

# 2024 E31 Vision Mātauranga Universal Datalogger

**Industrial Sponsor:** Vision Mātauranga (WRC initiative)

**Supervisors:** Phillipa martin, Lui Holder Pearson

**Project Group members:**

1. Logan Henderson,  
72740637, lhe68
2. Joshua Thompson-Holloway,  
29018037, jth142
3. Conor Dunlop,  
61689560, cdu58
4. Chloe McLaren,  
99594648, cmc335
5. Joshua McCorkindale,  
93040805, jmc670

## Executive Summary

The universal data logger is a research and development project for a system that can be used in varying Internet-of-Things (IoT) applications such as agricultural, and environmental, monitoring. The modularity of the system is designed based on market-ready devices for IoT, the design combines both long-range RF (LoRa) and long-term evolution (LTE-M), cellular modules to deliver the data from the field to the database via a cellular gateway, as well as implementing external MCU for data acquisition (DAQ). The user requirements achieved for the success of the vision Mātauranga project are seen in Table 1.

Table 1: Achieved high-level requirements

Functional	Project Requirements
	Connectivity and control with multiple sensors and outputs
	Implementation of communication technologies (LoRaWAN, 4G LTE)
Quality	
	Device is programmable over the air capable (OTA)
	Device receives information, Users can give commands, and interface with outputs through a web portal.
	Web portal functionalities (displaying data, showing status of data loggers)
	Data management (using a database to store data)
	Powering the device using battery/solar for rural deployment
Constraints	
	Maintaining cost of product under \$250 total and system costs under \$20 per month per data logger

### Tests and validations Conducted:

- LoRa-LoRa transmission on a separate platform, with a custom DAQ board to transmit ADC counts to the datalogger server using HTTP POST.
- HW debugging/verification for GPIO, SPI, and I2C for the DAQ board.
- Cellular: Dev-Kit HTTP POST, using TLS to servers, tested MQTT protocol.

### Limitations:

- Unsatisfactory integration of the entire system, from data logger and IoT transmission to the web server.
- Custom LPWAN device UART IC was broken, requiring more time/testing for full debugging.
- The Custom LoRa module was not tested with either a Zephyr dev kit or STM32 without an RTOS.
- An alternative proof of concept was demonstrated using LoRaWAN with a Wi-Fi gateway; however, no final prototype meeting all sponsor requirements were tested within this report.

While the hardware developed met the high-level requirements outlined in Table 1, the limited tests and validation of the entire hardware-software stack prevented the achievement of all custom implementation requirements. Nevertheless, the research and development conducted, once optimized, will significantly advance the concept. Recommendations are to continue developing the system, focus less on DAQ hardware due to successful implementation, and focus more on RF deployment and field test capability.

# Table of Contents

2024 E31 Vision Mātauranga Universal Datalogger .....	1
Executive Summary.....	2
Introduction .....	5
1. Purpose statement.....	5
1.1. Project Background.....	5
1.2. Scope.....	5
Group Member 1: Logan Henderson.....	7
The prototype design work.....	7
Power supply module design background.....	13
Test software programs .....	13
Discussion .....	15
Group Member 2 Joshua Thompson-Holloway.....	17
1. Research and Design contribution .....	17
2. Hardware design considerations .....	17
3. Hardware design implementation.....	18
3.1. Component selection.....	19
4. Implementation.....	21
4.1. Software implementation .....	23
5. Software tests/Validations.....	24
6. Hardware testing.....	25
7. Discussion.....	25
Group Member 3 Conor Dunlop .....	26
8. Introduction/Background.....	26
9. Work Undertaken.....	27
10. Discussion .....	31
Group Member 4: Chloe McLaren .....	33
Introduction .....	33
Work Undertaken.....	34
Database .....	34
Web API .....	36
User Interface .....	36
Communication Format.....	38
Discussion .....	40
Group Member 5 Joshua McCorkindale.....	41
Introduction .....	41
LoRaWAN Communications .....	41

LoRaWAN Classes.....	42
LoRaWAN Architecture .....	42
LoRaWAN Development Devices .....	43
LoRaWAN Networking Services .....	44
LoRaWAN Practical Applications.....	45
SX1262 Code development.....	46
Discussion.....	46
Sustainability .....	48
Triple bottom line analysis .....	48
LCA outline.....	48
Conclusions .....	50
Recommendations/Applications .....	51
References.....	52
Appendices .....	58

## Table of Abbreviations

The following table is a list of common IoT and electronic abbreviations which may prove useful for the contents of this report.

Abbreviation	Full Form
CAN	Campus Area Network
Cat-M1	Category M1
CO <sub>2</sub>	Carbon Dioxide
DAQ	Data Acquisition
EMI	Electromagnetic Interference
ESD	Electrostatic Discharge
GHG	Greenhouse Gas
HTTP	Hypertext Transfer Protocol
LiPo	Lithium Polymer
LPWAN	Low Power Wide Area Network
LTE	Long-Term Evolution
MPPT	Maximum Power Point Tracking
MQTT	Message Queuing Telemetry Transport
nRF	Nordic Semiconductor
OTA	Over-The-Air
PCB	Printed Circuit Board
RF	Radio Frequency
RTOS	Real-Time Operating System
TLS	Transport Layer Security
WAN	Wide Area Network

# Introduction

## 1. Purpose statement

The purpose of this project is to create a flexible Internet of Things (IoT) terminal device that can be deployed in a wide range of different scenarios, with a focus on environmental monitoring.

This project is specifically relevant to the Vision Mātauranga themes of indigenous innovation and Taiao, environmental sustainability and minor relevancy to Mātauranga, exploring indigenous knowledge [1]. This project allows iwi and other NZ agencies to make informed decisions using accurate environmental data and facilitates R&D into other areas. The flexibility of the device means that indigenous knowledge can be used alongside the device to further research goals. This project will be created in collaboration with iwi.

### 1.1. Project Background

This project focuses on the research, development, and deployment of a “Universal Modular Data Logger”. This project is following on from the Vision Mātauranga initiative, an MBIE framework aimed at supporting research beneficial to Māori communities [2]. Its objective is to develop a versatile Internet of Things (IoT) device tailored for environmental management, intended for use by iwi groups throughout New Zealand via agencies such as the Department of Conservation (DOC) and Landcare [14] as well as other possible clients. This project is important for the Vision Mātauranga initiative as this allows the local iwi to gather more information to allow them to apply their knowledge more effectively. The main design problem is that there is no “off the shelf” products that meet the current requirements as laid out by the client. Therefore, a new solution is to be designed. This new solution as previously outlined is to be an IoT-connected device focused on environment management using data collection with a range of different sensors. This data should then be displayed using a web-based interface. The data logger is also required to have a method of controlling outputs, such as LEDs, motors, or actuators. As stated in the name of the product this data logger is required to be modular allowing the user to swap out different sensors.

### 1.2. Scope

As described in the background the focus of this project is to assist with the gathering of data from the environment as well as remotely interfacing with the data. Therefore, the design problem of this project is: how to gather environmental data in a modular manner and access said data remotely. To solve this design problem the team has been tasked with the creation of a universal modular datalogger with IoT capabilities as previously outlined. This data logger can be broken down into two main parts, the physical data logger itself and the cloud-based access to the data. Both parts also have additional requirements set that can be found in Table 1.

These requirements are defined by their priority, a requirement with a hard priority must be met to ensure the success of the project. This contrasts with a soft priority that does not need to be met for the project to be deemed successful.

Table 2 : Universal Datalogger Requirements

Functional	Project Requirements
1.1	The datalogger shall interface with at least two sensors
1.2	The datalogger shall interface with at least two actuators
1.3	The datalogger shall all for implementation of multiple communication technologies (LoRa, 4G LTE)
Quality	
2.1	Device is programmable over the air capable (OTA)
2.2	Device receives information, Users can give commands, and interface with outputs through a web portal.
2.3	Web portal functionalities (displaying data, showing status of data loggers)
2.4	Data management (using a database to store data)
2.5	The hardware shall be powered from multiple power off grid power sources (batteries, solar)
Constraints	
3.1	The datalogger system shall have a cost of product under \$250 total
3.2	The datalogger system shall cost no more than \$20 per month per datalogger

Figure 1 details the core responsibilities of each member of the team, with each members contributions for the project presented following the order of:

- 1: Datalogger hardware, Logan Henderson
- 2: Communications hardware, Joshua Thompson-Holloway
- 3: Device-Device and sensor Software, Connor Dunlop
- 4: Web-server content software, Chloe McLaren
- 5: LoRa communications software, Joshua Mckorkindale.

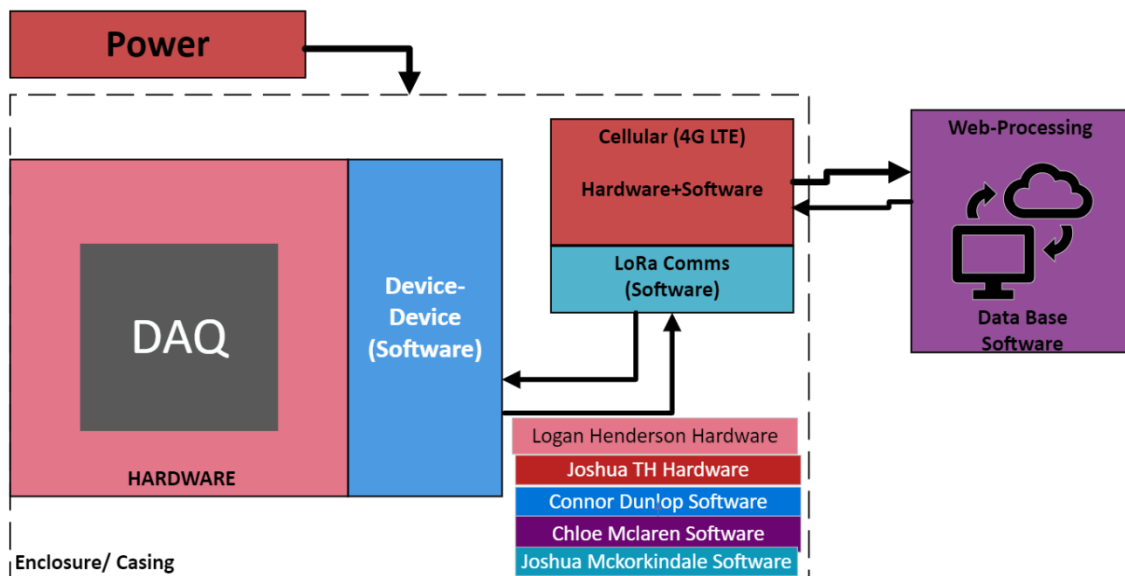


Figure 1: Block diagram of member research contributions and responsibilities

## Group Member 1: Logan Henderson Datalogger hardware

As outlined in the introduction, this section covers the work done on the datalogger hardware itself as well as the development of test programs. These test programs were created to not only test the hardware itself but also to emulate the hardware while the hardware was being developed. Custom hardware was chosen to be created for this project due to the complexity and battery life of the product. As previously outlined in the introduction the datalogger has a set of requirements, although not all the requirements apply to the data logger hardware therefor here are the requirements that directly apply to the data logger.

Table 3: Appropriate datalogger hardware requirements

#	Requirement
1.1	the data logger shall interface with at least two sensors
1.2	the data logger shall interface with at least two actuators
1.3	The datalogger shall all for implementation of multiple communication technologies (LoRaWAN, 4G LTE)
2.1	the data logger shall be reprogrammable over the air
2.5	The datalogger shall be powered from multiple power off grid power sources (batteries, solar)
3.1	The datalogger system shall have a cost of product under \$250 total
	Implied requirements
4.1	The datalogger shall be modular
4.2	The datalogger shall be universal

In addition to the requirements as layout within the introduction there are also two implied requirements, these are “The datalogger shall be modular” and “The datalogger shall be universal”. The former requirement means that the data logger will support multiple different modules whether different sensor modules, backpack modules, power modules etc. this is just an expansion to requirements 1.1, 1.2, 1.3 and 2.5. The latter requirement requires the data logger to support different voltage ranges, input impedances, output voltages, output loads, etc.

### The prototype design work

As outlined before two iterations of the datalogger were made, however this section will only cover the most recent prototype. This is due to the most recent prototype still implements the majority of the first prototype with minor hardware tweaks to fix the issue as previously outlined. Due to these board requirements, first a block diagram of the datalogger was created. This block diagram displayed the modular nature of the datalogger, as well as how the data logger was design to be universal by creating interface to support the way array of input requirements. Within Figure 3 as seen below, the data logger was broken in to 10 discrete design blocks, the power interface to meet requirement 2.5; the actuator, analogue and digital interface as well as the built in sensors for meeting requirements 1.1 and 1.2; the communications interface for 1.3; local storage for backing up data; the human interface for infield setup and maintenance of the device and finally the reset of the blocks are self-explanatory.

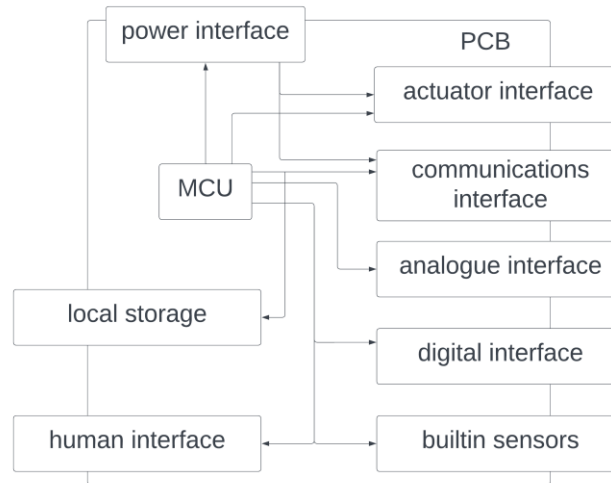


Figure 2: block diagram of the datalogger

### *Communication interface*

As previously outlined the communications interface exists to meet requirement 1.3. This interface needed to be designed to support the two specified interfaces (LoRaWAN, 4G LTE), as well as future interfaces, such as satellite communications or WiFi and bluetooth for shorter range network such as CAN [4].

This interface support SPI, UART and USART as well as 3 GPIO. These GPIO pins can be used for implementing features on such power a power cut off for if the communications modules have a high-power draw, chip select for SPI or even an IQR pin for a message ready. The interface was originally designed to only support SPI, this is the reason why the pins on the interface are only named with the SPI pin description. However, due to the capabilities of the STM32 micro controller pin multiplexing of both a UART and USART interface was also available on these pins. this allows for greater flexibility of supporting more communication devices.

The interface also has a power pass through from the power module. This was done to allow direct access to power rails for more regulation if a more stable voltage is needed for RF communications or if a higher power communications devices is required.

This communication interface allowing for a wide variety of communication modules to be used, meeting requirement 1.3. The interface also met requirement 4.1 and 4.2 by allowing for modules with different voltage, power requirements and interfaces to be used.

### *Power supply interface*

Due to the unknown power requirements of the sensors, actuator and communications interface, as well as the ability to support the unknown power supplies as specified by requirements 2.5, this power interface was made modular. This also allows the power interface to meet both requirements 4.1 and 4.2. although the power interface may have a few unknowns they are still knowns such as a 5v rail to allow for a regulated 3.3v for the microcontroller and a scaled battery voltage so the microcontroller can measure the battery level.

The power supply interface has power rails for 5v, battery voltage (VBAT), some  $\pm$ VDC voltage for custom voltage rails and a battery monitor voltage. This 5v power and the 5v power form the USB goes to a TPS2117 [5] power multiplexer. This power multiplexer swaps between the 5v supply from the power supply interface and USB. This was implemented using the schematic of the evaluation board [6], where a voltage divider will swap power supply inputs as soon as the



power supply interface reaches lower than 4.5v. This power multiplex prioritises power from the power supply over the USB.

The USB has a schlocky diode for reverse current protection as well as a fuse rated for up to 4 amps as this is the amperage that USB PD can supply at 5v [7]. As well as the USB having the specified 10uf decoupling capacitor as specified by the USB specification [8]. The USB also has TVS clamping diodes [9] to prevent ESD discharge.

This prioritized 5v power is then sent through a AMZ1117-3.3 [10] voltage regulator. This voltage regulator was chosen due to supporting up to 1 amp of current as well as prior experience using this regulator, finding it to be a robust option. This 3.3 voltage is then used to supply the microcontroller and other peripherals with power.

This power supply interface meets the requirements of 2.5, allowing multiple different power sources to be used. This interface too meets requirements 4.1 and 4.2, allow for multiple different modules to be used as well as allowing for custom voltage rails to be set for other modules, sensors or actuators.

#### *Actuators and sensor interfaces*

Due to these interfaces being inherently coupled as GPIO pins and other interfaces can handle both sensor inputs and actuator outputs, this section will cover how both requirements 1.1 and 1.2 were achieved. Just as previously implemented with both the power supply communication module interface this section too layout how interfaces were created to solve the design problem of supporting unknown sensor and actuator in a modular and universal manner to meet requirements 4.1 and 4.2.

#### *Analogue interface*

The datalogger is required to measure analogue voltage from such sensors like a phototransistor [11] or soil moisture sensor. This could be done with either external ADCs or internal ADCs on the microcontroller. The benefit to using external ADCs is they can measure closer to the source of the signal, which reduces the possible noise within the signal. However, as these data loggers are normally situated within the bush where there are few sources of EMI this noise would be limited. As well as these external ADCs adding extra cost to the system. Therefore using the internal ADCs would be an optimal solution. As determined the data logger uses the internal ADCs on the microcontroller. Although to meet user requirement 1.1 there need to be at least 2 ADC channels. The analogue interface also exposes the DAC on the STM32G4 to allow for analogue actuators to be used such as speaker for making bird noises. However, this was not a high priority therefore no extra amplification circuitry was created.

From research on sparkfun's website [12] most analogue sensors tend to require voltages 5v, this means that microcontrollers that operate on 3.3v could not accurately measure this voltage. As well as the output of these sensors, they may also be of a small voltage and require amplification. Due to this, the front end requires some analogue amplifier. It can be assumed that most of these analogue sensors output a voltage between 0-5v. As this voltage has no negative component as well as the internal ADC on the micro controller not being able to measure negative voltage, the amplifier architecture was selected to be non-inverting. The TL074 [13] was chosen as the opamp for this interface due to prior experience utilizing this opamp as well as being widely available and low noise. This also allows this interface to help to achieve requirement 4.2 as different input impedances can be selected as well as supporting a wide range of input signals.

## Digital interface

Yet again due to the unknowns of digital sensors such as pressure or humidity; or simple digital sensors like buttons; as well as unknown digital actuators such as smart servos or basic actuators that just require a PWM or a digital signal. The digital interface has multiple smaller interfaces, the first interface is conjoined with the analogue interface that exposes the internal ADC pins with these pins also to be available as GPIO with 7 exposed pins, and 3 extra pins that could not be routed easily. The digital interface also has a I2C bus that is buffered for offboard communications to items such as backpack modules, or singular I2C sensors. However, if it is found that the distance between the data logger and the sensor something like a P82B96 [14] can be used for shifting the logic level up to a possible 12v instead of 3.3v allow distances up to 20m to be achieved. Finally, the digital interface has a separate SPI bus from the communications interface to be a dedicated camera interface. This camera interface also has a mosfet power switch to allow the camera to only be turned on when an image should be taken conserving power. This design decision to use a separate SPI bus was chosen due to an external camera being more prone to failure with the possibility of the cable shorting and not allowing the communication to communicate. This created a more robust design allow for more points of failure before a data logger goes offline.

This interface helps to achieve requirements 1.1 and 1.2, as well as achieving 4.1. however, this interface did not quite achieve 4.2 as even though it can support a wide range of sensor and actuators, it does not support the more common 5v for these requiring either a logic level shifter to be used or mosfet to set this voltage to an appropriate range.

## Actuators interface

The actuator interface was designed around a power amplifier as the actual device that would connect to the other end is unknown. Since this could not be determined it would not meet requirement 1.2 if this actuator interface was designed to run an inductive actuator like a motor. Therefor this was designed to be a power amplifier for supplying either a high power PWM or just a constant high-power output.

The actuator interface was implemented using a simple high side switching power amplifier. This amplifier uses a basic through hole PMOS mosfet with a BJT to amplify the voltage from the STM32. This met the requirements of 1.2, and 4.2 however could be further improved.

## Built-in sensors

During talks the out client it was found that temperature was a very common environmental parameter to measure. This could be achieved with the internal temperature sensor on the STM32 however this would give imprecise reading if say the datalogger was left in the sun or had to do some computations beforehand warming up the MCU. Due to this imprecision a 4-channel thermocouple ADC was provisioned for. This ADC was implemented using the application schematic which required another temperature sensors to be used for cold junction compensation. This thermos couple interface was in addition to a separate environment sensor the BME680, this sensor allows to the measurement of temperature, humidity and pressure, all useful environmental parameters. this implementation of the interface helps to achieve requirement 1.1 and does not apply to any other requirements.

## Microcontroller

As previously stated, the microcontroller is a key part of this project. The microcontroller is needed to gather and collate the data to then send on to the communications module or stored

locally. The microcontroller needs some form of sleep mode, this is due to the limited power available as well as the microcontroller staying idle for long periods of time. This sleep mode also requires drawing as little power as possible due to it becoming more costly the more someone is required to be sent out to replace a battery within the data logger. As well as at least 2 ADCs channels as stated within the analogue interface although three is preferable to the ability to measure current battery voltage. The microcontroller also requires at least two SPI interfaces and one I2C interface to meet the specifications layout in the communications interface, digital frontend, and external storage sections. The micro controller should also have some USB interface to allow for easy powering during development as well as the option to plug in the data logger if a power source is available on site. The USB interface can also allow debug messages to be sent to an external device as well as the possibility of implementation as a USB mass storage device for direct communication with the local storage. The actuator interface also requires at least two GPIO pins that can support PWM. Using these already established specifications and the previously outlined user requirement a set of specifications can be derived form. These specifications can be seen in Table 4.

Using these specifications 5 possible candidates were decided on, however only 3 of these candidates are shown in Table 5. The two candidates not included were an Arduino based micro controller and a raspberry pi. An Arduino type micro controller was not considered due to the lack of features that were required as well as the poor software develop on atmega based processor for burning the boot loader on to a new MCU [15]. The raspberry pi was also not considered due to the high-power draw and not requiring all the compute it is capable of.

Table 4: microcontroller specifications

#	Specification	Notes
1.1	the microcontroller shall have a sleep mode	
1.2	where a sleep mode is included, the microcontroller shall have a deep sleep mode	
1.3	the microcontroller shall have at least 3 ADCs	
1.4	the microcontroller shall have at least 1 I2C interface	
1.5	the microcontroller shall have at least 1 SPI interfaces	
1.6	the microcontroller shall have a real time clock (RTC)	
1.7	the microcontroller shall have at least 2 PWM	
1.8	the microcontroller shall have a USB interface	
1.9	the microcontroller shall have at least 10 GPIO	
1.10	the microcontroller may have at least 1 DAC	Wanted if the person desires some audio out
1.11	the microcontroller shall be reprogrammed through use of another MCU	Desired for over the air programming

Table 5: analysis microcontrollers

MCU	Sleep?	ADC	I2C	SPI	USB?	PWM	DAC	GPIO	Ave Power	Cost \$
RP2040	Deep	4	2	2	Yes	16	0	30	25mA [16]	1.31 [17]
Stm32G4 [18]	Deep	15	2	2	Yes	14 <sup>1</sup>	2	52	25.5 <sup>2</sup> mA	11.67 [19]
ESP32	Deep	14	2	4	Yes*	16	2	21	~55mA [20]	5.00 [21]

<sup>1</sup> This is the number of timers the stm32g431cbxxx has, although it can have even more PWM channels attached to each timer

<sup>2</sup> This was calculated in stm32cubeMX

As can be seen in Table 5, all micro controllers bar one meets the specifications as laid out in Table 4. The RP2040 could still be used as the specification for the DAC is not required for the product to be successful. However, this microcontroller was not chosen due to the lack of GPIO interfaces when compared to the STM32. The RP2040 is also a dual core microcontroller, and this extra computing power is not required for this use case as there is no requirement for high-speed sampling. Although this microcontroller was the cheapest it was not decided on due to the lack of experience that the group had implemented anything on it. The other microcontroller that was not selected was the ESP32. The main reason this microcontroller was not selected was due to it having built in WIFI and blue tooth. This was deemed unnecessary due to the communications module already being able to handle this interface. The other reason this microcontroller was not selected was due to the poor performance of the internal ADCs [22]. This poor performance requires a correction algorithm to be applied to the measured value. For these reasons the ESP32 was not chosen. The STM32 was chosen mainly due to having prior experience with the microcontroller as well as the academic supervisor also having a large amount of experience. However, the STM32 was the most expensive although this microcontroller also came with a large set of libraries and example code to accomplish anything that was possible with it. The STM32G4 is also one of the few STM32s that support openboot loader [23]. This software can allow the STM32 to reprogram itself if there is enough storage this allow the datalogger to achieve requirement 2.1 [24]. The STM32G4 can also be programmed over almost all digital interfaces such as USART, USB, SWD and JTAG. This variety of the STM32 was the main reason for the choice. The STM32 also supports other digital interfaces such as SAI and I2S for audio output.

The STM32 uses an external 8MHz crystal oscillator as opposed to the internal one, this was chosen due to the relatively high-power draw of the internal oscillator compared to an external oscillator. The STM32 was also implemented as specified by the application schematics, however an extra zero-ohm resistor was added to the VDDA decoupling capacitors. This zero-ohm resistor was added to allow the creation of a PI filter to remove any high frequency noise that could be created by the power supply.

The STM32 also has a SWD debug interface to allow for a more pleasant debugging experience and direct integration with STM32CubeIDE through use of an STLink [25]. However, the SWD interface also uses a SDIO pin this extra capability required a stub to be added on to the camera SPI interface. Although the effective distance of this stub was limited using a solder jumper.

The STM32 can also be connected to power by and programmed from the USB type C header. This USB standard was selected due to prior experience using this as well as the STM32G4 having built in support for USB PD [18]. The STM32 also had its VBAT pin directly connected to the power supply interface. This choice was made because an ADC channel can be internally connected to this pin allowing for possible battery life estimation as well as allowing for better brown out detection.

#### *Human interface*

Although not part of any user requirement, a human interface for infield adjustment of parameters such as sample rate, battery usage, communications sanity checks are all very useful pieces of information to have before deploying one of these data loggers and leaving it. To start this interface just like the camera interface too has a power enable mosfet to allow for the disconnecting of power to save on battery life. The human interface implements 4 buttons as this is really the minimum needed to nicely navigate around menus. The interface also the same I2C bus as the digital interface and built in sensor interface, for its the display. This was done so an

another I2C bus would not be required. Although not specified due to not being a high priority, the display chosen should support extended character sets or bit mapped font to all the diacritical mark within the text to be displayed.

#### *Local storage*

Due to the possibility of the communication module going offline, or data that would be too large to otherwise be sent over the communications module such as images from the camera interface, it was decided that a local storage option would be required. This would store backups of critical and non-critical data as well as larger files. The storage medium chosen for this task was a micro-SD card. This storage medium should be removable, even though the stm32G4 does support the ability to act as a mass storage device [26], an external device should not have to be brought in to the bush in order to read the data off the data logger. Therefore the storage medium must be removable. As can be seen in Table 6, SD cards are the only easily removable device as well as storing up to 1Tb of data. This was also chosen due to SD card being able to be interfaced with over SPI [27]. However, unlike the camera this SD card can be on the same SPI bus as the communications module due to its protected nature on the PCB. The formfactor of the SD card was chosen to be micro-SD due to their abundance and ease of access.

Table 6: storage types

<i>Storage type</i>	<b>Removable</b>	<b>Size</b>
<i>Internal</i>	No	128Kb [18]
<i>Flash</i>	No	Upto 8Tb [28]
<i>EEPROM</i>	No*	Upto 32Mb [29]
<i>Sd card</i>	Yes	Upto 1Tb

#### *PCB*

The overall size of the data logger PCB was decided on being 100mmx100mm, this was due to being the maximum size that could be ordered from JLCPCB [30] before vastly increasing in cost. The PCB is 4 layers signal, power, ground, and signal. This was chosen over 8 layers due to requiring the larger area that 4 layers can provide at cheap price. The high-speed traces were routed on the bottom signal layer as it is next to the ground plane. Within the corner of the PCB there are m3 mounting holes. m3 size mounting sizes were chosen due to them being relatively common as well as already having this hardware on hand. These mounting holes were also placed in such a position to securely attach both the communications module and the power module to the main data logger without having to solely rely on 2.54mm pitch pin headers.

#### **Power supply module design background**

A single power supply module was developed for the datalogger. This power supply allowing power from either a single cell LIPO battery such as an 18650 or from some other DC power supply such as an MPPT solar controller. This implementation has over discharge, charge, and temperature protection for the LIPO battery. This module uses a buck boost converter to take the nominal voltage of the LIPO up to 5v to be sent to the data logger board. However, this module does not implement any custom voltage rails. This was a simple module designed to have a possible prototype power supply although time did not allow for this.

#### **Test software programs**

Three programs were created to test the basic functionality of the board, these were a simple program to send some changing data from the stm32 over UART to a communications module; the communications software that ran on an ESP32 to encapsulate the data in to a json format and send it to a test HTTP server using an HTTP post request, and finally the HTTP echo server that

would echo the sent package to the terminal for validation of the data [31]<sup>3</sup>. Separate from all these a test program was also written to act as an emulator for the data logger to allow for developed to the cloud infrastructure before the data logger was finalised.

#### *Data logger test program*

The first test program to send data over UART was only created for the first iteration of the PCB. This test program has a simple count that counts to 255 and rolls over; a count button that will increase/decrease with the press of the down and up buttons; an ADC value of ADC 1 channel 1 and finally a state string that changes with enter button. These values were chosen to ensure it would be reading new data in as well as checking for in order sending of packets. The program then formats this information using this given format string as can be seen in Figure 3. This data format is in a has separates all data points with a comma for easy separation and ends with a “\r\n” to show end of data for both UART and USB coms devices. This test program then sends the expanded format string over both USB as a virtual comms device as well as over UART on the communication interface. This interface was originally specified to use SPI however some other micro controllers are difficult to set up as SPI slave devices and therefor UART was used.



Figure 3: format string used

#### *ESP32 test program*

The second test program runs on an ESP32 WROOM-32 [32]. Firstly, this test program will connect to the given network over WiFi, this network is specified within a “secrets.h” file to ensure that personal SSID and passwords are not committed to the git repo. Once connected to WIFI this program will then start to read the data from UART using the UART 2 interface on the ESP32 as UART one is used for programming the device. Once read in the “\r\n” is stripped from the data. This new cleaned string is then formatted into an HTTP packet with a content type of JSON and sent to the specified HTTP server.

#### *Data logger emulator*

The data logger emulator has 2 scripts, one script (env.py) defines the packet type to be sent using a python class, and the other script (client.py) creates an https client then creates data using the former script and sends them to yet again another http echo server [31]. The former script (env.py) defines data within a python class. This architecture allows for methods to be added to the class to easily manipulate the data created, for example creating random data to send to the server. The script defines two data formats “chloeDataFormat” and “LoganDataFormat”, however “chloeDataFormat” is the one that is currently used by the server. These classes then get converted into JSON before being sent to the server using the “classToJson” as defined in the same script, this script converts the class to its dictionary definition, changes it to a string and replaces the single quotes with double quotes, in turn creating valid JSON. This JSON is then used by the former script (client.py) to create and send HTTP requests to a given server, allowing for objectives to be tested such as stress testing the server or just iterating quickly over new data formats.

---

<sup>3</sup> This server implementation is in git however the original example code is the code referenced

## Discussion

Currently two iterations of the data logger have been created. The reason for two iterations was to create a simple prototype to allow for development on hardware as soon as possible. However as expected this first version as can be seen in Figure 4 below, had three known hardware issues: a disconnected power plane between both the left- and right-hand sides of the board; The 3.3v regulated ground was not connected to the ground plane, and finally the SDA and SCL were the wrong way around. However, most of the test code was designed around these due to having these boards for the longest amount of time.

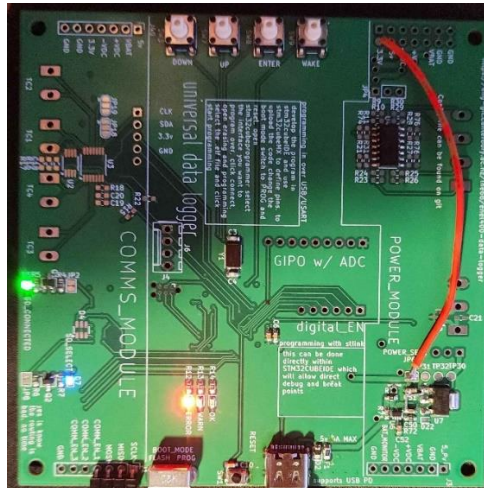


Figure 4: Fixed first iteration PCB

Due to these hardware issues a second iteration was made to fix these errors as can be seen below in Figure 5. The second iteration still had one hardware issue of the SDA and SCL being the wrong way around, however this was fixed on the assembled prototype PCB, with this issue being fixed on the schematic for future iterations. 7

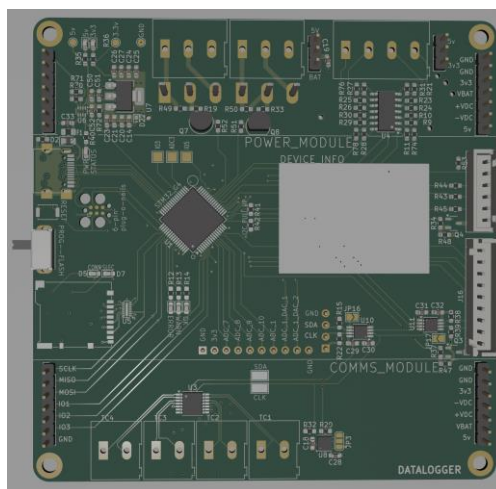


Figure 5: Render of second iteration PCB

Of these datalogger hardware iterations, 3 PCB were assembled of the first iteration and one prototype of the second iteration. This was done to allow for parallel work on the software to interface with the datalogger over the communication interface, as well as the sensors attached.

The second iteration of the datalogger met all requirements set for it bar one, however requirement 3.1 will be covered in a later section. Although as seen in **Appendix B** the rough cost for parts for the data logger hardware is only around \$48.



- The datalogger can sense at least two sensors through use of the digital, analogue interfaces achieving 1.1.
- The datalogger can actuate at least two actuators through use of the digital, analogue and actuator interfaces achieving 1.2
- The datalogger has a communication interface allowing for the expansion and use of different communication devices meeting achieving 1.3.
- The datalogger has a power interface allowing of the ability to add custom power board to support multiple battery chemistry and power inputs meeting 2.5.
- The datalogger can reprogram itself through use of openboot loader achieving requirement 2.1

Although the datalogger hardware did meet the requirements further improvements could be made to it. Such as current PCB is using a standard HASL finish [33]. This finish can wear away and corrode over time so a possible change to a more resistance finish like EING coating [33] could be beneficial. Although this EING would significantly increase cost.

The micro-SD card be changed to the now less common full size SD card. This argument revolves around the case if this micro-SD card is dropped during data retrieval within the bush it would be harder to locate on the forest floor due to its small form factor. A full-size SD card would solve this due to the larger size of them making them easier to find.

Another alteration that could be made is swapping to a low power STM32, this was not originally considered due to already having easy access to an STM32G4 as well as prior experience with this skew. However, this would not be a difficult transition to make as the HAL of the STM line of MCUs are constant allowing for the software to be transferred over.

An issue may arise from running an SPI interface off the PCB itself on the camera interface. This interface is high speed and tends to suffer from signal degradation if routed externally [34]. However this problem can be mostly mitigated by using an sending the SPI signal over an RS-422 data link [35].

As stated before currently the digital interface, only has one expansion interface for the I2C bus to move off board as well as only support signals up to 3.3v. This could become a problem due to a large amount of digital sensor still using 5v to send this data. The other problem that may arise from this is only having one I2C interface exposed, therefore adding one sensor to be mostly trivial. However, if more than one sensor wanted to be used this would require a backpack module to be created just to allow one extra digital sensor to be added. This would be wasteful and time consuming. Therefore, possible added more buffer I2C interfaces with logic level shifter would be a welcome addition.

Yet another problem area that should be looked in to is the humidity and condensation causing possible shorts on the PCB [36]. This could be remedied using a form of conformal coating [36] or possibly just through use of desiccant [36], such as silica gel [36].

In conclusion, the data logger met all requirement. However, there is room for further improvements as stated. However, these improvements are about expanding on the current functionality of the hardware instead of fixing the current design.



## Group Member 2 Joshua Thompson-Holloway Communications Hardware and Software

### 1. Research and Design contribution

As indicated in the introduction for this project, a core goal of this project was to implement a solution that combined standardized IoT RF technologies for remote data transmission. The core goal of this project, and of the supervisor, was to improve upon the WRC insect trapper design, to design a modular environmental device Appendix C. My core responsibilities and contributions were research and development of a device that combined two IoT RF protocols. Liaising with the sponsors of this project, the core requirements for the universal modular data logger, and communications hardware are shown in Table 7.

I designed and manufactured an LPWAN PCB based on the requirements in Table 7, and also Table 4 . The design of this device is ‘modular’ allowing it to replace the datalogger/DAQ board or supplement it, depending on a future project's budget, scope and power considerations; indirectly achieving ‘universal modularity’. Simple testing for IoT protocols was investigated using development kits implementing identical transceivers. A Python Flask server was implemented for testing purposes such as different security and data transmission. The LPWAN PCB was successfully manufactured, and schematics for a second improved version are prepared. For full software and hardware see Appendix A.

Table 7: Universal Datalogger Communications Requirements

Functional	Project LPWAN Requirements
1.1	Connectivity and control with multiple sensors and outputs
1.2	Implementation of communication technologies (LoRaWAN, 4G LTE)
Quality	
2.1	Device is programmable over the air capable (OTA)
2.2	Capable of being powered from Lithium-ion battery
2.3	System costs under \$20 to run per month per data logger
2.4	Capable of indicating status to operator

### 2. Hardware design considerations

One significant consideration was the power-stage of this device. Early development of a battery power-board was abandoned due to time constraints and health and safety considerations. The battery chemistry standard for IoT is lithium polymer (LiPo). The power characteristics are ideal for low-power operation for a device in the field. However, LiPo integration into the final design would require a lot of additional protection circuitry such as over-voltage protection (OVP). Power considerations, and other manufacturing considerations for a LPWAN device are shown in Table 8 and Table 9.

Table 8: Design considerations for a LPWAN, modular device

Design Consideration	Implementation	Rationale
<b>Modularity</b>	Combined LoRa + LTE-M modem with configurable SPI lines on PCB using solder-bridging	Ensures flexibility and adaptability in design.
<b>Reference design research</b>	Based on Nordic Semiconductors nRF9160 System-in-Package (SiP), SX1262 evaluation design, nRF9160 feather. <b>Invalid source specified.</b> [9]	Speeds up the design process and leverages proven designs.
<b>Integration</b>	Combined several reference modem designs, specifically the nRF9160 feather. Core component selected was the nRF9160 SiP with NB-IoT and Cat-M1 capabilities.	Provides a comprehensive solution with multiple connectivity options.
<b>Signal integrity</b>	50 ohms impedance, stitching vias to avoid reflections in the antenna.	Ensures efficient operation, extends the effective range of the device.
<b>Board stack-up</b>	4-Layer PCB. Signal/GND/PWR/Signal	Cost effective. Reduction of EMI/EMC utilising optimal stack-up
<b>Design for test</b>	Implemented test points, 0R resistors, solder bridges, and a debugging interface.	Facilitates troubleshooting and performance monitoring.
<b>Design for manufacturing</b>	Used well stocked components. Optimal passive components for RF applications.	Potential to mass produce.
<b>Modularity trade-offs</b>	Greater time investment for designing PCB. Routing and component placement is more complicated.	Balances flexibility, and cost efficiency for multiple devices.

Table 9: Power considerations between boards stickups

Board	Voltage target	Rationale
<b>Power Expansion (LPWAN)</b>	Output 3.7V (No Buck converter)	Simple. Cheap. Buck could potentially interfere with signal integrity/antenna. A linear power supply is preferred for RF applications. [37]
<b>LPWAN</b>	3V3	Low power draw.
<b>DAQ</b>	5-3V3, sensors may need additional power	Power considerations are context dependant (How many sensors, what type etc)

### 3. Hardware design implementation

The hardware implementation was one PCB designed, and two boards manufactured. One populated with both LoRa and Cellular modems, the second only populated with a LoRa modem and solder bridges connected, configured for external MCU interfacing. For the description of the design implemented, is shown in Figure 6 and Figure 7,. Table 10 covers additional rationale for the device.

Table 10: Block Diagram implementation of LPWAN device, iteration 1

Block	
<b>Board-Power</b>	Uses MOLEX connectors for power expansion board interface. Locking mechanism reduces reverse polarity risk. A diode provides reverse polarity protection.
<b>nRF9160 SiP</b>	Main design component with MCU, programmable via Zephyr RTOS. Interfaces with board-board connections, RF components, and debugging components.
<b>Board-Board Data and Signal</b>	Contains MOLEX connectors for SPI interface between expansion boards. Interfaces with nRF9160 SiP for battery voltage measurement (ADC GPIO), USB-UART, and external Flash memory.
<b>RF Connections and LoRa Module</b>	Sensitive to EMI. Requires careful component placement and test points for performance analysis.

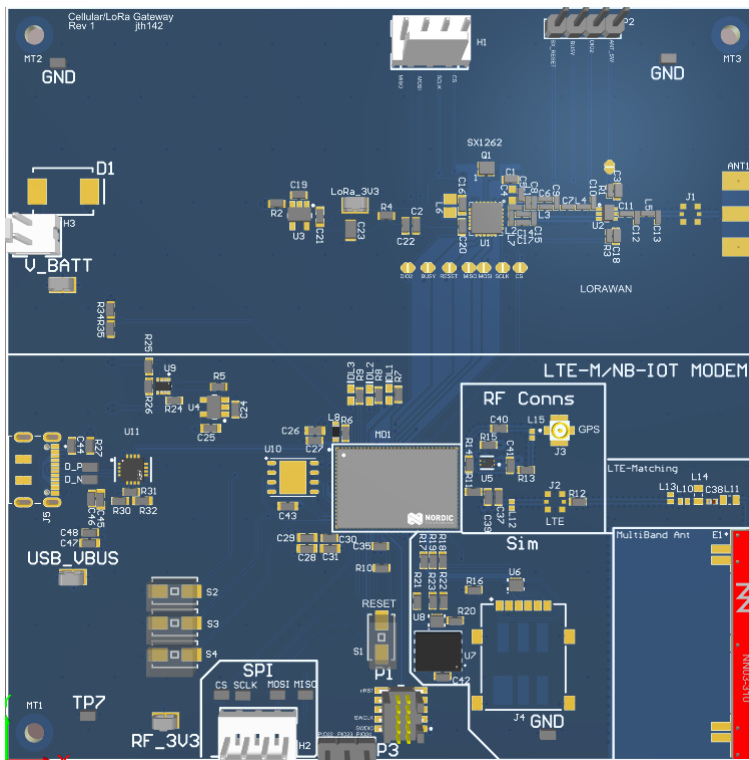


Figure 6: LPWAN PCB (Altium)

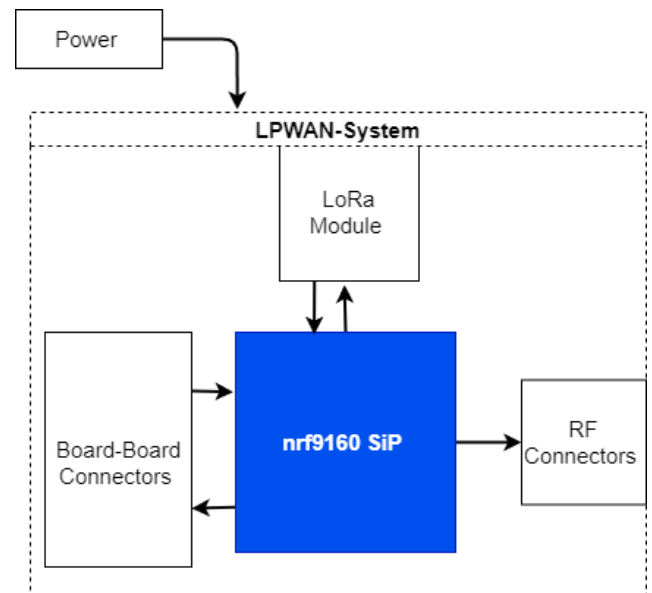


Figure 7: Simplified LPWAN block diagram with core components

### 3.1. Component selection

The core components selected are discussed in Table 11, A majority of the passive components, such as capacitors and resistors, have a uniform footprint and are available from the UC-Surface mount technology (SMT) lab. Further rationale for selecting core components to achieve requirements [1.4-1.5] are shown in Table 12-Table 13.

Table 11: Core Components Features for LPWAN PCB

Feature	SX1262IMLTRT [38]	NRF9160-SICA-B1A-R7 [39]
<b>Modulation</b>	LoRa and FSK	LTE-M and NB-IoT
<b>Maximum Link Budget</b>	170 dB (SX1262/68)	Not specified
<b>Transmit Power</b>	+22 dBm (high efficiency PA)	23-46 dBm
<b>Receive Current</b>	4.6 mA (low RX current)	Not specified
<b>Frequency Range</b>	150 MHz to 960 MHz (sub-GHz ISM bands)	Not specified
<b>Sensitivity</b>	Down to -148 dBm	Not specified
<b>Blocking Immunity</b>	88 dB at 1 MHz offset	Not specified
<b>Co-Channel Rejection (LoRa)</b>	19 dB	Not specified
<b>Application Processor</b>	N/A (dedicated modem)	Arm Cortex-M33
<b>Flash Memory</b>	N/A	1 MB
<b>RAM</b>	N/A	256 KB
<b>GNSS Receiver (GPS)</b>	N/A	Integrated
<b>Interfaces</b>	12-bit ADC, RTC, SPI, I <sup>2</sup> C, I <sup>2</sup> S, UART, PDM, PWM	12-bit ADC, RTC, SPI, I <sup>2</sup> C, I <sup>2</sup> S, UART, PDM, PWM
<b>Security</b>	Arm TrustZone technology, Arm CryptoCell	Arm TrustZone technology, Arm CryptoCell
<b>SIM/eSIM Support</b>	Both SIM and eSIM	Not specified

Table 12: Core Components features sets for LPWAN PCB

Feature	SX1262IMLTRT [38]	NRF9160-SICA-B1A-R7 [39]
<b>Modulation</b>	LoRa and FSK	LTE-M and NB-IoT
<b>Maximum Link Budget</b>	170 dB (SX1262/68)	Not specified
<b>Transmit Power</b>	+22 dBm (high efficiency PA)	23-46 dBm
<b>Receive Current</b>	4.6 mA (low RX current)	Not specified
<b>Frequency Range</b>	150 MHz to 960 MHz (sub-GHz ISM bands)	Not specified
<b>Sensitivity</b>	Down to -148 dBm	Not specified
<b>Blocking Immunity</b>	88 dB at 1 MHz offset	Not specified
<b>Co-Channel Rejection (LoRa)</b>	19 dB	Not specified
<b>Application Processor</b>	N/A (dedicated modem)	Arm Cortex-M33
<b>Flash Memory</b>	N/A	1 MB
<b>RAM</b>	N/A	256 KB
<b>GNSS Receiver (GPS)</b>	N/A	Integrated
<b>Interfaces</b>	12-bit ADC, RTC, SPI, I <sup>2</sup> C, I <sup>2</sup> S, UART, PDM, PWM	12-bit ADC, RTC, SPI, I <sup>2</sup> C, I <sup>2</sup> S, UART, PDM, PWM
<b>Security</b>	Arm TrustZone technology, Arm CryptoCell	Arm TrustZone technology, Arm CryptoCell
<b>SIM/eSIM Support</b>	Both SIM and eSIM	Not specified

Table 13: Rationale for core PCB Components of LPWAN PCB , 1<sup>st</sup> iteration.

Core Component	Rationale	Related User Requirement
<b>TPS2117DRLR (Power Select IC) [40]</b>	Power efficiency due to operational characteristics; Assists in field and product testing debugging by allowing for batteries to be switched out if USB power is connected, allowing for the device to remain powered.	1.13 (Use some form of battery/solar to power the device)
<b>SX1262IMLTRT (LoRaWAN Transceiver) [38]</b>	Well-stocked; economical in modular design; compatible with nRF9160 SiP or STM32 [41] board via an SPI-Bus.	1.4 (Implement LoRaWAN for communications), 1.3 (Ability to use multiple IoT technologies)
<b>nRF9160-SICA-B1A-R7 (Cellular IoT SiP) [39]</b>	Feature-rich as per X, good security for WPAN applications, well-stocked and well-supported.	1.5 (Implement cellular for communications), 1.3 (Ability to use multiple IoT technologies), 2.3 (Device is Programmable over the Air)
<b>Nano SIM Connector [42]</b>	Common component: necessary for cellular IoT as it operates in the licensed spectrum.	1.5 (Implement cellular for communications)
<b>Embedded-Sim</b>	Not necessary to populate, optimal for OTA, more research on implementation is necessary	
<b>CP2102N-A02-GQFN20R (USB to UART Bridge) [43]</b>	Used in nRF9160 Feather, allows for programming via UART over USB interface. [44] [45]	
<b>Molex Connectors</b>	Readily available in the UC store, mitigating costs; fixed and lock in place, providing robustness and reducing likelihood of incorrect interfacing.	2.1 (Hardware Designed for Modularity)
<b>Multiband Antenna (NN03-310) [46]</b>	Provides flexibility in design by supporting multiple frequency bands, trade-offs include lower efficiency and performance degradation due to interference.	1.3 (Ability to use multiple IoT technologies)

**Alternatives for this design:** Several alternative core components were considered for this design. However, due to the number of features, documentation, and support from Nordic Semiconductor, I opted to use the nRF9160, which uses Zephyr RTOS over less documented LPWAN modem devices such as the modem used in the Portenta GNSS shield. [47]. Alternative components, while similarly meeting the hardware requirements, had significantly less documentation, and publicly available products to evaluate success of custom design implementations. Such as the Arduino Portenta, this platform provided very few examples of code and applications for using both GPS and cellular. Most of my research for this design focused on the microcontroller's performance and features aligned with the project requirements, such as over-the-air programmability, and GPS, as requested by the client based on their experience with LoRa and the insect trapping.

## 4. Implementation

Table 14, summarizes the implementation steps and rationale for project management, device manufacturing and testing.

Table 14:LPWAN implementation

Step	Rationale
1. Custom Board Ordered	Ordered a custom board and began researching its software implementation. Followed references and read the nRF website to learn the implementation of Zephyr.
2. Planning and Development Kits	Planned both hardware and software integration. Used two development kits with Zephyr: one for testing SPI communications and the other for programming the custom board. This enabled parallel development. Other team members optimized software integration with the system stack (Datalogger-LPWAN or Datalogger-LoRa) <sup>4</sup> .
4. Initial Setup and Custom PCB Assembly	Set up the development kits, provisioned device certificates, and verified SIM connectivity. Assembled the custom PCB upon arrival.
5. Programming the Custom Board	Continued development of the custom board DTS (Include reference to the process). Once verification of simple MQTT using sample applications was verified, time was spent programming.
6. LED PIO and GPIO Implementation	Tested functionality and LED control libraries to achieve the required functionality (indicate status to user)
7. Server Connection Attempt	Attempted to connect to a custom server to interface with the datalogger server and upload JSON data.
8. Debugging and MQTT Proposal	Debugging bottlenecked progress with communications. Sought assistance from the server team and proposed implementing MQTT. HTTPS resulted in modem faults due to improper certificate provisioning, causing software faults and requiring extensive debugging. Rationale: MQTT broker example is easily programmable and simple to check message requests. Problem: The server/communications team decided not to implement it.
9. Flask Server Connection	Continued debugging, removed non-essential certificates, and reprovisioned the Flask server. Using a simple server validates connection to a secure server not predefined by example code.
10. Further Modifications and Validation	Further modified HTTPS to the custom server, using LED as status indicators. Near the end of the project, proved the concept for HTTPS and MQTT on the development kit. Needed validation on hardware. Rationale: Utilize the LoRa software team to validate LoRa on custom board. Verified custom board was programmable and GPIOs were operational with LED and buttons without verification of LoRa <sup>5</sup> .

**Debugging:** After success with the development board, the stages of development cycled around getting something to work and investing a lot of time in debugging and troubleshooting. Significant hindrances were the TLS provisioning to the datalogger server resulted in a modem fault which took several days to debug. After this, the final testing of hardware and software was planned. The following X represents the stage of making custom software-defined NRF boards. After the appropriate project configurations were tuned, HTTP code with implemented LED

<sup>4</sup> The software team did not manage to implement this by the final presentation.

<sup>5</sup> Software team did not implement software for LoRa hardware/ validate hardware on the custom board.

control was uploaded. However, the current USB implementation is unstable when powered off solely by USB, with recommendations of including a ferrite bead for stability as recommended in schematics. In a final attempt at debugging, (real-time-transfer) RTT debugging was implemented, but the execution was unsuccessful, as RTT is experimental on nRF9160 [48].

## 4.1. Software implementation

Because a Nordic Semiconductor product was chosen, all software development was done in VS Code using nRF-Connect, which offers features like automatic compiling and code execution with WEST as shown in Figure 10. The UI for the device tree, which is how peripherals are defined, simplifies creating and flashing applications to boards, as shown in **Error! Reference source not found**. Figure 10. Setting up a custom board requires more effort but defining it in nRF-Connect was straightforward because of the number of examples and similar designs implementing LoRa. The process involves configuring applications, using Kconfig files for debugging over UART, and mapping GPIOs for peripherals in nRF-Connect. I added initial support for LoRa, and other peripherals: flash, UART supervising the manufacturing of a second board for testing just the LoRa section with rationale that the code for the DTS can be used with the secondary development kit, to help the software team validate the proof of concept of combining LoRa in the Zephyr platform. However, the team did not decide to continue development with Zephyr or Nordic semiconductors, due to issues getting LoRa to work on ESP32 MCUs.

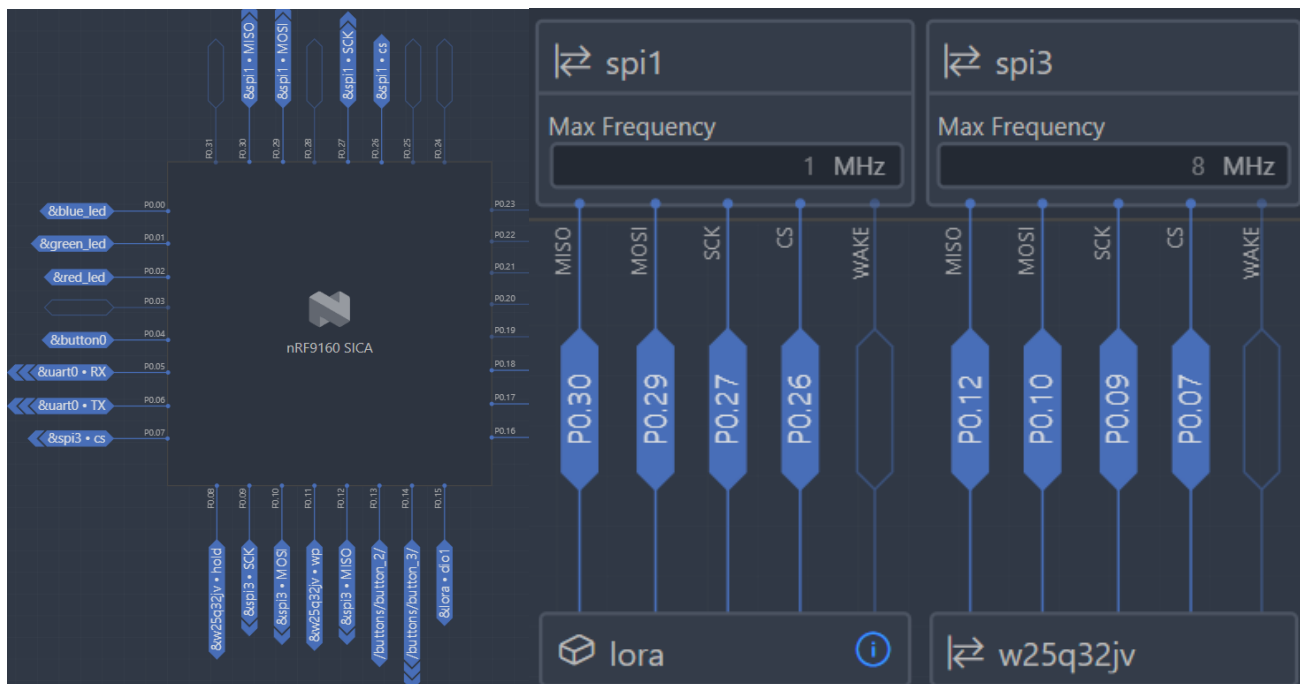


Figure 8: Right nRF Connect GPIO UI for Devicetree, Left nRF Connect peripheral UI for Devicetree



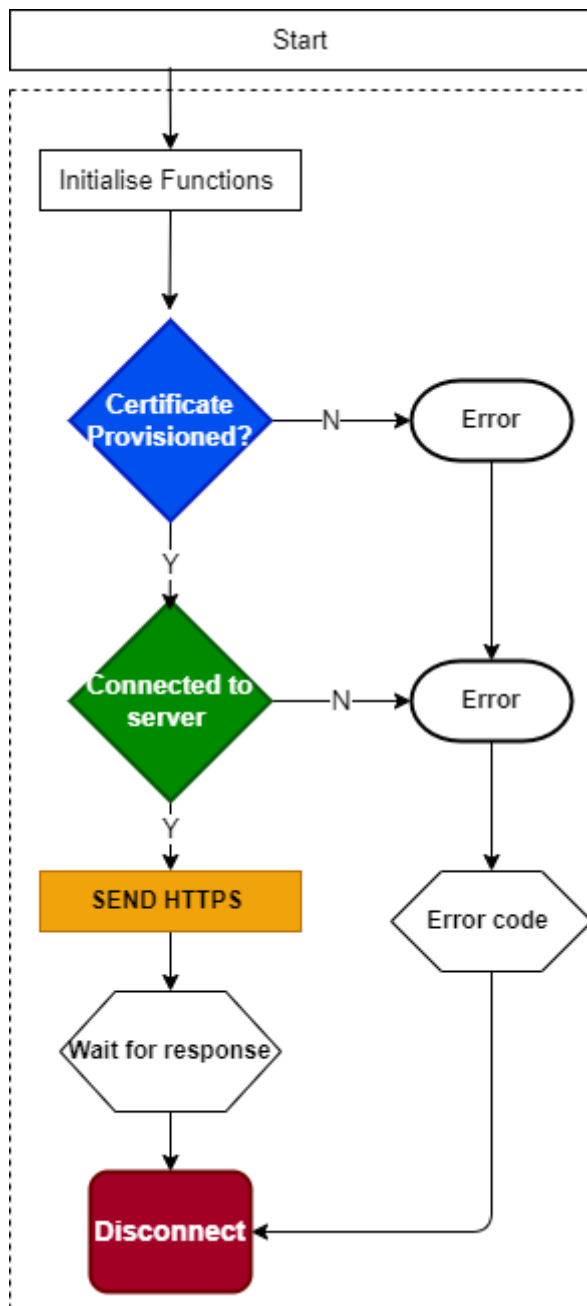


Figure 9: Flow diagram for all HTTP test applications

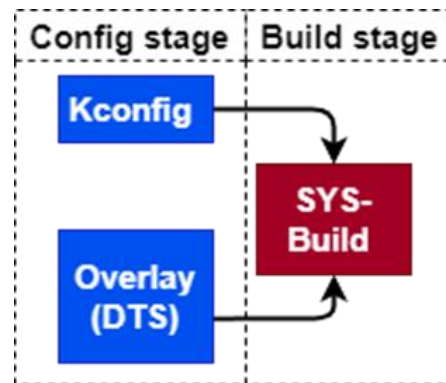


Figure 10: nRF Connect SDK simplified application process. [49]

## 5. Software tests/Validations

The execution for sending HTTP response to a server, for both the DK and custom board is demonstrated in **Error! Reference source not found.** with successful connection and transmission of the development kit shown in Figure 11

```

*** Booting nRF Connect SDK 4040aa0bf581 ***
HTTPS client sample started
Bringing network interface up
Provisioning certificate
Certificate mismatch
Provisioning certificate
Connecting to the network
+CEREG: 2,"B070","00194D02",7
+CSCON: 1
+CGEV: ME PDN ACT 0,0
+CNEC_ESM: 50,0
+CEREG: 5,"B070","00194D02",7,,,"11100000","11100000"
Network connectivity established and IP address assigned
Looking up joshuath.pythonanywhere.com
Resolved 35.173.69.207 (AF_INET)
Connecting to joshuath.pythonanywhere.com:443
Sent 77 bytes
Received 224 bytes

> HTTP/1.1 200 OK

Finished, closing socket.
+CGEV: ME PDN DEACT 0
+CEREG: 0
+CGEV: ME DETACH
+CSCON: 0
Network connectivity lost
  
```

Figure 11: Terminal output of HTTPS request to Flask server, using nRF9160DK



## 6. Hardware testing

Due to troubleshooting software and TLS, not much significant hardware validation was conducted, with the following significant hardware tests and debugging achieved, for lab setup refer to Appendix D.

- Tested compatibility with Li-ion battery supply with lab setup.
- Found that the current USB implementation is unstable when powered off USB and needs a decoupling capacitor and ferrite bead for stability as recommended in schematics.
- UART tested on oscilloscope, either dead or because of component footprint not connected correctly. Debugging UART on oscilloscope resulted in no data transmissions validating that the UART does not work on current PCB, and debugging custom board connection to server is challenging

## 7. Discussion

**Hardware:** The USB-power stage of the device could be improved by integrating a more efficient power management system, reducing the amount of LODs, to improve stability. Additionally, incorporating more robust protection circuitry for LiPo batteries would ensure safer operation in the field. The current USB implementation is unstable when powered solely by USB, and adding a decoupling capacitor and ferrite bead for stability is recommended, however USB with supplementary external power is stable. Revisions to the initial PCB that were manufactured, such as power optimisation are implemented in schematics on git, see Appendix A.

**Software:** The software could benefit from more extensive testing and debugging, particularly in real-world scenarios. Implementing automated testing frameworks and continuous integration could help identify and resolve issues more efficiently. Debugging bottlenecks, such as TLS certificate provisioning issues, JSON formatting, and UART/ RTT debugging need to be addressed in the next implementation.

**Team Management:** Improved communication and collaboration within the team could lead to better integration of hardware and software components. Supervisor feedback and guidance for communication issues should have been considered. Encouraging software team to develop on Zephyr/using one platform initially instead of multiple platforms could expedite development. The decision to not continue development with Zephyr using Nordic Semiconductors MCU due to issues with LoRa on ESP32 MCUs highlights the need for better coordination and decision-making processes.

## Acknowledgements

To enhance the clarity and logical flow of my report, I utilized generative AI to convert bulleted lists, such as my log of hours into tabular form. This approach helped me succinctly present a timeline of decisions, implementations, and rationales. For instance, I used prompts like “Convert this draft timeline into tabular form”. After generating the initial table, I manually modified it in Excel to ensure accuracy and coherence before incorporating it into my report. This iterative

process allowed me to maintain a consistent and professional presentation throughout the document.

## Group Member 3 Conor Dunlop

### 8. Introduction/Background

The focus of this project is to assist with the gathering of data from the environment as well as remotely interfacing with the data. Therefore the design problem of this project is: how to gather environmental data in a modular manner and access said data remotely. The specification is to create a universal modular datalogger with IoT capabilities as previously outlined. This data logger is broken down into two main parts, the physical data logger itself and the cloud-based access to the data. Both parts also have additional requirements set that can be found in Table 15: User requirements. These requirements are defined by their priority, a requirement with a hard priority must be met to ensure the success of the project. This contrasts with a soft priority that does not need to be met for the project to be deemed successful.

Functional		
1.1	Connecting two or more sensors, e.g. temperature, light, camera, etc.	Hard
1.2	Controlling two or more outputs, e.g. LEDs, solenoid/relay	Hard
1.3	Ability to use multiple IoT technologies	Hard
1.4	Implement LoRaWAN for communications	Hard
1.5	Implement cellular for communications	Hard
1.6	Have satellite (non-terrestrial) based IoT	Soft
1.7	Implement both Wi-Fi and Bluetooth communications	Soft
1.8	Receive information and give commands	Hard
1.9	Ability to interface and control outputs through a web portal	Hard
1.10	Web portal to display data collected from data loggers	Hard
1.11	Web portal to show status of data loggers	Hard
1.12	Use a database to store the data from the data logger	Hard
1.13	Use some form of battery/solar to power the device	Hard
Quality		
2.1	Enclosure to protect from the elements	Soft
2.2	Content management system to assist with layout of data	Soft
Constraints		
3.1	Preferable cost of product under \$100 per data logger	Soft
3.2	Absolute cost of product under \$250 total	Hard
3.3	System costs under \$20 to run per month per data logger	Hard

Table 15: User requirements

As my role was software for the data collection board the main requirements I worked to fulfil were 1.1 and 1.2 to interface with the inputs and outputs of the board. Additionally, as a mechatronics engineer, I was the most mechanically minded so I also took on 2.1 to specify the enclosure.

At the beginning of the project during the ideation phase, I helped develop the intended use case that we were primarily designing for which with the consultation of the sponsor came to be an occasional sampling of any connected sensors, sending that data over some coms channel then entering an energy mode for an extended period. Additionally, I assisted Logan in the decision-making process to select the MCU and relevant I/O to design for. This was important to ensure that the device would be able to capture all necessary information. While Logan did most of this, as a Mechatronics Engineering student, I have a deeper knowledge of what actuator and sensor I/O is most commonly used. My main focus was advocating for the introduction of many ADC channels as most common sensors have a variant that can run on 1 or 2 ADC channels such as temperature sensors [50], humidity [51], light [52] and pressure [53]. Another focus was on SPI and I2C but these are mostly software addressable so fewer inputs were needed for this they were still classed as necessary such as water Electrical Conductivity sensors [54] which were specifically requested by the sponsor to be compatible with. The other non-software task I had was to determine an appropriate housing type for the circuit boards.

## 9. Work Undertaken

Due to the selection of an STM microcontroller and Logan's familiarity with it STMCubeMX and STMCubeIDE were used to allow the schematics of boards to be automatically initialised and addressed as they should be with the STMCubeIDE creating its own skeleton code shell to work from. Extensive research was done into the Microcontroller we are using which is the STMG431RBTx. This has involved primarily reading the datasheet [55] to extract useful information on how to optimally utilise its features such as deciding on which low-power mode fits our application the best and Table 16: Table 29 of the STMG431RBTx datasheet. of the data sheet along with the different wakeup options available to each of the low power modes Stop Mode 1 [56] was chosen for use in the long delays expected for this project as it has a low power consumption as do all the Stop Modes but additional it can be woken by the Low-power timer (LPTIM1) which the others can't, leading to the lowest passive current drain.

Table 29. Current consumption in Stop 1 mode

Symbol	Parameter	Conditions		TYP						MAX <sup>(1)</sup>					Unit
		-	VDD	25°C	55°C	85°C	105°C	125°C	25°C	55°C	85°C	105°C	125°C		
IDD (Stop 1)	Supply current in Stop 1 mode, RTC disabled	RTC disabled	1.8 V	58.5	175	550	1050	1900	430	1400	3600	6500	11000	μA	
			2.4 V	58.5	175	550	1050	1950	430	1400	3600	6600	11000		
			3.0 V	59.0	175	555	1050	1950	430	1400	3700	6600	11000		
			3.6 V	59.5	180	560	1100	1950	430	1400	3700	6700	11000		
IDD (Stop 1 with RTC)	Supply current in Stop 1 mode, RTC enabled	RTC clocked by LSI	1.8 V	59.0	175	550	1050	1950	430	1400	3600	6500	11000		
			2.4 V	59.5	175	555	1050	1950	430	1400	3600	6600	11000		
			3.0 V	59.5	175	555	1050	1950	440	1400	3700	6600	11000		
			3.6 V	60.5	180	560	1100	1950	440	1400	3700	6700	11000		
		RTC clocked by LSE bypassed at 32768 Hz	1.8 V	58.5	175	550	1050	1900	-	-	-	-	-		
			2.4 V	59.0	175	555	1050	1950	-	-	-	-	-		
			3.0 V	60.0	180	555	1050	1950	-	-	-	-	-		
			3.6 V	62.0	180	565	1100	1950	-	-	-	-	-		
		RTC clocked by LSE quartz in low drive mode at 32768 Hz	1.8 V	58.5	150	445	890	-	-	-	-	-	-		
			2.4 V	59.0	150	445	890	-	-	-	-	-	-		
			3.0 V	59.5	150	445	890	-	-	-	-	-	-		
			3.6 V	61.0	150	450	895	-	-	-	-	-	-		
IDD (Stop 1 with RTC)	Supply current during wakeup from Stop 1 mode	Wakeup clock is HSI = 16 MHz,	3.0 V	1.39	-	-	-	-	-	-	-	-	-	mA	
		Wakeup clock is HSI = 4 MHz, (HPRE divider=4), voltage Range 2	3.0 V	0.93	-	-	-	-	-	-	-	-	-		

1. Guaranteed by characterization results, unless otherwise specified.

Table 16: Table 29 of the STMG431RBTx datasheet.

The possibility of using a Real-Time Operating System (RTOS) was considered with Microsofts FreeRTOS being the main one considered due to its integration with the STMIDE already established and available. An RTOS would allow for multiprocessing providing some marginal benefits to the speed of operation and would help greatly in debugging errors as the code could run as normal while handling real-time interaction ultimately an RTOS wasn't deemed worth using as speed isn't necessary and uses more power which is a large focus for this project. For ease of use, understandability and power efficiency a simple round-robin execution style was decided on which can utilise very long delays with the STMG4s low power timer that can be down-clocked to provide the very long time between potential sensor measurements and transmissions.

As my role is primarily embedded software-based which is only possible to do with the target to program I have also taken on a number of the smaller administrative tasks required to keep the project running such as the creation of the meeting lead and minutes taker roles each week. Once there was a more finalised main board design I began to develop the code that would run on it. STMCube programs were used to adapt the pinout of the MCU to pre-generated code creating and importing the Hardware Abstraction Layer (HAL), FatFS and other libraries.

The structure of a ports and adapters software structure was decided on as it fit well with the STM HAL library so The code has been made to work towards the structure shown in Figure

### 13: software architecture diagram

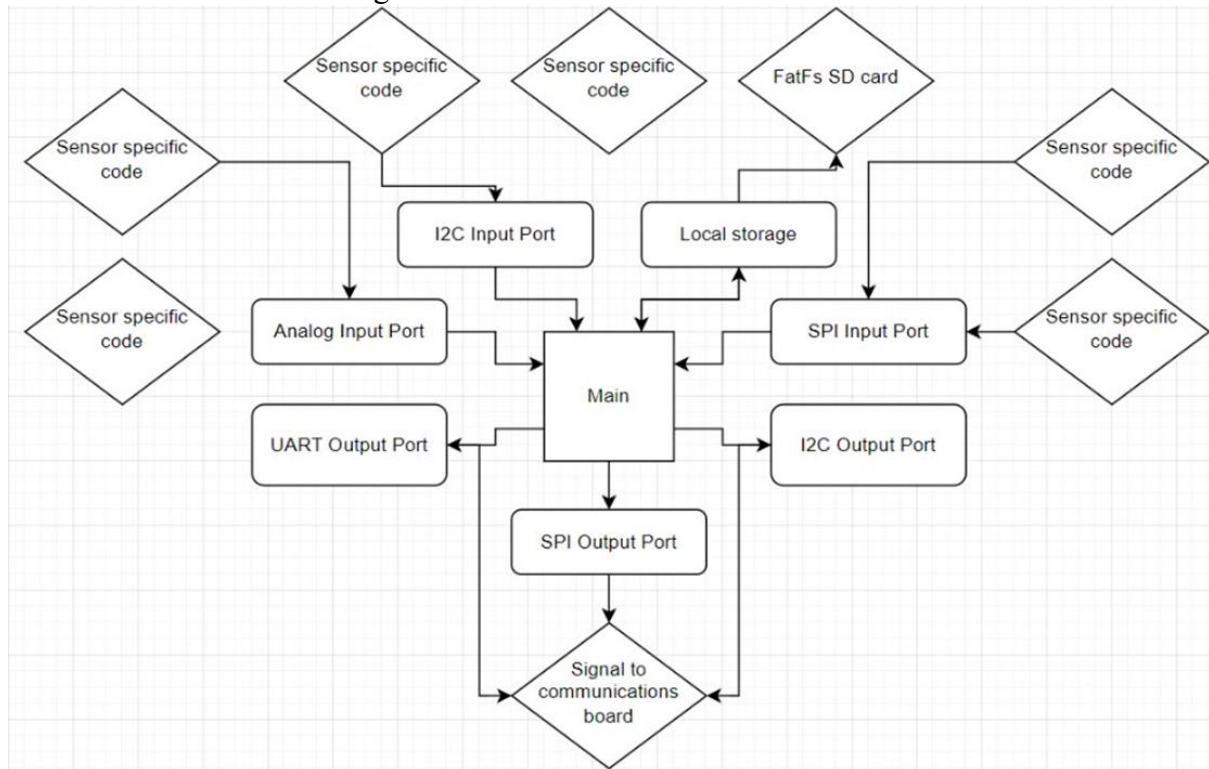


Figure 13: software architecture diagram

For the first board iteration testing for the I/O to find what components worked correctly and what needed changing for the second iteration was completed. Aside from the SPI which had intermittent errors and the I2C which did not work at all everything else worked as expected so this information was passed on for redesign.

For the second board code for the UART and SPI output blocks has been developed and tested successfully to interface with the various forms the coms board has taken, this also involved creating various skews of the STM32CubeMX file with the SPI coms output setting shown in Figure 14: STM32CubeMX SPI output skew

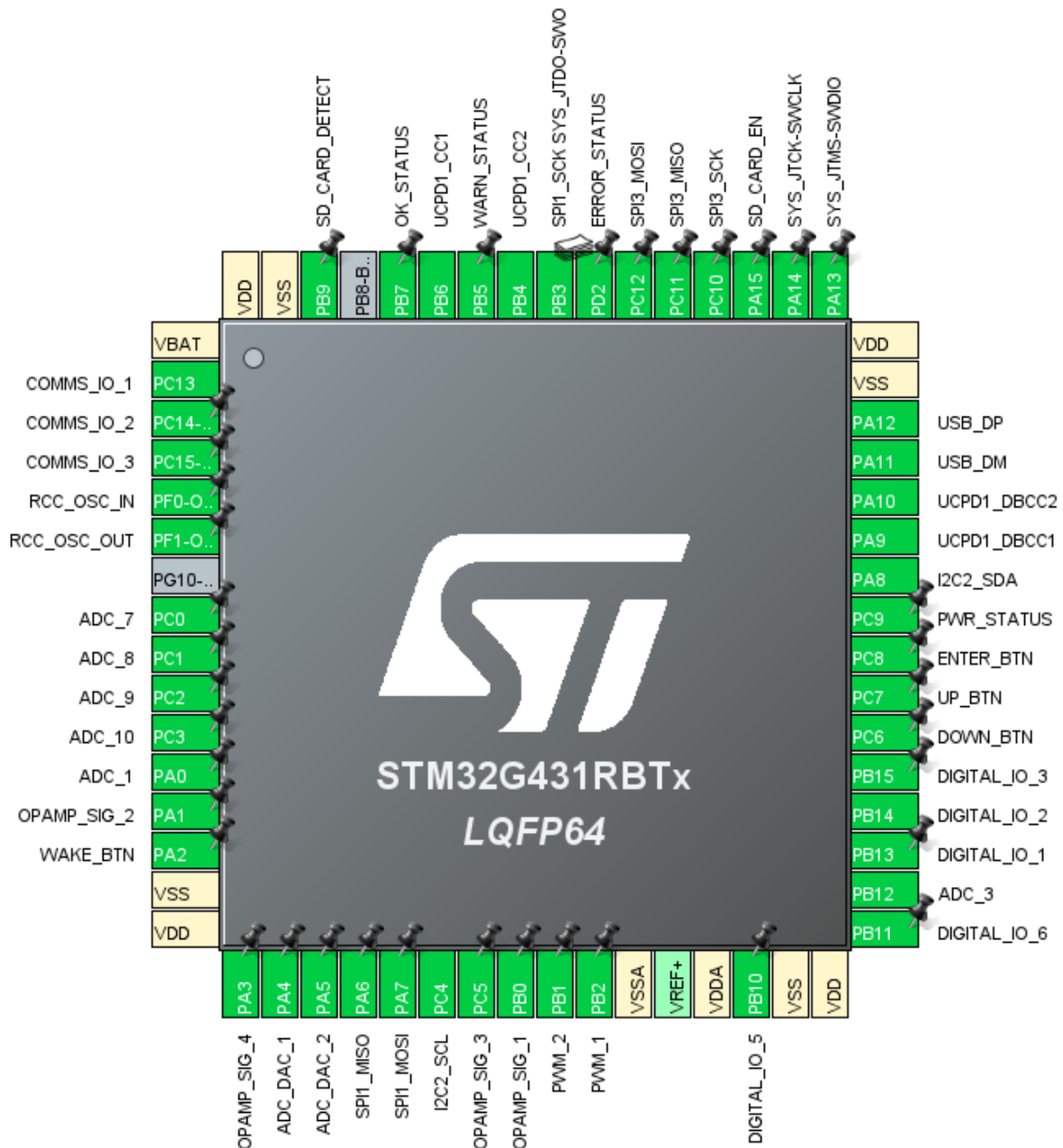


Figure 14: STM32CubeMX SPI output skew

I2C input and one sensor-specific code block have been developed and tested successfully to read an MCP9804 temperature sensor and convert its readings into the real temperature, this was used for one of the demonstrations/ inspections. Local storage code has been written using FatFS but has not been tested. Analogue input and output as well as internal and external ADC input have also been developed and tested successfully.

For the container, a suitable general-purpose solution was selected and some general requirements were selected. These requirements were water resistance specified to be at least IP55 to remain watertight even in heavy rain and resist dust or insect entry. The container material chosen is ABS as it is the ideal housing material as out of plastic containers it has the

greatest UV resistance [57] and has no issues in expectedly cold environments. This led to the decision to look for a container in that plastic which led me to the CU-1941 [58] which also has UL94 HB [59] flame resistance while the lowest classification of that flame resistance test still achieves a rating that is fine for most use cases, is RoHS [60] compliant and as a thermoplastic is exceptionally recyclable [61].

## 10. Discussion

I achieved 2.5 out of the 3 requirements I set out to complete as the device can read from 2 sensors with the dev kit version of the coms board successfully transmitting the expected data in the correct format and specifications for an oscilloscope reading if this can be seen in Figure 15: Readout of signal transmission.

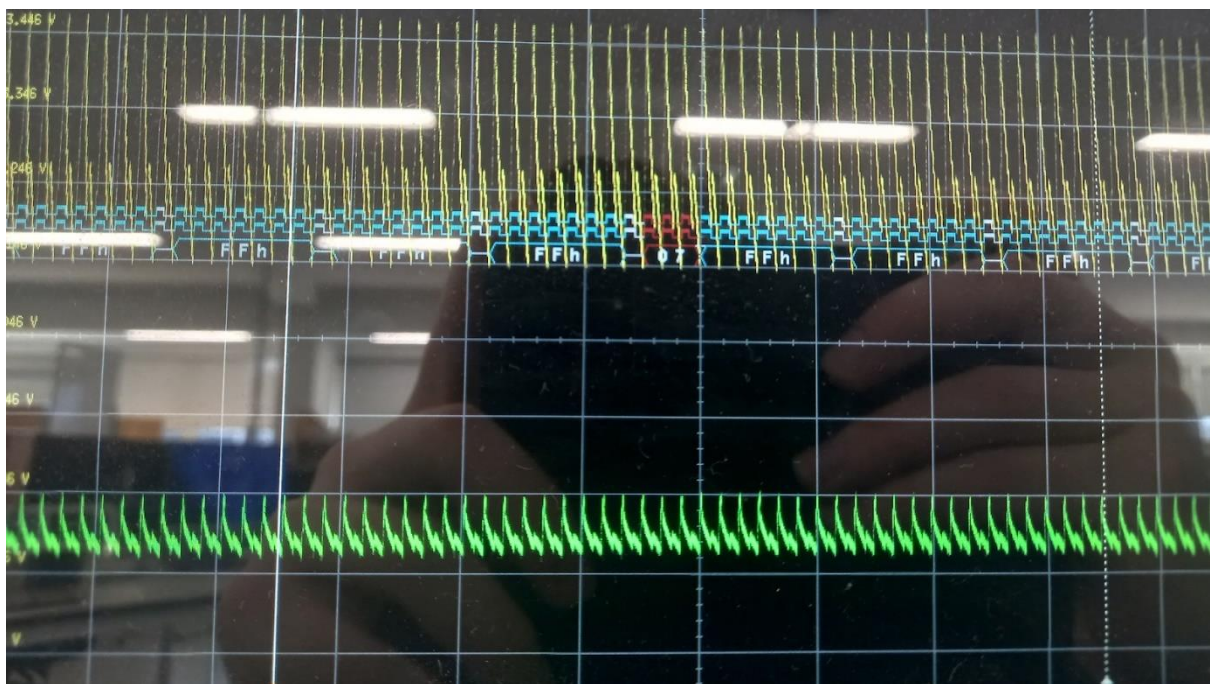


Figure 15: Readout of signal transmission

Overall I achieved most of the progress I expected to do aside from in-field testing which was outside of my control and am leaving my section of the project in a usable state that is ready for future improvements or use by any group that decides to take up this project. All of my software-based work is documented and laid out in the projects git repository in my section and is to my eye laid out well enough to be followed by any future groups.

An overview of the remaining tasks in this section of the project is to finish writing code to extract and parse data from more sensors and actuators to finish the specifications and make them meet the intended use case not merely the minimum requirement to build up a standard sensor and actuator suite for a wide variety of I/O for the main board. Changes to the existing working code could include switching to UART1 from UART4 to enable the use of LPUART1 the low-power UART on the G4 chip and enable low-power timer to be able to use stop mode 1 instead of delay functions.

Once the coms shields are produced integration with its requirements into the existing main board code to send out the gathered data. The code also currently should load any received data into a Micro SD card (not tested). Integration with the final comms modules will take some existing code from the summer project this FYP is based on and some code from MBED and Zypher which the original comms modules are designed for and integrate into the existing code to correctly send the data out of the board to whichever target is requested.

The software shields integrate well with the main board code and I have been looking into this step as the coms board is designed based on another existing shield that used Zypher [16] and MBED [62] for its LoRA communication code so I have also been testing this code and will need to incorporate some of it into the existing main code as well as likely assist with the coding the coms shields.

Following the in-field testing aside from fixing any unforeseen issues that arise the remainder of the time will be spent on user experience and documentation leading up to the final inspection and report. My tasks will be to add/ improve the UI for the device with the planned interaction buttons, feedback LEDs used to show error codes and a small screen for instructions or more detailed advice.

This project will require more documentation as it is intended to be used as a starting point for other people and organisations to start collecting data relevant to them without the significant upfront time or monetary investment that such a project would normally take.



## Group Member 4: Chloe McLaren

### Introduction

The area of the project I was responsible for was the cloud infrastructure and data visualisation for the Universal Modular Datalogger, the main requirements related to the cloud infrastructure and web-based portion of the project can be seen in Table 17. To meet these requirements the cloud infrastructure was split into three primary areas: the web API, the database and the user interface, along with a data format to transmit data between the dataloggers and web API.

The web API receives data from the dataloggers and processes it so it can be stored in the database and returns data to the dataloggers from the user.

The database stores data received from the dataloggers along with information about the datalogger such as whether it is functional and its location.

The user interface displays data to the user graphically and allows the user to control actuators remotely.

The data format encapsulates the data in an easily transmittable format with information such as the logger ID, sensor ID, timestamp, etc.

A block diagram for how these communicate with each other can be seen in Figure 16.

Table 17: Cloud Infrastructure Requirements

	Project Requirements
Quality	
2.2	Device receives information, Users can give commands, and interface with outputs through a web portal.
2.3	Web portal functionalities (displaying data, showing status of data loggers)
2.4	Data management (using a database to store data)
Constraints	
3.1	The datalogger system shall have a cost of product under \$250 total
3.2	The datalogger system shall cost no more than \$20 per month per datalogger

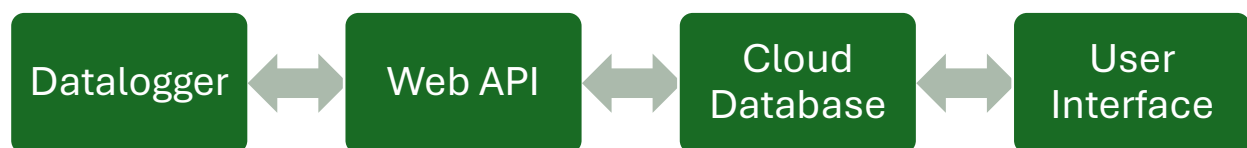


Figure 16: Cloud Infrastructure Block Diagram

To satisfy these requirements I have created a SQL database and web API hosted by Microsoft Azure with data visualization provided by Grafana cloud. This is completely free for our level of usage during development.

SQL was chosen due to its widespread use and integration with many services. Azure was chosen due to being free at expected levels of usage, having built-in IoT functionality and having an unlimited timeframe for free usage. Grafana cloud was chosen for data visualization as it provides an easy-to-use, fully functional interface to view and create graphs using data from the database.

## Work Undertaken

An Azure cloud SQL database has been created, a web API hosted on Azure app services has been created and a data visualizer setup using Grafana which automatically polls the database to generate graphs of inputted data. Data formats have been designed for communication between the API and dataloggers. At expected levels of use during development of the datalogger these options are completely free.

### Database

#### *Language*

SQL was chosen for the database language as this is most commonly used database language with 4 out of 5 of the most popular database languages being different versions of SQL [63], and most developers have experience with it as it's the second most used programming language with 54.1% of professional developers using it extensively over the past year [63] and is available with all major cloud service providers, allowing for easy switching between cloud providers (or a locally hosted database) if necessary.

#### *Cloud Provider*

To decide on an SQL cloud provider, their cost, storage and processing power was considered, the main three cloud providers examined were Amazon Web Services, Azure and Google Cloud due to their popularity and thus better documentation and integration with other services. Azure was decided on due to its low cost, good documentation, unlimited-length free tier database, integrated IoT services and wide array of other free services which could be of use to the project as seen in Table 18 [64]. Due to the modular nature of the device and simple web architecture (SQL database and webapp), a user could choose a different cloud provider or a local server instead and the only change to the datalogger would be which address it sends and receives data to/from.

As we were unlikely to be storing or transferring a large amount of data during the development phase of this project, I decided on a cloud provider based on their freely available SQL databases and length of time it remains free.

Table 18: Costs of Cloud SQL database

	Amazon Web Services [65]	Azure [66]	Google Cloud [67]
Free Storage	20GB	32GB	0GB
Free usage/month	750hr	100000vCore-seconds	None
Free for how long?	12 months	Forever + \$200 used within 1 month	\$300 used within 3 months
Cost of usage	\$1.81/hr (4 vCores, 16GiB ram)	\$1.3092/vCore/hr	\$0.091/vCore/hr
Cost of storage/month	\$0.138/GB/month	\$0.233/GB/month	\$0.23/GB/month
Data transfer costs	In: \$0	None	In: \$0

	Out: \$0.114/GB (free for <100GB/month)		Out: \$0.19/GB
Licensing costs	None	None	\$0.13/vCore/hr
Other features		Auto-pause, IoT services	
Estimated cost per month <sup>6</sup>	\$1329.72 [68]	\$14.90 [69]	\$328.52 [70]

Azure SQL database has a maximum storage of 32 GB at free tier, with 100,000 vCore seconds/month for an unlimited amount of time, in the unlikely event we go over this amount during development, pay-as-you-go can be enabled at a price of \$0.0003637/vCore-second, and more storage can be purchased for \$0.233/GB/month [71]. To stay below the \$20/month/datalogger requirement, each datalogger would need to stay below 55000 seconds (~15 hours) of 100% utilization of a core per month, this should be more than enough for each datalogger to send/receive data regularly. More storage can be purchased as necessary or older records could be deleted when more space is needed.

### Schema

The current schema for the database allocates a maximum of 50B of memory to data and ~100B to other information such as timestamp, logger ID, sensor ID, etc as shown in Figure 17. With the free limit of 32GB of storage this allows for ~210 million records to be stored at minimum. The maximum amount of data allocated to each record can easily be changed if necessary, but may lead to data loss so should be avoided when used in production. More specific applications such as images and video will need significantly more data (MB-GB range) so will need to be kept in a separate database table if future developers decide to implement camera connection (which may not be technically feasible due to connection speeds and packet size of communication protocols).

```
CREATE TABLE DataStorage
(
    id UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
    LoggerID [NVARCHAR](50) NOT NULL,
    SensorID [NVARCHAR](50) NOT NULL,
    ReceiveTime DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    ReceivedData [NVARCHAR](50) NOT NULL,
    DataType [NVARCHAR](50) NOT NULL
)
```

Figure 17: Current database schema

A future schema seen in Figure 18 has also been developed so that information about each datalogger such as their sensors, location, etc, can be stored more effectively. This schema also uses binary rather than variable characters for data storage so that data can be stored more

<sup>6</sup> Using 4 virtual CPUs, 32GB storage, 100% utilization, from each service's pricing calculator

efficiently. This schema was not implemented due to time constraints with manufacturing and integration of the datalogger.

```
CREATE TABLE LoggerInfo
(
    LoggerID [NVARCHAR](32) PRIMARY KEY,
    Location geostamp
)
```

```
CREATE TABLE SensorInfo
(
    LoggerID [NVARCHAR](32) PARTIAL KEY FOREIGN KEY REFERENCES(LoggerInfo),
    SensorID [NVARCHAR](32) PARTIAL KEY,
    DataType [NVARCHAR](32) NOT NULL
)
```

```
CREATE TABLE DataStorage
(
    UUID UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
    Sensor FOREIGN KEY REFERENCES(SensorInfo),
    LogTime datetime NOT NULL,
    Data [VARBINARY](128) NOT NULL
)
```

Figure 18: Future database schema

## Web API

The web API has been developed for Azure web services using a C# dotnet backend, this was chosen due to the extensive documentation and examples given by Microsoft, to prevent the need to use other service providers, and because it is included in the free tier of Azure. The web API processes HTTP requests from the datalogger gateway (or other service) to store data in or get data from the database. HTTP was chosen for this as it is more universal than other protocols, due to being the standard transfer protocol for internet communication. This means that the web API can easily integrate with any other internet-capable device to log data. The web API can handle logData, getLoggerData and getAllData requests, logData stores data in the database, getLoggerData gets data from a specific datalogger and getAllData gets all data from the database.

A datalogger emulator has been developed in conjunction with Logan to simulate sending data from the datalogger or getting data from the database so that the cloud infrastructure can be tested and to allow for new formats to be more easily developed.

## User Interface

A web-based app was decided on for the user interface as this can be accessed from anywhere and is more universal than a dedicated .exe application. It is significantly less work to update and develop for multiple devices (different app stores/programming languages, waiting on

validation for updates, paying license fees, etc), and is easier for the user to modify a webpage for added accessibility (screen readers, etc) or personal preferences (change colours, fonts, etc). Azure offers many options for deploying webapps such as App Service, Container Apps, Functions, Service Bus, Service Fabric and Static Web Apps.

Grafana Cloud was chosen to visualize data as this has a free tier and provides database integration, different graph types, data transformation, downloadable data, is available online, has automatic alerts and is very accessible for users. Some visualised test data and Grafana's easy-to-use query editor can be seen in Figure 19 and Figure 20.

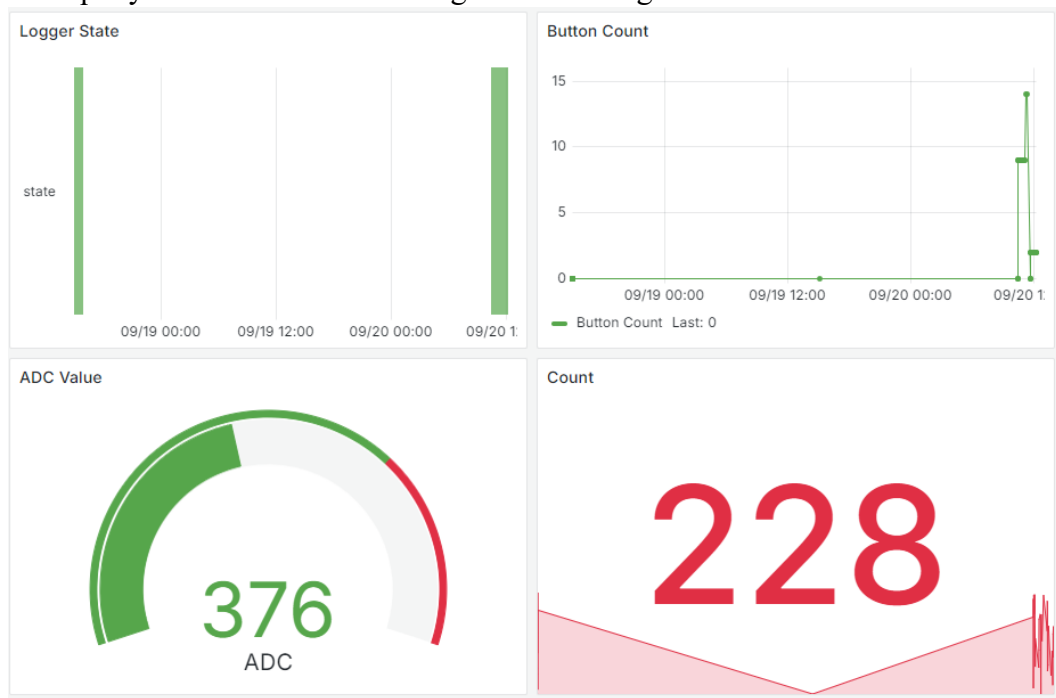


Figure 19: Data shown in different formats using Grafana

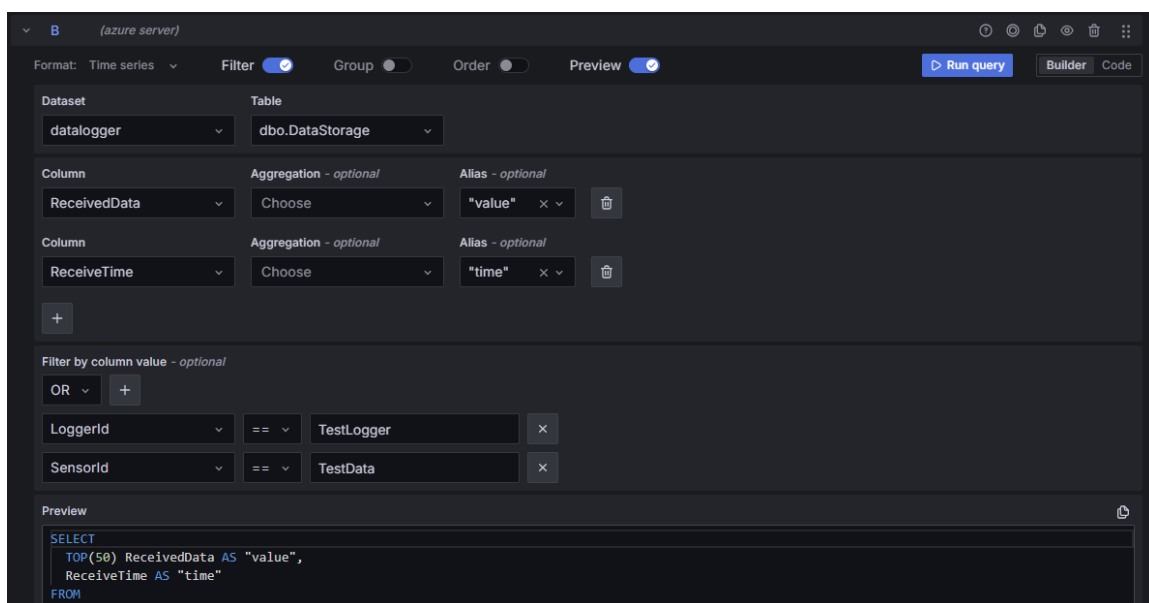


Figure 20: Grafana query editor

Data can be sent back to the database to allow actuators to be controlled with buttons or other HTML elements such as input fields as can be seen in Figure 21.

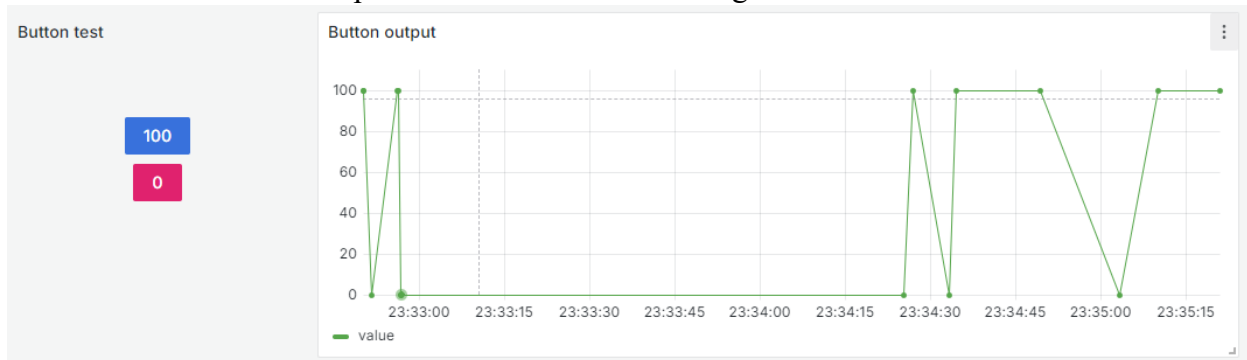


Figure 21: Sending data to dataloggers using Grafana

Locations of dataloggers is shown with the built in Grafana location visualiser which uses OpenStreetMap, an open-source mapping data provider [72]. This can be seen in Figure 22.

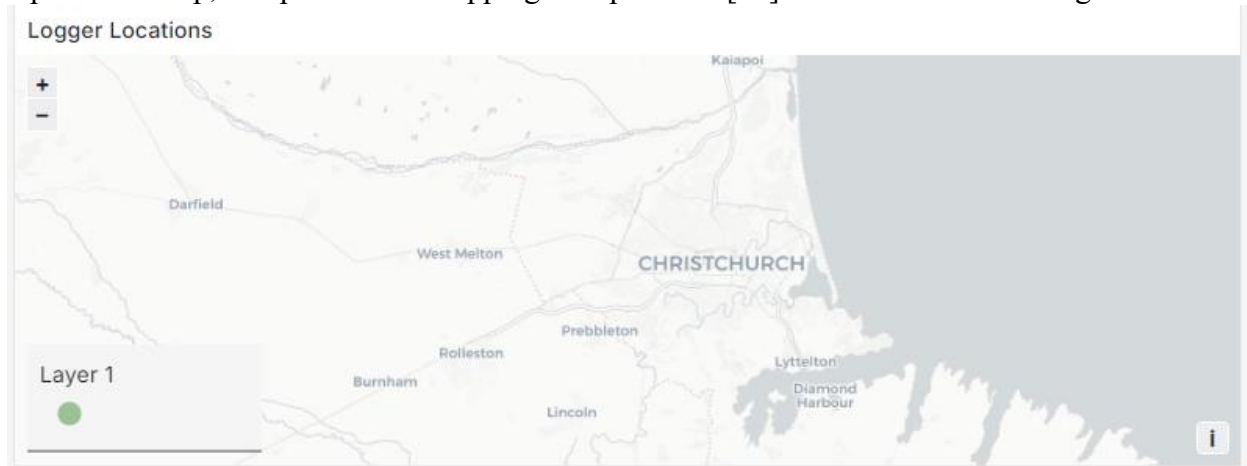


Figure 22: Datalogger location monitor<sup>7</sup>

### Communication Format

JSON has been chosen as the data format between the datalogger and the server as this is the most widely used format for IoT applications and there is plenty of documentation/example code. There is also built-in functionality with most SQL server providers that allows for simple conversion to/from JSON. The current JSON data format holds the data, Logger ID, Sensor ID, timestamp and data type. This provides everything necessary to determine important information such as when the data was sent, what sent it, and what type of data was sent. Timestamp can be omitted to instead use the time of arrival to the database.

<sup>7</sup> No dataloggers were online at this time

```

{
  "loggerID" : "string",
  "sensorID" : "string",
  "timestamp" : "datetime",
  "data" : "data",
  "datatype" : "string"
}

```

Figure 23: Current JSON format

A future JSON format was also designed for use with the future database schema to allow data such as the sensors, location and state of a datalogger to be more easily stored and need to be sent less often, such as only on startup. It would also allow multiple datapoints to be sent at the same time more easily and reduce the number of bytes needed to be sent per data transmission. This format can be seen in Figure 24 and Figure 25.

```

{
  "loggerID" : "string",
  "location" : "locationdata",
  "sensors" : [
    { "sensorID" : "datatype" },
    { "sensorID" : "datatype" },
    ...
    { "sensorID" : "datatype" },
  ]
}

```

Figure 24: Future JSON format, information sent on startup

```

{
  "loggerID" : "string",
  "data" : [
    { "sensorID" : "data", "timestamp" : "datetime" },
    { "sensorID" : "data" },
    { "sensorID" : "data" },
    ...
    { "sensorID" : "data", "timestamp" : "datetime" }
  ]
}

```

Figure 25: Future JSON format, information sent when logging data

## **Discussion**

The cloud infrastructure for the Universal Modular Datalogger project meets all relevant requirements, storing and showcasing data to the user while being completely free at current usage. To better meet the requirement to control two actuators the cloud infrastructure should automatically send output data to the datalogger when a user changes an actuator's state, rather than need the datalogger to check with the database to see if the state has changed. Future work should also implement the new schema and JSON format and implement security measures such as API keys to prevent malicious users from accessing data they should not be able to.

The cloud infrastructure has been tested with many different data formats using the datalogger emulator including integers, floats, text, location data, etc. As the data is stored as text theoretically all data types can be stored, however some such as video, photos and audio may be impractical to store due to the amount of storage required. The cloud infrastructure works as intended with the actual datalogger hardware, properly storing and showing the data received to the user.



## **Group Member 5 Joshua McCorkindale**

### **Introduction**

This part of the project is responsible for the design and implementation of the communication protocol LoRaWAN. It involves the research and design of LoRaWAN solutions, including testing with MultiTech and Heltec devices. Utilizing the SX1262 radio module and the STM32 microcontroller. The aim was to create a low-power, long-range communication solution which is suitable for Internet of Things (IoT) applications. This section details some background research and technical considerations that informed the development part of this project.

LoRaWAN is a popular communication protocol that is designed for long-range, low-power applications. It operates on the unlicensed 915 MHz frequency band in New Zealand making it ideal for IoT devices that require energy efficient communication without the need for subscription-based services.

LoRaWAN offers several advantages over other communication protocols including long-range transmission capabilities, low-power consumption being able to last year's depending on the class of the device. It also offers scalability being able to support thousands of devices at a time, additionally, it also has encryption techniques that ensures that all data is transmitted securely. The primary objective of this part of the project was to ensure it adhered to the project's user requirements, including low-power operation, and cost-effectiveness.

### **LoRaWAN Communications**

Low Power Wide Area Networking (LoRaWAN) is a form of radio communications which uses the 915 MHz band in New Zealand. It was specifically designed for battery-operated devices [73] and is widely used in Internet of Things (IoT) applications. It is capable of bi-directional communications being able to send and receive data, which makes it useful for devices that may need firmware updates or to be remotely controlled. LoRaWAN is also a part of the unlicensed radio spectrum, this means that there are no subscription-based services required to operate LoRaWAN devices [74]. Some key features of LoRaWAN include [74]:

- Long-Range Communications: It can transmit data over multiple kilometres in rural areas and a few kilometres in urban environments such as a city.
- Low-Power Consumption: LoRaWAN devices can operate for several years on small batteries due to its low power energy efficient design.
- Scalability: LoRaWAN networks can support thousands of devices using a single base station, making it suitable for very large IoT projects.
- Security: LoRaWAN has multiple types of encryption techniques that ensures confidentiality of the data transmitted
- Flexible Topology: It supports both public and private network setups and can be connected in multiple different ways.

## **LoRaWAN Classes**

LoRaWAN has 3 different classes of devices Class A, Class B, and Class C [75].

Class A devices are designed to maximize battery life, making them ideal for applications where low power consumption is a high priority, with a small amount of communication with the server. Class A devices are high-latency and can send uplinks whenever they want. Examples of Class A end devices include environmental monitoring, animal tracking, asset tracking, and waste management [75].

Class B devices have more frequent communications with the server for uplinks and downlinks compared to Class A but still prioritizes power efficiency. They open receive windows called ping slots that are scheduled to receive downlink messages and are relatively low latency. Class B devices are also used for firmware updates of devices. Examples of Class B devices include utility meters and streetlights [75].

Class C devices are designed for applications where the priority is responsiveness to downlink messages, with less concern for battery life. These devices are constantly listening for messages except for when they are transmitting. They are very low latency devices compared to Class A and Class B as a result they cannot be battery powered for very long, for this reason they are usually connected to the mains power. Examples of Class C devices include utility meters, streetlights, beacon lights, and alarms [75].

## **LoRaWAN Architecture**

LoRaWAN follows a star-of-stars network architecture which can be seen in Figure 25.

Comprising of the following elements [76]:

- **End Devices (Nodes):** These are battery operated devices or sensors that are equipped with LoRa radio modules that send data to gateways. End devices can be sensors measuring temperature, humidity, or any other form of environmental monitoring or sensor. These typically operate in the Class A, B, or C modes, which balance between power consumption and communication frequency.
- **Gateways:** Gateways receive data from the end devices and forward it to a centralised network server using IP based connections such as 4G LTE or ethernet. Gateways are also stateless this means that they simply relay information without storing or processing any of it.
- **Network server:** The network server is responsible for processing the data received from any gateways, they ensure proper routing, device authentication, and security. They also manage Media Access Control (MAC) commands to optimize efficient network communication.
- **Application Server:** The application server is where all of the collected data is stored and processed for end-user applications, such as dashboards, analytical systems, and alerts.

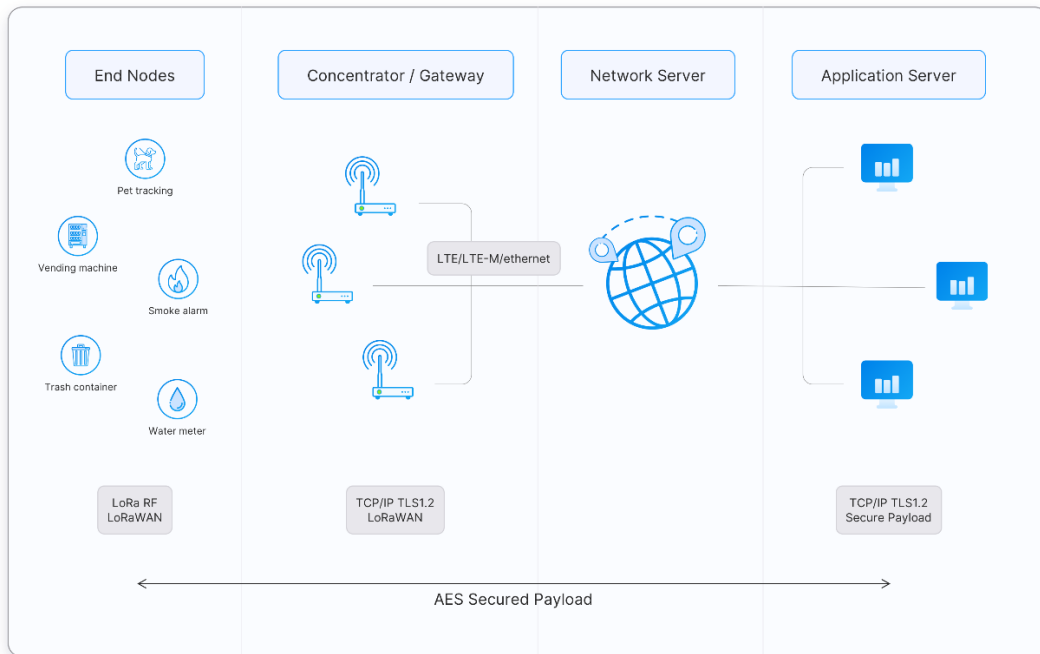


Figure 26: LoRaWAN Architecture

The end devices can be connected in multiple different topologies such as mesh and star topologies. Mesh topology is where multiple end devices are connected to each other before connecting to a gateway to send information to the server from multiple locations. A star topology is where multiple end devices are connected to a singular point before it connects to the server. Alternative methods of communications were also investigated such as satellite and Sigfox, but at this current stage satellite is very expensive requiring pricey chips [77]. It is also not unlicensed meaning that there are subscription services that require a monthly subscription to operate the device. Sigfox is another form of Low Power Wide Area Networking (LPWAN), although it is slower and has less security than LoRa [78]. This may have been a contender over LoRaWAN however in previous years the company has filed for bankruptcy and is now less commonplace in the IoT industry [79].

### LoRaWAN Development Devices

Multiple types of LoRaWAN devices such as the MultiTech xDot and the Heltec Wireless Stick V3 was investigated and tested.

The MultiTech xDot is a compact, low-power, ARM MBED programmable, LoRaWAN device. It contains an internal STM microcontroller and a SX1262 LoRa radio module. It has several useful features including [80]:

- AES-128 encryption to send and receive secure data transmission.
- Multi-band support allowing for multiple frequencies, making it adaptable for different countries.
- Long-range capabilities being able to transmit over several kilometres in rural and urban areas during line of site conditions.
- Low-power consumption, being designed to be battery operated for long periods of time.

The MultiTech comes with its own dedicated libraries and programs out of the box [81] which include several example programs. It runs off the Arm Mbed OS, it was recommended to use

Arm's online compiler called Keil Studio. However, during testing of this Keil Studio, it was found to have many issues compiling after initial setup. Instead of using Keil Studio, it was found that the best compiler to use was a desktop app called Mbed Studio. Mbed Studio made programming the xDot very simple, it also allows for AT commands to be run on the device more easily.

The Heltec Wireless Stick V3 is a small, compact LoRaWAN IoT development board designed primarily for IoT projects [82]. It is based on the ESP32 microcontroller which has both Bluetooth and Wi-Fi and it also contains a SX1262 LoRa radio module. It contains several useful features including [83]:

- An integrated OLED display which can be used for displaying sensor readings, network status or debugging information.
- Built-in battery support with a lithium battery changing circuit and battery voltage monitoring.
- Comprehensive GPIO and peripheral containing 21 GPIO and multiple interfaces such as I2C, SPI, UART, ADC, DAC, and PWM.
- Built in temperature and hall sensor on board the ESP32.

The Heltec Wireless Stick V3 also comes with vast dedicated libraries that contain many different programs. It was built in mind of being an easy-to-use device. This means that it is programmable using the Arduino IDE. It contains several examples including basic LoRa transmission to Over the Air Activation (OTAA) LoRaWAN transmission to a server.

### **LoRaWAN Networking Services**

LoRaWAN networking servers manage and handle the coordination of LoRaWAN devices by handling key functions such as data routing, device authentication, network management, and application integration. There were two networking server services that were investigated the Things Network (TTN) and ThingPark.

The Things Network (TTN) is a community-driven, open source LoRaWAN network server that aims to provide an accessible and affordable LoRaWAN infrastructure worldwide. It is designed to create public LoRaWAN networks that anyone can use, which makes it unique compared to other commercial solutions. It also offers private network capabilities for organizations. It is one of the largest LoRaWAN networks providing thousands of public gateways for up to 153 different countries [84]. It is best used for small to medium sized IoT projects.

ThingPark on the other hand is a commercial LoRaWAN network management platform created by Actility, it is aimed at enterprise and industrial IoT deployments. It provides a comprehensive set of tools and services to facilitate large-scale LoRaWAN networks being able to support millions of devices at a time. This service is offered by Spark in New Zealand, for student activities they offer a trial service that allows for up to 5 devices to be connected for free. Spark also sells several LoRaWAN end devices that are linked up with ThingPark [85]. This is a service that the WRC is currently using for their LoRaWAN insect traps.

## LoRaWAN Practical Applications

This knowledge has been applied in a few different applications, the first is an OTAA example using the Heltec Wireless Stick V3. This uses the example from the Heltec ESP32 Dev-Boards library [86] and uses the file called LoRaWAN. This program is set up to communicate with a LoRaWAN networking server, the networking server chosen to connect to was the Things network. The Things Network provided a devEUI which is a device identifier and an appKey which is an application password to confirm that it's the connected device. With slight alterations to the OTAA parameters and LoRaWAN channel mask code, packets were able to be sent from the Heltec Wireless Stick V3 to a local gateway a couple kilometres away which is registered on TTN. With further alterations to the code, different information can be sent, however, it is limited to 255 bytes as per LoRa protocol [87]. TTN allows for this information to be saved in a JavaScript format or be directly displayed using graphing tools.

The second application was also using the Heltec Wireless Stick V3. This application made use of the STM32 datalogger and 2 Heltec Wireless Stick V3 devices which can be seen in Figure 26. The STM32 sends a string of information from sensors over its UART to a Heltec Wireless Stick V3's UART. This Heltec board is set up in a transmitter configuration meaning that once it receives the UART string it then passes it into a buffer of which is then transmitted over LoRa on the 915 MHz band. The LoRa transmission is sent to another Heltec board which is setup in a receiver configuration, meaning that it will receive all incoming transmissions and store them into a buffer. After a transmission is received and it is stored in the buffer, it then uses a Wi-Fi connection to send the buffer which contains the string to the cloud. The code for this application can be found on the group git repository under Software.

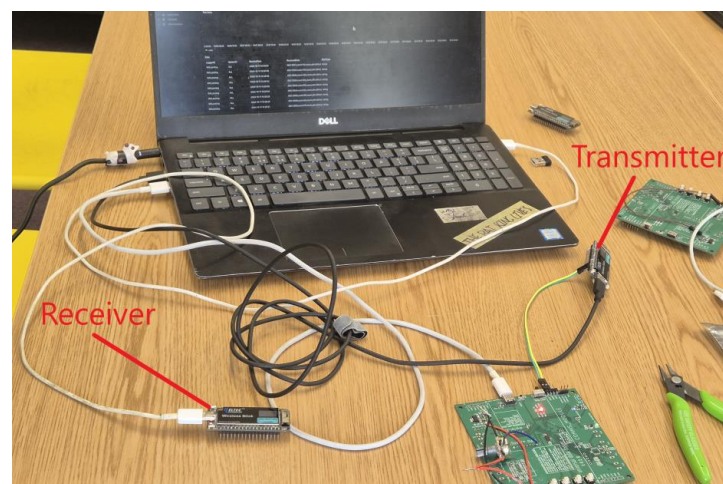


Figure 27: Application set up

The Semtech SX1262 LoRa RF radio module was the chip chosen for this project to go on the communications board. It is a high-performance, long-range transceiver which has been specifically designed for lower-power, long-range communications for IoT. It is capable of global applications being able to operate in the 150 MHz to 960 MHz frequency range. Its key features include [88]:

- Long-range communications that can transmit several kilometres in line-of-sight conditions and several hundred meters in urban areas making it suitable for wide area coverage.
- Low power consumption and is designed to be power efficient.
- Highly sensitive sensors allow the chip to detect weak signals which is a key part of long-range communications.
- Contains Spectrum Modulation which is used to provide communication in noisy environments.

This chip was chosen over other chips such as the MultiTech xDot chip which internally contains an SX1262. It was chosen over the MultiTech largely due to the price considerations as per the requirement of keeping it below 250\$. Although this did have a downside meaning that drivers for the SX1262 needed to be written for it to be programmed using the STM32 datalogger board.

### **SX1262 Code development**

Semtech provides a set of premade SX126X drivers [89] that provide all the basic functionality of features and functions that are referenced in the datasheet which communicates with internal buffers on the chip. However, these drivers require additional code to get it running. The remaining driver code needing to be written by the user was the hardware abstraction layer (HAL) files. This included a HAL write, read, reset, and wakeup functions [89]. Each of these functions needed to be implemented and linked to the STM32 HAL, GPIO, and SPI files. This was to ensure the correct routing and that it could interact with the STM32 correctly. This code was written to be as modular as possible with simple typedefs that can be changed to use different GPIO or SPI pins. These functions call the GPIO and SPI pins that are connected to the communications board. These tell it when the board is transmitting, when it's busy, and any interrupts that there may be.

Functions have also been written in the main which when called either sets the board into a receive or transmit mode. These functions all communicate via SPI with the communications board to send and receive information from its internal buffers. A series of LoRa parameters have also been set for it to be compatible with other LoRa devices and within the New Zealand operating frequency. Although a transmission and receive of this code has not been tested to check for its functionality.

## **Discussion**

This part of the project aimed to implement a LoRaWAN-based communication system using the SX1262 radio module and the STM32 microcontroller. The main objectives were to achieve low-power, long-range communication for IoT applications while keeping the system cost-effective and compatible with New Zealand's 915 MHz frequency band.

Throughout the development process, several key features were tested, including range and communication reliability. The SX1262 module's capabilities were validated through line-of-

sight testing and demonstrating successful data transmission over several kilometres. These results were consistent with the theoretical expectations of the SX1262 chip which allows for communication in noisy and obstructed environments.

Power consumption tests of the SX1262 module had not been tested while connected to the STM32 board, but the chip is designed to specifically to operate efficiently in low power modes. This aligns with the projects low-power requirement which is common among IoT devices. Early testing with the MultiTech xDot module revealed several issues related to the online compiler Keil Studio. Switching to Mbed studio improved the development and solved the issues in compiling. To meet the cost constraints, the SX1262 was chosen over the MultiTech xDot due to its lower price, despite it requiring additional development work for driver implementation.

Several iterations of hardware abstraction layer (HAL) code was required to integrate the SX1262 with the STM32 and make it as modular as possible. The SPI and GPIO calls were optimised to ensure seamless communication between the microcontroller and the radio module. Although the written transmit and receive functions were not tested, meaning further testing is required to verify the functionality of the LoRaWAN on the communications board using the STM32 board.

The primary project objectives of using LoRaWAN to achieve long-range, low-power communication and keeping costs low under 250\$ were successfully met. The SX1262 modules performance aligned with the initial project requirements of being power efficient. Writing custom driver code for the SX1262 ensured that the LoRaWAN section of the communications board was fully compatible with the STM32 and LoRaWAN protocol. Although it still requires further testing, the project has demonstrated that it can meet the basic requirements for an IoT communication network.

## Sustainability

### Triple bottom line analysis

The team carried out a simple triple bottom line analysis. This analysis was mainly focused on how Tura Ngati Te Nakau [90] would use this product would be use to assist them as a base case study. This case study will use the use case of the datalogger being used to monitor possum traps, as this was the main use case presented by Tura Ngati Te Nakau.

#### *Profit*

The universal modular datalogger was not created with the idea of being a profitable product in mind. However, the datalogger this was not the main goal of the datalogger, instead this datalogger allows Tura Ngati Te Nakau to expand on their profitability. During an interview with this iwi, it was found that possums that they would trap would be used to turn in to pelts and the meat be used for dog food. Although if the possum had consumed poison beforehand the meat could not be sold lowering the profitability of the trapping. Therefore, using the datalogger to effectively monitor these traps so poison would not have to be used would increase their profits.

#### *People*

As previously outlined in the profit these datalogger can be used to effectively monitor traps. however, this come with another benefit has these traps must be checked daily [91]. This monitoring of traps is a very tedious exercise for the iwi as they have over 16 thousand traps<sup>8</sup>. Therefore, these data logger too can be used to save both time and money as only the traps that are sprung need to be checked, lowering the amount of time people need to be within the bush.

#### *Planet*

Within the context of Tura Ngati Te Nakau this product assisted in creating a positive impact on the planet. This positive impact is created by assisting in the environmental management of the new Zealand forest through the effective management of pests. This environmental management also leads itself to the idea of Kaitiakitanga as it is about managing the environment. The data logger assists in this by allowing more information to be gathered for local iwi to use and apply their knowledge too and help in restoring the environment. Therefore also allow the monitoring of the forest the datalogger is in to check on things such as the health of the forest through its temperature [92].

### LCA outline

Due to the entire system of the data logger having several unknowns a complete life cycle analysis of the data logger cannot be currently done. However, the current prototype can be compared to other studies that measure the embodied carbon of PCBs can be used to approximate most of the output of the data logger. The estimated system for the life cycle of this PCB can be seen Figure 28. The PCB first must be manufactured creating GHG within the raw materials and fabrication stages. The next phase is the use phase of the PCB. There are several different emitters of GHG during this phase depending on how the data logger is used. Emission one is from the replacement of the battery on the data logger. However, it is unknown what battery would be in used and therefore these GHG cannot be models. Again, there is another emitter that cannot be modelled due to unknowns, is the travel it would take to check up on these data logger either to replace a battery, replace a module or even collect data from it. This is unknown due to the unknown location of where these data loggers would be, some may be deep within the bush requiring a trip of serval days to reach others may be within walking distance and require little to no travel. Due

---

<sup>8</sup> This information is from personal communications



to this, these emissions will not be included. The final stage of the data logger's life has three different outcomes, E-waste recycling, landfill, and environmental pollution. This environmental pollution end of life is due to some dataloggers being lost within the bush either due to a natural event such as an earthquake or storm or just being lost. The other two end of life are standard end of life for PCBs. However, this project has been made open source to combat the end of life of these data loggers when broken.

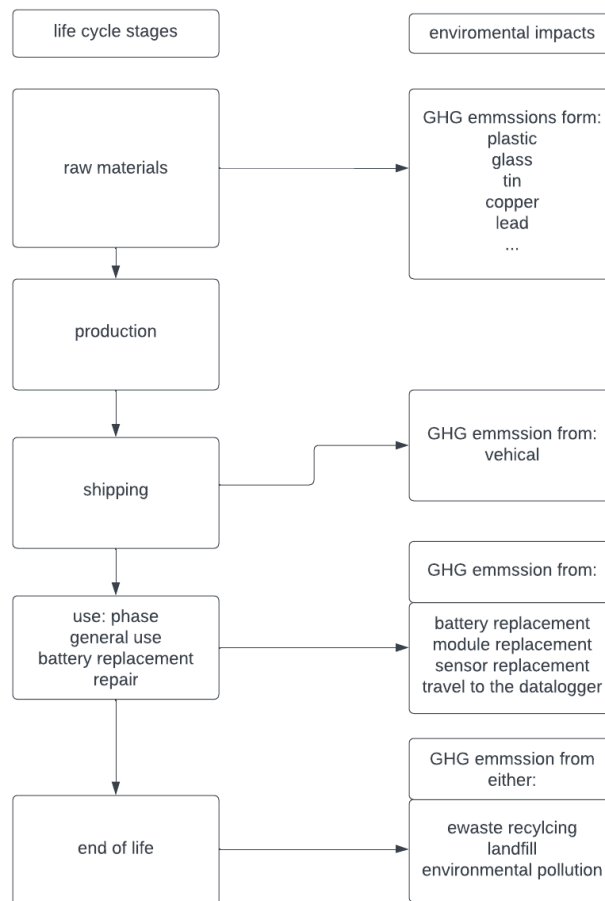


Figure 28: estimated life cycle of the datalogger.

Due to all these unknowns an accurate amount of embodied carbon cannot be determined. However a study only measuring the embodied carbon of PCBs to about 37 CO<sub>2</sub>-eq/kg [93]. This was also assuming they were disposed of correctly. Although a further exact figure of embodied carbon cannot be produced from this due to all the modules on the datalogger such as the power, communication interfaces, and sensors all having varying sizes and therefore weights. So, in conclusion an LCA of this magnitude is near impossible and should be carried out based on use case of the datalogger.

## Conclusions

The tests and validations conducted during this project demonstrated that the hardware design for the LPWAN device is feasible and capable of meeting the core requirements. The LPWAN PCB was successfully manufactured, and initial testing showed promising results for integrating LoRa and LTE-M communication technologies. However, the tests/validations also revealed areas for improvement, particularly in power management, software implementation, and component selection. Debugging and communication issues, such as TLS provisioning and server connectivity, were significant challenges.

The datalogger, data-acquisition (DAQ) hardware met all requirements specified of it within Table 3. This was achieved through the communication, power, sensor and actuator interfaces. Although this is in addition to the local storage and human interfaces, that allowed the device to be more pleasant to use. As well as the open sourcing of the schematics and PCB design allowing for more sustainability for repair. However further improvements to the design could be made as layout in the discussion. The test programs written for the data logger assisted with the parallel development of the datalogger.

While the project did not achieve complete success, the findings provide valuable insights for future iterations and highlight the potential of the modular design approach. The research and development efforts contributed significantly to advancing the concept of a universal modular data logger, laying the groundwork for further refinement and optimization.

## Recommendations/Applications

The following core points are considerations for improving the designs and implementations of the 'universal-modular-datalogger' to achieve all of the requirements.

1. **Device Architecture:** Rethink the device architecture. Focus on one platform, such as Zephyr, to expedite development for the core communications requirements. Instead of creating a datalogger/DAQ board, iterate on the LPWAN board to implement a modular IoT development board with an environmental module (temperature/humidity sensor). This IoT dev board should implement the datalogger hardware but remove all interfaces except for communications and power. These interfaces should then be implemented on an environmental datalogger module. This approach would meet the original project essentials while being more extensible and arguably, more universal.
2. **Datalogger Hardware:** Implement a third iteration of the datalogger hardware with the changes suggested in the discussions for both hardware designs. This should be done in conjunction with a redesign of the communications hardware to allow them to interface as a backpack onto the datalogger
3. **Power Supply Modules:** Design multiple power supply modules focusing on different battery chemistries, and functions for the device, such as the necessity for a linear power supply for RF boards, non-linear for DAQ expansion boards, and supporting solar power
4. **Field Testing:** Conduct field tests to benefit the datalogger hardware and software. Start with a small test within the university to prove it works within a CAN environment, and then expand to a larger area, such as the Port Hills.
5. **Cloud Infrastructure:** Future data formats should be implemented, automatic transmitting of actuator control should be added and better cybersecurity measures should be put in place.

## References

- [1] Ministry of Research, Science and Technology, “Vision Mātauranga Booklet,” 2007. [Online]. Available: <https://www.mbie.govt.nz/assets/vision-matauranga-booklet.pdf..>
- [2] MBIE, “Ministry of Business, Innovation and Employment,” [Online]. Available: <https://www.mbie.govt.nz.> [Accessed 2024].
- [3] S. a. T. Ministry of Research, “Vision Mātauranga Booklet,” 2007. [Online]. Available: <https://www.mbie.govt.nz/assets/vision-matauranga-booklet.pdf.> [Accessed July 2024].
- [4] C. S. Long, IP Network Design, Osborne/McGraw-Hill, 2001.
- [5] TI, “TPS2117 data sheet, product information and support,” [Online]. Available: <https://www.ti.com/product/TPS2117.> [Accessed 18 10 2024].
- [6] TI, “User’s Guide TPS2117 Evaluation Module,” [Online]. Available: [https://www.ti.com/lit/ug/sluct1/sluct1.pdf?ts=1690175707902&ref\\_url=https%253A%252F%252Fwww.ti.com%252Ftool%252FTPS2117EVM%253FkeyMatch%253D%2526tisearch%253Dsearch-everything%2526usecase%253Dhardware.](https://www.ti.com/lit/ug/sluct1/sluct1.pdf?ts=1690175707902&ref_url=https%253A%252F%252Fwww.ti.com%252Ftool%252FTPS2117EVM%253FkeyMatch%253D%2526tisearch%253Dsearch-everything%2526usecase%253Dhardware.) [Accessed 18 10 2024].
- [7] USB-IF, “USB Power Delivery,” 31 10 2023. [Online]. Available: <https://www.usb.org/document-library/usb-power-delivery.> [Accessed 18 10 2024].
- [8] USB IF, “USB 2.0 Specification,” [Online]. Available: <https://www.usb.org/document-library/usb-20-specification.> [Accessed 18 10 2024].
- [9] SEMTECH, “Product Details RClamp0582B,” [Online]. Available: <https://www.semtech.com/products/circuit-protection/low-capacitance/rclamp0582b.> [Accessed 18 10 2024].
- [10] EVVOSEMI, “AMS1117-3.3 EVVO,” [Online]. Available: <https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/6128/AMS1117-5272.pdf.> [Accessed 18 10 2024].
- [11] “Mini Photocell - SEN-09088 - SparkFun Electronics,” [Online]. Available: <https://www.sparkfun.com/products/9088.> [Accessed 18 10 2024].
- [12] SparkFun Electronics, “Environment - SparkFun Electronics,” SparkFun Electronics, [Online]. Available: <https://www.sparkfun.com/categories/305.> [Accessed 18 10 2024].
- [13] TI, “TL074 data sheet, product information and support,” [Online]. Available: <https://www.ti.com/product/TL074.> [Accessed 18 10 2024].
- [14] ti, “2-bit bidirectional 2- to 15-V 400-kHz I2C/SMBus buffer/cable extender,” [Online]. Available: <https://www.ti.com/product/P82B96.> [Accessed 18 10 2024].
- [15] arduino, “Burn the bootloader on UNO, Mega, and classic Nano using another Arduino,” arduino, [Online]. Available: <https://support.arduino.cc/hc/en-us/articles/4841602539164-Burn-the-bootloader-on-UNO-Mega-and-classic-Nano-using-another-Arduino.> [Accessed 18 10 2024].

- [16] SparkFun, “RP2040 Thing Plus Hookup Guide - SparkFun Learn,” SparkFun , [Online]. Available: <https://learn.sparkfun.com/tutorials/rp2040-thing-plus-hookup-guide/all>. [Accessed 18 10 2024].
- [17] DigiKey Electronics, “SC0914,” DigiKey Electronics, [Online]. Available: <https://www.digikey.co.nz/en/products/detail/raspberry-pi/SC0914-13/14306010>. [Accessed 19 5 2024].
- [18] ST, “This is information on a product in full production.,” ST. [Online]. Available: <https://www.st.com/resource/en/datasheet/stm32g431cb.pdf>. [Accessed 18 10 2024].
- [19] digikey, “STM32G431RBT6 STMicroelectronics,” digikey, [Online]. Available: <https://www.digikey.co.nz/en/products/detail/stmicroelectronics/STM32G431RBT6/10231565>. [Accessed 19 5 2024].
- [20] grillbaer, “grillbaer/esp32-power-consumption-test: Comparison of several ESP32 boards’ power consumptions in different sleep modes,” [Online]. Available: <https://github.com/grillbaer/esp32-power-consumption-test>. [Accessed 19 5 2024].
- [21] digikey, “ESP32-WROOM-32E-H4 Espressif Systems,” digikey, [Online]. Available: <https://www.digikey.co.nz/en/products/detail/espressif-systems/ESP32-WROOM-32E-H4/12696413>. [Accessed 18 5 2024].
- [22] kolban, “[Answered] What are the ADC input ranges?,” 17 1 2017. [Online]. Available: <https://www.esp32.com/viewtopic.php?t=1045#p4626>. [Accessed 18 10 2024].
- [23] ST Employee , “How to use the ST Open Bootloader for STM32 Microcontrollers,” st, 30 11 2024. [Online]. Available: <https://community.st.com/t5/stm32-mcus/how-to-use-the-st-open-bootloader-for-stm32-microcontrollers/ta-p/49896>. [Accessed 18 10 2024].
- [24] ST, “stm32-mw-openbl,” ST, [Online]. Available: <https://github.com/STMicroelectronics/stm32-mw-openbl>. [Accessed 18 10 2024].
- [25] ST, “Integrated Development Environment for STM32,” ST, [Online]. Available: <https://www.st.com/en/development-tools/stm32cubeide.html>. [Accessed 18 10 2024].
- [26] ST, “Introduction to USB with STM32 - stm32mcu,” [Online]. Available: [https://wiki.st.com/stm32mcu/wiki/Introduction\\_to\\_USB\\_with\\_STM32](https://wiki.st.com/stm32mcu/wiki/Introduction_to_USB_with_STM32).
- [27] SD Association, “SD Card Specification,” 10 2001. [Online]. Available: [https://www.sdcard.org/cms/wp-content/themes/sdcard-org/dl.php?f=PartE1\\_SDIO\\_Simplified\\_Specification\\_Ver1.00.pdf](https://www.sdcard.org/cms/wp-content/themes/sdcard-org/dl.php?f=PartE1_SDIO_Simplified_Specification_Ver1.00.pdf). [Accessed 18 10 2024].
- [28] “8Tbit Memory,” digikey, [Online]. Available: <https://www.digikey.co.nz/en/products/filter/memory/774?s=N4IgjCBcpgLATFUBjKAZAhgGwM4FMAaEAeygG0QB2AZgAZaxaQBdIgBwBcoQBIDgJwCWAOWDmIAL4SiISBTRYMOABYsJQA>. [Accessed 18 10 2024].
- [29] “32Mbit Memory,” [Online]. Available: <https://www.digikey.co.nz/en/products/filter/memory/774?s=N4IgjCBcpgLATFUBjKAZAhgGwM4FMAaEAeygG0QB2eABgGYAOAVhAF0iAHAFyhAGUuAJwCWAOWDmIAL5SiISBTx4Og4gFs2UoA>. [Accessed 18 10 2024].
- [30] JLCPCB, “standard PCB/PCBA,” JLCPCB, [Online]. Available: <https://cart.jlpcb.com/quote?orderType=1&stencilLayer=4&stencilWidth=100&stencilLength=100&stencilCounts=5>. [Accessed 18 10 2024].

- [31] nickjj, “webserver,” [Online]. Available: <https://github.com/nickjj/webserver/blob/master/webserver>. [Accessed 18 10 2024].
- [32] espressif, “ESP32-WROOM-32,” espressif, [Online]. Available: <https://www.espressif.com/en/producttype/esp32-wroom-32>. [Accessed 18 10 2024].
- [33] K. Knack, “PCB Design and Outer Layer Surface Finishes,” altium, 14 9 2020 . [Online]. Available: <https://resources.altium.com/p/pcb-design-and-outer-layer-surface-finishes>. [Accessed 18 10 2024].
- [34] TI, “Transmitting SPI Signals Over LVDS Interface Reference,” [Online]. Available: [https://www.ti.com/lit/ug/tidued8/tidued8.pdf?ts=1669575054987&ref\\_url=https%253A%252F%252Fwww.google.com%252F](https://www.ti.com/lit/ug/tidued8/tidued8.pdf?ts=1669575054987&ref_url=https%253A%252F%252Fwww.google.com%252F). [Accessed 18 10 2024].
- [35] T. Kugelstadt, “Extending the SPI bus for long-distance communication,” [Online]. Available: <https://www.ti.com/lit/an/slyt441/slyt441.pdf>. [Accessed 18 10 2024].
- [36] Z. Peterson, “Circuit Design Tips: PCB Moisture Protection for Humid Environments,” 12 9 2023 . [Online]. Available: <https://resources.altium.com/p/circuit-design-tips-pcb-moisture-protection-humid-environments>. [Accessed 18 10 2024].
- [37] “Linear Power Supply and. Switching Power Supply,” Cadence, 2020. [Online]. Available: <https://resources.pcb.cadence.com/blog/2020-linear-power-supply-vs-switching-power-supply-advantages-and-disadvantages>.
- [38] SEMTECH, “SX1262,oRa Connect™ Long Range Low Power LoRa® Transceiver +22dBm, global frequency coverage,” [Online]. Available: <https://www.semtech.com/products/wireless-rf/lora-connect/sx1262>.
- [39] Nordic Semiconductor, Nordic Semiconductor, [Online]. Available: [https://infocenter.nordicsemi.com/index.jsp?topic=%2Fps\\_nrf9160%2FnRF9160\\_html5\\_keyfeatures.html](https://infocenter.nordicsemi.com/index.jsp?topic=%2Fps_nrf9160%2FnRF9160_html5_keyfeatures.html).
- [40] Texus Instraments , “TPS2117DRLR 1.6-V to 5.5-V, 20-mΩ, 4-A low IQ power multiplexer with manual and priority switchover,” [Online]. Available: <https://www.ti.com/product/TPS2117/part-details/TPS2117DRLR>.
- [41] “STM32G4 Series of mixed-signal MCUs with DSP and FPU instructions - STMicroelectronics,” [Online]. Available: <https://www.st.com/en/microcontrollers-microprocessors/stm32g4-series.html>. [Accessed 14 03 2024].
- [42] Digikey, “SIM8060-6-0-14-00-A,SIM NANO HINGED, 6P, 1.40MM PROF,” [Online]. Available: <https://www.digikey.com/en/products/detail/gct/SIM8060-6-0-14-00-A/9859630>.
- [43] Digikey, “CP2102N-A02-GQFN20R,IC USB TO UART BRIDGE QFN20,” [Online]. Available: <https://www.digikey.com/en/products/detail/silicon-labs/CP2102N-A02-GQFN20R/9863478>.
- [44] J. Wolff, “The nRF9160 Feather,” 2020. [Online]. Available: <https://hackaday.io/project/171207/logs?sort=oldest>.
- [45] J. Wolff, “nrf9160:Programming and Debugging,” [Online]. Available: <https://docs.circuitdojo.com/nrf9160-programming-and-debugging.html>.
- [46] Digikey, “NN03-310,TRIO MXTEND MULTIBAND ANTENNA,” [Online]. Available: <https://www.digikey.co.nz/en/products/detail/ignion/NN03-310/10108235>.

- [47] P. Marquínez, “Arduino® Portenta Cat. M1/NB IoT GNSS Shield Cheat Sheet,” 09 May 2024. [Online]. Available: <https://docs.arduino.cc/tutorials/portenta-cat-m1-nb-iot-gnss-shield/cheat-sheet/>.
- [48] P. Kasanen, “How to setup Zephyr for RTT logging with nRF9160,” [Online]. Available: <https://devzone.nordicsemi.com/f/nordic-q-a/46083/how-to-setup-zephyr-for-rtt-logging-with-nrf9160>.
- [49] Nordic Semiconductor, “Build and configuration system,” [Online]. Available: [https://developer.nordicsemi.com/nRF\\_Connect\\_SDK/doc-legacy/latest/nrf/app\\_dev/config\\_and\\_build/config\\_and\\_build\\_system.html](https://developer.nordicsemi.com/nRF_Connect_SDK/doc-legacy/latest/nrf/app_dev/config_and_build/config_and_build_system.html).
- [50] A. Devices, “Using Analog Temperature Sensors with ADCs,” [Online]. Available: <https://www.analog.com/en/resources/design-notes/using-analog-temperature-sensors-with-adcs.html>. [Accessed 2024].
- [51] “SparkFun Humidity Sensor Breakout - H1H-4030 - SEN-09569,” SparkFun Electronics, [Online]. Available: [www.sparkfun.com](http://www.sparkfun.com). <https://www.sparkfun.com/products/9569>. [Accessed 26 May 2024].
- [52] “Adafruit FunHouse,” Adafruit Learning System, [Online]. Available: <https://learn.adafruit.com/adafruit-funhouse/analog-input-light-sensor>. [Accessed 26 May 2024].
- [53] “G1/4" Pressure Transducer Sensor, DC 5V Stainless Steel Pressure Sensor for Water,” Amazon, [Online]. Available: <https://www.amazon.co.uk/Pressure-Transducer-Sensor-Stainless-Diesel/dp/B07YZLCSRPT>. [Accessed 26 May 2024].
- [54] “ES-2 - METER Group,” metergroup, [Online]. Available: <https://metergroup.com/products/es-2/>. [Accessed 26 May 2024].
- [55] STM, “Arm® Cortex®-M4 32-bit MCU+FPU, 170 MHz /213 DMIPS, up to 128 KB Flash, 32,” October 2021. [Online]. Available: <https://www.st.com/en/microcontrollers-microprocessors/stm32g431r6.html>. [Accessed 2024].
- [56] “Getting started with PWR - stm32mcu,” wiki.st, [Online]. Available: [https://wiki.st.com/stm32mcu/wiki/Getting\\_started\\_with\\_PWR#Stop1\\_mode](https://wiki.st.com/stm32mcu/wiki/Getting_started_with_PWR#Stop1_mode). [Accessed 26 May 2024].
- [57] A. S. H. K. Y. Fatma Filiz Yildirim, “The effects of the weathering methods on the properties of the ABS, ASA and PMMA polymers, polymer,” *Testing*, vol. Volume 107, no. ISSN 0142-9418, 2022.
- [58] “Utilibox Style J Plastic Utility Box CU-1941,” UTILIBOX, [Online]. Available: [https://www.budind.com/product/general-use-boxes/utilibox-style-j-series-utility-boxes/cu-1941/#group=series-products&external\\_dimensions\\_group=0&internal\\_dimensions=0&general\\_use\\_mounting\\_style\\_group=0](https://www.budind.com/product/general-use-boxes/utilibox-style-j-series-utility-boxes/cu-1941/#group=series-products&external_dimensions_group=0&internal_dimensions=0&general_use_mounting_style_group=0). [Accessed 2024].
- [59] “Combustion (Fire) Tests for Plastics,” UL Solutions, [Online]. Available: <https://www.ul.com/services/combustion-fire-tests-plastics>. [Accessed 2024].
- [60] “What Is RoHS 3 Compliance?,” HQTS, 01 Mar 2022. [Online]. Available: <https://www.hqts.com/rohs-3/>. [Accessed 26 May 2024].

- [61] V. M. P. A. M. E. K. A. N., “Sustainable Additive Manufacturing: Mechanical Response of Acrylonitrile-Butadiene-Styrene over Multiple Recycling Processes,” *Sustainability*, no. 12, 2020.
- [62] “Free open source IoT OS and development tools from Arm | Mbed,” MBED, [Online]. Available: <https://os.mbed.com/>. [Accessed 2024].
- [63] Stack Overflow, “2024 Developer Survey,” 2024.
- [64] “Azure Pricing Calculator,” [Online]. Available: <https://azure.microsoft.com/en-us/pricing/calculator/>. [Accessed 23 May 2024].
- [65] Amazon Web Services, “AWS Free Tier,” May 2024. [Online]. Available: <https://aws.amazon.com/free/>. [Accessed 26 May 2024].
- [66] Microsoft Azure, “Azure free services,” 2024. [Online]. Available: <https://azure.microsoft.com/en-us/pricing/free-services/>. [Accessed 25 May 2024].
- [67] Google Cloud, “Free cloud features and trial offer,” 2024. [Online]. Available: <https://cloud.google.com/free/docs/free-cloud-features#storage>. [Accessed 26 May 2024].
- [68] Amazon Web Services, “AWS Pricing Calculator,” 2024. [Online]. Available: [https://calculator.aws/#/estimate?refid=ft\\_card](https://calculator.aws/#/estimate?refid=ft_card). [Accessed 26 May 2024].
- [69] Microsoft Azure, “Azure Pricing Calculator,” 2024. [Online]. Available: <https://azure.microsoft.com/en-us/pricing/calculator/>. [Accessed 23 May 2024].
- [70] Google Cloud, “Google Cloud Pricing Calculator,” 2024. [Online]. Available: <https://cloud.google.com/products/calculator>. [Accessed 26 May 2024].
- [71] Microsoft Azure, “Azure SQL database - pricing,” 2024. [Online]. Available: <https://azure.microsoft.com/en-us/pricing/details/azure-sql-database/single/>. [Accessed 25 May 2024].
- [72] OpenStreetMap, “About Page,” [Online]. Available: <https://www.openstreetmap.org/about>. [Accessed October 2024].
- [73] Amazon, “What is LoRaWAN?,” [Online]. Available: <https://docs.aws.amazon.com/iot-wireless/latest/developerguide/what-is-iot-wireless.html>.
- [74] T. T. Network, “What are LoRa and LoRaWAN?,” [Online]. Available: <https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/>.
- [75] T. T. Network, “Device Classes,” [Online]. Available: <https://www.thethingsnetwork.org/docs/lorawan/classes/>.
- [76] T. T. Network, “LoRaWAN Architecture,” [Online]. Available: <https://www.thethingsnetwork.org/docs/lorawan/architecture/>.
- [77] J. Brader, “Is Satellite Really Too Expensive?,” [Online]. Available: <https://www.sagenet.com/insights/is-satellite-really-too-expensive/>.
- [78] Narrowband, “NB-IoT Compared to LoRa and Sigfox: A Comprehensive Comparison,” [Online]. Available: <https://www.narrowband.com/nb-iot-vs-lora-vs-sigfox>.
- [79] C. O'Brien, “Inside Sigfox's Implosion,” [Online]. Available: <https://frenchtechjournal.com/inside-sigfox-implosion/>.



- [80] MultiTech, “MultiTech xDot® Long Range LoRa® Module,” [Online]. Available: <https://multitech.com/wp-content/uploads/86002182.pdf>.
- [81] armMBED, “MultiTech xDot (STM32L151),” [Online]. Available: <https://os.mbed.com/platforms/MTS-xDot-L151CC/>.
- [82] Heltec. [Online]. Available: <https://heltec.org/project/wireless-stick-v3/>.
- [83] Heltec, “LoRa Node Development Kit,” [Online]. Available: [https://resource.heltec.cn/download/Wireless\\_Stick\\_Lite\\_V3/HTIT-WSL\\_V3\(Rev1.1\).pdf](https://resource.heltec.cn/download/Wireless_Stick_Lite_V3/HTIT-WSL_V3(Rev1.1).pdf).
- [84] T. T. Network. [Online]. Available: <https://www.thethingsnetwork.org/>.
- [85] Acility. [Online]. Available: <https://www.actility.com/>.
- [86] Heltec, “Heltec\_ESP32,” [Online]. Available: [https://github.com/HelTecAutomation/Heltec\\_ESP32](https://github.com/HelTecAutomation/Heltec_ESP32).
- [87] T. T. Network, “Message Types,” [Online]. Available: <https://www.thethingsnetwork.org/docs/lorawan/message-types/>.
- [88] Semtech. [Online]. Available: <https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R000000Un7F/yT.fKdAr9ZAo3cJLc4F2cBdUsMftpT2vsOICP7NmvmMo>.
- [89] Semtech, “Lora-net,” [Online]. Available: [https://github.com/Lora-net/sx126x\\_driver/tree/v2.3.2](https://github.com/Lora-net/sx126x_driver/tree/v2.3.2).
- [90] Kaimai kaponga, “Te Arawa Waka Ngāti Kea / Ngāti Tuarā,” Kaimai kaponga, [Online]. Available: <https://www.kaimai-mamaku.org.nz/kaimai-kaponga>. [Accessed 20 10 2024].
- [91] NZ GOV, “Animal Welfare Act 1999,” 14 10 1999. [Online]. Available: <https://www.legislation.govt.nz/act/public/1999/0142/latest/whole.html#DLM50437>. [Accessed 20 10 2024].
- [92] P. Jonas Hamberg, “Temperature Check: Using Thermal Imaging to Assess Forest Health,” Toronto and Region Conservation Authority , [Online]. Available: <https://trca.ca/conservation/environmental-monitoring/thermal-imagery-assessing-forest-health/>. [Accessed 20 10 2024].
- [93] P. T. a. M. Kandlikar, “Comparing embodied greenhouse gas emissions of modern computing and electronics products,” *nvironmental Science and Technology*, vol. 47, no. 9, p. 3997–4003, 2013.
- [94] altium, “PCB Design and Outer Layer Surface Finishes,” altium, [Online]. Available: <https://resources.altium.com/p/pcb-design-and-outer-layer-surface-finishes>. [Accessed 18 10 2024].
- [95] Ministry of Research, Science and Technology, “Vision Mātauranga Booklet,” July 2007. [Online]. Available: <https://www.mbie.govt.nz/assets/vision-matauranga-booklet.pdf>.

## Appendices

### Appendix A: Git repos

<https://eng-git.canterbury.ac.nz/lhe68/enel400-data-logger>

[https://eng-git.canterbury.ac.nz/jth142/iot\\_device\\_project](https://eng-git.canterbury.ac.nz/jth142/iot_device_project)

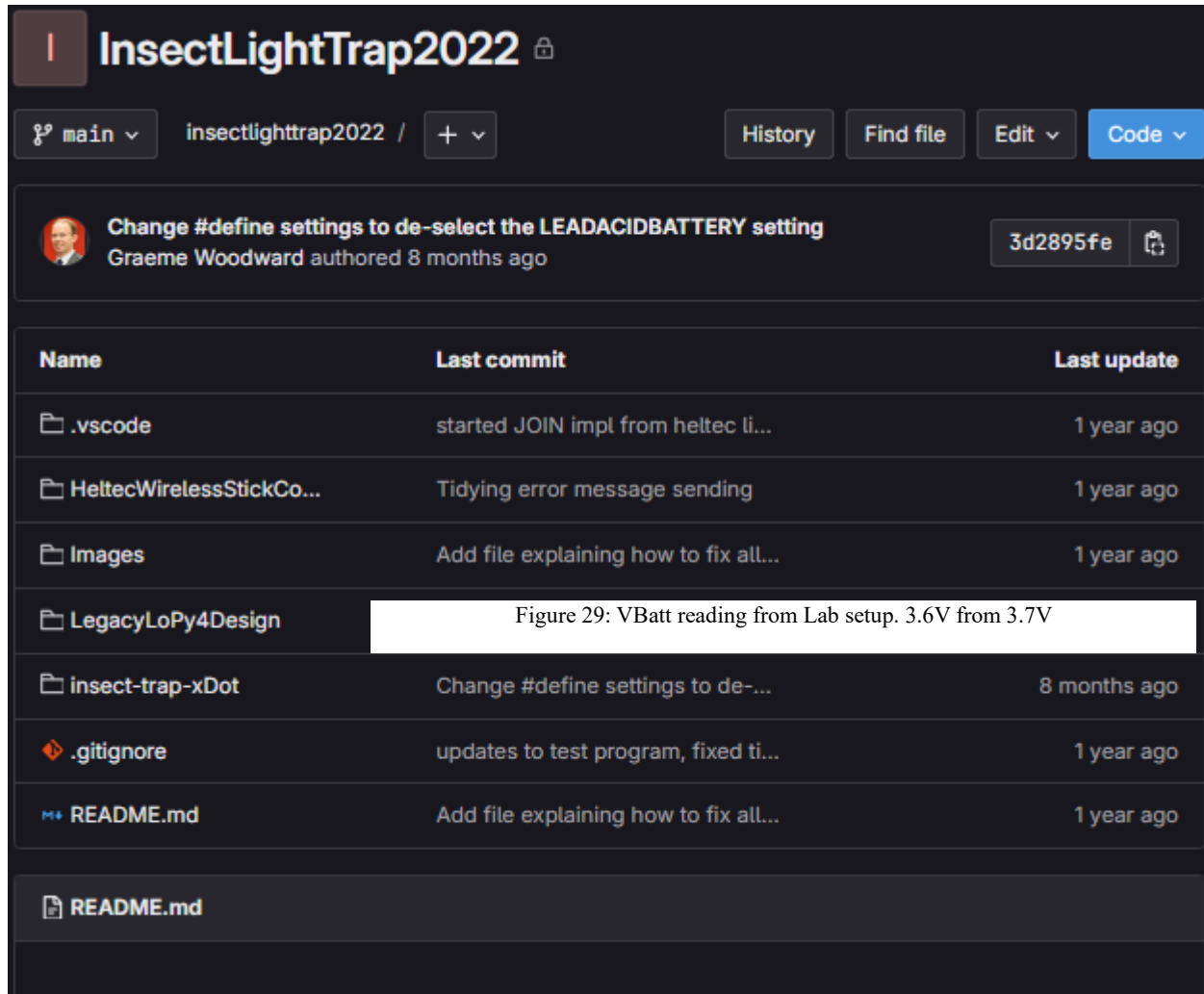
## Appendix B: Data logger hardware parts BOM

Reference	Value	Qty	price per 100	cost	link
C1	4.7u	1	0.1165	0.1165	<a href="https://www.digikey.co.nz/en/products/detail/yageo/CC0603KRX5R7BB475/5195193">https://www.digikey.co.nz/en/products/detail/yageo/CC0603KRX5R7BB475/5195193</a>
C2,C12,C14,C20,C21,C23,C24 ,C25,C26,C27,C52	1u	11	0.0245	0.2695	<a href="https://www.digikey.co.nz/en/products/detail/yageo/CC0201KRX5R5BB105/7164355">https://www.digikey.co.nz/en/products/detail/yageo/CC0201KRX5R5BB105/7164355</a>
C3,C4	26p	2	0.0556	0.1112	<a href="https://www.digikey.co.nz/en/products/detail/yageo/CC0603JRNPO9BN250/5883609">https://www.digikey.co.nz/en/products/detail/yageo/CC0603JRNPO9BN250/5883609</a>
C5,C6,C7,C8,C9,C10,C13,C18,C28, C29,C30,C31,C32,C22,C15,C16	100n	16	0.0331	0.5296	<a href="https://www.digikey.co.nz/en/products/detail/yageo/CC0603KRX7R7BB104/302822">https://www.digikey.co.nz/en/products/detail/yageo/CC0603KRX7R7BB104/302822</a>
C11	10n	1	0.0581	0.0581	<a href="https://www.digikey.co.nz/en/products/detail/yageo/CC0603KRX7R7BB123/5883754">https://www.digikey.co.nz/en/products/detail/yageo/CC0603KRX7R7BB123/5883754</a>
C17,C19,C33,C50	10u	4	0.1081	0.4324	<a href="https://www.digikey.co.nz/en/products/detail/yageo/CC0603MRX5R6BB106/5195224">https://www.digikey.co.nz/en/products/detail/yageo/CC0603MRX5R6BB106/5195224</a>
C51	22u	1	0.5125	0.5125	<a href="https://www.digikey.co.nz/en/products/detail/yageo/CC0603MRX5R6BB226/5195226">https://www.digikey.co.nz/en/products/detail/yageo/CC0603MRX5R6BB226/5195226</a>
D1,D3	RCLAMP0582B	2	0.9313	1.8626	<a href="https://www.digikey.co.nz/en/products/detail/semtech-corporation/RCLAMP0582B-TCT/4626644">https://www.digikey.co.nz/en/products/detail/semtech-corporation/RCLAMP0582B-TCT/4626644</a>
D5,D7	LED	2		0	optional, just used for status and will consume power
D6,D12	SP0502BAHT	2	0.2579	0.5158	<a href="https://www.digikey.co.nz/en/products/detail/diodes-incorporated/DESD5V2S2UT-7/2192644">https://www.digikey.co.nz/en/products/detail/diodes-incorporated/DESD5V2S2UT-7/2192644</a>
D8,D9,D10,D13,D14,D15	LED	6		0	optional, just used for status and will consume power
D22, D2	PTVS5V0Z1USK	2	0.2221	0.4442	<a href="https://www.digikey.co.nz/en/products/detail/nexperia-usa-inc/PTVS5V0Z1USKYL/6576000">https://www.digikey.co.nz/en/products/detail/nexperia-usa-inc/PTVS5V0Z1USKYL/6576000</a>
J8	USB_C_Receptacle_USB2.0	1	0.8757	0.8757	<a href="https://www.digikey.co.nz/en/products/detail/amphenol-cs-commercial-products/12402012E212A/13683192">https://www.digikey.co.nz/en/products/detail/amphenol-cs-commercial-products/12402012E212A/13683192</a>
J9	Micro_SD_Card_Det1	1	3.1592	3.1592	<a href="https://www.digikey.co.nz/en/products/detail/hirose-electric-co-ltd/DM3BT-DSF-PEJS/1982749">https://www.digikey.co.nz/en/products/detail/hirose-electric-co-ltd/DM3BT-DSF-PEJS/1982749</a>
Q1,Q2	NPN	2	0.2561	0.5122	<a href="https://www.digikey.co.nz/en/products/detail/diodes-incorporated/BSS127S-7/3451451">https://www.digikey.co.nz/en/products/detail/diodes-incorporated/BSS127S-7/3451451</a>
Q3,Q4,Q5,Q6	PMOS, TDB	4		0	TBD, pick based on load of the acuator interface
Q7,Q8	2N3904	2	0.0858	0.1716	<a href="https://www.digikey.co.nz/en/products/detail/lumimax-optoelectronic-technology/2N3904/22116349">https://www.digikey.co.nz/en/products/detail/lumimax-optoelectronic-technology/2N3904/22116349</a>
R1,R49,R50,R51,R52,R72	10k	6	0.0127	0.0762	<a href="https://www.digikey.co.nz/en/products/detail/yageo/RC0603FR-0710KL/726843">https://www.digikey.co.nz/en/products/detail/yageo/RC0603FR-0710KL/726843</a>
R2,R3	5.1k	2	0.0127	0.0254	<a href="https://www.digikey.co.nz/en/products/detail/yageo/RC0603FR-075K1L/726843">https://www.digikey.co.nz/en/products/detail/yageo/RC0603FR-075K1L/726843</a>
R4,R5,R6,R7,R46, R12,R13,R14,R35,R36,R40	TBD	11	0.0127	0.1397	using the same price as above due to all resistors at this scale being roughly the same price
R8,R17,R18,R43,R44,R45	0	6	0.0127	0.0762	<a href="https://www.digikey.co.nz/en/products/detail/yageo/RC0603FR-070L/726843">https://www.digikey.co.nz/en/products/detail/yageo/RC0603FR-070L/726843</a>
R9,R10,R11,R21,R23,R24, R25,R26,R27,R28,	1k	18	0.0127	0.2286	<a href="https://www.digikey.co.nz/en/products/detail/yageo/RC0603FR-071KL/726843">https://www.digikey.co.nz/en/products/detail/yageo/RC0603FR-071KL/726843</a>








R29,R30,R31,R47,  
R48,R74,R76,R78

R15,R20,R22,R32,R38,R39,R41,R42, R53	4.7k	9	0.0127	0.1143	<a href="https://www.digikey.co.nz/en/products/detail/yageo/RC0603FR-074k7L/726843">https://www.digikey.co.nz/en/products/detail/yageo/RC0603FR-074k7L/726843</a>
R16	200	1	0.0127	0.0127	<a href="https://www.digikey.co.nz/en/products/detail/yageo/RC0603FR-07200L/726843">https://www.digikey.co.nz/en/products/detail/yageo/RC0603FR-07200L/726843</a>
R19,R33,R34,R37	10	4	0.0127	0.0508	<a href="https://www.digikey.co.nz/en/products/detail/yageo/RC0603FR-0710L/726843">https://www.digikey.co.nz/en/products/detail/yageo/RC0603FR-0710L/726843</a>
R70	16.9k	1	0.0127	0.0127	<a href="https://www.digikey.co.nz/en/products/detail/yageo/RC0603FR-0717kL/726843">https://www.digikey.co.nz/en/products/detail/yageo/RC0603FR-0717kL/726843</a>
R71	5k	1	0.0127	0.0127	<a href="https://www.digikey.co.nz/en/products/detail/yageo/RC0603FR-075kL/726843">https://www.digikey.co.nz/en/products/detail/yageo/RC0603FR-075kL/726843</a>
SW1	RESET	1	1.0967	1.0967	<a href="https://www.digikey.co.nz/en/products/detail/alps-alpine/SKRKAHE020/19529210">https://www.digikey.co.nz/en/products/detail/alps-alpine/SKRKAHE020/19529210</a>
SW2	SW_SPDT	1	0.8328	0.8328	<a href="https://www.digikey.co.nz/en/products/detail/c-k/OS102011MA1QN1/1981430">https://www.digikey.co.nz/en/products/detail/c-k/OS102011MA1QN1/1981430</a>
U1	STM32G431R6Tx	1	6.47256	6.47256	<a href="https://www.digikey.co.nz/en/products/detail/stmicroelectronics/STM32G431RBT6/10231565">https://www.digikey.co.nz/en/products/detail/stmicroelectronics/STM32G431RBT6/10231565</a>
U2	MCP9804_MSOP	1	6.3978	6.3978	<a href="https://www.digikey.co.nz/en/products/detail/microchip-technology/MCP3428-E-ST/2179237">https://www.digikey.co.nz/en/products/detail/microchip-technology/MCP3428-E-ST/2179237</a>
U3	MCP3428x-xST	1	2.8236	2.8236	<a href="https://www.digikey.co.nz/en/products/detail/microchip-technology/MCP9804T-E-MS/2178719">https://www.digikey.co.nz/en/products/detail/microchip-technology/MCP9804T-E-MS/2178719</a>
U4	TL074	1	0.2841	0.2841	<a href="https://www.digikey.co.nz/en/products/detail/texas-instruments/TL074CDR/276926">https://www.digikey.co.nz/en/products/detail/texas-instruments/TL074CDR/276926</a>
U5	CM1624	1	0.5049	0.5049	<a href="https://www.digikey.co.nz/en/products/detail/nexperia-usa-inc/IP4252CZ16-8-TTL-1/2677644">https://www.digikey.co.nz/en/products/detail/nexperia-usa-inc/IP4252CZ16-8-TTL-1/2677644</a>
U6	tps2117	1	1.2951	1.2951	<a href="https://www.digikey.co.nz/en/products/detail/texas-instruments/TPS2117DRLR/21737520">https://www.digikey.co.nz/en/products/detail/texas-instruments/TPS2117DRLR/21737520</a>
U7	AZ1117-3.3	1	0.4116	0.4116	<a href="https://www.digikey.co.nz/en/products/detail/evvo/AMS1117-3-3/22482148">https://www.digikey.co.nz/en/products/detail/evvo/AMS1117-3-3/22482148</a>
U8	BME680	1	12.5819	12.5819	<a href="https://www.digikey.co.nz/en/products/detail/bosch-sensortec/BME680/7401317">https://www.digikey.co.nz/en/products/detail/bosch-sensortec/BME680/7401317</a>
U10,U11	TCA9800	2	0.9196	1.8392	<a href="https://www.digikey.co.nz/en/products/detail/texas-instruments/TCA9800DGKR/7653208">https://www.digikey.co.nz/en/products/detail/texas-instruments/TCA9800DGKR/7653208</a>
Y1	8MHZ, 18PF	1	0.9502	0.9502	<a href="https://www.digikey.co.nz/en/products/detail/abracon-llc/ABM3-8-000MHZ-D2Y-T/2344570">https://www.digikey.co.nz/en/products/detail/abracon-llc/ABM3-8-000MHZ-D2Y-T/2344570</a>
			total	45.81086	

## Appendix C Insect light trap Git



The screenshot shows the GitHub interface for the repository 'InsectLightTrap2022'. At the top, the repository name is displayed with a lock icon. Below it, the current branch is 'main' and the path is 'insectlighttrap2022 /'. Navigation buttons include 'History', 'Find file', 'Edit', and 'Code'. A recent commit by Graeme Woodward is shown, titled 'Change #define settings to de-select the LEADACIDBATTERY setting', with commit hash '3d2895fe' and a timestamp of '8 months ago'. Below the commit list, a table displays the repository's file structure and commit history.

Name	Last commit	Last update
 .vscode	started JOIN impl from heltec li...	1 year ago
 HeltecWirelessStickCo...	Tidying error message sending	1 year ago
 Images	Add file explaining how to fix all...	1 year ago
 LegacyLoPy4Design	Figure 29: VBatt reading from Lab setup. 3.6V from 3.7V	
 insect-trap-xDot	Change #define settings to de-...	8 months ago
 .gitignore	updates to test program, fixed ti...	1 year ago
 README.md	Add file explaining how to fix all...	1 year ago

Below the table, a section for the 'README.md' file is visible, showing a document icon and the filename.

Appendix D: Lab Setup for LPWAN PCB

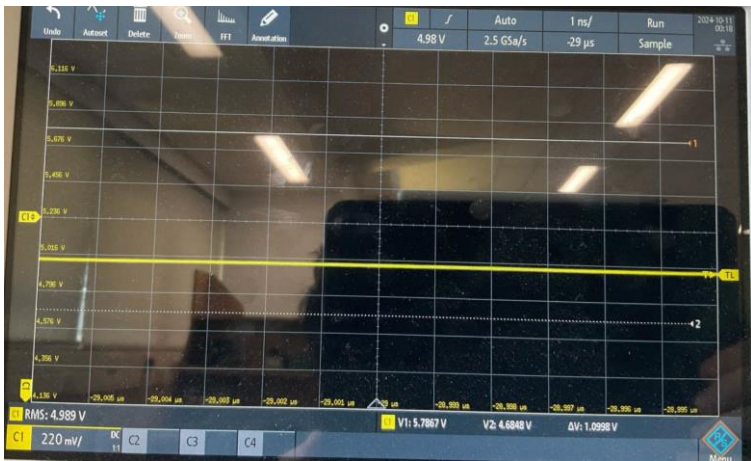
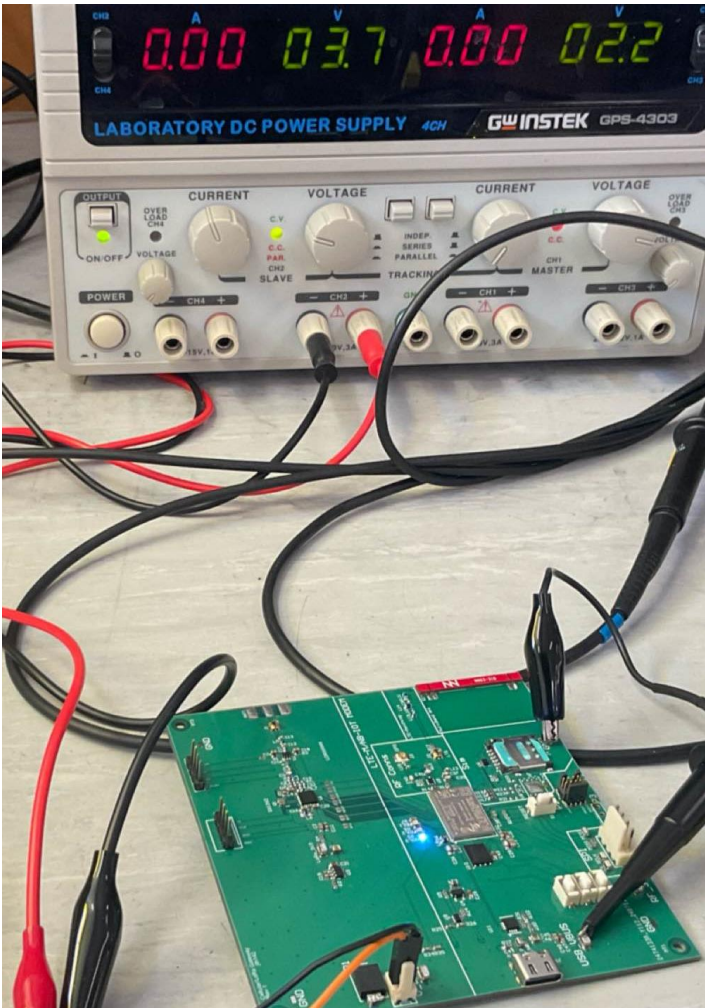


Figure 31: Lab setup for powering LPWAN PCB

Figure 30: VBUS\_USB output, when powered with external power (linear power supply)