



# FreshByte IoT: Fresh Food Telemetrics

October 2019

School of Engineering

OENG1168 – Engineering Capstone Project Part B

Assignment Task 2 – Final Report

## Academic Supervisor:

Dr. Samuel Ippolito

## Team Name:

Raven

## Team Members:

Kevin Totikidis - s3601377

Huy Nguyen – s3603441

Li Hao Soh – s3587920

Jong Yiing Yang – s3457782

# Contents

# Statement of Problem

According to the 2017 National Food Waste Strategy [1], food wastage has become a worldwide epidemic as 1 billion tons of food produced for human consumption is wasted on earth each year. Economically, food waste costs approximately US$940 billion globally, while costing the Australian government A$20 billion annually [2]. The Australian fresh food industry plays a significant role for the Australian economy, with horticulture being one of the largest export groups in the industry. 618 Million Dollars was the annual cost of transporting horticulture products in 2017, which proposes a very large target market [3]. Horticulture products include fresh fruit/vegetables, fresh produce, meat, wheat, plants and flowers. Environmentally, food that is harvested but wasted, uses about one-quarter of all water used by agriculture every year [4].

After the food waste produced in households; which contributes to a fraction of food waste [5], a large percentage of food is wasted due to spillage and degradation during handling, storage and transportation from farmgate to distribution channels [6]. This spoilage of food mostly goes unnoticed only until it reaches its intended destination. $350 million tons of fresh food is wasted during the handling and storage phase alone in 2015, equating to a $120 billion lost [7]. Identifying the cause of food degradation becomes a challenge as many factors come into play while transporting fresh food. Said cases of supply chain failure and damaged goods or packaging may appear to occur sporadically or in isolated exceptions, but additional data acquisition and analysis on transit information may tell a different story. Stakeholders of the fresh food supply chain require a solution to probe into the black box of this supply chain system, assisting them in isolating and pinpointing the exact location and time of abnormalities occurring within the transport process. The ability to quickly visualize data and improve its transparency between stakeholders opens doors to accelerating problem rectification and streamlining future shipments. As of time of writing, few specifically designed off-the-shelf solutions have been developed for the fresh food supply chain industry [8].

Hence, the aim of this project is to increase the transparency of fresh food data during the transportation phase of the supply chain. This can be done by monitoring metrics such as temperature and gas emissions of the fresh food remotely for all stakeholders. The proposed solution is an autonomous monitoring device placed within food packaging prior to the transport process. The device will independently measure and track the individual package's location and internal conditions from source to destination. The device then uploads vital transport information onto an online database, for data analysis and performance tracking, via Telstra's Cat M1 or NBIoT network.

# Executive Summary

This final report aims to demonstrate the advancements and adjustments made to the initial prototype established in semester 1, used to compete in the 2019 Telstra Innovation Challenge.

This product intends to more easily acquire and represent data to help benefit those working in the fresh food industry, whether that be farmers or transport and logistics companies.

The project is still focused in the area of IoT, with the FreshByte module continuing to provide its major goals of both freight tracking and data transparency. During the packing stage, the module itself can be placed within packaging; both at the farm or during provisional stages such as at itinerary warehouses. The module can then simply be retrieved at the end of its journey, recharged if needed, and placed back into the next shipment.

A primary objective is to reduce the number of incidents occurring during the transport process itself; by providing control of data which is transparent and easy to understand. This can not only reduce costs for those involved in the transport process, but also their total contribution to global food wastage as well.

The autonomous device tracks the freight's temperature, humidity, oxygen concentration, carbon dioxide concentration, acceleration, ethylene concentration and GPS coordinates.

Furthermore, the product is now able to provide a fully functioning warning system, which notifies authorities through the cloud, when a threshold such as temperature or humidity has been breached. This is made possible with the addition of enhanced software upgrades to make data visualization simpler to navigate; whilst being packaged up to be much more robust in terms of size and appearance.

Moreover, the product has a unique shelf life prediction capability which can predict the shelf life of specific foods. This unique feature was implemented to separate this product from other freight tracking systems out there in the market.

The major changes and enhancements made for this project include power efficiency, GPS improvements, the introduction of data logging, the change of network communication, addressing operating temperature issues, hardware appearance and data visualization improvements. In addition, the total elimination of hardware and software constraints limited to us from Telstra created more freedom in terms of the changes that could be made. This further allowed the product to evolve into a more rounded one, which could target a larger audience.

# Background and Literature Review

## Market Comparison

Parcel and shipment tracking technology have also seen a significant amount of advancement – from barcode scanners [9] to physical tracking devices like ParceLive [10]. However, the former has been met with a fair share of logistical mishaps, from misplacement and mishandling inside and out of warehouses, to shipments lost in transit [11]. On the other hand, the latter device and service, along with its array of on-board sensors, may not be tailored to suit the temperature and environmental extremes of the cold food chain, and is more well-suited for conventional consumer-based parcel tracking [12]. Alternatives such as the TREK-120 cold chain sensor [13]. have recently made its way into the market but does not provide the added functionality of shelf life estimation. In the case of fresh food shipments, also known as the cold food chain, shipments freighted around Australia are typically done in refrigerated containers [14], whereby a chiller unit measures the container's internal ambient temperature and subsequently adjusts its cooling output, keeping said container of fresh food within a temperature threshold depending on the type of food being freighted. Unfortunately, even with the industry's best attempts to speed up processes and protocols to minimize transit time and preserve the shipments' freshness; 75 percent of Australia's estimated $20 Billion spent due to food waste [15] occurs when the food was being handled between farmgate and consumer [16].

Table 1. Market Research Summary.

| Implementation | Description | Weaknesses | Remedy/ Proposed Functionality |
|---|---|---|---|
| ParceLive tracking | Tracking unit providing real-time location and information. At the end of the shipment, the parcel recipient returns the unit into any postal mailbox. | Product operating conditions unknown. | Selection of more robust, temperature-tolerant sensors. |
| TREK-120 LoRa wireless cold chain sensor | Tracking unit for the cold-food chain network. | Product does not contain enough sensors to achieve our required functionality. No GPS tracking. | Additional gas sensors and shelf-life estimation algorithms implemented. |
| Cold Storage + Portable Refrigeration Chillers | Distribution centers and trucks in-transit contain chiller units to keep temperatures below a safe range. | Sensors merely track the internal ambient temperature rather than specific temperature of produce and may not be accurate. | Place sensors as close to the shipment as possible. This allows accurate measurements and eliminates false positives when alerting authorities on abrupt measurement changes. |
| Warehousing and Logistics Control | Optimizing the operating procedures of warehouses and logistics to reduce time taken to freight shipments. | Prone to human errors. | Implement a solution that requires minimal human effort and intervention. |

## Gap Analysis

From the research done above, a more detailed gap analysis was done on the products and methods currently available in the supply chain logistics industry. Focus areas we deemed important within the industry such as network connectivity, telemetry data measurement method, automation and data transparency were chosen. We then compared the current state of these focus areas against an ideal or better target, laying out the functionalities of our product needed to achieve those targets.

Table 2. Gap Analysis

| Gap Analysis | | | | |
|---|---|---|---|---|
| Focus Areas | Current Market State | Future State | Identified Gap | Actioned Functionality |
| **Network Connectivity** | Other available products on the market use shorter range or less ubiquitous wireless networks and technologies such as LoRa [13] and XG network [10] | Products that uses more widely accessible networks such as Cat M1 and NB-IoT. | Using shorter range wireless technology would not allow data to be transmitted remotely to be monitored. While niche technology would be phased out with newer cheaper technology. | The Arduino MKR NB1500 on the FreshByte utilizes a ublox modem that can connect to Cat M1 and NB1 bands. |
| **High Resolution Temperature Data** | Current temperature sensors on refrigeration units of trucks only measures ambient temperature of the fridge. | Temperature measurement tools that can measure temperature of different crates or different types of fresh food within the same shipment. | Temperatures of different fresh produce may vary through its journey. Few products offer specific temperature measuring capabilities | The FreshByte module is designed to be portable and attachable to single crates allowing the module to probe the temperature of specific fresh produce. |
| **Automation** | Human interaction still plays a vital role in the logistics industry throughout the supply chain. Tasks such as inventory management is usually done by a worker using technology such as barcode scanners. | A process that involves minimal human interaction in acquiring a product's logistics data throughout its journey. | Human interaction especially subcontractor interaction is becoming an issue in the logistics industry as errors may arise at any point of the supply. | Vital data of fresh produce such as location, temperature and inventory info are all autonomously transmitted onto a database without the need for a human input. |
| **Data Transparency** | Logistics companies usually employ subcontractors to complete shipments. During this phase of the supply chain, shipment information is usually lost between transfer of good and destination. | The transmission of telemetry data on the condition of fresh produce shipments to be periodically collected and transmitted for all stakeholders of the supply chain to access. | Condition of the fresh produced is only discovered at the end of its journey or if deliberately check on during the shipment. No information is obtained otherwise. Identification of when and where an issue arises becomes an arduous task. | |

## SWOT Analysis of Proposed Solution

A SWOT (Strength, Weakness, Opportunity and Threats) analysis was done on the designed product as shown below.

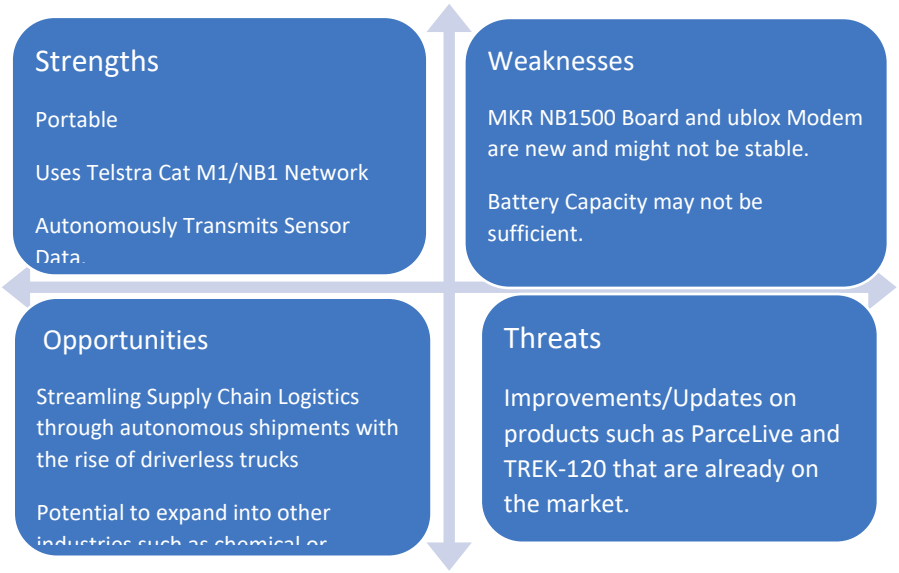



Diagram 1 SWOT Analysis of FreshByte Module Design

Overall, apart from weaknesses stemming from components and unusual operating conditions, our product design addresses most of the gaps identified in the supply chain logistics industry. These weaknesses can be improved on in future iterations and through more testing. With a more robust system, there is potential for this product to be integrated in different industries depending on the sensors required.

# Methodology and Engineering Design

As the bulk of a working prototype has already been developed and tested in Semester 1 for the Telstra Innovation Challenge, improvements such as website aesthetics, housing, power management and extra functionality were worked on in Semester 2. As mentioned in the SWOT analysis, connecting to the IoT networks using the Arduino MKR NB1500 can be temperamental and unreliable at times. That is why the board is being swapped for an ESP32 that connects to the internet through WI-FI instead of cellular data. This serves only as a proof-of-concept and all functionality of the module remains the same. The following Gantt Chart shows the timeline of the project to incorporate all the improvements made to the FreshByte module.

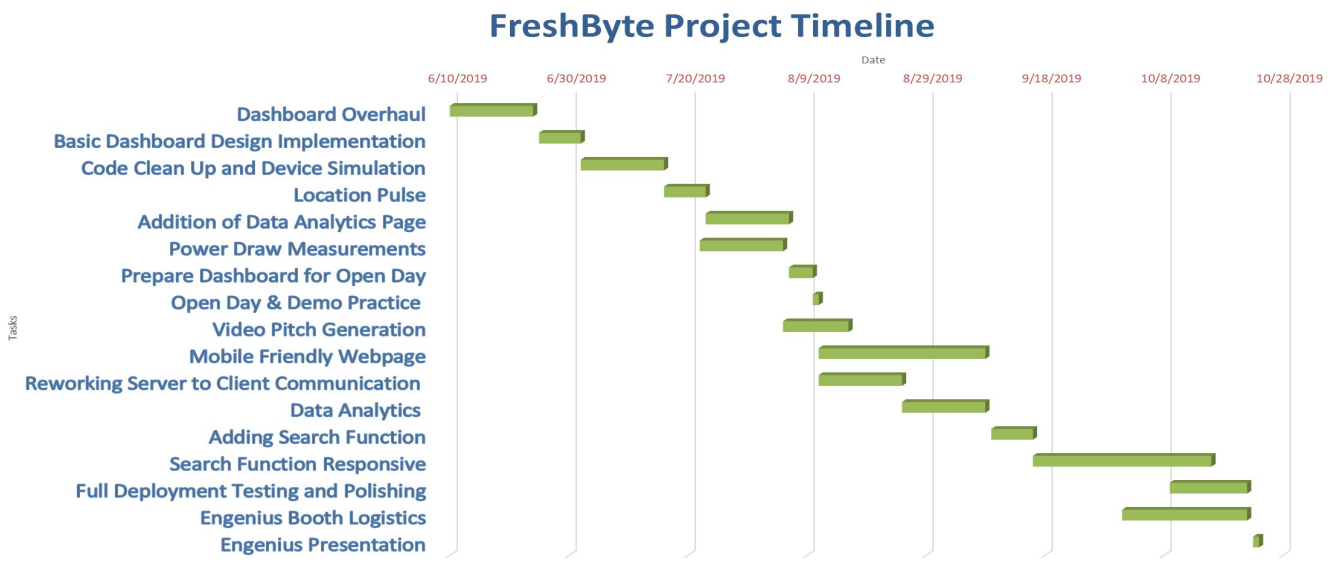



*Figure 1 Semester 2 Project Timeline*

## System Block Diagram

A high-level system diagram as shown below was drawn to assist in visualizing how the system communicates and transmits data. The sensors will all communicate with the ESP32 through different protocols whilst uploading data to Azure through MQTT.

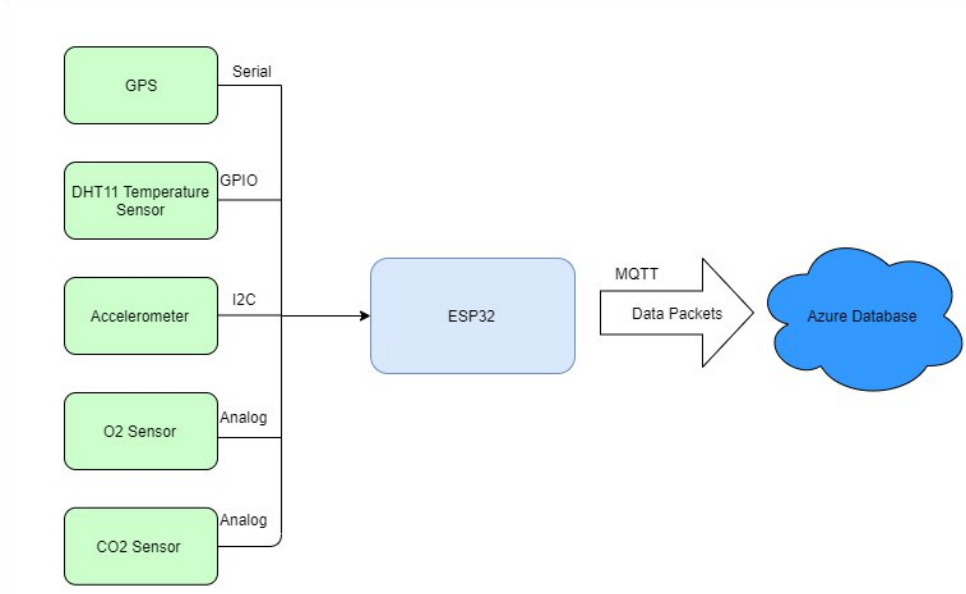



*Figure 2 High Level System Block Diagram*

## Temperature Considerations

The storage temperature of fresh produce must be taken into account, especially when considering factors such as sensor performance. Fresh produce has an optimal temperature to keep them as fresh as possible, with different kinds of produce having different optimal temperatures.

The figures below come from the Department of Primary Industries and Regional Development Government of Western Australia. [17] They detail the optimal temperature for cold food storage.

Fruit

Table 1 Storage conditions for fruit

| Fruit | Temperature range (°C) | Relative humidity (%) | Storage time | Short-term temperature (°C) | Comments |
|---|---|---|---|---|---|
| Apple** | -1-4.5<br>2-4.5 | 90-95<br>95 | 4-32 weeks | 0 | — |
| Apricot** | -0.5-0 | 85-95 | 1-3 weeks | 0 | — |
| Avocado | 0-2<br>4.5-13 | 90-95<br>85-95 | 10 days<br>2-4 weeks | 2<br>7 | Green<br>Ripe |
| Banana** | 13.5-15<br>12.5-21 | 85-95 | 2-5 days<br>4-21 days | 13 | Ripe<br>Green |
| Blackberry | -0.5-0 | 85-100 | 2-3 days | 0 | — |
| Blueberry | -0.5-0 | 90-100 | 2 weeks | 0 | — |
| Boysenberry | 0 | 90-100 | 2-3 days | 0 | — |
| Carambola | 10-15 | 90 | 5 weeks | — | Starfruit |
| Cherry | 0 | 90-95 | 3-7 days | 0 | Sour |
| Cherry | -1-0 | 85-95 | 2-4 weeks | 0 | Sweet |
| Chico | 15 | 85-95 | 2 weeks | — | — |
| Coconut | 0-2 | 80-95 | 4-8 weeks | — | — |

*Figure 3 Optimal Fruit Storage Temperatures.*

# Storage of fresh fruit and vegetables

Page last updated: Wednesday, 10 August 2016 - 8:02am

## Vegetables

Table 2 Storage conditions for vegetables

| Vegetable | Temperature range (°C) | Relative humidity (%) | Storage time | Short-term temperature (°C) | Comments |
|---|---|---|---|---|---|
| Artichoke, globe** | 0 | 90-100 | 3-4 weeks | 0 | - |
| Artichoke, Jerusalem | -0.5-0 | 90-95 | 8-20 weeks | - | - |
| Asparagus* | 0-2.5 | 85-100 | 2-4 weeks | 2 | - |
| Bean* | 4-10 | 85-100 | 1-3 weeks | 7 | Green, French |
| Bean, Lima | 0-4.5 | 90 | 1-2 weeks | - | - |
| Beetroot, bunched | 0 | 95 | 1-2 weeks | - | - |
| Beetroot, topped | 0 | 90-100 | 4-20 weeks | 0 | - |
| Broccoli* | 0 | 90-100 | 1-2 | 0 | - |

*Figure 4 Optimal Vegetables Storage Temperatures.*

The most extreme cases are what we need to focus on as our gas sensors can handle up to approximately -20 degrees Celsius. The figures below show that most extreme cases where a pair (fruit) is to be -2 to 0 degrees Celsius and horseradish (vegetable) to be -1 to 0 in Celsius. This means that our sensors will be able to handle it.

**Minimum Temperature for Fruit Storage of In Celsius**

| | | | | | |
|---|---|---|---|---|---|
| Pear** | -2-0 | 90-95 | 8-28 weeks | 0 | — |

*Figure 5 Minimum Temperature for Fruit Storage of In Celsius*

**Minimum Temperature for Storage of Vegetables in Celsius**

| | | | | | |
|---|---|---|---|---|---|
| Horseradish | -1-0 | 90-100 | 40-48 weeks | - | - |

*Figure 6 Minimum Temperature for Storage of Vegetables in Celsius.*

It is important to note the storage times of the foods as well from the figures above. The most extreme maximum storage times of foods can range anywhere from a week to almost a year, with the average from the figures above approximately in the 1 to 8-week range. Therefore, trucks transporting produce for example, will differ in the distance and times they have to travel based on the foods. Raspberries and blackberries are in the 2-3 days range and thus will need to be transported as quickly as possible from farmgate to consumer.

## Temperature (Hardware) Design Considerations

In terms of the battery, we have chosen a standard lithium polymer (LiPo) variant which can handle -2 degrees Celsius. It has an operating temperature Charge range of 0~45℃ and an operating temperature discharge range of: -20~60℃ (Li-ion Polymer Battery MODEL: GMB042035) [18].

The ESP32 board we are using can also handle this temperature quite comfortably. The ESP32 is capable of functioning reliably in industrial environments, with an operating temperature ranging from –40°C to +125°C [19].

More detailed testing on battery consumption such as current and power draw is specified in the experimental/findings section.

## Hardware

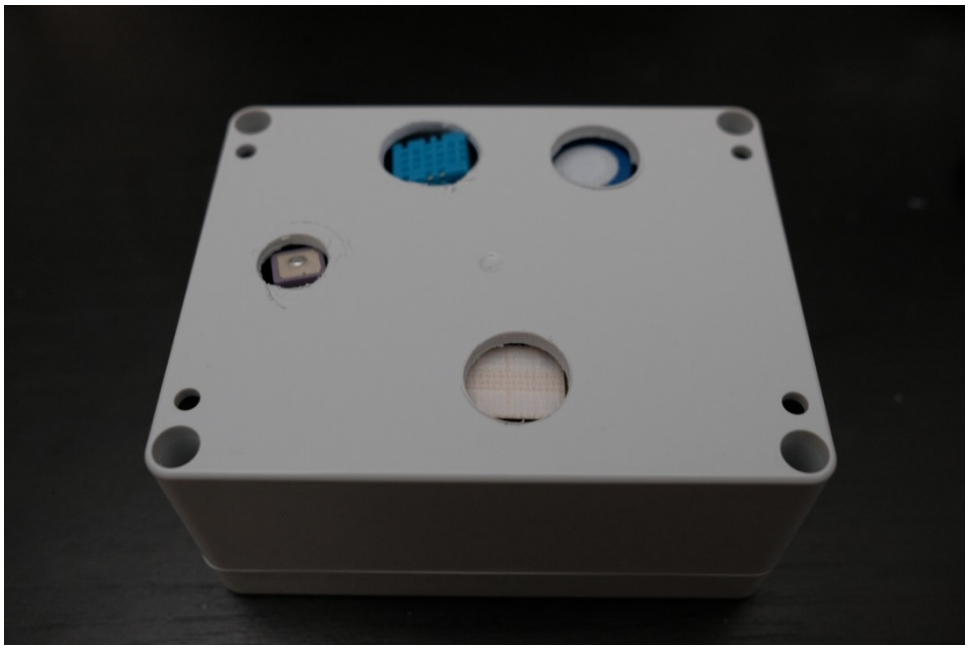


*Figure 7 Underside of the FreshByte Module Prototype.*

Table 3. Summary of hardware used.

| **Name:** | **Cost($AUD):** | **Function:** | **Communication Protocol:** |
|---|---|---|---|
| Arduino MKR NB 1500 | 119 | Main Board | N/A |
| Adafruit Ultimate GPS | 42 | GPS Location | Hardware Serial |
| DHT11 | 6 | Temperature & Humidity | GPIO |
| LIS3DH | 3 | Accelerometer | I2C |
| MQ-131 | 16 | O2 Sensor | Analog |
| MH-Z14A PWM NDIR | 23 | CO2 Sensor | Analog |
| 4000mAh LiPo | 17 | Additional Power Source | N/A |
| **ESP32** | **12** | **WI-FI Enabled MCU** | **N/A** |
| Total Cost: | 238 | | |

The main board (Arduino MKR NB 1500) was provided by Telstra along the following sensors: Adafruit Ultimate GPS, DHT11, LIS3DH. The Arduino MKR NB 1500 was used as main board in Semester 1 but, for Semester 2 the ESP32 replaces the Arduino MKR NB1500 and was added onto the components list from last semester as previously mentioned, where all the sensors and the battery will be connected to it. (Figure 8).

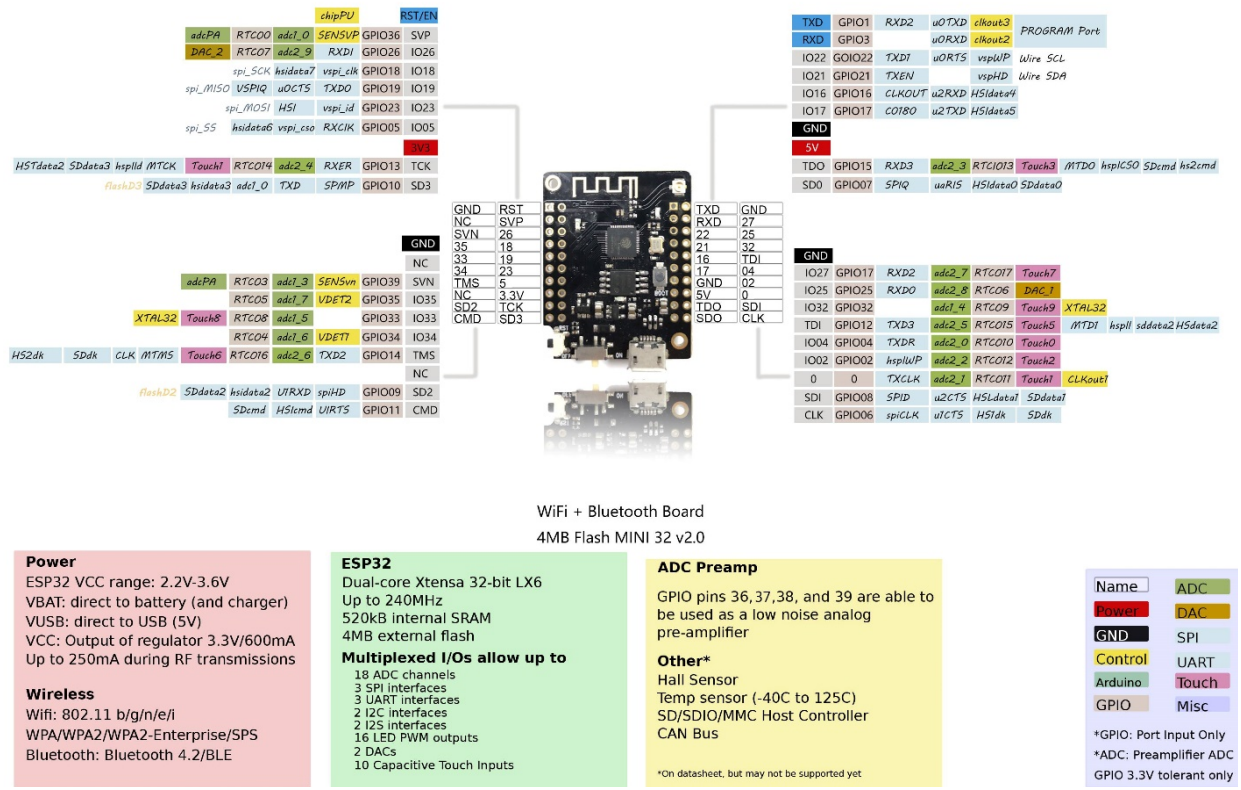



*Figure 8 ESP 32 Board.*

**Arduino MKR NB 1500** (Board used in semester 1)
The Arduino MKR NB 1500 is a suitable main board due to the follow features:

- SPI
- I2C
- Hardware Serial
- Analog
- Wireless Radio
- SAMD21 Cortex-M0+ 32bit Low Power ARM MCU
- Small and Compact Form Factor

SPI, I2C, Hardware Serial, and Analog allow for communication to the sensors in Table 3. The small form factor is useful for ease of handling and transport. The wireless radio (UBLOX SARA-R410M-02B) provides Narrow Band Communication capable of Cat M1 and NB1. Furthermore, the microcontroller (SAMD21 Cortex-M0+ 32bit Low Power ARM MCU) can support multiple tasks which are useful for the accelerometer (LIS3DH), which needs to be read a much higher frequency. This will be discussed later in the Software Section.

The O2 and CO2 sensors were chosen based on the operating temperature as they can handle -20 °C. However even though certain sensors such as the DHT11 do not work below 0 °C, they will still be sufficient as the most extreme case is –2 °C (as discussed in the temperature hardware design consideration section). The majority of food is stored above this extreme case

and above 0C, averaging around the 5 to 10 °C range for cold storage. For more extreme cases, such as pairs and horseradish (-2 °C to 0 °C and –1 °C to 0 °C respectively), better sensors will need to be used.

**Adafruit Ultimate GPS**

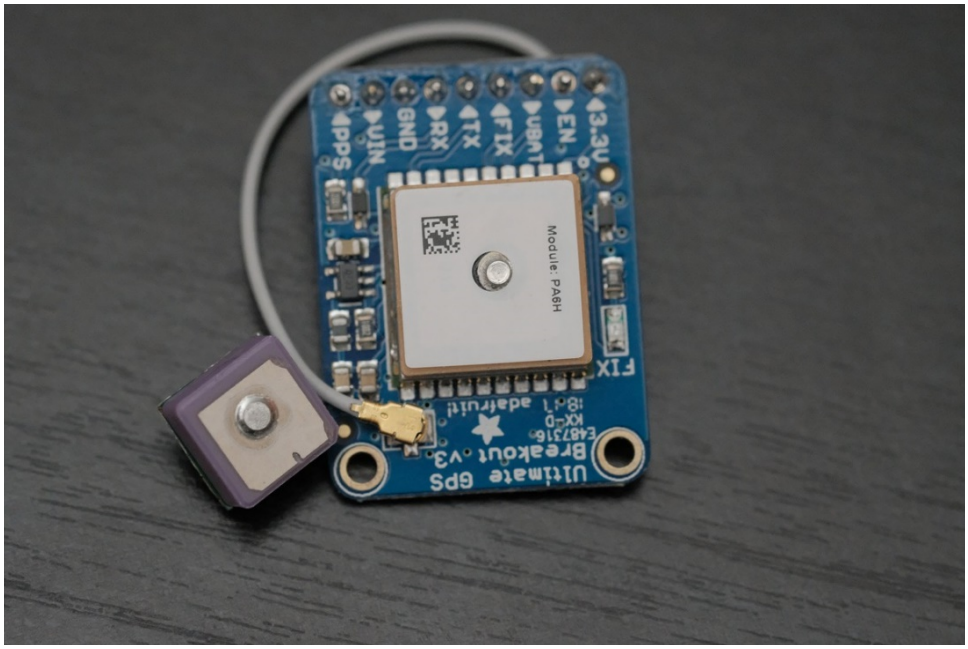



*Figure 9 Adafruit Ultimate GPS.*

The GPS module requires little setup and configuration. This make this GPS module ideal for fast prototyping and little to no maintenance. It is attached to a breakout board which provides a power and serial interface, which reduces the time to integrate this module into the main board.

**DHT11 (Temperature & Humidity)**

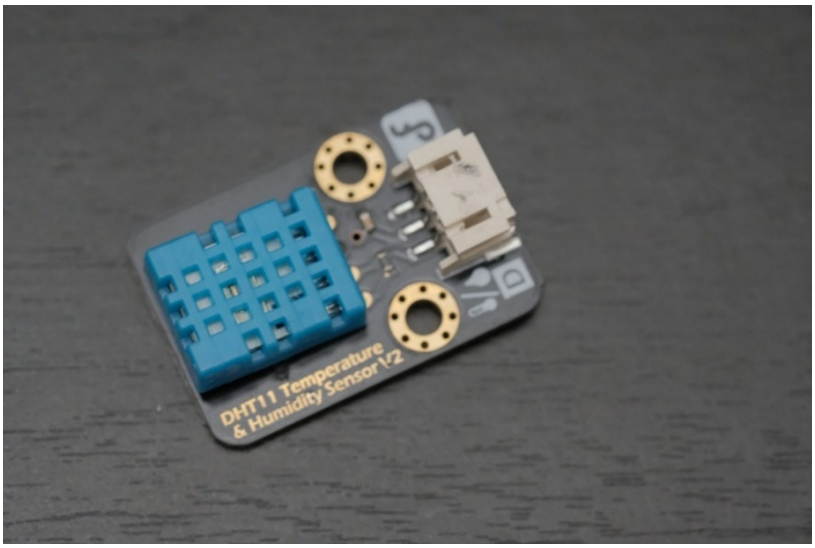



*Figure 10 Temperature and Humidity Sensor.*

Table 4. Summary of DHT11 Specifications [20]

| | |
|---|---|
| **Temperature Range** | 0 – 50°C ± 2°C |
| **Humidity Range** | 20 – 80% ~ 5% accuracy |
| **Sampling Rate** | 1 Hz |
| **Operating Voltage** | 3 – 5V |

Like the Adafruit Ultimate GPS, the DHT11 requires minimal setup. The temperature and humidity sensor are attached to a breakout board which provides a simple one wire interface. The temperature is measured in degrees Celsius, whilst humidity is measured in percentage. Temperature is measured using an NTC Temperature Sensor Thermistor, which increases in resistance as temperature rises. Similarly, humidity is measured using a moisture holding substrate that is sandwiched between two electrodes that have changing resistance depending on the amount of moisture. These changing resistances are interpreted as a percentage value.
The temperature and humidity range (Table 4) is not ideal for sub-zero conditions, and it cannot be sampling faster than once per second.

### LIS3DH (Accelerometer)

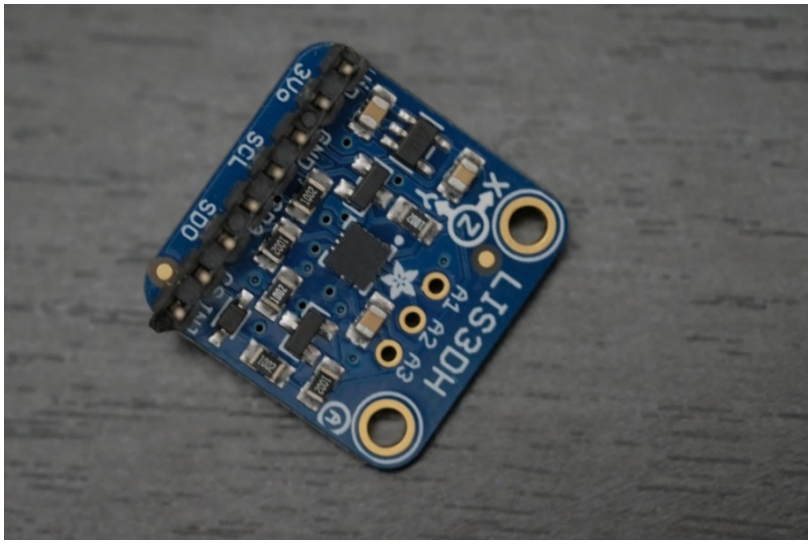



*Figure 11 Accelerometer*

Table 5. Summary of LIS3DH Specifications. [21]

| | |
|---|---|
| **Axis Sensing** | 3 with 10-bit precision |
| **Scaling Options** | ±2g / ±4g / ±8g / ±16g |
| **Sampling Rate** | 1 Hz – 5KHz |

| | |
|---|---|
| **Operating Voltage** | 3.3V regulated + level shifting |

The accelerometer provides 3 axis of acceleration measurements with 10-bit precision (Table 5). The accelerometer supports both I2C and SPI (Table 3), but I2C is used in this application. The accelerometer supports many scaling options. However, 2G is chosen by default. The sampling rate can be as fast as 5000 times per second.
Schematic Available as Appendix 3.

**MQ-131 (O2 Sensor)**

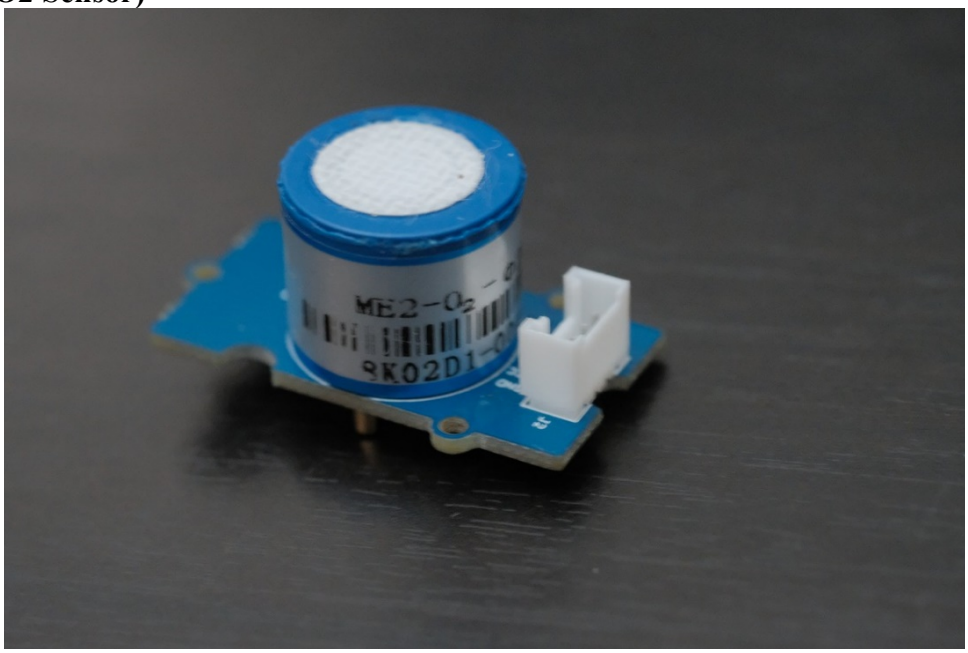



*Figure 12 O2 Sensor*

Table 6. Summary of MQ-131 Specifications.

| | |
|---|---|
| **Measurement Range** | 0 – 25% |
| **Operating Temperature Range** | -20 – 50°C |
| **Operating Voltage** | 3.3V / 5V |

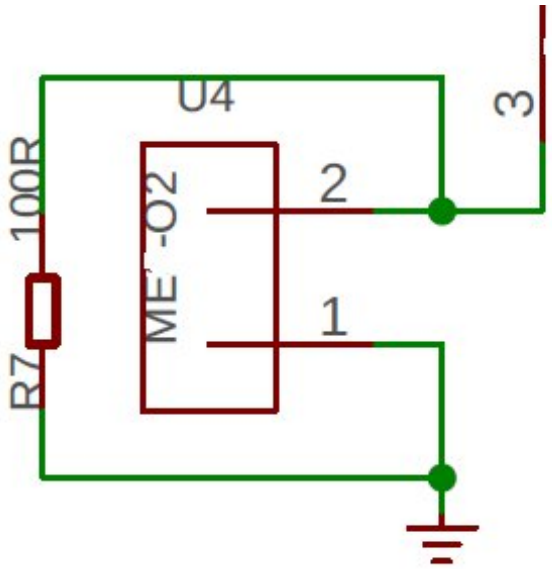



*Figure 13 Concentration measured as a current*

The O2 sensor measures O2 and outputs it as a voltage. This can then be converted to a percentage concentration.

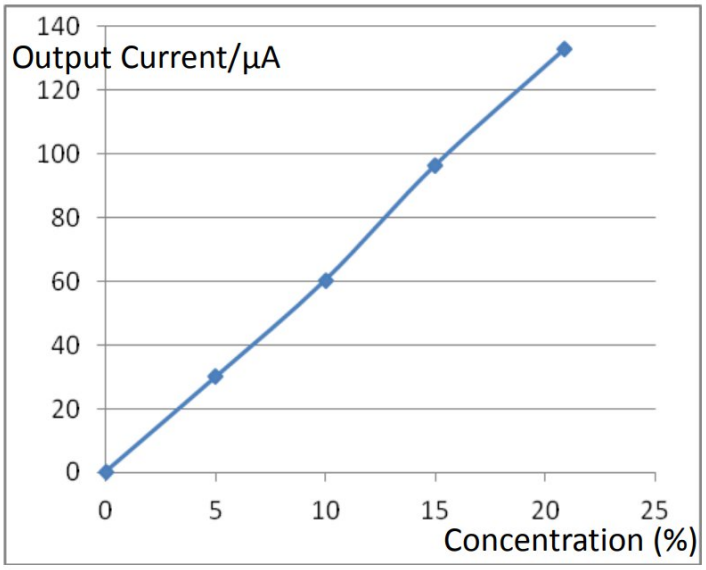



*Figure 14 Current vs Concentration of O2. [22]*

The O2 sensor measures O2 concentration in the air as current value ranging from 0.05 – 0.15mA. As current increases, so does concentration. The Arduino cannot measure current; thus, it must be converted to a voltage. This voltage can then be interpreted as a O2 concentration. Unlike the DHT11, the O2 sensor can operate in subzero conditions up to -20°C (Table 6). This makes this sensor suitable to be in transporting fresh produce conditions where temperatures can reach –2 degrees Celsius (most extreme case).

## MH-Z14A PWM NDIR (CO2 Sensor)


*Figure 15 CO2 Sensor*

Table 7. Summary of MH-Z14A Specifications.

| | |
|---|---|
| **CO2 Measurement Range** | 0 – 5000ppm ± (50ppm + 3% reading) |
| **Operating Temperature Range** | 0 - 50°C |
| **Operating Voltage** | 4.5 – 5.5V |

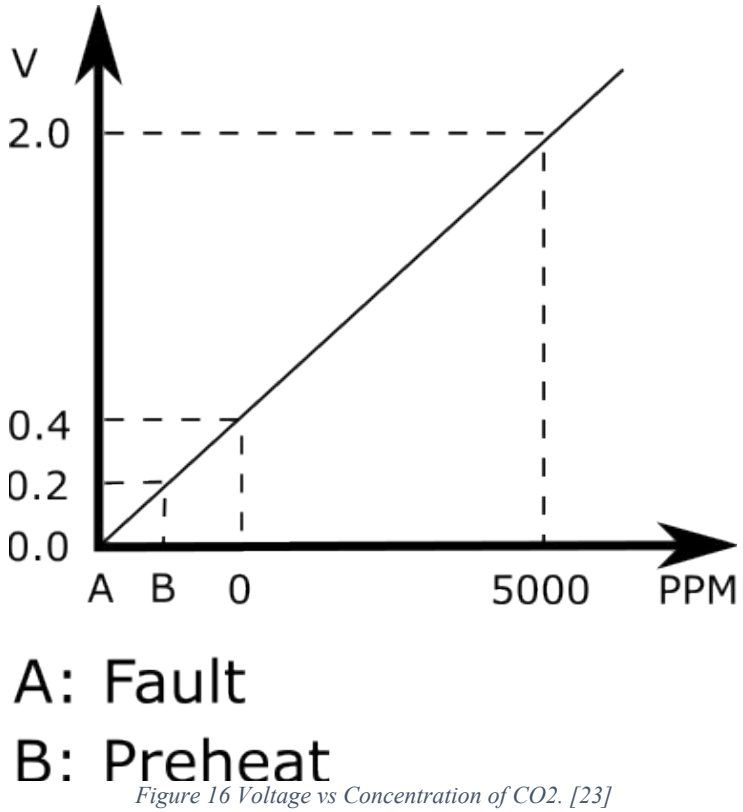



*Figure 16 Voltage vs Concentration of CO2. [23]*

The CO2 sensor also outputs the CO2 readings as a voltage, which can be converted to a concentration (and displayed as a percentage) (Figure 16). The operating temperature is from 0 to 50°C (Table 7), which is not ideal from some extreme transport conditions where the temperature can fall below 0°C (-2C for pairs is most extreme case). However, it will satisfy requirements for most fresh produce which lies around the 5 to 10-degree mark for cold storage.

**4000mAh LiPo**

The main board is unable to provide adequate voltage to radio module with multiple sensors attached. The LiPo battery serve a secondary source of power to allow the sensors and the radio module to be provided with adequate power. Power consumption and management testing was done for the module to run solely on battery. Further information can be referred to in the Numerical Results section.

**ESP 32 Mini Kit (Updated Microcontroller Board for Semester 2)**

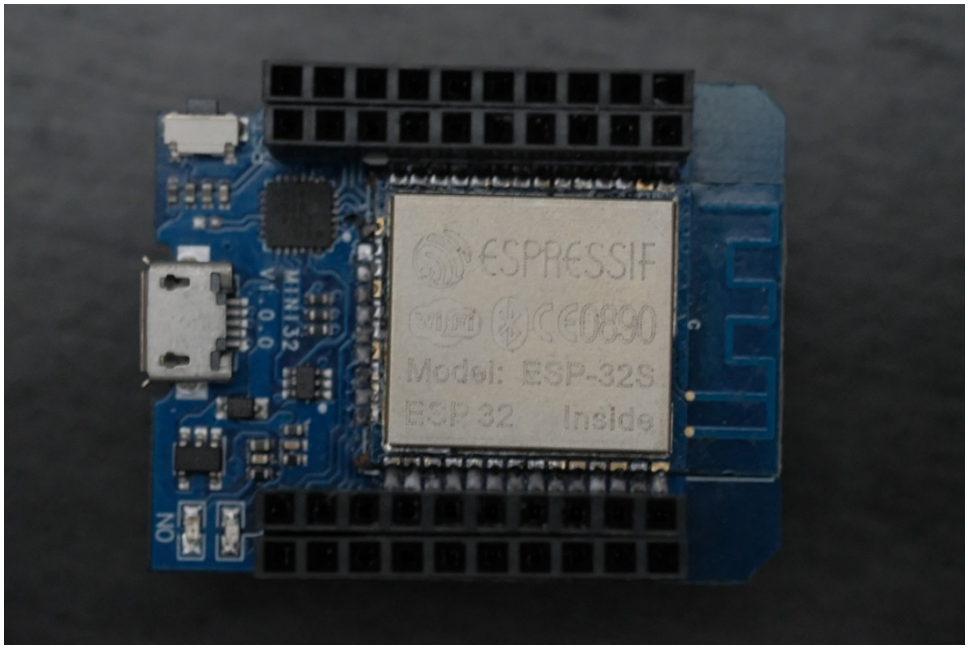



*Figure 17 ESP32 Board*

The ESP 32 Mini Kit provides a similar amount of IO and has a smaller footprint. Due to reliability issues with the UBLOX module of the Arduino MKR NB 1500, Wi-Fi is used instead. Wi-Fi however is not optimal for this IoT application but was used for testing/demonstration purposes. The ESP 32 features the following:

- ESP32 ESPRESSIF Chipset.
- Wi-Fi and Bluetooth.
- 12-bit SAR ADC (18 channels).
- 2 × 8-bit D/A converters.
- 10 × touch sensors.
- Temperature sensor.
- 4 × SPI, 2 × I2S, 2 × I2C, 3 × UART.
- 1 host (SD/eMMC/SDIO) and slave (SDIO/SPI).
- Ethernet MAC interface with dedicated DMA and IEEE 1588 support.
- CAN 2.0.
- IR (TX/RX).
- Motor PWM, LED PWM up to 16 channels.
- Hall sensor.
- Ultra-low power analog pre-amplifier.

Many of these features above are not used. But it provides similar advantages to the Arduino MKR NB 1500, and can support all the sensors by itself. It is important to note that the power supply of the sensors is separated from the ESP32.

## Software

The software consists of two parts; hardware output, and the dashboard. The hardware can be programmed by using Platform IO. Whilst the dashboard was programmed in Visual Studio Code. The dashboard will be explained in detail later.

**Main Board**
The software of the main board operates as follows:

1. Setup Serial.
2. Setup Sensors.
3. Schedule Secondary Loop.
4. Split Connection String.
5. Connect to NB-IOT Network.
6. Create SAS Token.
7. Set MQTT client and device ID.
8. Check of GPRS and NB Access status.
9. Connect to MQTT.
10. Read the accelerometer in separate loop.
11. Publish Message.
    a. Read sensors data.
    b. Format sensor data as JSON.
    c. Send message to IoTHub.

On a high-level view, the main board takes the readings of the sensors and sends it as a message to Azure IOT Hub via MQTT. Meanwhile, the dashboard listens to these messages that are sent to IoT Hub. An important operation step to note is the sensors and accelerometer should be read separately. The accelerometer must be read a much higher rate than other sensors to accurately capture the acceleration. Thus, a secondary scheduled loop must be used to separate the accelerometer from the main loop.

Further details can be found in the Code section located in the appendix.

- Data uploaded via 4G CatM1 using MQTT protocol due to its efficiency, although it is noted that the SIM card that came with the device has a daily limit of only 8000 MQTT messages.
- We made a dashboard as the main page for end users to be able to get vital information at a glance.

- Clicking into each graph gave additional information regarding the measured unit, plotting them over time to show its history.
- GPS mapping of each ping and connecting them with a solid colored line to see where data is logged when something happened.
- Later implemented a second page that provides tracker selection to accommodate more than one tracker.

**Old Dashboard**

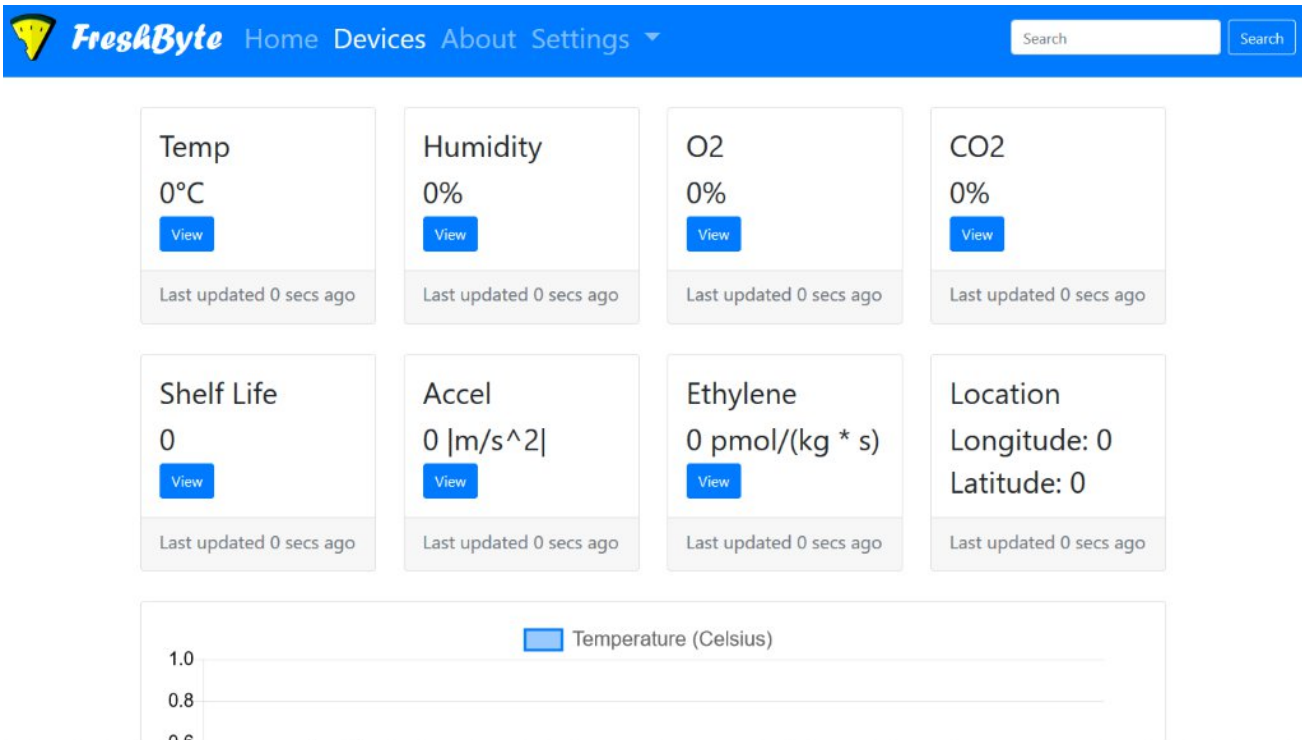



*Figure 18 Dashboard UI (Home Page).*

Above is a screenshot of our old website dashboard. It displays the sensors readings, a graph for each sensor as well as GPS coordinates. **The dashboard has been upgraded and restyled and can be referred to in the Discussions Section**. The updated dashboard can be viewed via this link here at: https://fresh-byte.azurewebsites.net.

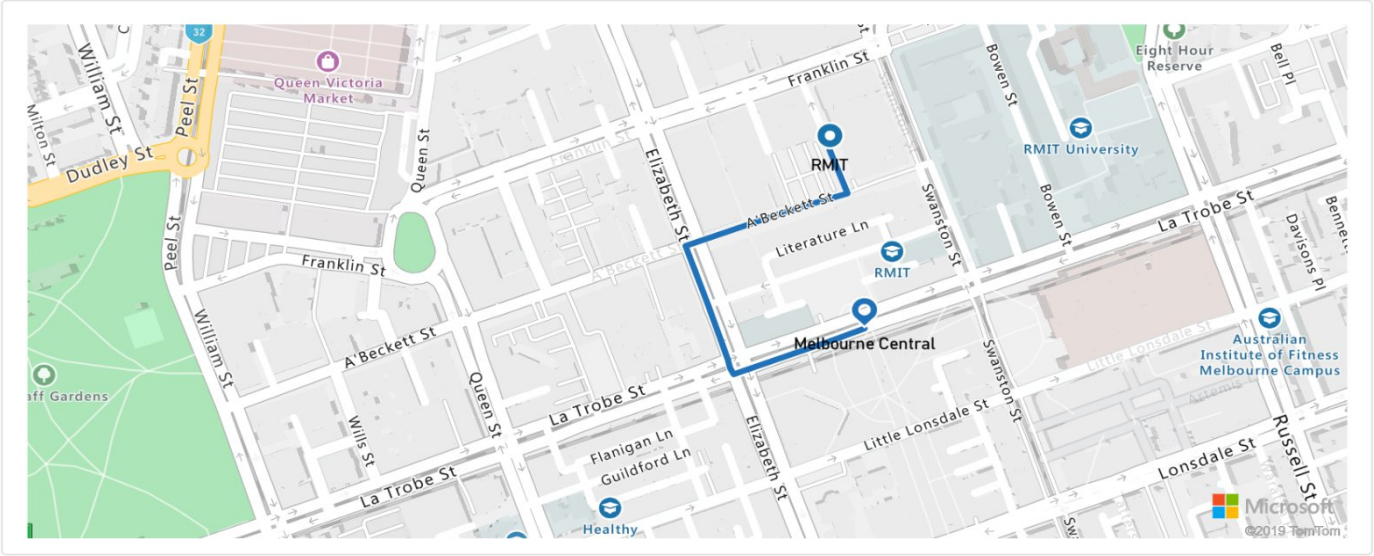



*Figure 19 Map Location Tracking (GPS).*

The dashboard UI requires Bootstrap CSS framework. This simplifies the process of designing a UI from the ground up. Each parameter, such as temperature and humidity can be viewed at the top of the dashboard (Figure 18). To view the historical data of each parameter, the view button can be pressed to change the graph above (Figure 18). Each parameter has a footer which shows when was the last value was received in seconds. The dashboard is a simple one-page design, that has the important information at the top to be easily accessed at a glance. The daily limit of MQTT messages is limited to 8000, because a free account is used. Thus, sending the sensor data must be limited.

Later implementation includes a more detailed Azure map, that maps each point with the relevant sensor measurements (Figure 19). A second page will be added, to provide further information of multiple devices. More interestingly, the shelf life and ethylene are estimated based on rough models. Due to budget constraints, an ethylene sensor could not be purchased.

# Findings

## Predicted Shelf Life

One of the primary features that sets the FreshByte module apart from other freight tracking systems, is it has a pair of gas sensors responsible for determining the shelf-life of foods. The way it does this is using the oxygen and carbon dioxide gas sensor values to first measure the gas concentrations of the package, and then using a shelf life model, it determines the shelf life of the specific food.

The shelf life model as a proof of concept (based on formulas/models from academic papers), predicts the estimated shelf life of the foods based on ethylene concentration [24].

When fresh produce begins ripening, they released a gas known as ethylene and this continually speeds up the ripening process until the food spoils. We can roughly deduce the volume of ethylene gas inside a package from the model used below, which takes the oxygen and carbon dioxide values to deduce the ethylene concentration. Then we can see on our dashboard when the ethylene spike is beginning to occur based on the calculations below.

Below is an example using pears, showing how we determine the ethylene concentration from our gas sensors and thus the expected shelf life of the food. This model is specifically being used for pears, with the ethylene concentration formulas coming from well-established academic papers [24].

As a proof of concept, the shelf life of the produce can be estimated. The following assumptions [25] must be made:

1. Atmospheric pressure is at sea level.
2. The produce is of typical human diet.
3. The produce values are based on pears.

To estimate ethylene, the following formula [24] is used:

$$V_{C_2H_4} = \frac{V_{C_2H_4\,max} \cdot p^e O_2^h}{K_{O_2\,max}^{C_2H_4} + p^e O_2^h},$$

$V_{C_2H_4}$: ethylene production rate (pmol $kg^{-1}\ s^{-1}$).
$V_{C_2H_4\,max}$: max ethylene production.
$p^e O_2^h$: external oxygen partial pressure.
$K_{O_2\,max}^{C_2H_4}$: ethylene production for half of oxygen partial pressure.
**h**: equal to 1.

Secondly, the partial pressure of O2 must be estimated based on the measured O2 and CO2 concentrations. Note that each additional estimation, introduce further inaccuracy. To estimate the partial pressure of O2, the following formula [25] is used:

$$\boldsymbol{p^e O_2^h = \left(P_{atm} - P_{H_2O}\right) FI_{O_2} - \frac{P_{CO_2}}{RQ}},$$

$$\boldsymbol{P_{atm}} = 760mmHg$$

$$\boldsymbol{P_{H_2O}} \approx 45mmHg$$

$$\boldsymbol{FI_{O_2}} = 0.21 + (0.0004 \cdot O_2\%)$$

$$\frac{\boldsymbol{P_{CO_2}}}{\boldsymbol{RQ}} = \frac{\frac{CO_2\%}{100}}{0.82} \cdot 760mmHg = 9.268 \cdot CO_2\%$$

With the above formulas, the estimated ethylene content can be calculated. Shelf life cannot be calculated or estimated without significant resources since fresh produce ripening is depended on the conditions [26] it is exposed to. However, shelf life can be associated with the amount of ethylene produced. Again, this is a proof of concept.

Assuming this is a pear (based on the papers on pears) we can predict the shelf life. To find shelf life the following formulas [26] [27] are used:

Projected Shelf Life of Pears:

$$\text{Gradient} = \frac{5\ days - 0\ days}{(0.3696 - 0.0528)}$$

$$\text{Number of Days} = \text{Gradient} \times ethylene\ production$$

Pears after harvest:

$$V_{C_2H_4} = \frac{0.14 \cdot p^e O_2^h}{0.99 + p^e O_2^h},$$

$$p^e O_2^h = 715(0.21 + (0.0004 \cdot O_2\%)) - (9.268 \cdot CO_2\%)$$

$$As\ \downarrow O_2\%\ and\ \uparrow CO_2\% \quad ,\uparrow V_{C_2H_4}.$$

$$Increases\ in\ V_{C_2H_4}\ can\ indicate\ ripening,$$

$$thus\ reduced\ shelf\ life.$$

An example using the model above but from avocado data is:

$$p_{o2} = 715 \times (0.21 + (0.0004 \times o_2)) - \left(9.26 \times \left(\frac{Co_2}{10000}\right)\right)$$

$$Ethylene = \frac{(0.14 \times p_{o2})}{(0.99 \times p_{o2})}$$

$$Days = 15.7829 \times ethylene$$

The following code was then used applied on the database to calculate the project shelf life:

```
float pO2 = 715 * (0.21 + (0.0004 * o2)) - (9.26 * (co2 / 10000));
float ethylene = (0.14 * pO2) / (0.99 * pO2);
float days = 15.7829 * ethylene;
```

These papers give the relationship between ripening of the food and time in days. Our gas sensors are specifically measuring both oxygen and carbon dioxide as our inputs, and from the formulas we can calculate the amount of ethylene gas from these inputs using the formulas above. From this we can deduce approximately when the food begins to ripen too quickly due to an ethylene production spike and thus when it is most likely to spoil.

We are aware this is a proof of concept as a lot more testing needs to be carried out on all foods to have a complete reliable model. But by focusing in on specific foods; whilst using the academic literature concerned with gas chemistry; we can get a rough estimation on the predicted shelf life of the produce based on the gas sensor readings.

## Numerical Results

During the testing phase of the prototype, the module was powered with 5V USB power supply. This allowed for long durations of testing to achieve the required functionalities. Once the baseline functions were achieved, the prototype had to be working off a battery as the ready version of the product on the market will be deployed on long trips, possibly multiple trips back-to-back. Therefore, a robust power management system is required.

An important factor to consider is power consumption. It determines how long the device can operate on a battery with limited capacity. To determine the power usage, the current consumption can be measured indirectly with a shunt resistor.

To begin measuring the voltage, a shunt resistor is placed between the positive terminal of the power supply and the FreshByte module (load). The probes of the oscilloscope are then attached to the terminals of the shunt resistor. In this case, a 0.22Ω resistor with a 5W rating is used. The setup is as follows:

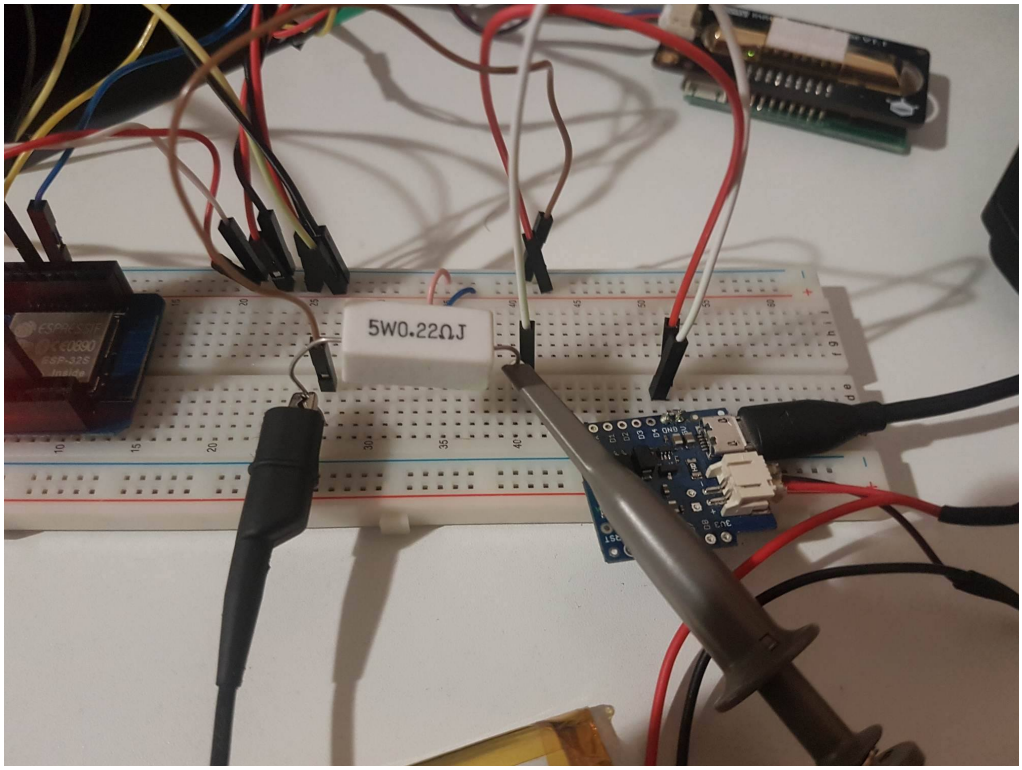





*Figure 20 Measuring Voltage through Sensor.*

The setup above can be simplified and represented as follows:

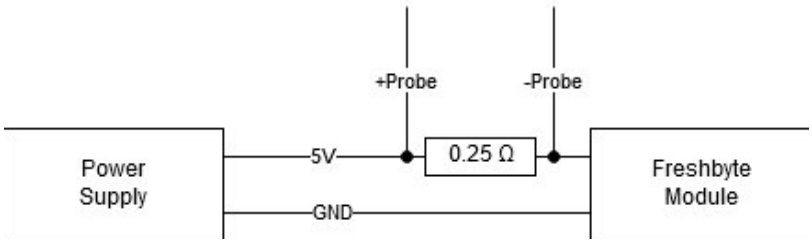



*Figure 21 Simple Diagram of Voltage Measuring.*

It is important that the resistor used is large enough so that there is enough voltage to the load whilst being small enough that it does not change the behavior of the load.

The current consumption of the load can now be measured on the oscilloscope.

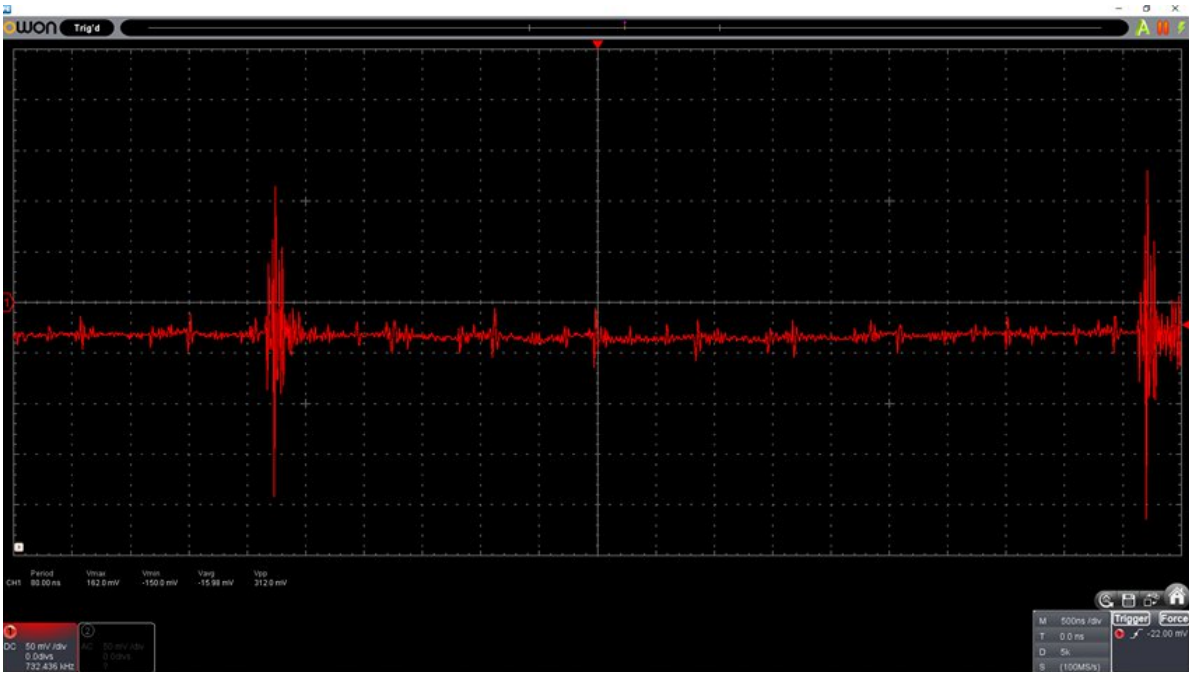


*Figure 22 Current Consumption.*

The maximum voltage measured across the shunt resistor is 162.0mV. Thus, the maximum current is $\frac{162.0mV}{0.22\Omega} = 736.36mA$ and the maximum power is $\frac{(162.0mV)^2}{0.22\Omega} = 119mW$. The current consumption is very high, but this is expected. There are two gas sensors that rely on heating elements, which require a high current in order to function properly.

If a battery with a capacity of 2000mAh is used, the expected battery life would be $\frac{2000mAh}{736.36mA} = 2.72hrs$. Since the purpose of the FreshByte module is to be shipped long distances over an extended period, this battery life is not enough. It is also important to note that these calculations are assuming that the device is continuously sending and reading measurements.

Realistically, the device will be reading and sending measurements every 30 minutes. If we assume that the device in deep sleep, it will draw a current of 0.15mA [28]. The battery life can be extended if the device reads and sends measurements every 30 minutes and then goes to a deep sleep state. If we also assume that the time to send and read measurements takes 5 minutes (gas sensors need to heat up to operating temperatures), then we can calculate the power consumption of the device for each day as follows:

$$Frequency\ of\ sending\ and\ reading\ measurements:$$

$$\frac{24\ hours\ \times\ 60\ mins}{30\ mins} = 48\ times\ per\ day.$$

$$Total\ time\ spent\ sending\ and\ reading:$$

$$48\ times\ per\ day\ \times\ 5\ mins\ =\ 240\ minutes.$$

$$Total\ time\ spent\ sleeping:$$
$$(24\ hours\ \times 60\ mins) - (240\ mins) = 1200\ mins.$$

$$Total\ power\ consumed\ per\ day:$$
$$\left(\frac{240\ mins}{60\ mins} \times 736.36mA\right) + \left(\frac{1200\ mins}{60\ mins} \times 0.15mA\right) = 2945.45\ mAh + 3\ mAh = 2948.45mAh.$$

Therefore, to read and send messages every 30 minutes with deep sleeping in between, a battery with a minimum of 2948.45mAh is required just for one day of operation. This is not ideal since the device is required to be compact and portable. The deep sleep mode added in can save a large amount of power and is suitable for short to medium trips. However, longer trips will draw a lot more power. A suggested recommendation for future improvements to combat this (longer trips) is explained more in detail in the recommendations section of this report.

## Projected Cost and Revenue (Real World - Business Case)

As the product was envisioned as a solution to the woes of the supply chain logistics industry, rough projections of costs and revenue involved in creating such hardware was taken into account to further study the feasibility of our project in the real world.

With preliminary information regarding the 618 million dollars spent on 150000 horticulture road freight shipments annually [29], we estimate each trip, including the vehicle's service life and fuel costs to be about 3000 dollars. With that being set as a baseline, we believe our product to be financially viable when adopting the subscription model [30].

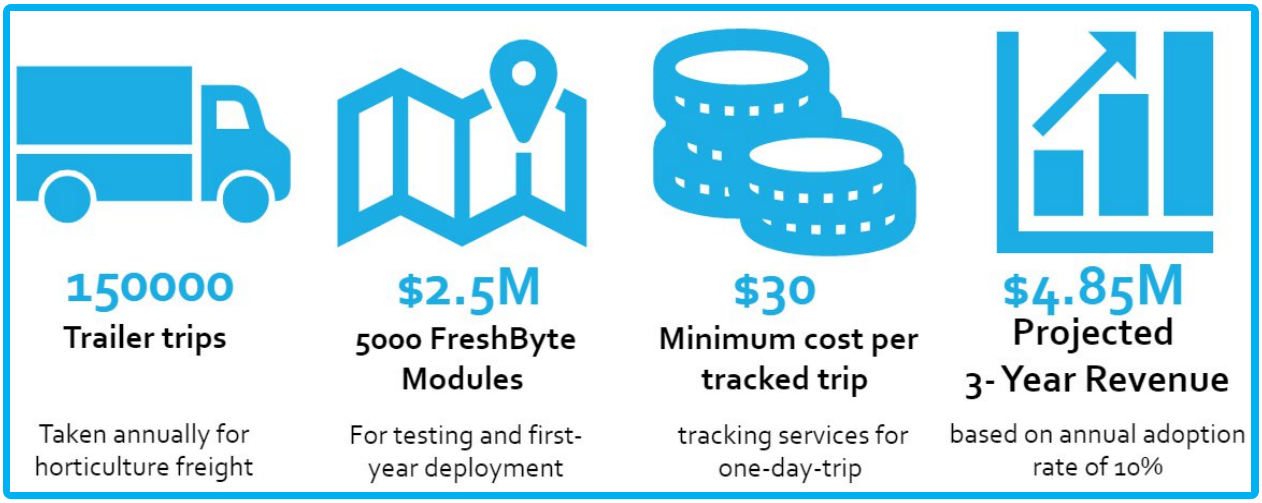



*Figure 23 Projected Costs and Revenue Prepared for Telstra Innovation Challenge.*

Our calculations assumed the following instances [31]:

- Based on online bulk ordering quotes, the hardware cost per module is generously estimated between 187 and 243 Australian dollars, with a 20 percent overhead included for potential lost or failing modules.
- After a limited test run on a small sample of modules, about 5000 modules will be injected into the market each year for two years, totaling more than 10000 modules by the third year, estimated to cost about 2.5 million dollars. The hardware costs will be fully covered by Raven.
- Raven will be potentially charging as low as 30 dollars per single-day tracking, on top of access to the online monitoring interface. This equates to a very reasonable one percent of the freight cost - all while providing a value package to our clients, from product location tracking to shelf life prediction.
- With the expenses and costs laid out, as well as assuming an adoption rate of 10% (one out of 10 shipments) per annum, we deduce that FreshByte will be able to break even and achieve a cumulative 3-year revenue of 4.85 million Australian Dollars.

# Discussions of Results/Products

In the field of IoT, battery life and power consumption are important factors which we have tried to address in this capstone project. Thus, our latency and performance are the tradeoffs we have chosen to sacrifice. If we consider that the components of the FreshByte module are off-the-shelf and are designed to be more general purpose and appeal to a more general audience, the results are of an unoptimized product. But even in the context of freight tracking and with the factors discussed above, the FreshByte module continues to meet its goals of freight tracking and data transparency. Ultimately, our product performance and results are satisfactory to our expectations and barring a few potential further refinements on code efficiency and marketing strategy. We believe the FreshByte module will require further improvements in battery and sensor technology before the idea of real-time data transparency can be fully realized.

## Updates and Improvements

The software upgrades for the dashboard is also an important improvement that was made. The addition of colour-coded sensor reading values with a more user-friendly user interface is a large improvement from the uninspiring dashboard created in semester 1.

**Updated Dashboard**

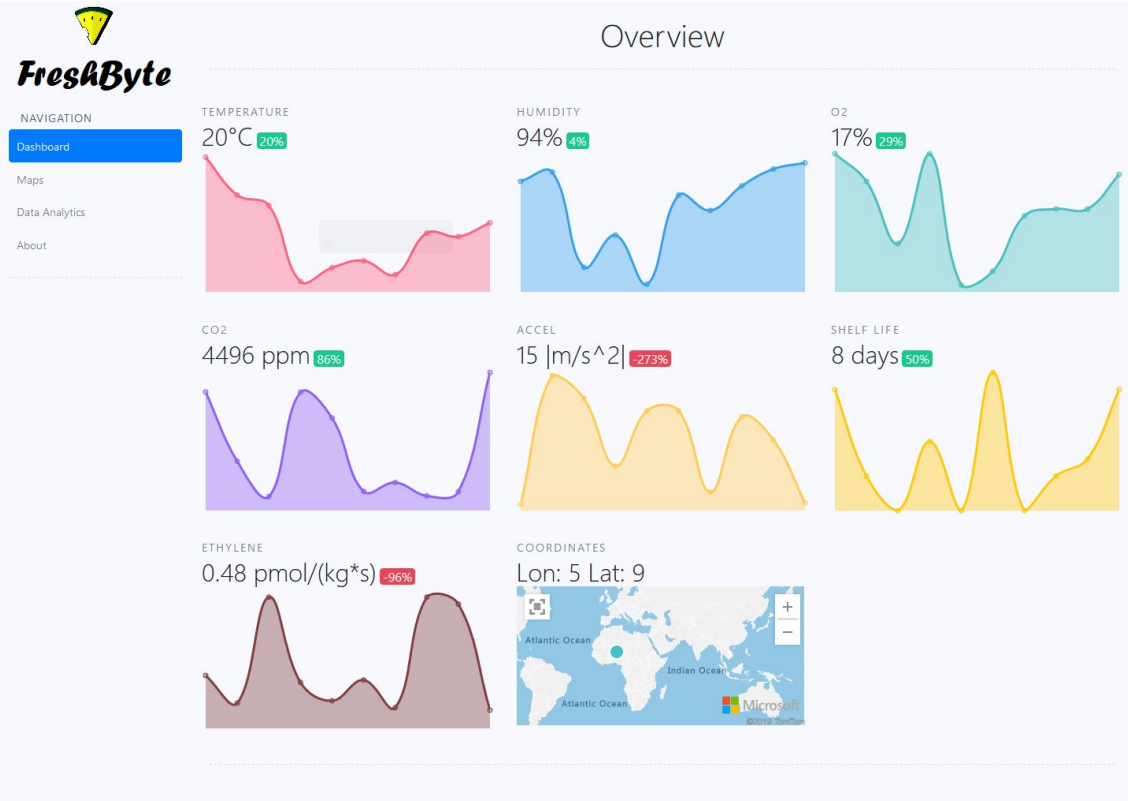



*Figure 24 Updated FreshByte Dashboard.*

The navigation panel was designed to be easy to use with the dashboard describing the sensor values, the maps section having the GPS coordinates of the map on full screen and a data analytics section (figure 25) which shows the data logging values coming from the database.

The dashboard section has temperature in degrees, humidity displayed as a percentage, oxygen concentration displayed as a percentage, carbon dioxide displayed as a percentage, acceleration displayed in metres per second squared, shelf life in days, ethylene concentration in pmol/kg*s and GPS coordinates of the freight.

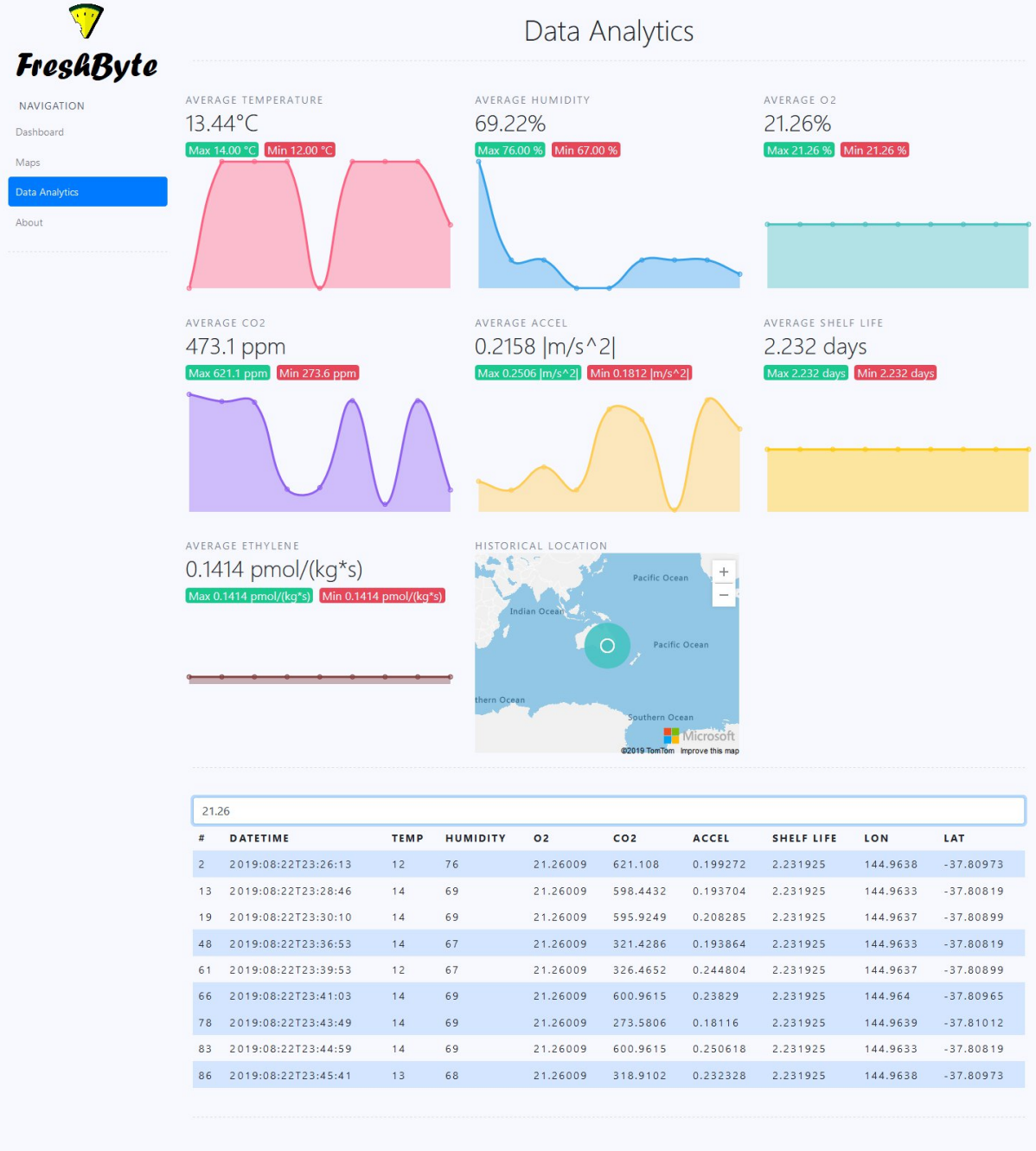



*Figure 25 Raw Data Collected Displayed on Dashboard.*

The updated version of the dashboard allows for easy monitoring of data at one glance. The source code of the new dashboard can be viewed here (https://github.com/HuyNguyenAu/FreshByteDashboard).

We have also added a search functionality to the database where any searched input will list everything associated with it. For example, searching the number 3 will display any result with a 3 in it.

The database also implements the data sent to it through data logging functionality and can backlog data even if network connection is lost. The device attempts to send data every 15 seconds and can locally backlog data at every time interval. It can then pass the data back to the cloud whenever reception is successful. These capabilities can also have the potential to explore power saving mechanisms. For example, it is possible to set the device so that it is not sending data at every time interval but maybe at multiples of the time interval, further reducing power consumption.

### Hardware Appearance

The figures below detail the adjustments to the visual appearance of the hardware itself. The hardware has been packaged up to be much more robust in terms of practical use. The prototypes used in semester 1 were used to test the functionality of the product and once this was completed steps were taken to remove wires from hanging out of the hardware visually. They consisted of a breadboard with multiple connecting wires for testing purposes. Figure 29 displays the finalized FreshByte module product.

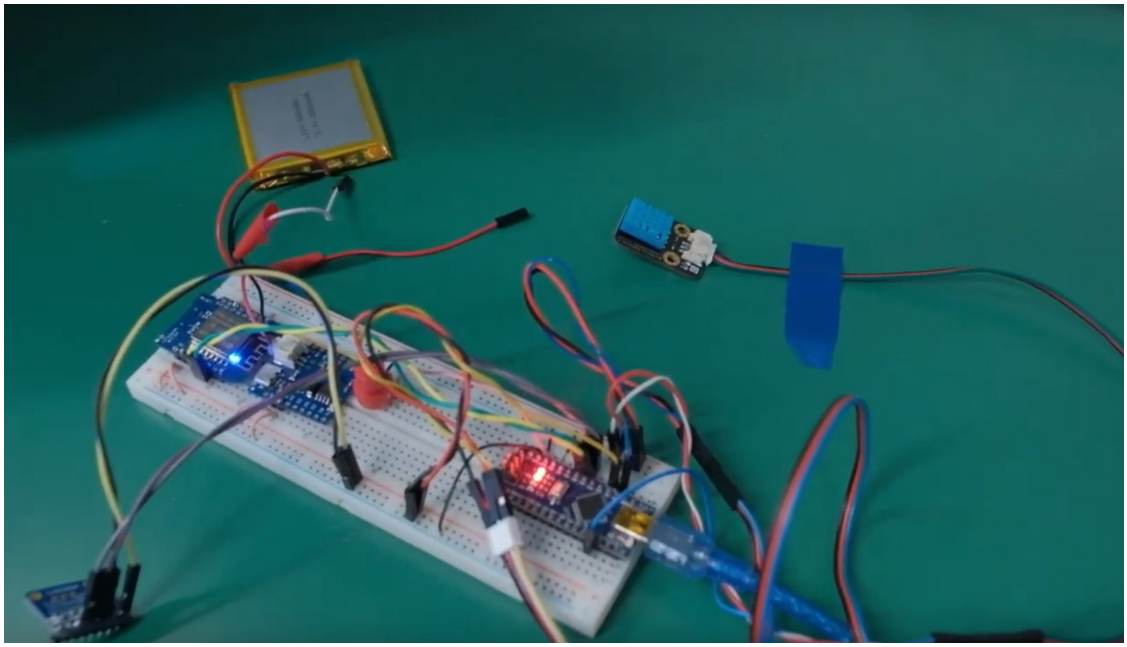


*Figure 27 Semester 1 Hardware - Prototype*

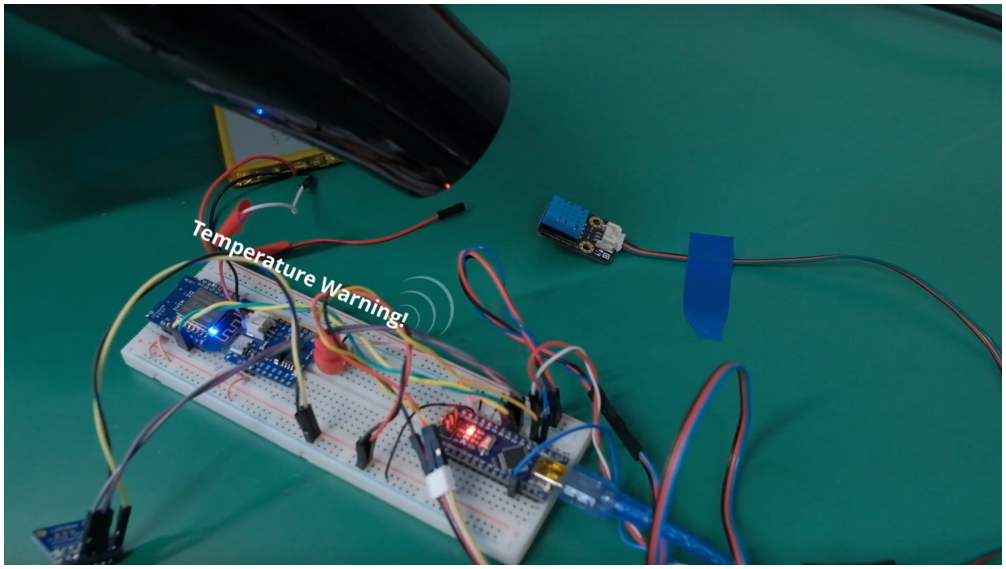



*Figure 28 Semester 1 Hardware Prototype with Hairdryer Temperature Test*

Below is the latest revision of the FrehsByte module which contains all sensors packaged up into the rectangular case shown below.

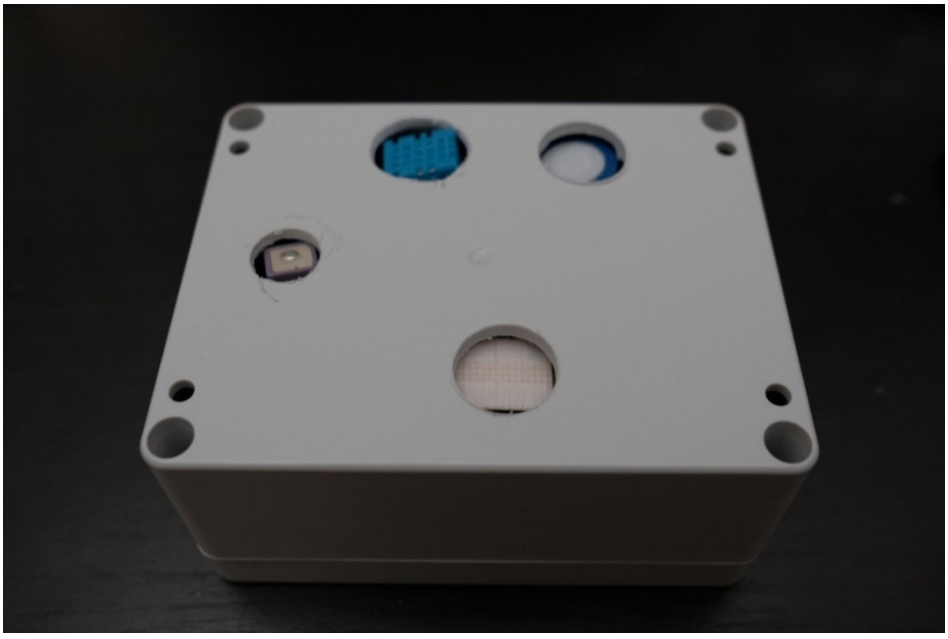


*Figure 29 FreshByte Module*

# Conclusions

To summarize the team's performance for the year, we believe we have achieved the primary objectives we set out at the start of the year, which was to have an autonomous IoT tracking device which provides freight tracking and data transparency.

In achieving these objectives there were milestones and targets we needed to hit such as having full functioning hardware and a data visualization platform for the dashboard. In addition, an important milestone we wanted to achieve was to implement the predicted shelf life feature which we were able to incorporate into the project. This feature manages to set our project apart from current freight tracking systems out there in the market.

The weaknesses of the project include overcoming RF blocking in metal trucks, overcoming temperature cellular modem as well as battery and power management. For future revisions and to enhance the project further these factors need to be addressed. In addition, future implementations could include more advanced inclusions such as the presence of machine learning, artificial intelligence and deep learning. This would allow for a device to be even more removed from human interaction, whereby the device can learn to calibrate on its own and improve over time based on a given data set.

# Recommendations

### Overcoming RF Blocking in Metal Trucks

One of the problems, that we were aware of was packages being transported by metal trucks specifically. Our standalone device transmits using cellular IoT networks to the cloud but transmitting outside of metal or even metal packages poses an obstacle. To overcome this in future implementations, a wireless gateway (WI-FI) could be added on top of the transport truck for example, and the device can transmit to this device (gateway) with a stronger signal that can penetrate more. The gateway can then continue to use the IoT cellular network. Current trucks, don't have a large amount of metal which could degrade the RF signal greatly, but if there was to be a large amount of metal or materials that significantly distort the signal, then extra signal boosters such as repeaters or access points may have to be used.

### Overcoming Temperamental Cellular Modem

As mentioned, after identifying that the Ublox SARA-R410M cellular modem on the Arduino MKR NB1500 was very temperamental, the board was replaced with an ESP32 Wi-Fi module. This is to mitigate the risk of hardware failure allowing for the testing of other functionalities of the system as well as ensuring proper functionality during Engenius. The MKR NB1500 is still a relatively new product on the market and not much support is available for the board. The MKR NB1500 was provided by Telstra to be used for the Telstra Innovation Challenge. Therefore, other more stable hardware options can and will be explored to be integrated into our product in the future allowing for a more robust system. Ublox produces many reliable IoT band modems such as the SARA-R4, hence building custom board with a Ublox modem also becomes an option in the future.

### Battery and Power Management

A 4000 mAh battery was proposed to be paired with the final device, but even a battery of that size would only be adequate for a little more than one day of battery life. As the power-hungry gas sensors play a critical and central role to FreshByte's functionality, we were not able to simply remove it from the design to save power – which narrows future suggestions down to either pairing it with bigger batteries or to reduce the transmission frequency, each with their accompanying limitations and drawbacks as well. A bigger battery would lead to the increase in size of the unit, as well as a significant increase in the severity of a potential fire hazard. On the other hand, although reducing the frequency of data transmission will remove the "real-time" aspect of the system, the idea would appear as the most sensible option before sensor and battery technology is to be further improved in the future.

# Future Works

As we move into the next technological phase, IoT becomes more of an important topic. The fourth industrial revolution, also known as Industry 4.0, is affecting almost every industry worldwide and is rapidly transforming how businesses operate [32]. Specifically, the fourth industrial revolution is becoming much more relevant with the demands for lower latency and longer lasting devices is increasingly common in device requirements. In addition, the fourth industrial revolution is concerned with fields that are not normally categorized within industry, such as smart cities, home automation and driverless vehicles.

Listed below are the current trends from research conducted by the Australian Government's Department of Industry, Innovation and Science. These include:

- advanced automation and robotics (including collaborative robots or 'cobots')
- machine-to-machine and human-to-machine communication
- artificial intelligence and machine learning
- sensor technology and data analytics

Four key drivers enable this trend:

- rising data volumes, computational power and connectivity
- emerging analytics and business-intelligence capabilities
- new forms of human-machine interaction, such as touch interfaces, augmented and virtual reality systems
- improvements in transferring digital instructions to the physical world, such as robotics and 3D printing.

[32].

Considering the factors above shows that there is a lot of room for improvement for our capstone project. Particularly, a tracking system which can involve artificial intelligence (AI) or machine learning would be a major enhancement to our project that could set the industry standard for freight tracking.

There is a lot of opportunity for our project to incorporate some form of AI or machine learning which where the autonomous device learns to improve or calibrate itself to different situations.

For example, if a box of Bananas is to become blemished due to temperature abuse at certain distances or journey times, then the device can probe the transport truck's cooling system to increase its output (reduce the truck's temperature).

This example can go for any other mechanisms used in transport trucks as well as when food is being stored in transit warehouses. The device simply just needs to be hooked up to the functioning mechanism and be able to reliably learn from its data set (machine learning).

In addition, another example could be if you have a crate of apples moving from Brisbane to Melbourne and the device can figure out that the food is repining too quickly. If it knows that it likely to be spoiled when it comes to Melbourne; then it can autonomously on its own send a warning to the driver to direct it to a closer destination instead such as Sydney.

This would almost take human to device interactions out of the equation which could free up workers to focus more on fault tolerance and recovery, rather than constantly observing the freight.

With the future already looking to welcome IoT products, our project fits within the current framework. However, adding the mechanisms discussed above will take some time to adapt to. The workforce adjusting itself to reduce the amount of human to device interaction will take some time. This will be one of the challenges we face as we move through the Industry 4.0 revolution, where the presence of AI, machine learning and autonomous devices are very prevalent. This will not only change the current working landscape, but also increase performance requirements such as low latency and devices which can support longer ranges.

# References


[1] Environment.gov.au. (2017). *National Food Waste Strategy*. [online] Available at: http://www.environment.gov.au/system/files/resources/4683826b-5d9f-4e65-9344-a900060915b1/files/national-food-waste-strategy.pdf [Accessed 14 Oct. 2019].

[2] wearethenest.com.au, T. (2019). *Food Waste Facts - OzHarvest*. [online] OzHarvest. Available at: https://www.ozharvest.org/what-we-do/environment-facts/ [Accessed 14 Oct. 2019].

[3] Agriculture, "Protfoilio-facts for Horticulture: [Online]. Available: http://www.agriculture.gov.au/about/commitment/portfolio-facts/horticulture

[3] Champions123.org. (n.d.). *SDG Target Progress Report*. [online] Available at: https://champions123.org/wp-content/uploads/2017/09/champions-123-sdg-target-123-2017-progress-report.pdf [Accessed 14 Oct. 2019].

[4] Kummu, M., de Moel, H., Porkka, M., Siebert, S., Varis, O. and Ward, P. (2012). Lost food, wasted resources: Global food supply chain losses and their impacts on freshwater, cropland, and fertiliser use. *Science of The Total Environment*, 438, pp.477-489.

[5] Environment.gov.au. (n.d.). *National Food Waste Baseline Final Assessment*. [online] Available at: https://www.environment.gov.au/system/files/pages/25e36a8c-3a9c-487c-a9cb-66ec15ba61d0/files/national-food-waste-baseline-final-assessment.pdf [Accessed 14 Oct. 2019].

[6] Fao.org. (n.d.). *Global Food Losses and Food Waste - Extent, Causes and Prevention*. [online] Available at: http://www.fao.org/3/mb060e/mb060e01.pdf [Accessed 14 Oct. 2019].

[7] Bcg.com. (2018). *TACKLING THE 1.6-BILLION-TON FOOD LOSS AND WASTE CRISIS*. [online] Available at: https://www.bcg.com/en-au/publications/2018/tackling-1.6-billion-ton-food-loss-and-waste-crisis.aspx [Accessed 14 Oct. 2019].

[8] Food Logistics. (2019). *Tracking The Future Of Food Logistics*. [online] Available at: https://www.foodlogistics.com/3pl-4pl/article/12144413/tracking-the-future-of-food-logistics [Accessed 14 Oct. 2019]. [9] Barcoding.com, "Package Tracking" [Online]. Available: https://www.barcoding.com/technology/software/package-tracking/

[10] Hanhaa Ltd, "Parcelive" [Online]. Available: https://hanhaa.com/parcelive/

[11] SupplyChain247.com, "Warning Signs of Asset Mismanagement, Supply Chain Technologies" [Online]. Available: https://www.supplychain247.com/article/5_warning_signs_of_asset_mismanagement/apex_supply_chain_technologies

[12] Paper on Avocadoes: [Online]. Available: http://www.plantphysiol.org/content/plantphysiol/53/1/45.full.pdf

[13] Advanced Technology Services, "Products" [Online]. Available: https://www.advantech.com/products/a4ec3c08-0bb5-4e12-af9c-d0c45d8661bb/trek-120/mod_b7cafa00-7027-4b0a-bb40-9bb023bab735

[14] Linfox, "Refrigerated Transport Freight" [Online]. Available: https://www.linfox.com/transport-freight/refrigerated/

[15] Department of the Environment, Australian Government, “National Food Waste Strategy [Online]. Available: https://www.environment.gov.au/protection/waste-resource-recovery/publications/national-food-waste-strategy

[16] Department of the Environment, Australian Government, “National Food Waste Baseline Executive Summary” [Online]. Available: http://www.environment.gov.au/system/files/pages/25e36a8c-3a9c-487c-a9cb-66ec15ba61d0/files/national-food-waste-baseline-executive-summary.pdf

[17] Agric.wa.gov.au. (2019). *Storage of fresh fruit and vegetables | Agriculture and Food*. [online] Available at: https://www.agric.wa.gov.au/fruit/storage-fresh-fruit-and-vegetables?page=0%2C1 [Accessed 14 Oct. 2019].

[18] GMBPower ,“li-ion Polymer Battery Specification”, Available at: https://www.powerstream.com/lip/GMB042035.pdf [Accessed 14 Oct. 2019].

[19] Espressif.com. (2019). *ESP32 Overview | Espressif Systems*. [online] Available at: https://www.espressif.com/en/products/hardware/esp32/overview [Accessed 14 Oct. 2019].

[20] Industries, A. (2019). *DHT11 basic temperature-humidity sensor + extras*. [online] Adafruit.com. Available at: https://www.adafruit.com/product/386 [Accessed 14 Oct. 2019].

[21] Industries, A. (2019). *Adafruit LIS3DH Triple-Axis Accelerometer (+-2g/4g/8g/16g)*. [online] Adafruit.com. Available at: https://www.adafruit.com/product/2809 [Accessed 14 Oct. 2019]. \

[22] Wiki.seeedstudio.com. (2019). *Grove - Gas Sensor($O_2$) - Seeed Wiki*. [online] Available at: http://wiki.seeedstudio.com/Grove-Gas_Sensor-O2/ [Accessed 14 Oct. 2019].

[23] Wiki.dfrobot.com. (2019). *Gravity__Analog_Infrared_CO2_Sensor_For_Arduino_SKU__SEN0219-DFRobot*. [online] Available at: https://wiki.dfrobot.com/Gravity__Analog_Infrared_CO2_Sensor_For_Arduino_SKU__SEN0219 [Accessed 14 Oct. 2019].

[24] Journal of Experimental Botany, “Carbon dioxide action on Ethylene Biosynthesis of Preclimacteric and Climacteric Pear Fruit” [Online]. Available: https://academic.oup.com/jxb/article/54/387/1537/540315

[25] Sandeep Sharma; Deepa Rawat, “Partial Pressure of Oxygen (PO2)”, Baptist Regional Medical Center, February 23, 2019. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK493219/

[26] Mikal E. Saltveit, University of California, Davis 95616, “Measuring Respiration”. [Online]. Available: http://ucce.ucdavis.edu/files/datastore/234-20.pdf

[27] Paper on Ethylene production: [Online]. Available: https://www.researchgate.net/publication/316169265_Respiration_rate_and_ethylene_metabolism_of_'Jonagold'_apple_and_'Conference'_pear_under_regular_air_and_controlled_atmosphere

[28] "Insight Into ESP32 Sleep Modes & Their Power Consumption", Last Minute Engineers, 2019. [Online]. Available: https://lastminuteengineers.com/esp32-sleep-modes-power-consumption/. [Accessed: 13- Oct- 2019]

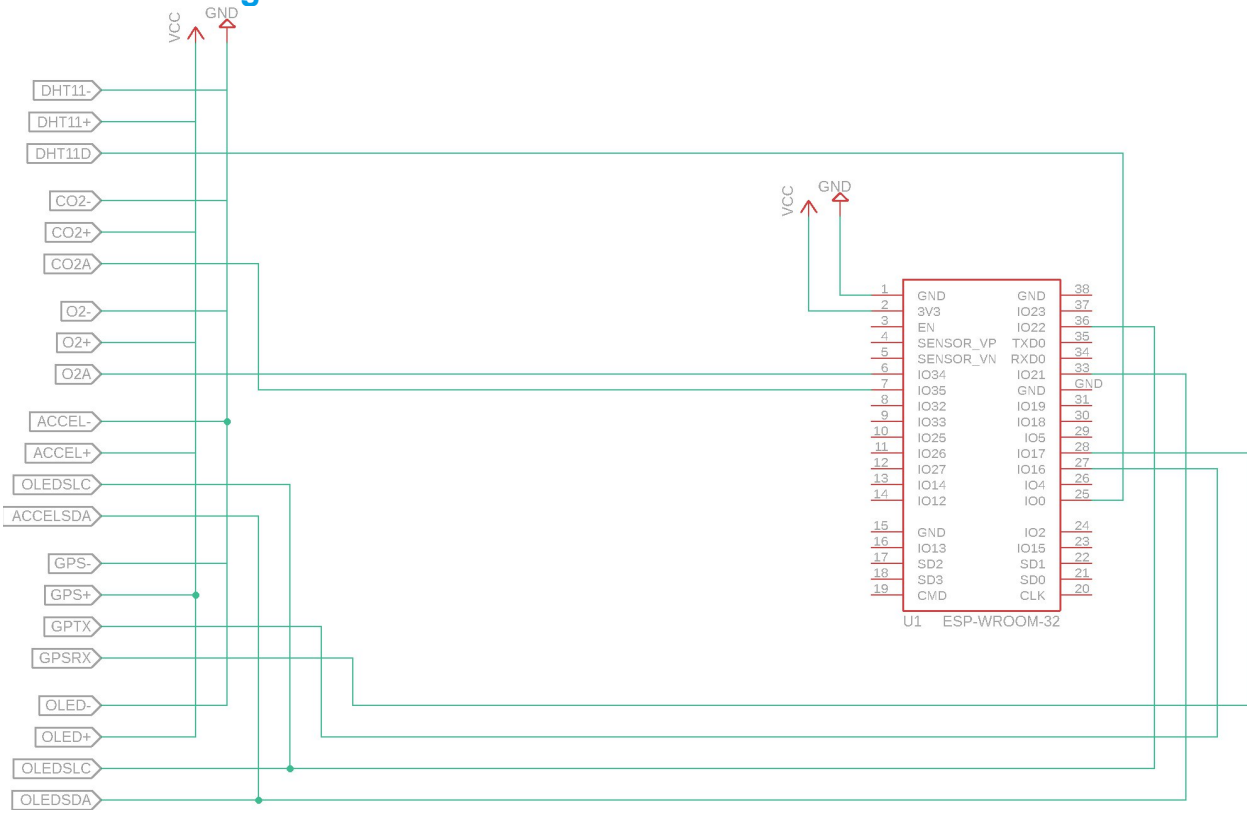
[29] Department of Agriculture and Water Resources, Australian Government, “Transit Agriculture Final Report” [Online]. Available: http://www.agriculture.gov.au/SiteCollectionDocuments/ag-food/publications/transit-agriculture-final-report.pdf

[30] Stanford University, “Why Every Business Will Soon Be a Subscription Business” [Online]. Available: https://www.gsb.stanford.edu/insights/why-every-business-will-soon-be-subscription-business
[31] – FreshByte Costs in Business Model. [Online]. Available: https://rmiteduau-my.sharepoint.com/:x:/g/personal/s3457782_student_rmit_edu_au/EcAPtoKftEJDrWTOQVMP24UB8dZ7PKSGQ0W1x2U8emmj3w?e=2Qjdym

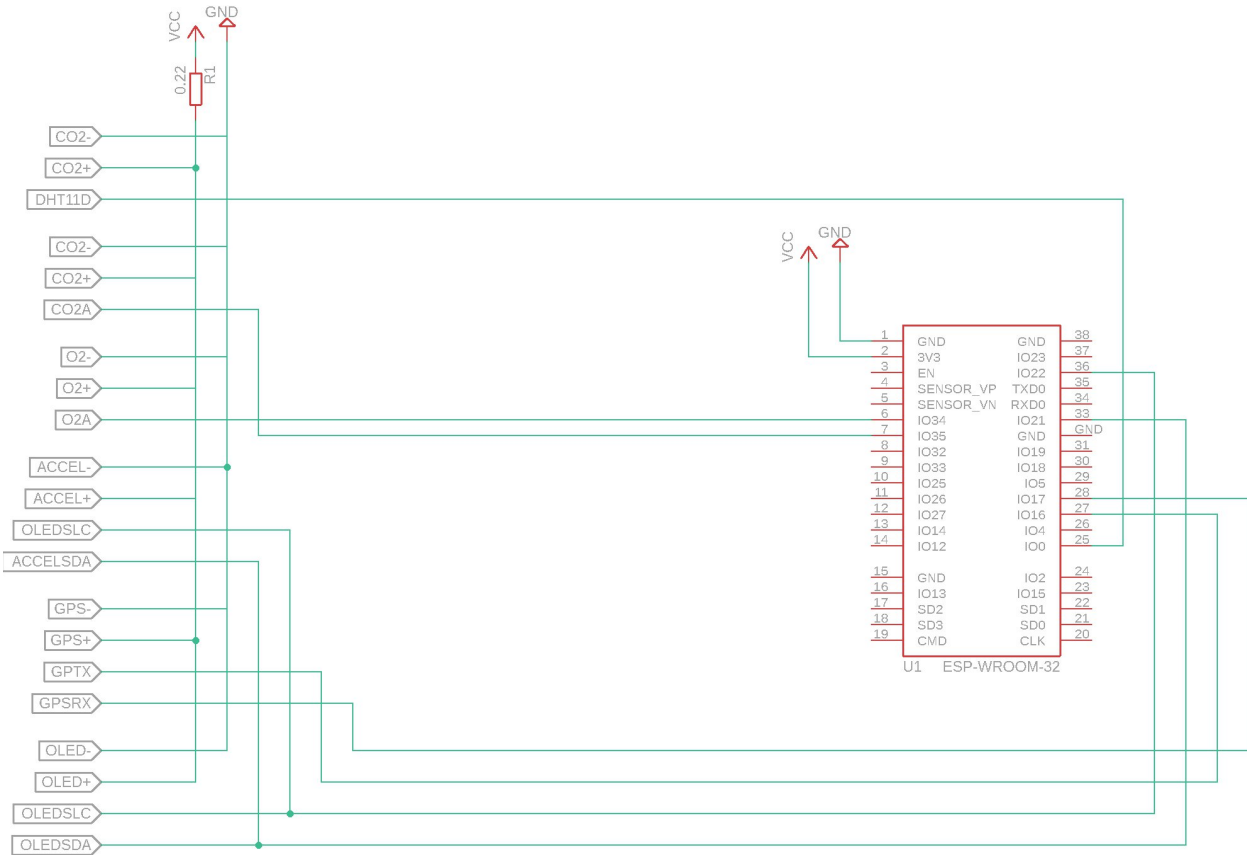
[32] Department of Industry, Innovation and Science. (2019). *Industry 4.0*. [online] Available at: https://www.industry.gov.au/funding-and-incentives/manufacturing/industry-40 [Accessed 14 Oct. 2019].

# Appendix

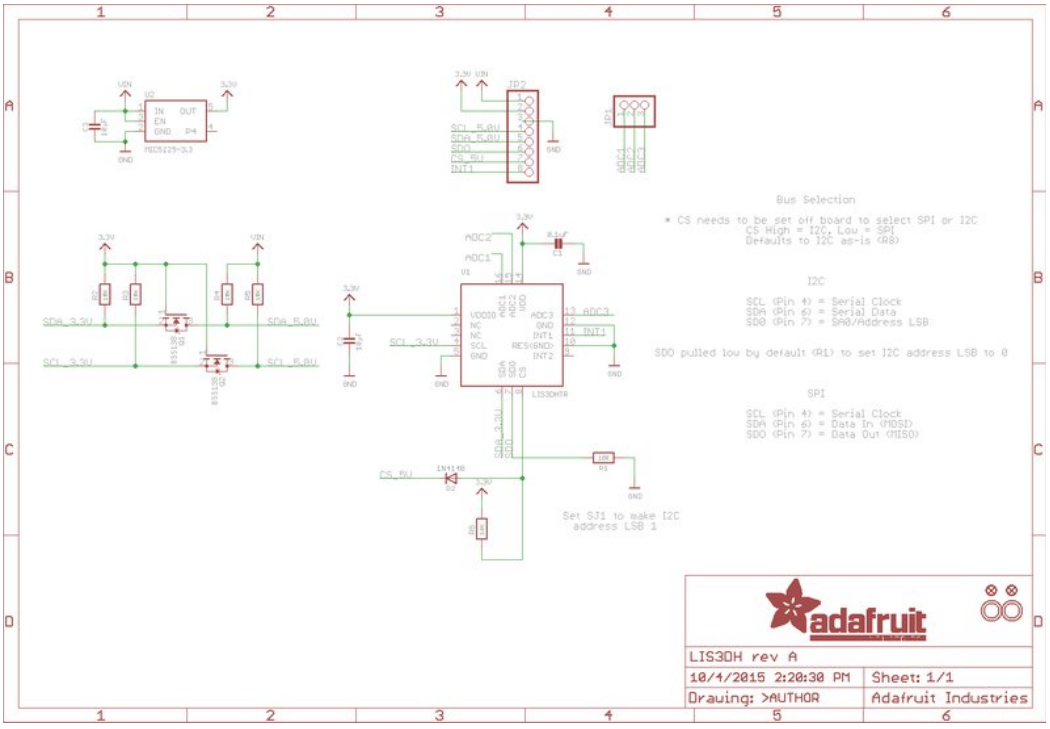
## Schematic Diagram




*Appendix 1 FreshByte Hardware Schematic*




*Appendix 2 FreshByte Hardware Schematic with Test Resistor R1*




*Appendix 3 LIS3DH Schematic*



FreshByte

RAVENTL

## Code

**Code:**
**Accel.ino**

```
#include <Wire.h>
#include <Adafruit_LIS3DH.h>
#include <Adafruit_Sensor.h>

#define LIS3DH_CS 10

Adafruit_LIS3DH lis = Adafruit_LIS3DH();

void setupAccel()
{
  Serial.println("LIS3DH test!");

  if (!lis.begin(0x18))
  { // change this to 0x19 for alternative i2c address
    Serial.println("Couldnt start");
    while (1)
      ;
  }
  Serial.println("LIS3DH found!");

  lis.setRange(LIS3DH_RANGE_4_G); // 2, 4, 8 or 16 G!

  Serial.print("Range = ");
  Serial.print(2 << lis.getRange());
  Serial.println("G");
}

float getAccelMag()
{
  sensors_event_t event;
  lis.getEvent(&event);
  // Offset by 13.91.
  return sqrt(sq(event.acceleration.x) + sq(event.acceleration.y) + sq(event.acceleration.z)) -
9.0;
}
```

**Co2.ino**

```
int sensorIn = A2;
```

```
void setupCO2() {
  analogReference(AR_DEFAULT);
}

float co2Concentration() {
  int adcVal = analogRead(sensorIn);
  float voltage = adcVal * (5 / 4095.0);
  float voltageDiference = voltage - 0.4;
  float concentration = (voltageDiference * 5000.0) / 1.6;
  return (-1.0 * concentration);
}
```

**Dh11.ino**

```
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>

#define DHTPIN 6
#define DHTTYPE DHT11

DHT_Unified dht(DHTPIN, DHTTYPE);

uint32_t delayMS;

void setupDHT11() {
  dht.begin();
  sensor_t sensor;
  dht.temperature().getSensor(&sensor);
  dht.humidity().getSensor(&sensor);
  delayMS = sensor.min_delay / 1000;
}

float getTemp() {
  delay(delayMS);
  sensors_event_t event;
  dht.temperature().getEvent(&event);
  if (isnan(event.temperature)) {
    return 0.0;
  } else {

    return event.temperature;
  }
}
```

```
float getHumidity() {
  delay(delayMS);
  sensors_event_t event;

  dht.humidity().getEvent(&event);
  if (isnan(event.relative_humidity)) {
    return 0.0;
  } else {
   return event.relative_humidity;
  }
}
```

**Gps.ino**

```
#include "TinyGPS.h"

/* This sample code demonstrates the normal use of a TinyGPS object.
   It requires the use of SoftwareSerial, and assumes that you have a
   4800-baud serial GPS device hooked up on pins 4(rx) and 3(tx).
*/

TinyGPS gps;

void setupGPS()
{
  Serial1.begin(9600);
}

float getLat()
{
  bool newData = false;
  unsigned long chars;
  unsigned short sentences, failed;
  // For one second we parse GPS data and report some key values
  for (unsigned long start = millis(); millis() - start < 1000;)
  {
    while (Serial1.available())
    {
      char c = Serial1.read();
       Serial1.write(c); // uncomment this line if you want to see the GPS data flowing
      if (gps.encode(c)) // Did a new valid sentence come in?
        newData = true;
    }
  }
```

```
  float flat, flon;
  unsigned long age;
 if (newData)
 {
  gps.f_get_position(&flat, &flon, &age);
 }
 return (flat == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flat, 5);
}


float getLon()
{
 bool newData = false;
 unsigned long chars;
 unsigned short sentences, failed;

 // For one second we parse GPS data and report some key values
 for (unsigned long start = millis(); millis() - start < 1000;)
 {
  while (Serial1.available())
  {
   char c = Serial1.read();
   // Serial1.write(c); // uncomment this line if you want to see the GPS data flowing
   if (gps.encode(c)) // Did a new valid sentence come in?
    newData = true;
  }
 }
  float flat, flon;
  unsigned long age;
 if (newData)
 {
  gps.f_get_position(&flat, &flon, &age);
 }
 return (flon == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flon, 5);
}
```

**O2.ino**

```
const float VRefer = 5;
const int pinAdc = A0;

float readO2Vout()
{
  long sum = 0;
  for(int i=0; i<32; i++)
```

```
  {
    sum += analogRead(pinAdc);
  }

  sum >>= 5;

  float MeasuredVout = sum * (VRefer / 1023.0);
  return MeasuredVout;
}

float readO2Concentration()
{
  // Vout samples are with reference to 3.3V
  float MeasuredVout = readO2Vout();

  //float Concentration = FmultiMap(MeasuredVout, VoutArray,O2ConArray, 6);
  //when its output voltage is 2.0V,
  float Concentration = MeasuredVout * 0.21 / 2.0;
  float Concentration_Percentage=Concentration*100;
  return Concentration_Percentage;
}
```

**Main.ino**

```
/*
 Azure IoT for Arduino MKR NB 1500

 Author: Matt Sinclair (@mjksinc)

 This sketch securely connects to either Azure IoT Hub or IoT Central using MQTT over NB-IoT/Cat-M1.
 The native NBSSL library is used to securley connect to the Hub, then Username/Password credentials
 are used to authenticate.

 BEFORE USING:
 - Ensure that SECRET_BROKER and CONN_STRING in arduino_secrets.h are completed
 - Change msgFreq as desired
 - Check ttl to change the life of the SAS Token for authentication with IoT Hub

 If using IoT Central:
 - Follow these intructions to find the connection details for a real device: https://docs.microsoft.com/en-us/azure/iot-central/tutorial-add-device#get-the-device-connection-information
 - Generate a connection string from this website: https://dpscstrgen.azurewebsites.net/
```

```
 Full Intructions available here: https://github.com/mjksinc/TIC2019

*/

// Libraries to include in the code
#include <Scheduler.h>
#include <ArduinoMqttClient.h>
#include <MKRNB.h>
#include "./base64.h"
#include "./Sha256.h"
#include "./utils.h"

// Additional file secretly stores credentials
#include "arduino_secrets.h"

// Enter your sensitive data in arduino_secrets.h
const char broker[] = SECRET_BROKER;
const char pinNumber[] = SECRET_PINNUMBER;
// CONN_STRING: connection string from Hub;

String iothubHost;
String deviceId;
String sharedAccessKey;

int msgFreq = 1000; //Message Frequency in millisecods
long ttl = 864000;   //Time-to-live for SAS Token (seconds) i.e. 864000 = 1 day (24 hours)

NB nbAccess;
GPRS gprs;
NBSSLClient sslClient;
MqttClient mqttClient(sslClient);
NBScanner scannerNetworks;

unsigned long lastMillis = 0;

int ledRed = 7;
int ledGreen = 8;
bool warning = false;

bool reading = false;
float accel_mag = 0.0;

/*
```

```
  Establish connection to cellular network, and parse/augment connection string to generate credentials for MQTT connection
  This only allocates the correct variables, connection to the IoT Hub (MQTT Broker) happens in loop()
*/

void setup()
{
 delay(1000);

 Serial.begin(9600);
 while (!Serial)
  ;

 pinMode(ledRed, OUTPUT);
 pinMode(ledGreen, OUTPUT);

 setupDHT11();
 setupCO2();
 setupGPS();
 setupAccel();

 Scheduler.startLoop(loop2);

 Serial.println(F("******Splitting Connection String - STARTED*****"));
 splitConnectionString();

 //Connects to network to use getTime()
 connectNB();

 Serial.println(F("******Create SAS Token - STARTED*****"));
 // create SAS token and user name for connecting to MQTT broker
 String url = iothubHost + urlEncode(String("/devices/" + deviceId).c_str());
 char *devKey = (char *)sharedAccessKey.c_str();
 long expire = getTime() + ttl;
 String sasToken = createIotHubSASToken(devKey, url, expire);
 String username = iothubHost + "/" + deviceId + "/api-version=2018-06-30";

 Serial.println(F("******Create SAS Token - COMPLETED*****"));

 // Set the client id used for MQTT as the device id
 mqttClient.setId(deviceId);
 mqttClient.setUsernamePassword(username, sasToken);
```

```
  // Set the message callback, this function is
  // called when the MQTTClient receives a message
  mqttClient.onMessage(onMessageReceived);
}

/*
   Connect to Network (if not already connected) and establish connection the IoT Hub (MQTT
Broker). Messages will be sent every 30 seconds, and will poll for new messages
   on the "devices/{deviceId}/messages/devicebound/#" topic
   This also calls publishMessage() to trigger the message send
*/
void loop()
{
  if (nbAccess.status() != NB_READY || gprs.status() != GPRS_READY)
  {
    connectNB();
  }

  if (!mqttClient.connected())
  {
    // MQTT client is disconnected, connect
    connectMQTT();
  }

  // poll for new MQTT messages and send keep alives
  mqttClient.poll();

  // publish a message roughly every 30 seconds.
  if (millis() - lastMillis > msgFreq)
  {
    lastMillis = millis();
    publishMessage();
  }
}

void loop2()
{
  if (!reading)
  {
    float accel_mag_new = getAccelMag();

    if (accel_mag_new > accel_mag)
    {
      accel_mag = accel_mag_new;
```

```
   }
  }
  delay(100);
  yield();
}

/*
   Gets current Linux Time in seconds for enabling timing of SAS Token
*/
unsigned long getTime()
{
  // get the current time from the cellular module
  return nbAccess.getTime();
}

/*
   Handles the connection to the NB-IoT Network
*/
void connectNB()
{
  Serial.println(F("\n******Connecting to Cellular Network - STARTED******"));
  //pinNumber, true, true  true
  while ((nbAccess.begin() != NB_READY) ||
        (gprs.attachGPRS() != GPRS_READY))
  {
    // failed, retry
    Serial.print(F("."));
    delay(500);
  }
  Serial.println(F("******Connecting to Cellular Network - COMPLETED******"));
}

/*
   Establishses connection with the MQTT Broker (IoT Hub)
   Some errors you may receive:
   -- (-.2) Either a connectivity error or an error in the url of the broker
   -- (-.5) Check credentials - has the SAS Token expired? Do you have the right connection string
copied into arduino_secrets?
*/
void connectMQTT()
{
  Serial.print(F("Attempting to connect to MQTT broker: "));
  Serial.print(F(broker));
  Serial.println(F(" "));
```

```
 while (!mqttClient.connect(broker, 8883))
 {
  // failed, retry
  Serial.print(F("."));
  Serial.println(F(mqttClient.connectError()));
  delay(1000);
 }
 Serial.println();
 Serial.println(F("You're connected to the MQTT broker"));
 Serial.println();

 // subscribe to a topic
 mqttClient.subscribe("devices/" + deviceId + "/messages/devicebound/#"); //This is for cloud-
to-device messages
 mqttClient.subscribe("$iothub/methods/POST/#");                    //This is for direct methods +
IoT Central commands
}

/*
  Calls getMeasurement() to read sensor measurements (currently simulated)
  Prints message to the MQTT Client
*/
void publishMessage()
{
 Serial.println(F("Publishing message"));

 String newMessage = getMeasurement();

 // send message, the Print interface can be used to set the message contents
 mqttClient.beginMessage("devices/" + deviceId + "/messages/events/");
 mqttClient.print(newMessage);
 mqttClient.endMessage();
}

/*
  Creates the measurements. This currently simulates and structures the data. Any sensor-
reading functions would be placed here
*/
String getMeasurement()
{
 if (warning)
 {
  digitalWrite(ledRed, HIGH);
```

```
}
else
{
  digitalWrite(ledGreen, HIGH);
}

float temp = getTemp();
float humidity = getHumidity();
float o2 = readO2Concentration();
float co2 = co2Concentration();
float lon = getLon();
float lat = getLat();
// Very very very inaccurate and super rough estimation of ethylene. Based off avocados.
float pO2 = 715 * (0.21 + (0.0004 * o2)) - (9.26 * (co2 / 10000));
float ethylene = (0.14 * pO2) / (0.99 * pO2);
float days = 15.7829 * ethylene;

if (temp >= 30 || humidity >= 90)
{
  warning = true;
}
else
{
  warning = false;
}

digitalWrite(ledRed, LOW);
digitalWrite(ledGreen, LOW);

// Begin network scan to get signal strength
scannerNetworks.begin();

// String signalStrength = scannerNetworks.getSignalStrength();

String formattedMessage = "{\"temp\": ";
formattedMessage += String(temp);

formattedMessage += ", \"humidity\": ";
formattedMessage += String(humidity);

formattedMessage += ", \"ethylene\": ";
formattedMessage += String(ethylene);

formattedMessage += ", \"shelflife\": ";
```

```
  formattedMessage += String(days);

  formattedMessage += ", \"o2\": ";
  formattedMessage += String(o2);

  formattedMessage += ", \"co2\": ";
  formattedMessage += String(co2);

  formattedMessage += ", \"lon\": ";
  formattedMessage += String(lon);

  formattedMessage += ", \"lat\": ";
  formattedMessage += String(lat);

  reading = true;
  formattedMessage += ", \"accel\": ";
  formattedMessage += String(accel_mag);
  reading = false;
  accel_mag = 0.0;

  formattedMessage += "}";

  Serial.println(formattedMessage);
  return formattedMessage;
}

/*
   Handles the messages received through the subscribed topic and prints to Serial
*/
void onMessageReceived(int messageSize)
{

  String topic = mqttClient.messageTopic();

  // when receiving a message, print out the topic and contents
  Serial.print(F("Received a message with topic '"));
  Serial.print(topic);
  Serial.print(F("', length "));
  Serial.print(F(messageSize));
  Serial.println(F(" bytes:"));

  // use the Stream interface to print the contents
  while (mqttClient.available())
  {
```

```
   Serial.println(F((char)mqttClient.read()));
  }

  // Responds with confirmation to direct methods and IoT Central commands
  if (topic.startsWith(F("$iothub/methods")))
  {
   String msgId = topic.substring(topic.indexOf("$rid=") + 5);

   String responseTopic = "$iothub/methods/res/200/?$rid=" + msgId; //Returns a 200 received
message

   mqttClient.beginMessage(responseTopic);
   mqttClient.print("");
   mqttClient.endMessage();
  }
}

/*
   Split the connection string into individual parts to use as part of MQTT connection setup
*/
void splitConnectionString()
{
  String connStr = CONN_STRING;
  int hostIndex = connStr.indexOf("HostName=");
  int deviceIdIndex = connStr.indexOf(F(";DeviceId="));
  int sharedAccessKeyIndex = connStr.indexOf(";SharedAccessKey=");
  iothubHost = connStr.substring(hostIndex + 9, deviceIdIndex);
  deviceId = connStr.substring(deviceIdIndex + 10, sharedAccessKeyIndex);
  sharedAccessKey = connStr.substring(sharedAccessKeyIndex + 17);
  Serial.print(F("******Splitting Connection String - COMPLETED*****"));
}

/*
   Build a SAS Token to be used as the MQTT authorisation password
*/
String createIotHubSASToken(char *key, String url, long expire)
{
  url.toLowerCase();
  String stringToSign = url + "\n" + String(expire);
  int keyLength = strlen(key);

  int decodedKeyLength = base64_dec_len(key, keyLength);
  char decodedKey[decodedKeyLength];
```

```
 base64_decode(decodedKey, key, keyLength);

 Sha256 *sha256 = new Sha256();
 sha256->initHmac((const uint8_t *)decodedKey, (size_t)decodedKeyLength);
 sha256->print(stringToSign);
 char *sign = (char *)sha256->resultHmac();
 int encodedSignLen = base64_enc_len(HASH_LENGTH);
 char encodedSign[encodedSignLen];
 base64_encode(encodedSign, sign, HASH_LENGTH);
 delete (sha256);

 return "SharedAccessSignature sr=" + url + "&sig=" + urlEncode((const char *)encodedSign) +
"&se=" + String(expire);
}
```

## Group work member contribution table

| **Section**<br>*(add, remove and rename sections as needed)* | **Person(s) responsible and percentage**<br>*(e.g. Person A 70% Person B 20% Person C 10% OR All members contributed equally)* |
|---|---|
| Executive Summary | Kevin 50% Li 50% |
| Problem Statement | Kevin 50% Li 50% |
| Background and Literature Review | Kevin 50% Li 50% |
| Methodology and Engineering Design | Huy 50% Jong 50% |
| Findings | Kevin 25% Li 25%<br>Huy 25% Jong 25% |
| Discussions of Results/Products | Huy 50%<br>Jong 50% |
| Conclusions | Kevin 25% Li 25%<br>Huy 25% Jong 25% |
| Recommendations/Future Works | Huy 50%<br>Jong 50% |
| References/Appendix | Kevin 25% Li 25%<br>Huy 25% Jong 25% |
| Appendix | Kevin 25% Li 25%<br>Huy 25% Jong 25% |