. Last accessed 1 June 2019.
- [25] HuaHuaCaoCao, “Flower careTM smart monitor,” 2019. Last accessed 2 June 2019.
- [26] The Dark Sky Company, LLC, “Dark sky api,” 2019. Last accessed 4 May 2019.

- [27] Bluetooth, “Bluetooth: Website,” 2019. Last accessed 5 June 2019.
- [28] B. H. Laekemariam, “Rssi-based indoor localization system,” 2012.
- [29] J. Petäjärvi, K. Mikhaylov, M. Pettissalo, J. Janhunen, and J. Iinatti, “Performance of a low-power wide-area network based on lora technology: Doppler robustness, scalability, and coverage,” 2017.
- [30] Zigbee Alliance, “Zigbee: Website,” 2019. Last accessed 29 May 2019.
- [31] C. M. Ramya, M. Shanmugaraj, and R. Prabakaran, “Study on zigbee technology,” 2011.
- [32] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of things: A survey on enabling technologies, protocols, and applications,” 2015.
- [33] Electronics Notes, “Ieee 802.11af white-fi technology,” 2019. Last accessed 21 May 2019.
- [34] SigFox, “Sigfox: Website,” 2019. Last accessed 29 May 2019.
- [35] Haystack, “Haystack: Website,” 2019. Last accessed 29 May 2019.
- [36] Dash7 Alliance, “Dash7: Website,” 2019. Last accessed 29 May 2019.
- [37] Hoperf Electronic, “Rfm95/96/97/98(w) - low power long range transceiver module v1.0,” 2019. Last accessed 18 May 2019.
- [38] Banggood, “Raspberry pi zero 512mb ram 1ghz single-core cpu support micro usb power and micro sd card with noobs,” 2018. Last accessed 17 September 2018.
- [39] Banggood, “Geekcreit® 30 pin esp32 development board wifi+bluetooth ultra low power consumption dual cores esp-32 esp-32s board,” 2018. Last accessed 17 September 2018.
- [40] SparkFun, “Sparkfun lora gateway - 1-channel (esp32),” 2018. Last accessed 17 September 2018.
- [41] Adafruit, “Adafruit rfm95w lora radio transceiver breakout - 868 or 915 mhz - radiofruit,” 2018. Last accessed 17 September 2018.
- [42] Banggood, “Geekcreit® uno r3 atmega328p development board for arduino no cable,” 2018. Last accessed 17 September 2018.
- [43] Semtech, “B-l072z-lrwan1,” 2018. Last accessed 17 September 2018.
- [44] JB-HiFi, “Cygnett charge up sport 4400mah powerbank (black),” 2019. Last accessed 10 April 2019.
- [45] Amazon Web Services, “The freertos™ reference manual api functions and configuration options,” 2017.
- [46] Espressif Systems, “Esp-idf programming guide,” 2019. Last accessed 29 April 2019.
- [47] B. Marty, “Github repository: Esp32 lm32,” 2017. Last accessed 2 May 2019.
- [48] IBM Corporation, “Ibm long range signaling and control, ibm lorawan in c (lm32),” 2015.
- [49] O. Khutsoane, B. Isong, and A. M. Abu-Mahfouz, “Iot devices and applications based on lora/lorawan,” 2017.
- [50] LoRa Alliance, “Lora alliance: Website,” 2019. Last accessed 5 June 2019.
- [51] LoRaServer.io, “Lora server, open-source lorawan network-server,” 2019. Last accessed 29 April 2019.
- [52] MongoDB, Inc., “Iot reference architecture.”
- [53] Node.JS Foundation, “Express,” 2019. Last accessed 25 May 2019.
- [54] Google, “Angular,” 2019. Last accessed 25 May 2019.

- [55] Node.JS Foundation, “Node.js,” 2019. Last accessed 25 May 2019.
- [56] A. Russell and J. Song, “Service workers, w3c first public working draft 08 may 2014,” 2014.
- [57] Mongoose.JS, “Mongoose,” 2019. Last accessed 25 May 2019.
- [58] TensorFlow, “Tensorflow,” 2019. Last accessed 25 May 2019.
- [59] Ngrok, “Ngrok,” 2019. Last accessed 25 May 2019.
- [60] Home Assistant, “Mi flora plant sensor,” 2018. Last accessed 2 June 2019.
- [61] Banggood, “Xiaomi flora 4 in 1 flower plant light temperature tester garden soil moisture nutrient monitor,” 2018. Last accessed 17 September 2018.