Abstract

The need for resilient and sustainable agricultural solutions has been on the increase in recent years following the upward trend of economic instability and climate change challenges, reflected by the recent strategies released by the Isle of Man Government. This study presents the design and implementation of a cost-effective, modular and secure wireless weather monitoring network powered by Raspberry Pi, tailored for local Manx farmers to enhance their agricultural output and sustainability. The system collects key meteorological metrics including atmospheric pressure, air and ground temperatures and humidity, alongside designs for wind speed monitoring which is transmitted wirelessly over long distances via a secure and encrypted 433Mhz network, optimised for rural, local environments with high amounts of vegetation and humidity.

Key issues addressed include radio wave propagation, attenuation, frequency analysis and selection, antenna design, security methods such as hashing, encryption and frequency hopping and compliance with local regulations. Power management strategies, including battery banks and solar power are addressed to enable reliable off-grid operations.

Testing demonstrates robust, reliable data transmission over distances ranging from 0 - 250 metres utilising handmade dipole antennas with recommendations on how to improve range and reliability, supporting proactive instead of reactive decision-making for farmers and contributing to the island's goal on increasing food security and smart farming techniques.

This work provides a scalable foundation for a secure, off-grid IoT-based weather monitoring network with the potential for further expansions in similar, more specific contexts.

Chapter 2: Literature Review

2.1. Radio Wave Propagation in Rural Environments

Wave propagation has always proved to be more difficult to overcome in environments with high amounts of vegetation, which is often exacerbated throughout Spring and Summer due to the increase of foliage (Chee et al., 2011). This phenomenon is known as wave attenuation, which is the reduction of a wave's intensity as it propagates throughout the atmosphere, similar to how a current flowing through a wire naturally degrades as the distance increases, which is further increased by the wire being damaged. As a signal is transmitted from point A to point B, it may encounter physical barriers, such as leaves, trees, water vapour and precipitation which act as conductors, causing the signal to scatter and weaken (Tamir, 2003).

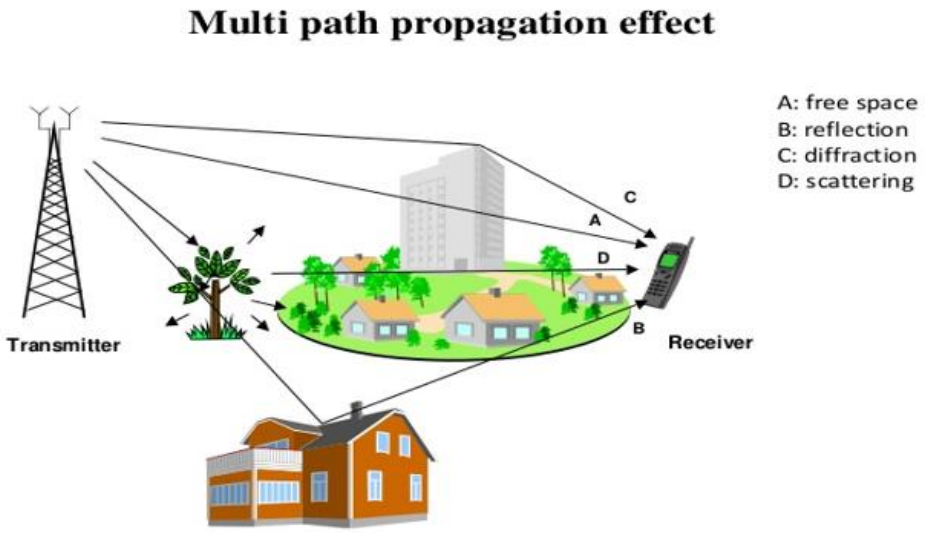


Figure 1: A visual representation of scatter, resulting in signal degradation (RATEL, n.d.)

This poses a potential problem for a system aiming to transmit real-time data throughout farmlands and possible rough terrain. Add on to this.

Wave propagation can be categorised into two types: ground and sky waves. Ground waves (further split into space and surface waves) are waves that have been transmitted from Earth to another point in Earth without entering the upper atmosphere, whereas skywaves perform ionospheric propagation due to reflecting off of the charged ions and

electrons in the ionosphere. This phenomenon allows skywaves to cover significantly greater distances than what would be possible with normal line-of-sight and refraction down on Earth's lower atmosphere and enables wireless communication in environments that typically wouldn't be possible, such as in forests (Hum, n.d.; Saakian, 2020).

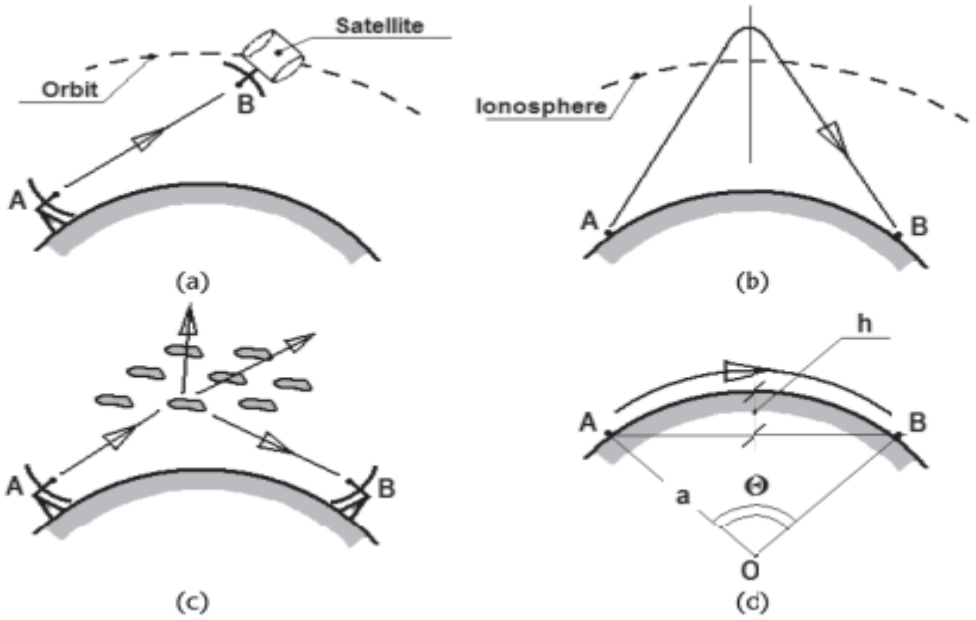


Figure 2: (a) Direct radio wave, (b) reflected radio wave, (c) scattered radio wave, and (d) diffracted radio wave (Saakian, 2020)

2.2. Frequency Strength Over Distance

The propagation of radio waves is significantly influenced by the frequency, as well as the atmospheric conditions. Generally, as we decrease the frequency of a wave, we increase the propagation conditions, allowing the wave to travel longer distances and through obstacles more effectively (Saakian, 2020). This is because frequencies and wavelengths are interlinked (the construction of an antenna revolves around capturing the wavelength), with lower frequencies offering higher wavelengths that penetrate through objects better than lower wavelengths. As Loxley (2022) says, "At 300 GHz, the wavelength is 1 mm, which is shorter than a grain of rice, while at 30 Hz; the wavelength is 10,000 kilometres".

Although decreasing the frequency allows for longer transmissions, we also minimise the maximum bandwidth of data we can transmit. This is due to technical limitations (such as the antenna size increases as the frequency is lowered) and the increased interference and power demand needed to maintain a suitable signal-to-noise (SNR) ratio (RF Spectrum Allocation | Cadence, 2024). Figure 3 shows the relationship between bandwidth and range, specifically how the range of a troposcatter signal is increased when the bandwidth is decreased.

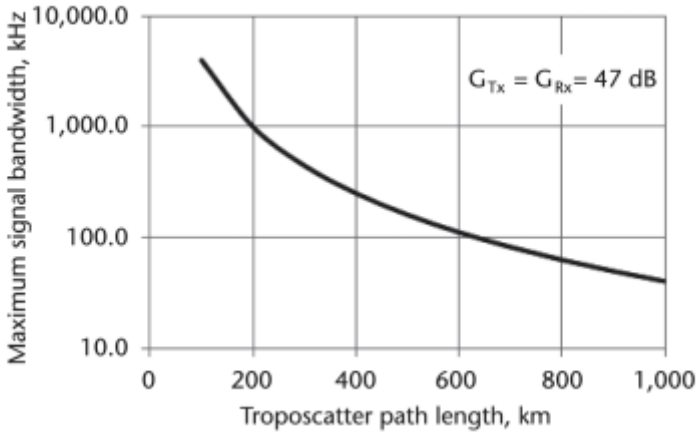


Figure 3: Example of the distance dependence of maximum bandwidth for troposcatter signal transmission (Saakian, 2020)

2.3. Antenna Types & Radiation Patterns

According to the Institute of Electrical and Electronics Engineers (2019), an antenna is "that part of a transmitting or receiving system that is designed to radiate or to receive electromagnetic waves." It is the critical component used in transmitting or receiving electromagnetic waves by converting guided waves (electronic pulses on a wire) into free waves for propagation that is typically received by the same type of antenna for conversion back into guided waves (Balanis, 1992).

As the demand for higher bandwidth and further distances increase, the choice and design of antennas must evolve to meet these increasing requirements effectively. One of the major properties of an antenna is its radiation pattern, which is a mathematical or graphical model of how an antenna radiates or receives energy into or from free space (Cisco, 2007). These models can be used to determine which antennas provide appropriate coverage and gain. For example, dipole antennas (a type of omnidirectional

antenna) are commonly used within radio stations as their radiation patterns allow for equal coverage over a certain area which resembles a doughnut shape, as shown in Figure 4:

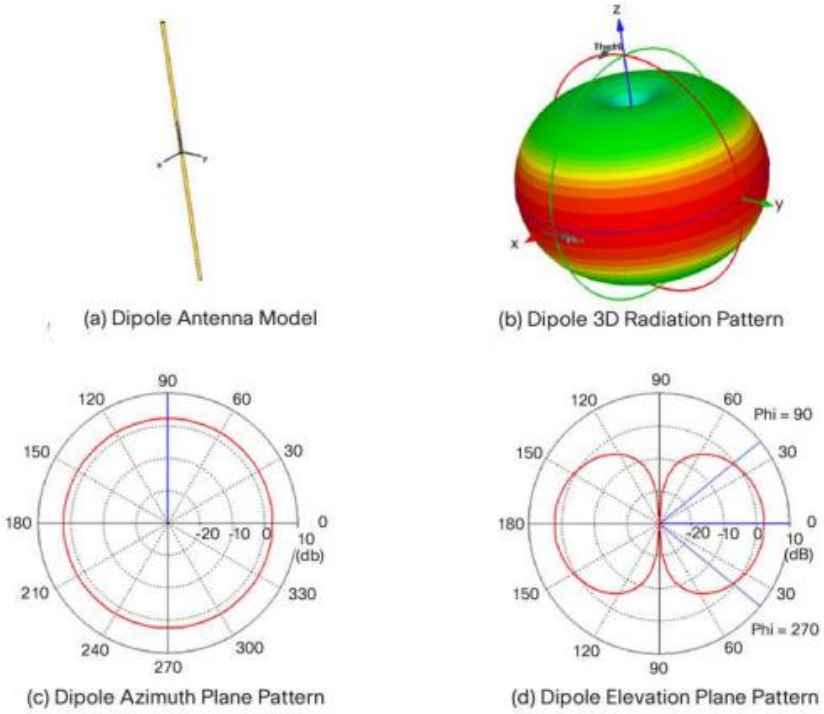


Figure 4: Dipole antenna radiation pattern (Cisco, 2007)

Furthermore, directional antennas can be implemented in systems where longer ranges are required, such as for television broadcasting or in point-to-point (P2P) wireless networks. A commonly used antenna for this purpose is known as a Yagi-Uda antenna, a type of antenna array, which utilises a main beam that is fixed to a certain direction with multiple directors at the front to amplify the signal and a reflector behind the main dipole to reflect the signal back to the dipole, further increasing gain and minimising interference by focusing on one small degree of free-space (Dai et al., 2011;Salanitro, 2024).

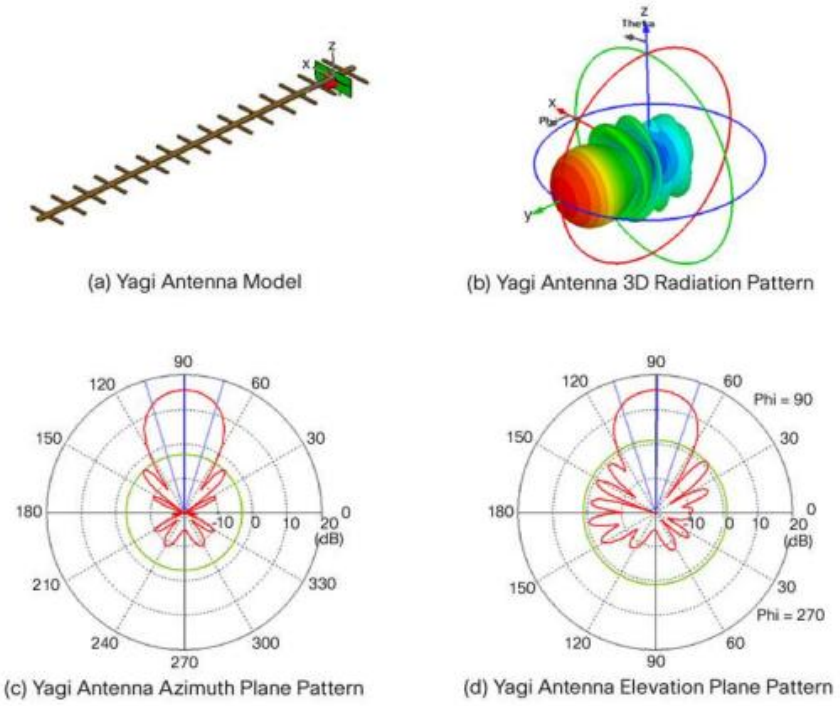


Figure 5: Yagi-Uda radiation pattern (Cisco, 2007)

2.4. Frequency Spectrum Management

Electromagnetic waves cross international borders, requiring global coordination to avoid interference. The International Telecommunication Union (ITU) manages global spectrum allocation and sets technical standards (ITU, 2019). In the UK, including the Isle of Man, Ofcom regulates frequency use across sections like military, government, commercial and broadcasting (Communications Commission, 2024; Ofcom, 2024).

Ofcom's Frequency Allocation Table (FAT) helps users identify available bands, for example 433 MHz which is commonly used for IoT purposes. As per section 1.1.2, frequencies in this range are free for civilian use: *"Responsibility for assigning frequencies to this Allocation rests with Ofcom. Frequencies in this Allocation are assigned for civilian use except where permissions for military use are agreed with the Ministry of Defence. No protection can be claimed for these assignments and undue interference must not be caused to other users"* (Ofcom, 2024):

430 – 440 MHz FIXED MOBILE RADIOLOCATION Amateur-satellite Amateur 5.138, 5.282	UK 2.1 UK 2.1, UK 1.1.1 UK 2.1 UK 1.1.2 UK 1.1.2	430 – 432 MHz AMATEUR RADIOLOCATION 5.271 5.272 5.273 5.274 5.275 5.276 5.277	430 – 432 MHz RADIOLOCATION Amateur 5.271 5.276 5.277 5.278 5.279

Figure 6: Referencing Ofcom's FAT for determining license-free bands (Ofcom, 2024).

Effective Radiated Power (ERP) is extremely regulated to minimise interference (Bensky, 2019), with transmitters exceeding 6.1W ERP needing to meet specific compliance rules (Ofcom, 2021).

2.5. Practical Implementations

Tamir (2003) tried and tested two approaches to wave propagation in a forest environment. One test utilised ionospheric propagation over one hop and a lateral-wave method which skims across the treeline. The ionospheric propagation method was disregarded due to the narrow frequency band required, resulting in only extremely low data rates.

A study conducted in the rainforests of India found that transmission loss was decreased with an increase in antenna heights in the presence of foliage, suggesting the height enhances the antenna's gain, allowing for improved signal transmission. It was also found that there was no distance dependence of height gain found, meaning that the same increase in performance was found independent of distance. Lastly, the data loss rate in the wet season were higher than the dry season, indicating that the increase of green leaves in the wet season further hindered propagation (Tewari et al., 1990).

A system was proposed for environmental monitoring based on Raspberry-Pi by Cabaccan et al. (2017) based on Wireless-Sensor-Network (WSN). This study showed the use of WSN over Wi-Fi for short-range wireless communication, which unfortunately wouldn't be plausible over a maximum range of a few miles.

U et al. (2022) designed a weather monitoring system that captures temperature, humidity, moisture, ultra-violet and carbon-dioxide which is then stored in a cloud environment for remote access and analysis. The project also includes an alert application to warn people and farmers about any current changes in the weather and utilises an API for complex weather predictions with accurate results.

A paper by Blokhin & Blokhina (2024) presents a hybrid wireless sensor network for agricultural monitoring set up to facilitate the monitoring of crops, weather and soil characteristics in real time using ZigBee, Wi-Fi and LTE 4G. This network consisted of a base station operating as the network coordinator and two sensor nodes; a mobile sensor node with an optical camera and a further node equipped with soil moisture and temperature sensors. A Raspberry Pi Pico is used to collect the readings before being sent to a Raspberry Pi 3B for processing and analysis which is then stored on a server.

A paper released by Prodanović et al. (2020) focuses on the security of wireless sensor networks. The paper discusses how agriculture is one of the last systems in the world that hasn't been completely digitalised and develops a model of security for sensor networks independent from communications infrastructure, fulfilling the security requirements of a distributed system such as authentication, data integrity, authorisation, availability and non-repudiation by focusing on encrypted, digital envelopes, digital signatures and public key infrastructure.

2.6. Security

Securing real-time weather data is crucial for reliable data-driven decision-making within agriculture. If systems are insecure, attackers could take advantage and tamper with or block data, leading to poor decisions, crop losses and wasted resources, The Isle of Man's unpredictable and often harsh weather further increases the risk of data loss or disruption. Protecting the integrity and availability of weather data is therefore an essential consideration to ensure system robustness.

Securing such real-time weather data is an important consideration in maintaining the reliability and trustworthiness of a system used for vital decision making. Insecure systems could be vulnerable to malicious actors who may attempt to modify or block the flow of critical data, which could lead to misinformed responses by farmers. This could ultimately lead to crop losses, inefficiencies in resources and incorrect sowing and harvesting times. Another consideration is the unpredictability and harsh conditions of Manx weather, particularly high humid and heavy rain conditions resulting in data loss. Protecting the integrity and availability of weather data is therefore a consideration that needs to be implemented to ensure the robustness of such a system.

If such a system were to be implemented on a national scale, it could become a main target for threat actors to threaten food security. The World Economic Forum lists both extreme weather events and cyber insecurity in the top 10 global risks for the next 2 and 10 years when asking participants "Please estimate the likely impact (severity) of the following risks over a 2-year and 10-year period.", further demonstrating the need for technological solutions to increase food production in a secure manner (World Economic Forum, 2024).

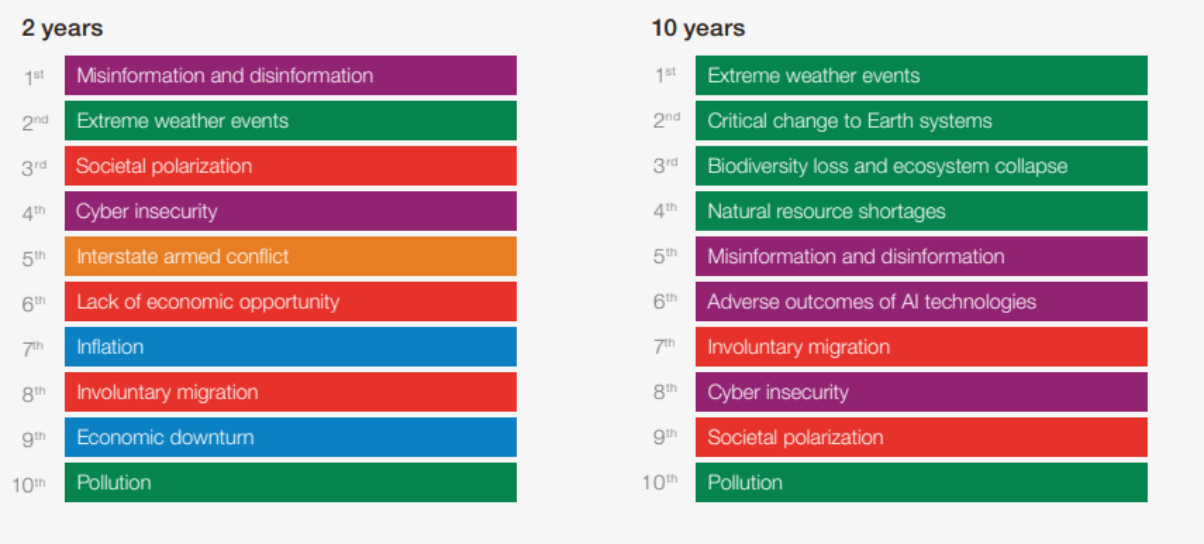


Figure 7: Results of the global risks ranked by severity over the short and long term (World Economic Forum, 2024).

2.7. Hashing

One of the most simple and efficient security methods is the hashing of data. Hashing helps prove the integrity of data by generating a one-way cryptographic string (the hash value) using the data as the input for the hash function. The hash can then be sent alongside the data needed to be checked, with the receiver then using the approach on the original data and comparing the two. If any altering of the data has occurred (malicious or due to data-loss), the hashes will be different (Krawczyk, 1994).

2.8. Frequency Hopping Spread-Spectrum

Frequency hopping spread-spectrum (FHSS) is a frequency hopping technique used for improving signal performance by reducing radio interferences and preventing jamming attacks and detection of signals by malicious actors, used by many military applications due to its Low Probability of Intercept (Hossein, 2011). This is accomplished by spreading

a signal across a wide band of frequencies at a fast rate, utilising a code sequence generator to pseudo-randomly generate a list of frequencies to hop between, which reduces the likelihood of interference and makes it significantly more difficult to detect and jam the signals (Ristić et al., 2022).

2.9. Frequency over Security

Various security protocols are available for wireless transmissions, however the level of security implemented will need to be considered as this will become a factor in the frequency used. This is because many security mechanisms, for example, hashing and encryption, will increase the amount of data needed to transmit, in turn increasing the necessary bandwidth, frequency and lowering the range.

Furthermore, frequency hopping spread-spectrum (FHSS) works most effectively when spread out over a large bandwidth (Ristić et al., 2022), which, as already discussed, isn't plausible in lower frequency bands and is severely limited by regulations and interference. Due to this, higher frequency bands are required for more effective use of FHSS.

Moreover, the encryption algorithm utilised has a significant impact on remote, wireless communications as power optimisation is critical alongside ensuring adequate levels of security are in place (Aboshosha et al., 2019). Below is a table comparing different types of symmetric and asymmetric algorithms to determine the most practical candidate in this type of system:

Algorithm	**Pros**	**Cons**
RSA	Extremely secure, utilising both a public and private key for encryption that is, by modern standards, almost impossible to crack without the use of quantum computing which doesn't currently exist (Amos, 2024).	While RSA is mostly used for encrypting small amounts of data, it is still inefficient in terms of time and computational power for use in IoT as compared to modern symmetric alternatives (Franklin, 2022).

AES	AES is also extremely secure; however it is a symmetric algorithm which is more efficient for low power devices. AES is also commonly used in wireless technologies due to its high-security and low power demands, such as WiFi (Hübschmann, 2021).	Whilst even the smallest 128 bit AES keys are extremely difficult to brute-force, the fact that symmetric key cryptography such as AES utilise the same key for encryption and decryption means that key secrecy needs to be ensured as an attacker could physically get hold of a weather station node and analyse the code to find the secret-key (Cooper, 2024).

Table 3: Table comparing RSA and AES encryption algorithms.

2.9.1 Conclusion

To conclude, it appears AES offers the most practical encryption algorithm due to its low-power demands while still remaining secure. The concerns with key management can be mitigated by enhancing physical security measures on the systems to prevent a bad actor from gaining access to the AES key.

2.10. Analogue vs Digital Transmission

Wireless communication is always fundamentally analogous, due to the carrier being electromagnetic waves that are analogue in nature. We can, however, encode our data to work in a digital environment (using bits for representing data instead of sine waves) (Sunil Bhooshan, 2021).

Frequency-shift keying (FSK) is a very well-known modulation scheme that represents the binary 0 and 1 by two different frequencies. There are many practical reasons why digital encoding is preferred to a complete analogue system. Twp such reasons would be limitation of interference and a further maximum range. This is due to the larger discrepancy between the two waves representing 1 and 0, so noise is less apparently and can be easily repeated to cover larger distances (Moon, 2024).

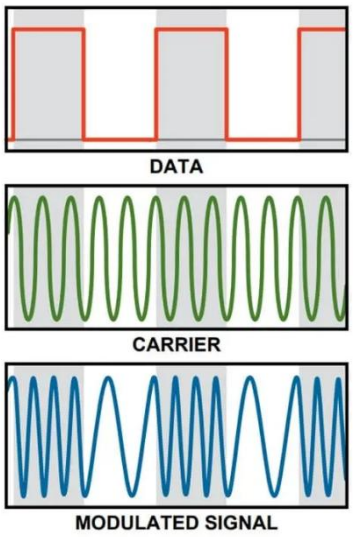


Figure 8: The frequency-shift keying modulation scheme visualised (Baby, 2024).

Meanwhile, analogue signals are more susceptible to noise and interference due to the infinite number of values that can be contained within a wave (Monolithic Power Systems, 2024).

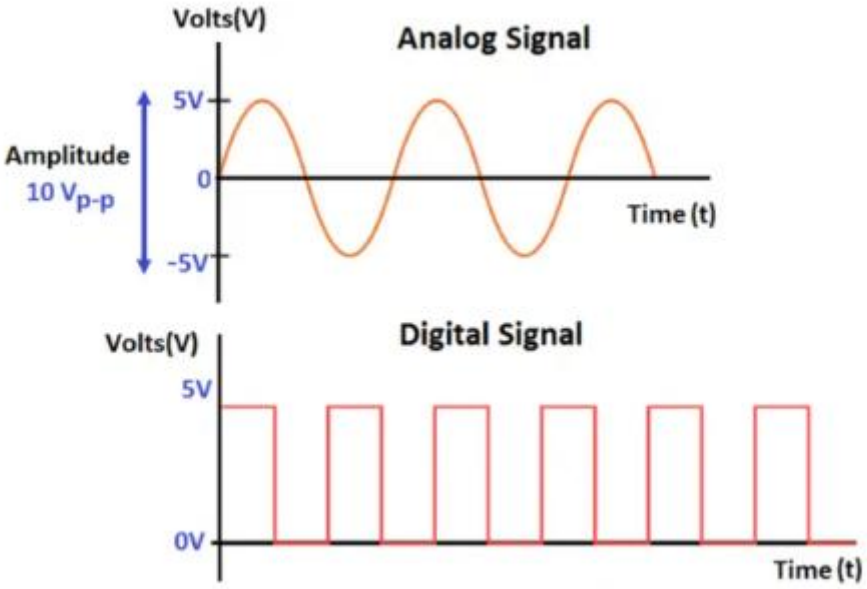


Figure 9: Analogue vs digital signal (QuarkTwin, 2024).

2.11. Summary

This literature review examines the technical and practical challenges involved in creating reliable, secure and wireless data transmission for IoT-based weather systems, particularly for agriculture where high-vegetation, differing terrain and moisture levels are

a major challenge that causes signal attenuation. Lower frequencies penetrate these barriers easier than higher-frequency frequencies and allow for further range communications, with a downside of reduced data capacity.

Furthermore, the antenna design is crucial in maximising range depending on the network topology used. Omnidirectional antennas provide equal 360° coverage with reduced range, with directional antennas providing significantly extended ranges with precise alignment.

Data integrity and confidentiality are two other important factors to consider. Techniques such as hashing, encryption and frequency hopping are recommended to protect against interference, eavesdropping and other cyber related threats. In terms of encryption, AES is the most efficient popular algorithm for symmetric encryption.

Compliance with international and local regulations is essential. The 433 MHz band offers both high suitability for wave propagation and regulatory availability.

Finally, there appears to be a lack of research in long-distance radio communications in IoT agricultural projects without the use of mainstream protocols such as Wi-Fi, Zigbee and LoRa which this dissertation aims to solve.

Chapter 4: Design

4.1. Project Scope

The Met Office (2016) records various meteorological parameters, including air temperature, pressure, wind, rainfall, humidity, cloud height and visibility. As the UK's national weather authority, these metrics support accurate forecasting.

Furthermore, Yoder (2014) identifies the four key parameters in crop growth as temperature, precipitation, solar radiation and humidity.

Due to the time and cost constraints, this dissertation will aim to satisfy air temperature, atmospheric pressure, wind speed and humidity. This satisfies both weather forecasting and crop growth. The design will ensure that any additional functionality or parameters can be easily adopted in future iterations.

4.1.1. Functional and non-functional requirements

Functional requirements	**Non-functional requirements**
The system shall collect real-time weather data (temperature, atmospheric pressure, humidity and wind speed) from a variety of sensors at regular intervals.	Reliability: The system must achieve high data success rates over varying distances in rural environments.
The collected weather data should be sent over a custom-made secure 433 MHz RF network.	Accuracy: The sensor readings must be within an acceptable tolerance (for example, ±1°C).
The system shall enable modular support for the addition and replacement of sensors without major system changes.	Security: Encryption must be used to protected against unauthorised access, tampering and sniffing attacks with access controls in place.
The data transmission must adhere to local radio frequency regulations.	Compliance: The system must adhere to current relevant legal and regulatory standards.

Packet retransmission should be implemented to resend packets in the event that an issue-in-transmission occurs, and a lost packet is detected.	Maintainability: The system must be easy to update and repair, providing a cheap alternative to other high-cost weather stations available on the market.

Table 5: Functional and non-functional requirements identification

4.2. System Architecture

The proposed systems will use two Raspberry Pi devices acting as the central processing unit, equipped with various weather instruments for real-time weather monitoring. The system will utilise Raspberry Pi OS, the default and recommended operating system for Raspberry Pi, along with Python for the data collection, processing and transmission.

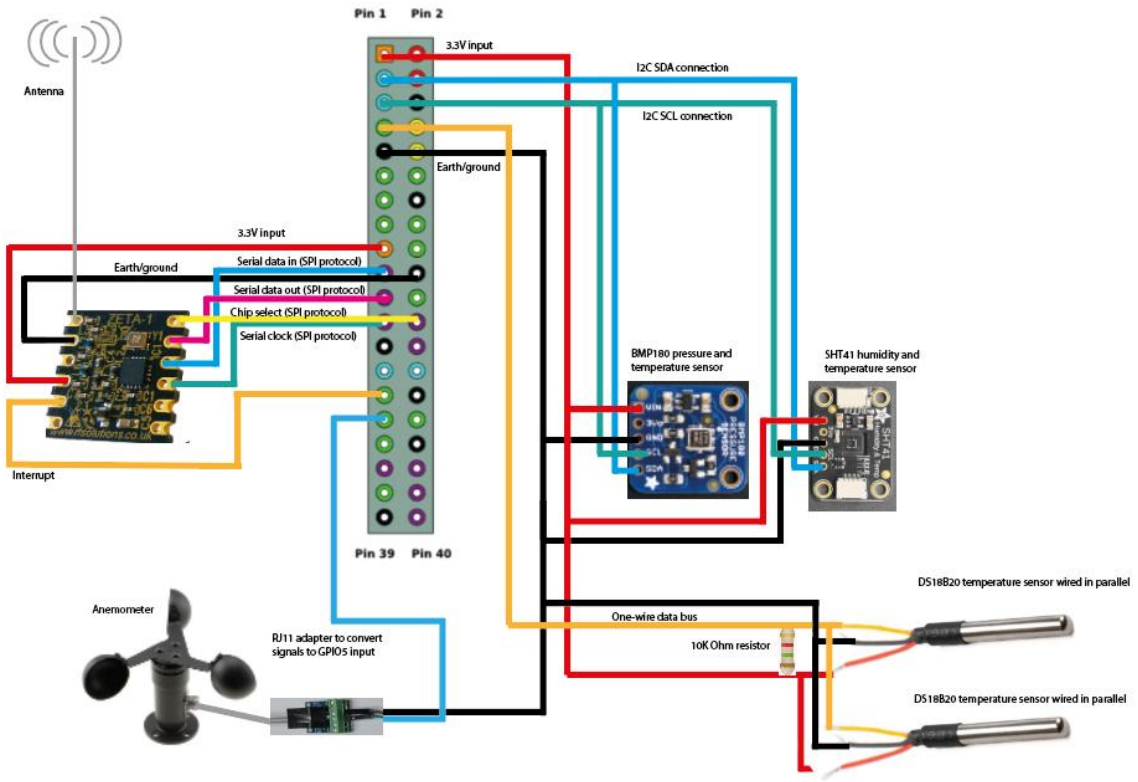


Figure 12: Initial system architecture design based on common weather monitoring system requirements and the outcome determined in Chapter 2: Literature Review

Component/ feature	**Protocol**	**Connection Details**	**Data type**	**Justification**

BMP180	I^2C	3.3v, GND, shared SDA and SCL wired in parallel	Pressure, temperature	Shares I^2C bus, accurate pressure and temperature readings
SHT41	I^2C	3.3v, GND, shared SDA and SCL wired in parallel	Humidity, temperature	Also shares I^2C bus with accurate humidity readings
DS18B20	1-Wire	3.3V, GND, 1-Wire bus, 10k resistor wired in parallel	Temperature	Cheap, accurate and supports a large-scale network due to parallel wiring with the 1-Wire protocol
Anemometer with RJ11 adapter	Digital input	GND, GPIO	Wind speed	Simple wiring with accurate results that can be easily calibrated
ZetaRF	SPI	3.3V, GND, MISO, MOSI, SCLK, CS, Interrupt	Wireless transceiver	Fast data transfer with low latency
Shared power and ground	N/A	3.3v, GND	Power distribution	Reduces wiring complexity, adds a layer of modularity and helps maintain stable voltage levels
Parallel 1-Wire and I^2C data busses	1-Wire, I^2C	3.3v, GND, SDA, SCL, 1-wire bus	Humidity, pressure, temperature	These two protocols allow for easy parallel wiring, which simplifies the whole wiring setup, reduces wiring issues and adds an easy layer of

				modularity by allowing devices to attach onto the rest of the circuit without any further physical configurations.

Table 6: Table discussing the different components, features, protocols and data types used.

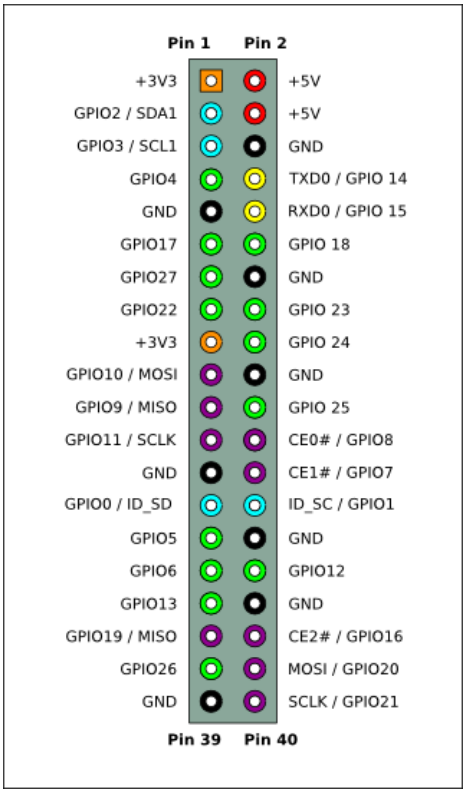


Figure 13: Raspberry Pi pinout graphical representation used for identifying appropriate GPIO and power pins.

4.2.1. Components

4.2.1.1. BMP180

BMP180 sensors are used for measuring atmospheric pressure, altitude and temperature. This sensor was chosen as it has been frequently categorised as the best low-cost sensor for measuring these parameters with accurate readings at +- 0.03hPa (Adafruit Industries, n.d.).

Atmospheric pressure is an indicator of changing weather, as lower pressures indicate windy conditions, precipitation and cloudiness whereas higher pressure is an indicator of calmer weather (National Geographic, 2022).

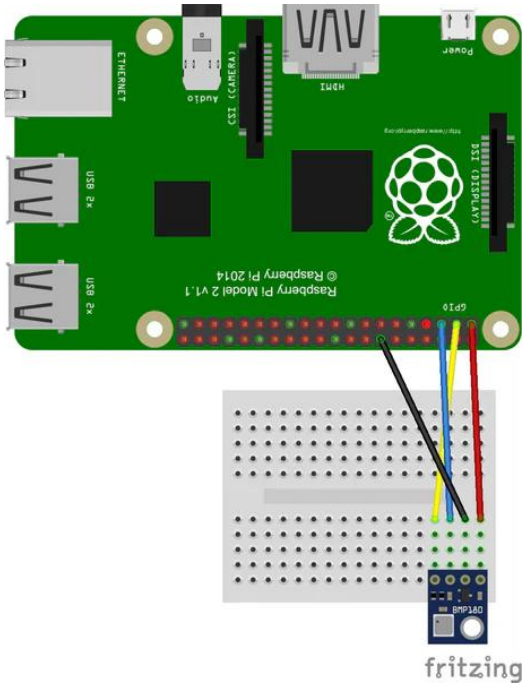


Figure 14: BMP180 wiring diagram (The Pi Hut, 2015).

4.2.1.2. DS18B20

The Waterproof DS18B20 sensors equipped are suited for measuring temperatures in humid environments, both air temperature and underground for soil temperature readings. Chosen as they are cheap, reliable (+-0.5°c) and utilise the 1-wire protocol which is simple to use on Raspberry Pi devices operating within 3.0-5.0v (DFRobot, n.d.).

If wired in parallel (as in the wiring diagram on Figure 12), a network of these devices can be placed in different areas of the ground for collecting samples of soil temperature over a large area.

Soil temperature is an important factor in weather readings and agricultural output, as it is directly linked to the growth of temperate crops (such as most fruits and many vegetables) (Sabri et al., 2018). Furthermore, increased soil temperatures have proven to be a significant disruption to the microbial community present within the soil, negatively affecting the humidity, nutrient availability and overall plant growth (Zhao et al., 2024).

Secondly, air temperature is another important element in weather predictions and forecasting. By monitoring temperature variations, shifts in weather patterns can be identified, such as extreme weather conditions like droughts and heavy rain spells (Zidar, 2024).

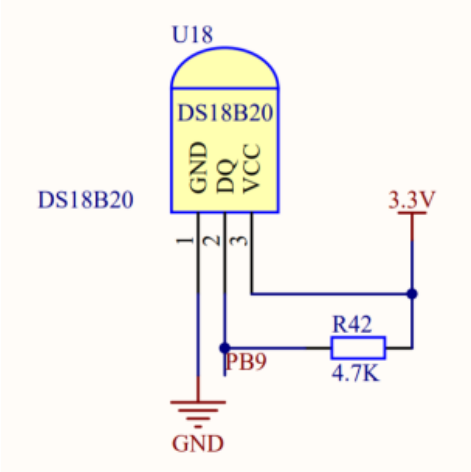


Figure 15: Wiring diagram for the DS18B20 sensor (Hu et al., 2022).

4.2.1.3. SHT41

The SHT41 sensor is employed for precise temperature and humidity readings, offering reliability with +-1.8% relative humidity at a low cost. The SHT41 also utilises the I^2C protocol so they can be easily daisy chained (wired in parallel) with the BMP180 making it simplistic to install and replace (Rembor, 2025).

Humidity readings are important in weather monitoring, forecasting and agricultural production as they can be a valuable indication of upcoming storms, severe rain, heatwaves, flooding and wildfires (Tuononen, 2023). By recording humidity, the early detection of these weather phenomena can be detected, assisting Manx farmers to take timely action.

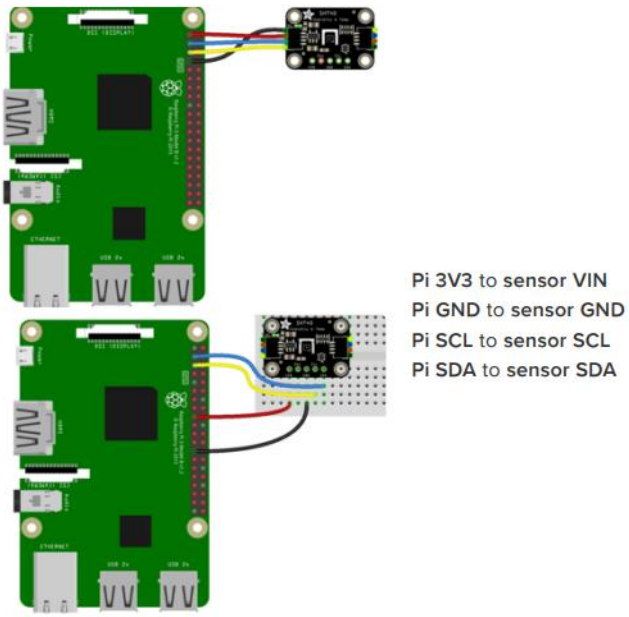


Figure 16: Wiring diagram for SHT41 (Rembor, 2025).

4.2.1.4. Anemometer

An anemometer is a device used for measuring wind speed. On a Raspberry Pi, this can be achieved by calculating the distance a magnet is rotated due to the wind by connecting it to a series of reed switches which sends a pulse of current to the Raspberry PI GPIO every partial revolution which, when accumulated over a period of time (for example 30 seconds) can be used to determine how much distance the magnet has moved, which is correlated to the wind speed (Raspberry Pi Foundation, 2023).

Wind speed is another factor that needs to be considered in weather forecasting, as it plays a key role in the movement of weather systems across the globe, driven by high and low-pressure air interactions (National Oceanic and Atmospheric Administration, 2019).

Figure 17: Basic anemometer (Raspberry Pi Foundation, 2023).

4.2.1.5. Zeta RF Transceiver module

The Zeta RF Transceiver modules are a cost-effective, high-performance series of wireless transceiver modules capable of reaching up to 2000 metres at 500kbps (RF Solutions, n.d.). These sensors conform with current UK regulations on radio communication for the 433mhz band, with the declaration of conformity available in Appendix 5. Moreover, GIPdA (2016) released a GitHub project with example code to easily interact with the modules with Raspberry Pi devices.

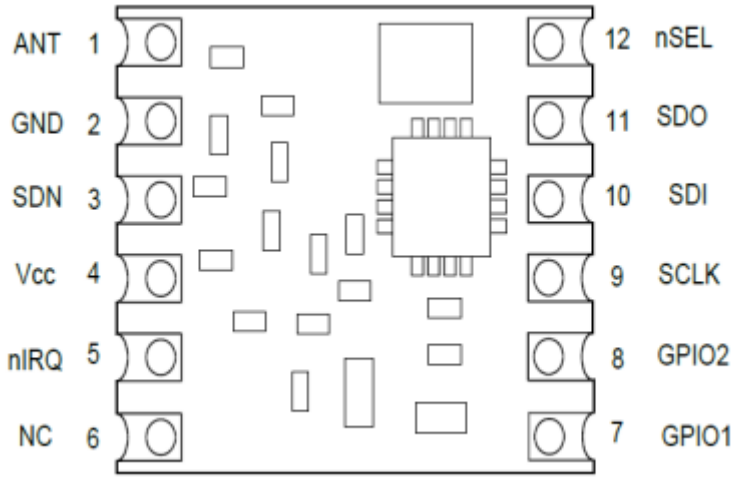


Figure 18: Zeta RF pinout diagram (RF Solutions, n.d.).

4.2.2. Software

The software enables reliable wireless transmission of weather data in a simple, open and extensible format. Developed in Python for compatibility with Raspberry Pi and the hardware outlined in section 4.2.1, it uses a modular design with each component containing its own module, linked through the main file for data processing and transmission. This simplifies testing and future updates without disrupting future code. This software follows the structure below:

```
weather_station/
│
├── main.py
├── bmp180_module.py
├── ds18b12_module.py
├── sht40_module.py
├── wind_module.py
└── rf_module.py

weather_receiver/
│
├── main.py
```

Figure 19: Software structure.

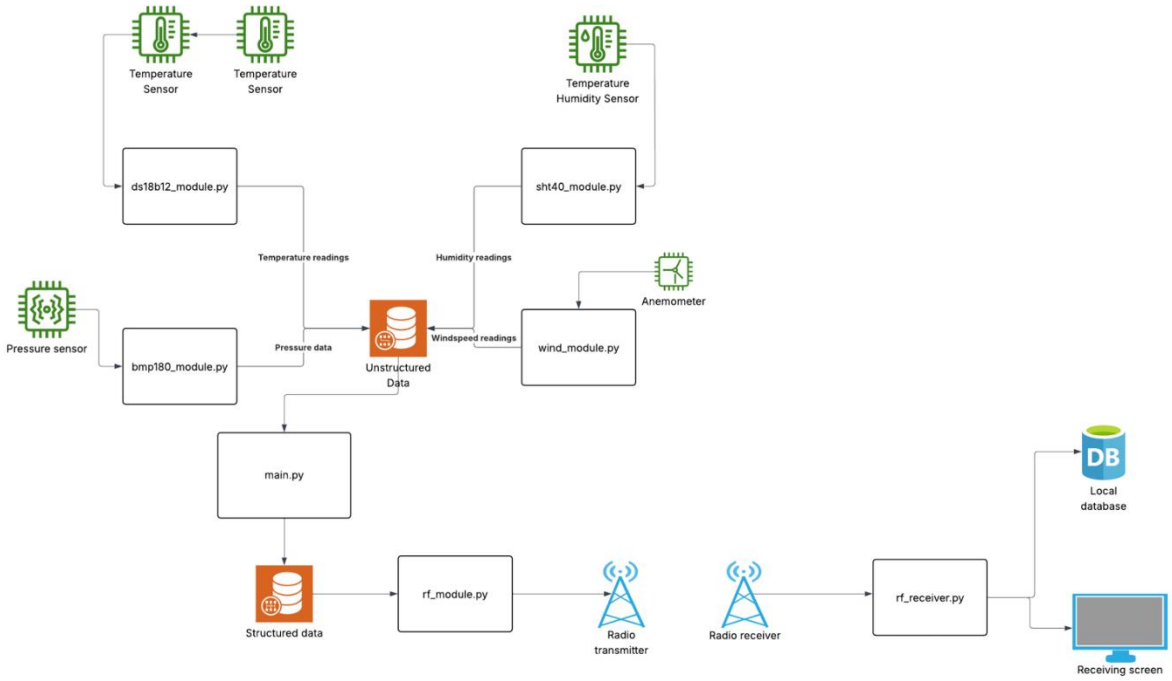


Figure 20: Software diagram of the proposed weather collection and retrieval systems.

4.2.2.1. Dependencies

The following external libraries have been imported into the project for easily integrating the hardware together:

4.2.2.1.1. Adafruit_Python_BMP

The Adafruit Python BMP library allows easily access to the BMP series of pressure and temperature sensors (Adafruit, 2018).

4.2.2.1.2. Adafruit_SHT4X

Similarly to the BMP library provided by Adafruit, their SHT4X library enables easy code integration with the Adafruit series of SHT4X devices (Adafruit, 2024).

4.2.2.1.3. Board

A library available via pip which enables the SHT library to access the physical I^2C interface on the Raspberry Pi (Golden, 2019).

4.2.2.1.4. RPi.GPIO

The RPi.GPIO pip package provides a simplistic approach to controlling Raspberry Pi GPIO pins with Python (Croston, 2022).

4.2.2.1.5. ZetaRF

A library for interacting with the SiLabs Low Current Sub-GHz wireless transceivers, such as the ZetaRF (GIPdA, 2016).

4.2.3. Network Design

4.2.3.1. Topology and Antenna Strategy

The proposed network employs a star topology with a hybrid antenna configuration to balance coverage and signal integrity. The weather stations utilise Yagi-uda antennas focused on the receiver which utilises an omnidirectional dipole antenna to radiate RF and receive data from a 360° radius.

This setup should be optimal for a star topology for this use-case, however one downfall with the use of an omni-directional receiver would be a hindrance of the maximum range between transmitter and receiver. The use of multiple directional antennas would increase this distance (SRFS Teleinfra, 2024).

The network operates at 433.1Mhz to minimise foliage attenuation and comply with Ofcom regulations.

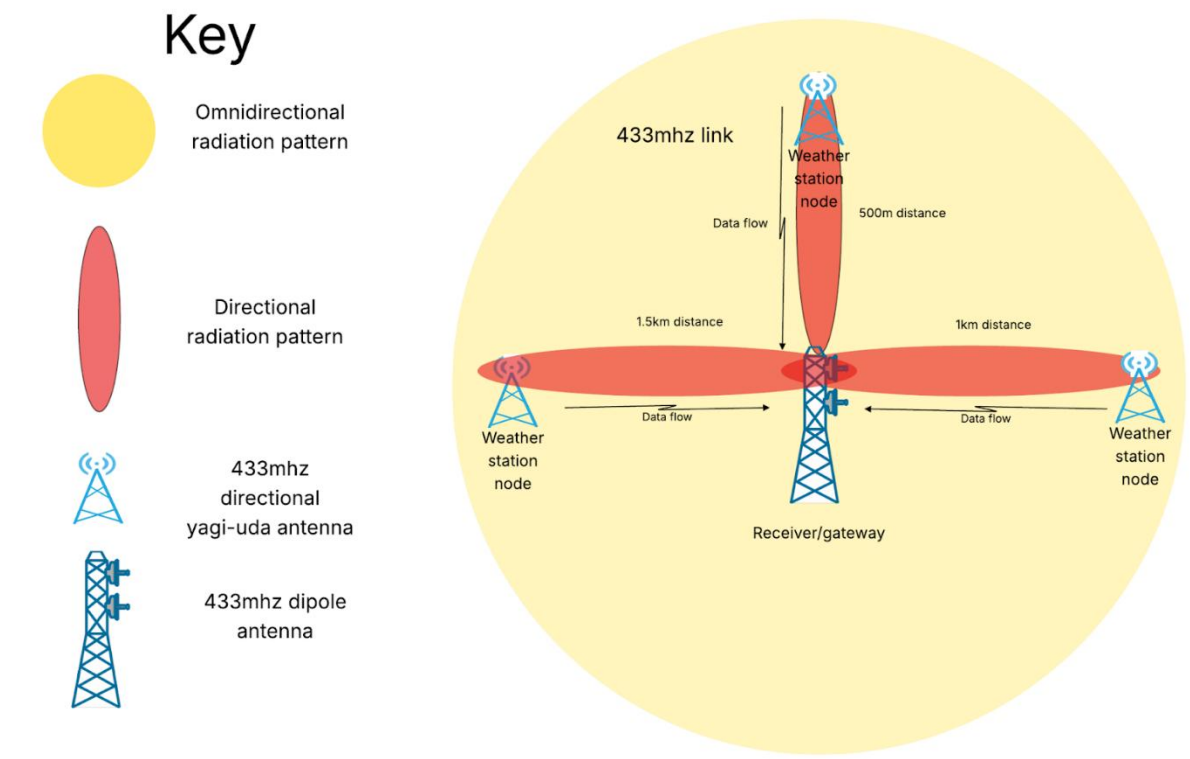


Figure 21: Proposed network diagram utilising the Star topology.

4.2.3.2 Security Considerations

Data integrity will be maintained through packet retransmission and AES-128 encryption, protecting against unauthorised access. FHSS is also considered as this reduces interference and interception. As shown in Table 3 (Section 2.9), AES offers strong security while using less resources and power compared to RSA. Although 128-bit keys will be used, they remain secure and vigilant against brute-force attacks and are considered extremely secure by current standards. The main drawback is the need to hardcode encryption keys into both the weather station and receiver, so physical security will also need to be considered. Key access requires removing the SD card, as the system can only be accessed through local protected SSH access.

4.2.4. Power requirements

Each weather station node in the network will be powered by a portable battery bank enclosed within the case, providing a simple and flexible solution to power management off-grid. Optionally, the devices could be externally powered however due to the nature of the field deployments, accessing on-grid power would be challenging. The Raspberry Pi Zero W's employed use around 100ma idle, with 140ma at full load (Kiwi Electronics,

n.d.). With all the sensors attached, it is estimated that they could use up to ~220mA. With a standard 10000mAh battery, such as the one decided in Figure 24, this would last up to 45.4 hours in continuous, 24/7 use. The system would only be active during data collection and transmission, which has been decided to be 5 minutes every hour, however this could be easily increased or decreased. At 10 minutes an hour, the battery could run for ~ 227 hours or 11.3 days before needing recharging, as per the calculation below:

$$Power\ consumption\ in\ hours = \frac{Battery\ capacity\ (mAh)}{Current\ draw\ (mA)} = \frac{10000}{220} = 45.45\ hours$$

$$10\ minute\ operation\ consumption = 220mAh \times \frac{10}{60} = 36.67mAh = \frac{10000}{36.67} = 272.5\ hours$$

Figure 22: Formula for calculating power consumption.

Power demands		
		Estimated consumption (mA)
Device	BMP180	0.005
	SHT40	0.15
	DS18B12 * 2	2
	RPi Zero W	120
	Zeta RF Tx	85
	Zeta RF Rx	13
		220.155

Figure 23: Estimated power consumption per device (mA)

Figure 24: Acquired battery banks for off-grid operations, providing up to 227 hours at 10000mAh (left) and 454 hours at 20000mAh (right)

4.2.1. Solar Power Integration

To reduce maintenance on power and enhance sustainability, solar panels can be easily integrated into the power delivery system in future iterations.

A common approach is to use a hybrid power model, whereby solar power is harvested throughout periods of high sunlight intensity to power the system and excess power is stored in batteries for use in low sunlight intensity and nights. By implementing a voltage reader into the system, the Raspberry Pi can also determine when the available power is too low to operate at regular intervals (Kumar et al., 2024).

4.2.5. Regulatory Compliance

The system will comply with Ofcom regulations regarding frequency allocation and use. Refer to Appendix 5 for the adherence to compliance.

Additionally, the project also considers broader legal implications. The Telecommunications Act 1984 and Wireless Telegraphy Act 2006 are two important pieces of legislation in this field which govern the acceptable use of radio frequency spectrum (The National Archives, 2025; The National Archives, 2022).

Furthermore, the European Telecommunications Standards Institute (ETSI) produces criteria for Short Range Devices, including a 10 mW Effective Radiated Power limit and <=10% duty cycle in the 433.050 – 434.790Mhz band, which aligns with Ofcom's IR2030/1/10 regulations.

F	169,4875 MHZ to 169,5875 MHz	10 mW e.r.p.	≤ 0,001 % duty cycle except for 00:00 h to 06:00 h local time where the duty cycle limit is ≤ 0,1 %	100 kHz
G	169,5875 MHz to 169,8125 MHz	10 mW e.r.p.	≤ 0,1 % duty cycle	225 kHz
H	433,050 MHz to 434,790 MHz	10 mW e.r.p.	≤ 10 % duty cycle	1,74 MHz
I	433,050 MHz to 434,790 MHz	1 mW e.r.p. -13 dBm/10 kHz power spectral density for bandwidth modulation larger than 250 kHz	No requirement	1,74 MHz

Figure 25: Maximum allowed Effective Radiated Power at 433.05 – 434.79Mhz

Future updates (e.g, live weather cameras and AI prediction models) must comply with the Data Protection Act 2018 if personal data is captured.

4.3. Testing methods

Various tests have been designed for validating performance, reliability and security, divided into two phases.

4.3.1. System testing

A full system test will be conducted in fields to assess the end-to-end performance of the network. Tests will be performed across varying distances to assess signal integrity and the maximum wireless range of different custom-built antennas.

4.3.2. Security testing

Additionally, security tests will be conducted where live captured signals will be analysed comparing encrypted and unencrypted transmissions.

Chapter 5: Implementation

5.1. Hardware implementation

5.1.1. Antenna construction

To test, three types of omni-directional antennas were acquired or constructed to measure and compare their efficiency over a range of distances.

Wireless communications over a long-range during testing can be found in Appendix 2I and 2J.

5.1.1.2. Monopole and spring antennas

Figure 27 shows two basic wire/monopole antennas for transmitting and receiving, each made from 6.8” stripped jumper wire, ideal as a quarter-wave antenna for 433 MHz. These provide better range than spring antennas, as can be calculated using Zaborowska’s (2025) formula below:

$$Wavelength\ (\lambda) = \frac{c\ (speed\ of\ light)}{f\ (frequency)\ (Hz)}$$

$$\lambda = \frac{300{,}000{,}000}{433{,}00{,}000} = 0.693\ metres$$

$$¼\ \lambda = \frac{0.693}{4} = 0.173\ metres = 6.8"$$

Figure 26: Antenna length formula.

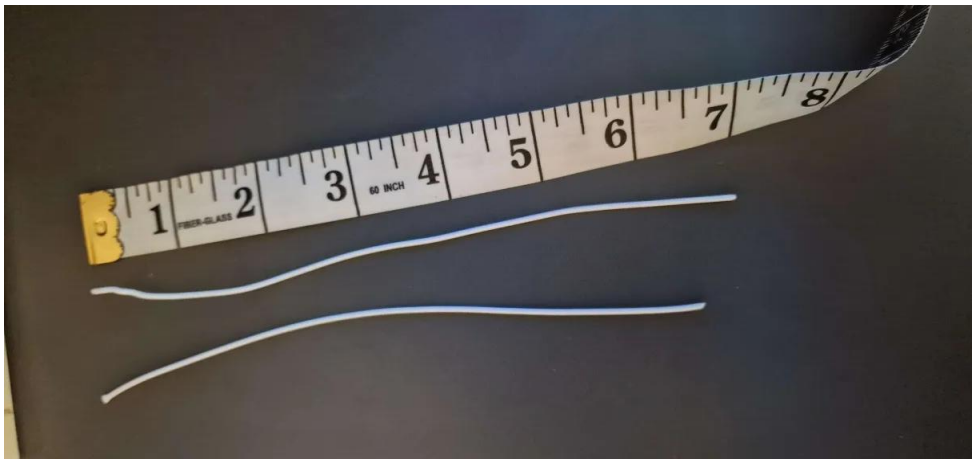


Figure 27: Two pieces of wire measuring ~6.8" acting as basic wire antennas.

Figure 28: Two spring antennas for low distance communication tuned to 433mhz.

5.1.1.3. Dipole antenna

Additionally, two dipoles were also constructed to serve as the main type of omnidirectional antenna used.

Dipoles are usually preferred over simple wire antennas due to their independence on a physical ground plane for operation (usually Earth which isn't always flat and reliable), as dipoles utilise a secondary conductor for the negative sides of a signal. This allows a

dipole to be mounted up in an elevated position, increasing gain and maximum coverage (Cadence PCB Solutions, 2024).

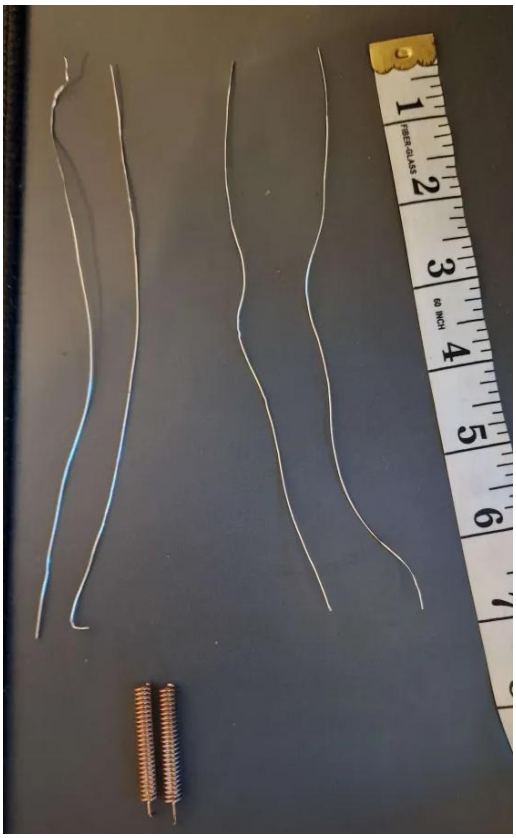


Figure 29: Two sets of ~6.8" wires used for the separate sides of the dipole antennas.

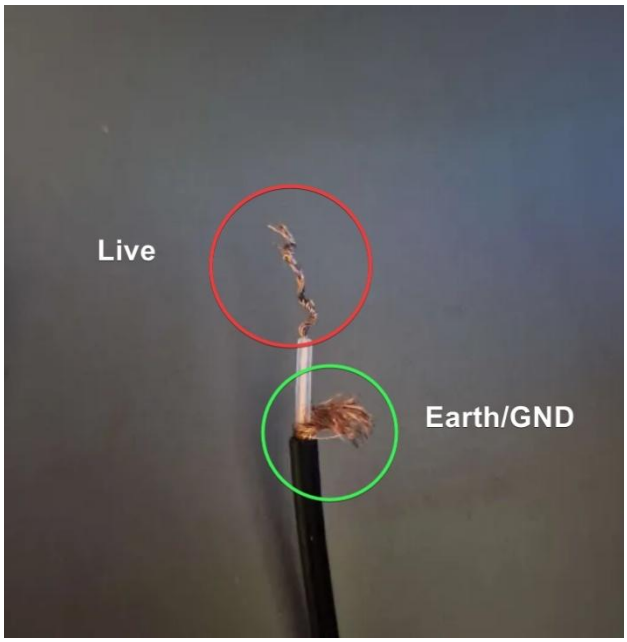


Figure 30: Stripped coaxial cable with separately spliced live and earth wires.

Figure 31: Soldered dipole antenna utilising coaxial cable and two 6.8” jumper wires to create a homemade dipole.

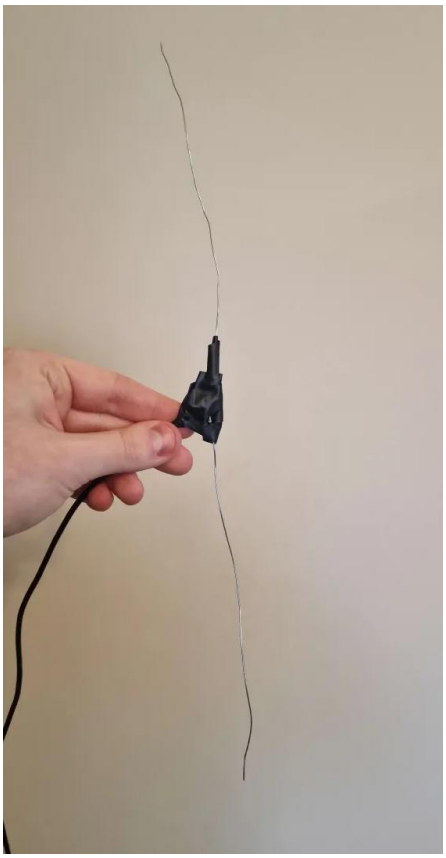


Figure 32: Completed dipole antenna.

Figure 33: Mounted dipole antenna in use.

5.2. Software implementation

From a software perspective, the project can be separated to two main sections which manage the data collection, processing and transmission of the weather station and then the section for receiving weather data:

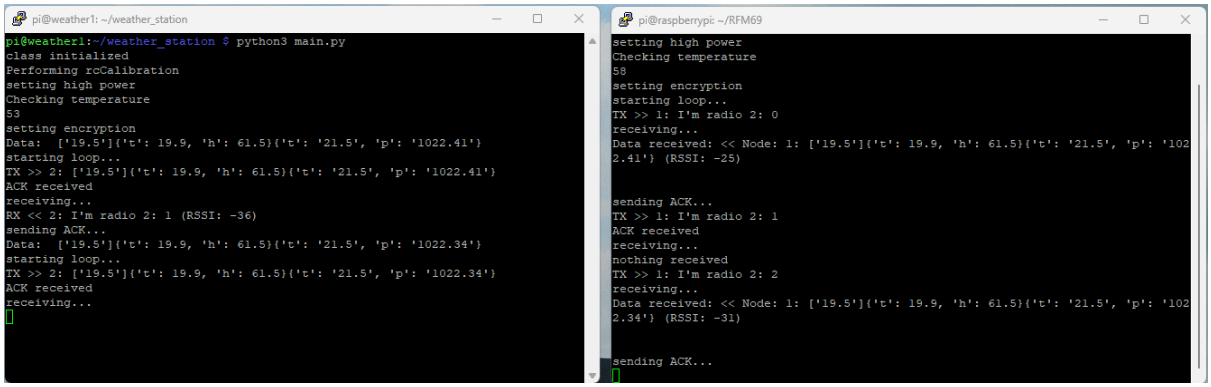


Figure 34: Weather station terminal output (left) and receiving endpoint terminal (right) communicating with each other.

5.2.1. Weather station node core modules

5.2.1.1. bmp180_module.py

The BMP180 module first imports the Adafruit_BMP.BMP085 library to easily interact with the hardware.

```
import Adafruit_BMP.BMP085 as BMP085
```

Table 7: Code snippet importing the Adafruit BMP085 library.

Next, two functions are created: initialise_sensor and read_bmp180. Initialise sensor is imported into the main file where it is accessed directly. Read_bmp180 utilises the Adafruit library to directly access the temperature and pressure readings from the device, where it is formatted before being returned:

```
def initialise_sensor():
    sensor = BMP085.BMP085()
    return sensor

def read_bmp180(sensor):
    temp = sensor.read_temperature()
    pressure = sensor.read_pressure() / 100
    # alt = sensor.read_altitude()
    tData = "{0:0.1f}".format(temp)
    pData = "{0:0.2f}".format(pressure)
    return {'t': tData, 'p': pData}

sensor = initialise_sensor()
```

Table 8: Code snippet of the initialise_sensor and read_bmp180 functions.

5.2.1.2. ds18b20_module.py

This module sets up some base variables. “Sensors” is a hardcoded array of 1-Wire addresses for the DS18B20 sensors and base_dir which is the directory of the 1-wire devices. Finally, a count loop is initialised for future use in read_temps:

```
sensors = ["28-01193a919c84/"]

base_dir = '/sys/bus/w1/devices/'

count = 0
```

Table 9: Code snippet of initial variables in the ds18b20_module.py script

Next, two main functions are defined, getData and read_temps. getData processes the raw data from the 1-Wire devices by looping over each sensor and reading the file that is created with the value which is then appended to a data array to passed onto read_temps:

```
def getData():
    #sleep(1)
    data = []
    for sensor in sensors:
        tempFile = open(base_dir + sensor + "w1_slave")
        tempText = tempFile.read()
        tempFile.close()
        value = sensor + tempText
        data.append(value)
    return data
```

Table 10: Code snippet of the getData function

Read_temps is the function that handles the raw value and formats it into a usable temperature. This is necessary due to the DS18B20 providing raw values such as:

```
f3 01 4b 46 7f ff 0c 10 7e : crc=7e YES
f3 01 4b 46 7f ff 0c 10 7e t=31875
```

Table 11: Raw output of the DS18B20 sensor

The temperature values are encoded in the first two bytes in the little-endian byte order (in this case f3 and 01) which generates the hex 0x01f3 which is 499. This value is then multiplied by 0.0625 as it is 1/16 of the temperature, which equals 31.1875 degrees. The temperature is also included at the end of the output “t=31875”.

This is where read_temps comes in. Read_temps loops over every line of data from the various sensors outputted by getData and searched for the key value “t=” where the temp_string value is equal to the position of “t=” + 2, so all of the values two characters after t, which is 31875. This value is then divided by 1000 to show the temperature at 31.875 degrees:

```
def read_temps():
    lines = getData()
    count = 0
    data = []
    for line in lines:
        sPosition = lines[count].find("t=")
        temp_string = (float(lines[count][sPosition+2:]))
        count = count + 1
        temp_c = temp_string /1000
        temp_c_formatted = "{:.1f}".format(temp_c)
        data.append(temp_c_formatted)
    return data
```

Table 12: Analysis of read_temps function

5.2.1.3. rfm69_module.py

This module imports all the required libraries from the RFM69 library. Next, the main variables are configured, such as configuring the network (explained in detail in the Network implementation section). After this, the code sets up the correct registries, frequency, power level, encryption keys and calibrates the device before running the main loop:

```
radio.rcCalibration()
radio.encrypt(KEY)
radio.setFrequency(433100000)
radio.writeReg(0x11, 0x5C)
radio.writeReg(REG_BITRATEMSB, RF_BITRATEMSB_1200)
radio.writeReg(REG_BITRATELSB, RF_BITRATELSB_1200)
```

```
radio.writeReg(REG_FDEVMSB, RF_FDEVMSB_5000)
radio.writeReg(REG_FDEVLSB, RF_FDEVLSB_5000)
```

Table 13: Analysis of rfm69_module.py

The main loop consists of sending the weather data packets and retrying 3 times, waiting for an acknowledgement each time with a 500ms delay. Conditions are implemented for both timeouts and acknowledgements where the RSSI value is printed to the screen alongside the packet's receiver:

```
  if radio.sendWithRetry(OTHERNODE, msg, 3, 500):
            print("ACK received")

            print("receiving...")
            radio.receiveBegin()
            timedOut = 0
            while not radio.receiveDone():
                timedOut += TOSLEEP
                time.sleep(TOSLEEP)
                if timedOut > TIMEOUT:
                    print("nothing received")
                    break

            if timedOut <= TIMEOUT:
                sender = radio.SENDERID
                msg = "".join([chr(letter) for letter in radio.DATA])
                ackReq = radio.ACKRequested()
                print(f"RX << {sender}: {msg} (RSSI: {radio.RSSI})")
                if ackReq:
                    print("sending ACK...")
                    time.sleep(0.05)
                    radio.sendACK()
                time.sleep(TIMEOUT / 2)
```

Table 14: analysis of main rfm69_module.py loop

Finally, a KeyboardInterrupt is implemented so when the program is exited by the user, the radio can shut down and clean the GPIO pins up gracefully:

```
except KeyboardInterrupt:
        pass
        print("shutting down")
        radio.shutdown()
```

Table 15: KeyboardInterrupt analysis

5.2.1.4. sht40_module.py

Similarly to the bmp180_module, sht40_module imports a third party Adafruit library (adafruit_sht4x) with two methods, one for initialising the sensor that is utilised in the main.py script and a further method for reading and processing the data. The only data processing in this module is rounding the temperature and humidity to one decimal place to limit the payload size:

```
import adafruit_sht4x
def initialise_sensor():
    sht = adafruit_sht4x.SHT4x(board.I2C())
    return sht
def read_sht40(sht):
    temperature = round(sht.temperature, 1)
    humidity = round(sht.relative_humidity, 1)
    return {'t': temperature, 'h': humidity}
```

Table 16: Initialise_sensor and read_sht40 function analysis.

5.2.1.5. main.py

Main.py acts as the central data aggregation and processing module. The previous modules are accessed in this module and initialised where their methods are called to collect each component's data:

```
import ds18b12_module
import bmp180_module
import sht40_module
import rfm69_module
import time
sht40 = sht40_module.initialise_sensor()
 bmp180 = bmp180_module.initialise_sensor()
```

Table 17: Main program module importation and initialisation

The data is passed through a large try except loop where error handling is in place in the event of hardware failure without crashing the whole program. The result of each components data is then added to its own variable:

```
 while True:
            # Read DS18B12 sensors
            try:
                temperature_ds18b12 = ds18b12_module.read_temps()
            except Exception as e:
                temperature_ds18b12 = None
                print(f"Error reading DS18B12: {e}")


            try:
                pressure = bmp180_module.read_bmp180(bmp180)
            except Exception as e:
                data_bmp180 = {'temperature_bmp180': None, 'pressure':
None,    'altitude': None}
                print(f"Error reading BMP180: {e}")


            try:
```

```
                humidity = sht40_module.read_sht40(sht40)
            except Exception as e:
                sht40 = {'temperature_sht40': None, 'humidity': None}
                print(f"Error reading SHT40: {e}")
```

Table 18: Analysis of main data aggregation and error handing loop

After the data has been collected, it is appended together as a single string before being passed onto the RFM69HCW module:

```
data = str(temperature_ds18b12) + str(humidity) + str(pressure)
rfm69_module.send(str(data))
```

Table 19: Analysis of appending data as a single string to be passed onto transmission module

Finally, keyboard interrupt loop is created for the main thread so when ctrl+c is executed, the program can terminate safely (specific GPIO cleanups happen in the above module files)

```
if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        print("Program terminated by user")
    except Exception as e:
        print("An error occurred" + e)
```

Table 20: Main calling loop with error handling and keybord interrupt handling

An updated wiring diagram, based on the modules discussed in this section and updated based on chapter below has been created to reflect the changes to the system requirements:

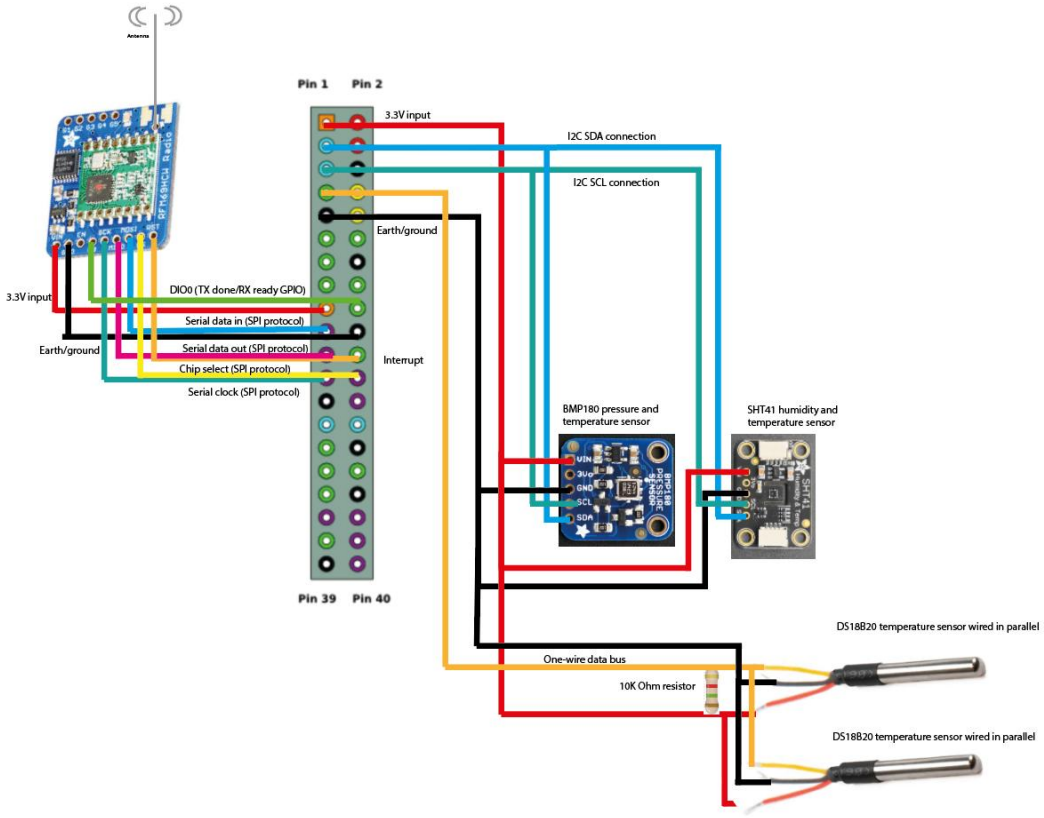


Figure 35: Updated weather station wiring diagram based on changes implemented in section.

5.2.2. Receiver

5.3.1. main.py

The receiver only has one module, the main.py script which is almost identical to the rfm69_module.py used in the weather station.

The same registers, frequency, power and encryption keys are setup as these need to match to form a connection, with the only difference being the NODE and OTHERNODE variables:

```
NODE = 2
OTHERNODE = 1
NET = 1
KEY = "1234567890123456"
TIMEOUT = 6
TOSLEEP = 0.1

radio = RFM69.RFM69(RF69_433MHZ, NODE, NET, True)
```

```
radio.rcCalibration()

print(radio.readTemperature(0))

radio.encrypt(KEY)

radio.setFrequency(433100000)

radio.writeReg(0x11, 0x5C)

radio.writeReg(REG_BITRATEMSB, RF_BITRATEMSB_1200)
radio.writeReg(REG_BITRATELSB, RF_BITRATELSB_1200)

radio.writeReg(REG_FDEVMSB, RF_FDEVMSB_5000)
radio.writeReg(REG_FDEVLSB, RF_FDEVLSB_5000)
```

Table 21: Analysis of main receiving code

The same loop is created, however this time instead of packets of weather data being transmitted, it is essentially creating a ping request to the weather station, so it knows that the receiver is still active and listening for data:

```
       msg = "I'm radio %d: %d" % (NODE, sequence)
        sequence += 1

        print(f"TX >> {OTHERNODE}: {msg}")
        if radio.sendWithRetry(OTHERNODE, msg, 3, 500):
            print("ACK received")
```

Table 22: Analysis of pinging loop

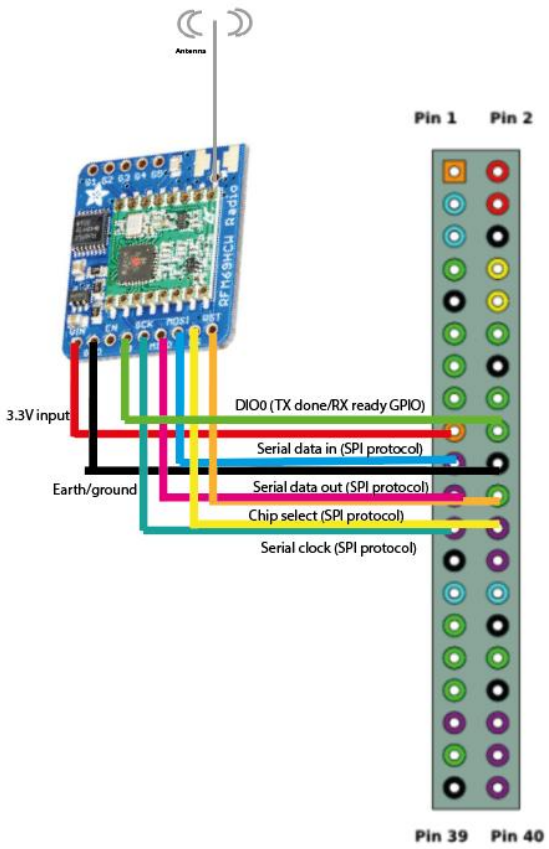


Figure 36: Receiver wiring diagram

5.3. Network implementation

As RFM69HCW's are based around the star topology, a simple network has been configured utilising the RFM69 library available on Github by Trombly (2021).

Python code	**Explanation**
NODE = 1	Creates a variable for the current device's node ID. In the case of the weather station, this has been set to 1.
OTHERNODE = 2	The other node variable has been set to represent the receiving device, which has been set to 2. As only two devices are configured on this network, it would be

	advisable to set the base station/receiver to node 1 with all of the weather stations continuing after that.
NET = 1	This is the network ID for the initial weather station network. This can be set from 1-255 if a larger farm would prefer to split each section into its own network, or if multiple nearby farms are operating on the same system to prevent interference.
radio = RFM69.RFM69(RF69_433MHZ, NODE, NET, True)	The radio class is initialised utilising the node and network ID.
if radio.sendWithRetry(OTHERNODE, msg, 3, 500):	Some logic which transmits data to the OTHERNODE variable (the receiver), sending the same packet 3 times with a 500ms delay whilst it awaits for an acknowledgement.
print(f"Data received: << Node: {sender}: {msg} (RSSI: {radio.RSSI})")	This sample is from the receiver, where it utilises the sender's transmitted node ID to distinguish itself from other nodes on the network. For further development, this could be used for implementing each node's entries separately into a database.

Table 23: Network implementation code analysis

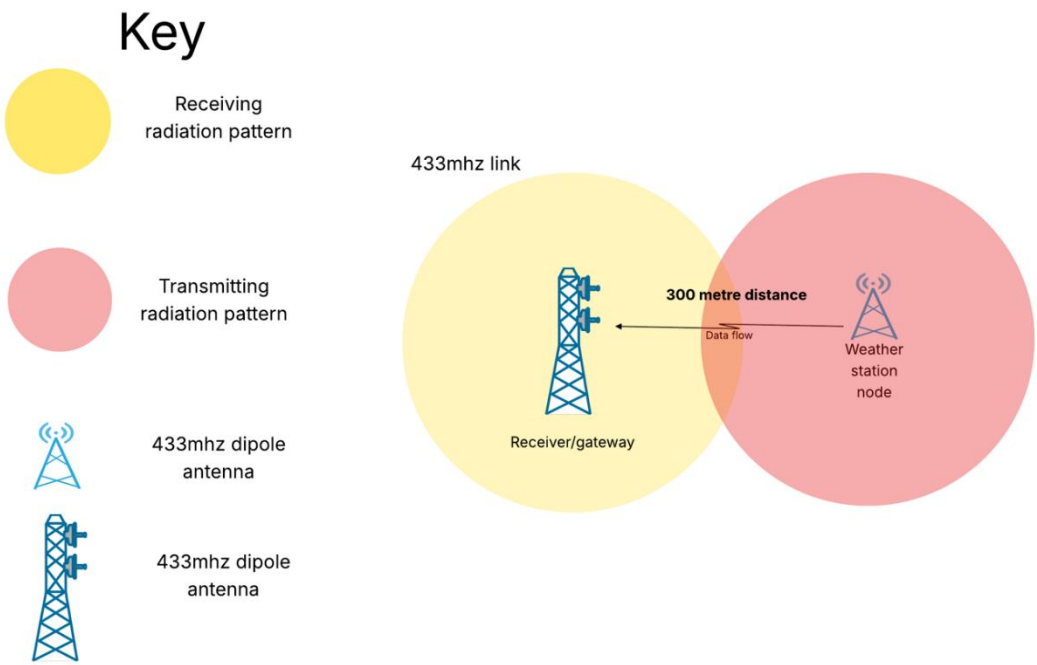


Figure 37: Updated network diagram based on the updated network topology and antenna strategy, as covered under Deviations

5.4. Implementation deviations, problems and rationale

Throughout the course of the implementation phase, it became evident that several initial strategies developed in the Design chapter were ineffective and impractical in practice. Due to the Agile nature of this project, this was efficiently dealt with through multiple iterative cycles, allowing for continuous improvement and adaptation as new knowledge was gained and understood.

5.4.1. Deviations

Original plan	**Change implemented**	**Rationale for change**
1: The use of a hybrid antenna approach for the network, utilising a mix of omni-directional and	During implementation, omni-directional antennas were used for both the weather station and receiving base station.	**Time constraints**: the construction and alignment of directional antennas proved more time consuming than would be feasible throughout the course of this project.

directional antennas.		**Resource limitations**: Directional antennas and appropriate mounting hardware were more expensive and harder to acquire than their omni-directional counterparts. **Technical complexity**: The construction of appropriate and efficient directional antennas added a layer of complexity which increased the chances of signal loss and misconfiguration which was not feasible with limited time allocated for antenna design and construction.
2: The use of ZetaRF modules for the transmission and receiving of weather data	RFM69HCW's were implemented instead of the ZetaRF modules.	**Technical complexity and limited support**: Online libraries for use of the Zeta RF were mainly available for Arduino based microcontrollers, with limited support available to get working on Raspberry Pi. Various tests were performed to attempt to transmit data over-the-air, to no avail. The RFM69HCW's are mostly tailored for Raspberry Pi's, with plenty of libraries to choose from online. These chips are also built for star topology networks, where each device requires and transmits its network and node ID by default, allowing a single network of up to 254 devices.

3: Measuring wind speed	Wind speed monitoring was removed from the project.	**Hardware damage**: The original anemometer used was damaged, resulting in one of the three cups detaching from the rest of the device (see Figure 38). This could've been glued and taped back on, but the added weight would've resulted in recalibrating the equipment with a chance that the same issue could occur with one of the other two cups. **System change**: It was decided that a new anemometer would be used as a replacement, this time using a harder, sturdier model made out of metal to prevent damage (see Figure 39). After acquiring the new anemometer, it was discovered that it sends the wind speed in an analogous format (increasing speeds resulting in a higher voltage output) unlike the previous digital system that would send a signal down to the GPIO pin at each 1/3rd of a revolution. Unfortunately, without an analogue to digital converter such as the MCP3008, the wind speed could not be monitored using this on a Raspberry Pi.
Implementing Frequency Hopping Spread Spectrum	This security technique was omitted.	FHSS was originally considered for its vast security benefits, however this was omitted due to the complexity not

		outweighing the practical use of this technique. FHSS requires two or more networked devices to pseudo-randomly alternate between different frequencies over a certain range, which requires the number generators to be in-sync, which, with the possibility of the weather station not being powered 24/7, would be difficult to track, resulting in the weather station transmitting out of sync with the receiver. Whilst effective and worthwhile for critical data transfers, this was simply not worth the additional time and workload for the minimal security benefit it possesses on weather data.

Table 24: Project deviations

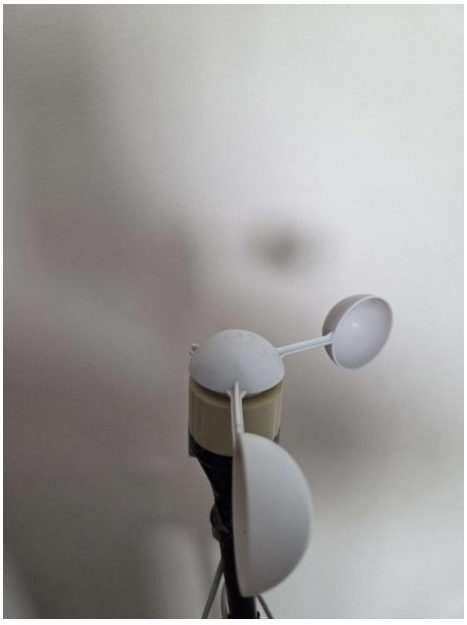


Figure 38: Original digital anemometer with one of the cups missing due to damage.

Figure 39: Replacement anemometer utilising an analogous transmission scheme.

5.4.2. Problems

Problem	**Description**	**Fix**
SSH in an off-grid environment	The initial project was developed using SSH over a LAN to deploy and execute code. It was realised quite late into the project that SSH in a remote environment without a LAN would be a severe issue, as access to the devices terminal wouldn't be possible due to the headless setup.	Various attempts were made to combat this problem, including the transition to a Raspberry Pi Pico microcontroller for direct access to the system over a USB (see Figure 40 below). It was later discovered that an open source RNDIS driver is available on Github by dukelec (2019) that allows the Raspberry Pi to be detected as a network device over USB, which can be directly connected to over SSH using Putty and the hostname of the Pi.
Payload cut off	The initial payload contained too much data and needed to be minimised to increase the amount of weather data that	Referring to the spec sheet available by HopeRF Electronic (n.d.), the FIFO max payload size is 66 bytes.

	can be sent via a single packet.	The original data format was 111 bytes long in UTF-8, resulting in almost half of the payload being ignored: {'t_ds18': [19.687, 31.437], 'sht40': {'t_sht40': 19.34, 'h': 54.36}, 'p': {'t_bmp': '20.60', 'p': '1020.23'}} This can be fixed by splitting the data up into multiple packets (a process known as fragmentation), however this adds extra complexity (such as implementing reassembly logic) and the need to send extra data. The fix implemented in this project was to instead encode the weather data in the most efficient format. Multiple iterations have been performed on this data structure, with the following being the currently used version which is 59 bytes: ['18.2']{'t': 18.5, 'h': 59.3}{'t': '19.7', 'p': '1016.22'} In future expansions of this project, a more efficient solution would simply be the following (at only 31 bytes long, nearly ¼ the size of the original string): [18.2, 18.5, 59.3, 19.7, 10162] This is due to each character taking up a single byte. This method reduces unnecessary data while still ensuring its decodability. For example, all data variables are limited to 1 decimal place with pressure being scaled by 10 to reduce the singular byte required by the decimal point. As long as the

		order of the data is known and structured in the same order on every transmission, the receiver would be able to decode this back into a more usable format.
Antenna element damage	The use of 22AWG Dupont wire presented an issue when splicing the rubber shields. It was discovered that performing the removal of the rubber shields with a wire stripper would sometimes cause slight indents in the wire which negatively affect its propagation characteristics.	The wire could be replaced with pre-stripped 22AWG Dupont wire to prevent the damage caused, or an entirely different material could be selected to act as the antenna.
Flexing antenna elements	The antenna elements for the dipole and wire antennas use 22AWG wire which is typically used for prototyping circuits.	The characteristics of 22AWG wire for radio wave propagation is completely fine, the issue arises when the antenna is in an outdoor, open environment with wind. Throughout testing, it was realised that the antenna arms would flex considerably in the wind, negatively affecting its radiation pattern, thus leading to reduced range and inconsistent results. A simple fix would be the use of 3mm metal rods (such as copper, brass or aluminium) which are study enough to withstand windy conditions.

Table 25: Problems encountered and fixes

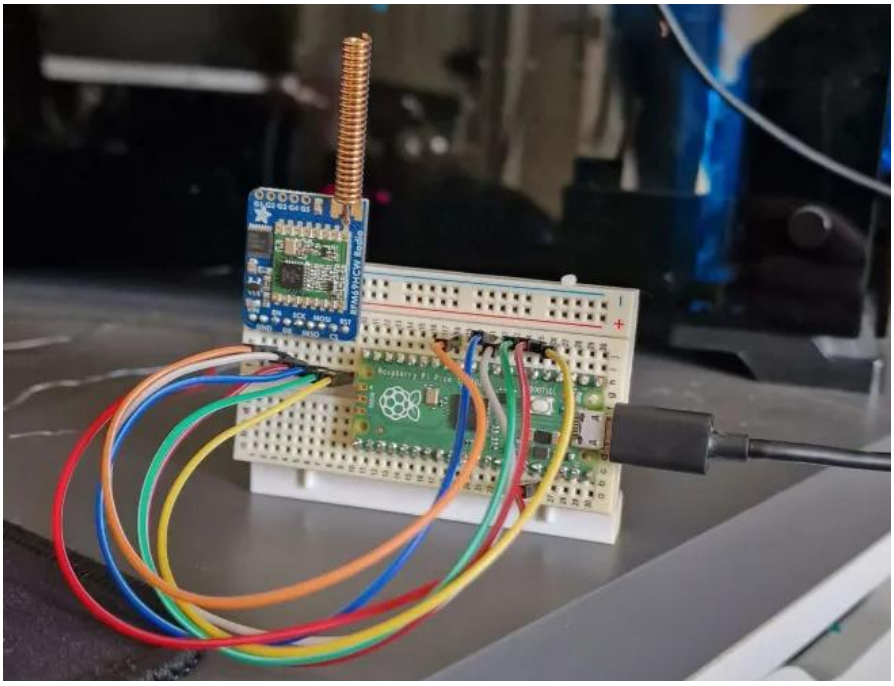


Figure 40: Testing the feasibility of transitioning over to a Raspberry Pi Pico microcontroller.

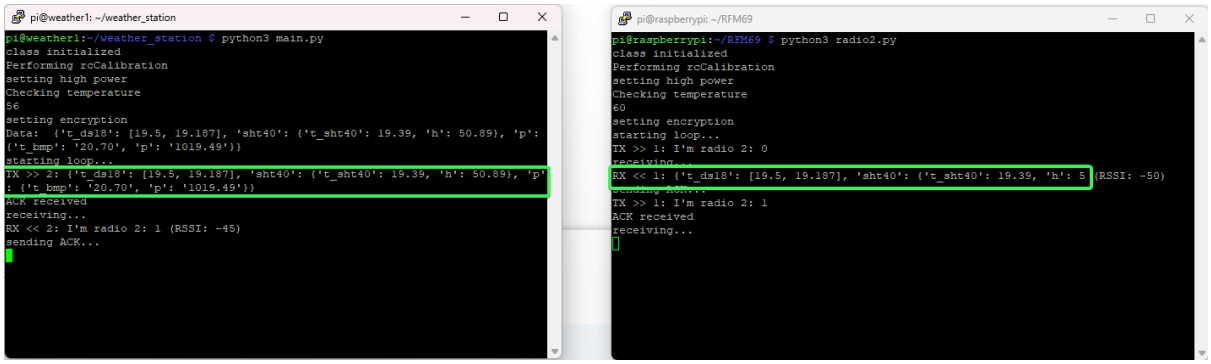


Figure 41: Visual example of the payload being cut off with the original encoding used. Note: everything after "h": 5 is omitted.

Chapter 6: Testing and Evaluation

This chapter details the testing and evaluation approach used to validate the wireless weather monitoring system.

Signal analysis

Alongside testing the end-to-end communication, a manual live analysis of the communications has been captured to understand the data structure in transit and attempt to perform a security test by analysing encrypted packets. This has been performed using a Software Defined Radio (SDR), specifically the RTL-SDR v4 (see Figure 42), which is a USB device that can monitor frequencies between 24 – 1766 MHz when paired with an appropriate antenna. When paired with software such as SDR++ which was used in this demonstration, a PC can monitor these frequencies in a live environment.

Figure 42: An RTL-SDR hooked up to a custom-made dipole antenna for real-time analysis of wireless communications.

Unencrypted signal analysis

By recording the raw IQ data into a .wav file, Universal Radio Hacker (URH) can be used to analyse and decode most waveforms and convert the raw data into hexidecimal and ASCII for easy analysis, alongside detecting preambles, headers and other meta data (Pohl, 2022). A simple, unencrypted packet can be seen being recorded in Figure 43 and decoded into ASCII in Figure 44.

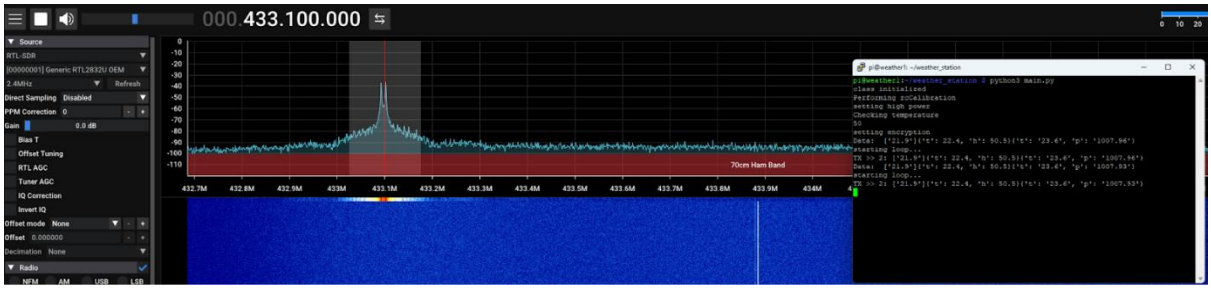


Figure 43: Spectrum analysis and waterfall display of an unencrypted transmission using SDR++ (Rouma, 2022).

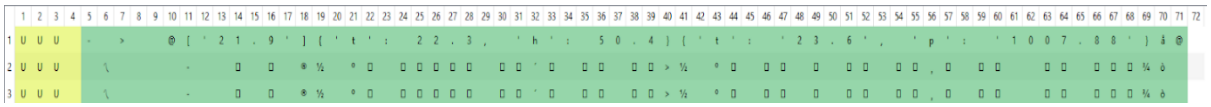


Figure 44: ASCII representation of an unencrypted weather station packet

As is expected, the raw unencrypted transmission can be easily eavesdropped against and decoded. It is worth noting that the 2nd and 3rd lines of Figure 44 also contain the 2nd and 3rd retransmissions of weather data, however sometimes a stray bit is implemented at the beginning of the transmission which causes the ASCII to be offset, looking completely jumbled, however this is normal behaviour which the RFM69HCW will perform bit shifting on to correct.

In the binary example below, the unshifted bit sequence for an unencrypted transmission correlates to:

“UUU□□ □□□□®½□º□□ ´□□¾½□º□□□□ ¸□□□□□□¾Ëì”

in ASCII, however, by removing the first bit of the transmission in a binary to ASCII editor reveals the following:

```
“ªªª->@['21.9']{'t': 22.3, 'h': 50.3}{'t': '23.6', 'p': '1007.91'} Ù”
```

The bit sequence for this example is below and can be reproduced:

101010101010101000101101000000010011111000000010000000010100000001011011001
001110011001000110001001011100011100100100111010111010111101100100111011101
000010011100111010001000000011001000110010001011100011001100101100001000000
010011101101000001001110011101000100000001101010011000000101110001100110111
110101111011001001110111010000100111001110100010000000100111001100100011001
1001011100011011000100111001011000010000000100111011100000001001110011101000
1000000010011100110001001100000001100000011011100101110001110010011000100100
1110111110110010111110110011111

Encrypted signal analysis

Whilst the above system works for unencrypted transmissions, this is significantly more difficult to perform on an encrypted transmission. From an initial viewport, the transmission looks the exact same on a spectrum analyser, as can be seen in Figure 45:

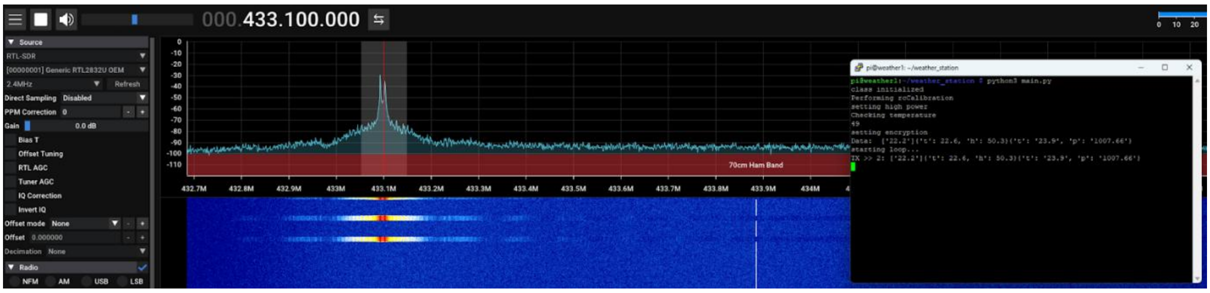


Figure 45: Spectrum analysis and waterfall display of an encrypted transmission using SDR++ (Rouma, 2022).

The differences are only in the payload section of the packet. This is because meta data and the packet structure need to remain unencrypted so that the receiver can determine if the packet is for them, and from the correct network. Other data such as preambles need to remain unencrypted so the packet can be decoded and the payload unencrypted, which would be difficult without knowing where the payload starts and finishes.

Figure 46: ASCII representation of the raw encrypted binary captured from the transmission using Universal Radio Hacker (Pohl, 2022).

As can be seen in Figure 46, the ASCII representation doesn't show a single piece of usable data, with or without bit shifting in place. To find and decrypt the weather data,

the payload section needs to be identified and unencrypted with the AES key that was used to encrypt it.

Code alterations

Throughout the course of testing, multiple issues and necessary adjustments were identified and addressed to improve the overall performance and reliability of the system.

Power amplifier

It was identified throughout testing that the transmitted signal strength was not adequate for long-range communication. Initial testing from a few metres distance identified that the signal strength should be much higher.

Initially, it was thought that the antennas were the culprit, however two pre-purchased spring antennas perfectly manufactured for 433Mhz were also tested with the same issue.

Various tweaks of the power amplifier registry were performed, which is the manual attempt at controlling the output power. The RFM69HCW documentation shows that 0x11 is the registry address for RegPaLevel, where the value 0x5C was written to (which is the power level 92 in decimal) on PA1 (Power Amplifier 1) (HopeRF Electronic, n.d.). This is to remain compliant with local regulations, as combining power amplifier 1 and 2 together boosts the maximum ERP to +20 dBm, with the legal limit being +10 dBm on the 433 MHz spectrum which is accomplished with just PA1 (refer to 4.2.5 for an in-depth review on this topic).

```
radio.writeReg(0x11, 0x5C)
```

Figure 47: Final registry edit for the power amplifier to output the maximum legal ERP

Pa0On	Pa1On	Pa2On	Mode	Power Range	Pout Formula
1	0	0	PA0 output on pin RFIO	-18 to +13 dBm	-18 dBm + *OutputPower*
0	1	0	PA1 enabled on pin PA_BOOST	-2 to +13 dBm	-18 dBm + *OutputPower*
0	1	1	PA1 and PA2 combined on pin PA_BOOST	+2 to +17 dBm	-14 dBm + *OutputPower*
0	1	1	PA1+PA2 on PA_BOOST with high output power +20dBm settings (see 3.3.7)	+5 to +20 dBm	-11 dBm + *OutputPower*
Other combinations			Reserved		

Figure 48: Power Amplifier Mode Selection Truth Table (HopeRF Electronic, n.d.).

The signal strength can be increased or decreasing by altering these registry values as per the documentation, which can be visually noticed on the waterfall plot in Figure 45 which shows the difference between the hex value 0x5C and 0x50 set to the power amplifiers, noting the red boxed signal is significantly stronger than the green:

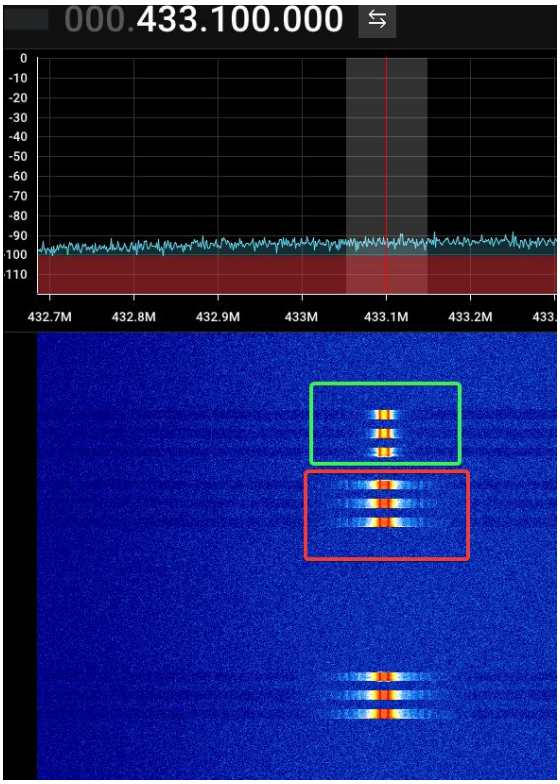


Figure 49: Waterfall analyser showing the signal strength variations when changing the power amplifier registry. Note: 0x5C (red) and 0x50 (green).

Payload minimisation

As discussed in section 5.4.2 under “Payload cutoff”, various iterations were made for minimising the payload size of the weather data for more optimised transmissions and

enabling future iterations to use unallocated bytes in the RFM69HCW's 66byte FIFO packet engine.

The first iteration consisted of a 111-byte structure:

{'t_ds18': [19.687, 31.437], 'sht40': {'t_sht40': 19.34, 'h': 54.36}, 'p': {'t_bmp': '20.60', 'p': '1020.23'}}

This first structure needed to be minimised to be within the maximum 66-byte payload, so various code changes were performed:

Original code	**Updated code**	**Justification**

`tData = temp()`
`pData = pressure()`
	`tData ="{0:0.1f}".format(temp)`
	`pData = "{0:0.2f}".format(pressure)`
	`return {'t': tData, 'p': pData}`
		This code adds a maximum decimal point value onto the pressure and temperature data which rounds them up to 1 and 2 decimal places respectively. This maintains an accurate, acceptable level of precision whilst reducing the byte count.

`data = [`
`temperature_ds18b12,`
`humidity,`
`pressure`
`]`
 	
`data = str(data)`
	`data = str(temperature_ds18b12) + str(humidity) + str(pressure)`	Originally, the weather data was structured into an array, which was then destructed into a string before being transmitted and reconstructed into an array on the receiving end.

		This method added extra unnecessary bytes by stringifying an array, so the process was simplified and encoded all of the weather data into a simple concatenated string.
`return {'bmp180_t': tData, 'bmp180_p': pData}`	`return {'t': tData, 'p': pData}`	Initially, each sensor would return an object with its name and data for each key-value pair. Whilst potentially useful in a system that utilises many different sensors all performing the same job, it was impractical to transmit the sensor names for this project as they are already known and can be distinguished due to the transmissions containing an ordered list of data that stays the same.

Figure 50: Table comparing original and update code with justification

Error handling

The lack of error handling in the initial program code would cause the whole program to crash if one sensor was unable to be read (for example, if the sensor needed replacing or the wiring came loose).

The initial program code shows a while True loop which runs continuously until the program is halted by the user, it simply calls the methods to retrieve data from each sensor:

```
while True:
    temperature_ds18b12 = ds18b12_module.read_temps()
    print("DS18B12 Temperature: ", temperature_ds18b12)

    pressure = bmp180_module.read_bmp180(bmp180)
    print("BMP180 Pressure: ", pressure)

    humidity = sht40_module.read_sht40(sht40)
    print("SHT40 Humidity: ", humidity)
```

Figure 51: Initial main.py program flow

Throughout testing, one of the DS18B20 sensors stopping polling information to the ds18b20_module.py code, which resulted in a hard crash within that module that resulted in the main process also being affected, which crashed the system and prevented all weather data from being resent before needing a manual reset.

Due to this, rigorous exception handling was implemented for each module in the main process, where the system calls "try" to attempt to call the methods and if this fails, the "except" block is executed where the error is printed to the terminal and a default value of "None" is applied to that variable, preventing hard crashes and allowing the system to remain robust:

```
while True:
    try:
      temperature_ds18b12 = ds18b12_module.read_temps()
    except Exception as e:
      temperature_ds18b12 = None
      print(f"Error reading DS18B12: {e}")
```

```
try:
    pressure = bmp180_module.read_bmp180(bmp180)
except Exception as e:
    data_bmp180 = {'temperature_bmp180': None, 'pressure': None, 'altitude': None}
    print(f"Error reading BMP180: {e}")
try:
    humidity = sht40_module.read_sht40(sht40)
except Exception as e:
    sht40 = {'temperature_sht40': None, 'humidity': None}
    print(f"Error reading SHT40: {e}")
```

Figure 52: Exception handling implemented in the main process

Transmission range results

Transmission range testing was carried out to evaluate the effectiveness of the wireless system across varying distances and antenna configurations within a realistic agricultural environment. For more detailed line graphs on each test, refer to Appendix 4.

Three tests were conducted over the course of a month with varying levels of wind, humidity and temperature with spring, wire and dipole antennas being used to compare against each other.

Distance (m)	Spring (T1)	Wire (T1)	Spring (T2)	Wire (T2)	Dipole (T2)	Spring (T3)	Wire (T3)	Dipole (T3)
0	-50	-34	-49	-32	-34	-46	-38	-44
25	-56	-54	-72	-50	-44	-67	-54	-44
50	-66	-73	-	-65	-50	-	-61	-53
75	-71	-76	-	-76	-59	-	-75	-61
100	-	-79	-	-82	-67	-	-83	-70
125	-	-85	-	-	-65	-	-	-76
150	-	-	-	-	-72	-	-	-84
175	-	-	-	-	-73	-	-	-77

200	-	-	-	-	-79	-	-	-80
225	-	-	-	-	-79	-	-	-80
250	-	-	-	-	-80	-	-	-82

Transmission range analysis

Antenna type	Maximum Range	Notes
Spring	~75 metres	Suitable for short distances or enclosed environments due to the compact size of the antenna
Wire	~125 metres	Improvement over the spring antenna, however still limited in maximum distance
Dipole	~250 metres	Best performing antenna with the most stable and reliable communication over long distances.

Table 26: Transmission range analysis

Findings

Spring antennas provided minimal range, with severe signal disruption past 25 metres, however unreliably maintaining ranges past 100 metres in some conditions. This makes spring antennas unsuitable for field deployments but could be useful in compact environments where the range between transmitter and receiver is ~20 metres.

Wire monopoles extended the range to ~100 metres, however this is with degraded signal quality.

The home-made dipole antenna setup provided the most reliable furthest range communication, with distances up to 250 metres recorded.

Adjustments to the power amplifier registry assisted in amplifying the signal whilst remaining compliant with local regulations.

Finally, the low-quality Dupont wire used for the antenna arms was not practical in real-world scenarios where wind would cause flexing in the wire, heavily distorting the radiation pattern of the antennas resulting in unreliable and unpredictable performance.

This can be evidenced with the wire antenna receiving -85 dBm RSSI in test one at 125 metres but receiving -83 dBm RSSI at the same distance in test 3. Using higher quality materials such as 3mm diameter brass rods which offer superior radiation characteristics and lack flexing due to their solid structure would be more practical. Equally, cheaper metals such as aluminium would also be suitable.

Chapter 7: Conclusion

Alignment with Original Aims

This project successfully met its core objectives by designing and implementing a cost-effective, modular and secure wireless weather monitoring network tailored towards Manx agriculture.

Aim	Outcome
Design and assemble a low-cost Raspberry Pi-based weather station	Achieved utilising various sensors within £5-£15 price range on a Raspberry Pi Zero
Integrate RF communications for data transmission	Achieved, with transmissions tested between 0 and 250m.
Implement lightweight encryption and test data integrity/reliability	Achieved using AES-128 built into the transceiver chip for optimal power and resource efficiency. Encrypted data packets were compared with unencrypted packets to determine if the encryption was sufficient.
Evaluate most efficient transmission ranges and sees the power consumption & cost-effectiveness of the system	Achieved, with battery life theoretically lasting up to 11 days without the use of solar power.

Table 27: Alignment with original aims

Reflection

The iterative, Agile-based approach to the development of this work allowed the project to adapt to real-world challenges, such as signal attenuation, hardware limitations and hardware failure, resulting in practical solutions such as enhanced error handling and the replacement of critical components such as the transceiver.

Strengths	Weaknesses

All of the initial project aims and objectives found in Chapter 1 were met successfully.	Hardcoded 1-wire addresses caused maintenance bottlenecks, as the user would have to manually update the array of serial numbers if additional 1-wire devices were added or removed, along with if any devices failed. This would cause the whole 1-wire implementation to fail as the system would try to access a device no longer on the system.
Secure data transmission using AES-128.	SSH access uses password authentication and should be replaced with key-based authentication using RSA or Ed25519.
Features a modular, scalable design with the ability to easily add and remove sensors and functionality.	Antenna elements made from Dupont wire flex in windy conditions, affecting performance
Real-world field testing was performed validating the system up to 150 metres.	Lack of solar charging, meaning a manual battery replacement and recharge will be required every ~11 days.
Compliance with ETSI and Ofcom regulations.	Limited transmission range compared to commercial solutions, which could be improved by using higher quality materials for the antenna, such as 3mm brass or aluminium rods.
Power-efficient, with theoretical operation lengths of up to 11 days.	Lack of physical security measures such as a lock and key.

Table 28: Reflection

Improvements and recommendations for future work

Upgrades to antenna materials, such as brass or aluminium rods, as opposed to Dupont wire, would improve durability and signal consistency in harsh weather.

Further integrations could include the use of solar charging with smart power management for increasing times between transmissions and data readings while on low power.

Additionally, linking the system to a database for long-term storage and analysis for future use in machine learning could enable smarter farm management via AI recommendations and alerts on current and upcoming weather events.

Final Thoughts

This dissertation provides a scalable blueprint for secure, off-grid weather monitoring solutions for agriculture. Its successful real-world deployment showcases the subject's feasibility in implementing smart-farming measures and increasing food security on the Isle of Man.

References

Aboshosha, B. W., Dessouky, M. M., & Elsayed, A. (2019). Energy Efficient Encryption Algorithm for Low Resources Devices. *The Academic Research Community Publication*, *3*(3), 26–37. https://doi.org/10.21625/archive.v3i3.520

Adafruit. (2018, August 30). *GitHub - adafruit/Adafruit_Python_BMP: Python library for accessing the BMP series pressure and temperature sensors like the BMP085/BMP180 on a Raspberry Pi or Beaglebone Black*. GitHub. https://github.com/adafruit/Adafruit_Python_BMP/tree/master

Adafruit. (2024). *Adafruit SHT4X Temperature and Humidity Sensor Breakout*. https://github.com/adafruit/Adafruit_SHT4X

Adafruit Industries. (n.d.). *BMP180 Barometric Pressure/Temperature/Altitude Sensor-5V ready*. Www.adafruit.com. https://www.adafruit.com/product/1603

Adobe. (2023). *Popular project management methodologies*. Adobe.com. https://business.adobe.com/blog/basics/methodologies

Amos, Z. (2024, November 25). *5 best encryption methods for IoT devices*. IOT Insider. https://www.iotinsider.com/iot-insights/technical-insights/5-best-encryption-methods-for-iot-devices/

Baby, R. (2024, March 6). *What is Frequency Shift Keying (FSK)?* RF Page. https://www.rfpage.com/what-is-frequency-shift-keying-fsk/

Balanis, C. A. (1992). Antenna theory: a review. *Proceedings of the IEEE*, *80*(1), 7–23. https://doi.org/10.1109/5.119564

Bensky, A. (2019). Regulations and standards. *Short-Range Wireless Communication*, 237–271. https://doi.org/10.1016/b978-0-12-815405-2.00010-5

Blokhin, Yu. I., & Blokhina, S. Yu. (2024). Wireless hybrid sensor network for agriculture monitoring. *BIO Web of Conferences*, *141*, 02025. https://doi.org/10.1051/bioconf/202414102025

Cabaccan, C. N., Cruz, F. R. G., & Agulto, I. C. (2017). Wireless sensor network for agricultural environment using raspberry pi based sensor nodes. *2017IEEE 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*. https://doi.org/10.1109/hnicem.2017.8269427

Cadence PCB Solutions. (2024, March 20). *Monopole vs Dipole Antenna*. Resources.pcb.cadence.com. https://resources.pcb.cadence.com/blog/monopole-vs-dipole-antenna

Chee, K. L., Torrico, S. A., & Kurner, T. (2011). Foliage Attenuation Over Mixed Terrains in Rural Areas for Broadband Wireless Access at 3.5 GHz. *IEEE Transactions on Antennas and Propagation*, *59*(7), 2698–2706. https://doi.org/10.1109/tap.2011.2152340

Cisco. (2007). *Antenna Patterns and Their Meaning*. https://www.industrialnetworking.com/pdf/Antenna-Patterns.pdf

Communications Commission. (2024). *Consultation on the allocation of spectrum in the 3.4 GHz band*. Www.gov.im. https://www.gov.im/media/1357471/consultation-on-the-allocation-of-spectrum-in-the-34-ghz-band.pdf

Cooper, V. (2024, November 12). *AES Encryption: How it works, Benefits, and Use Cases*. Splashtop.com. https://www.splashtop.com/blog/aes-encryption

Croston, B. (2022, February 2). *RPi.GPIO: A module to control Raspberry Pi GPIO channels*. PyPI. https://pypi.org/project/RPi.GPIO/

Dai, H.-N., Ng, K.-W., Li, M., & Wu, M.-Y. (2011). An overview of using directional antennas in wireless networks. *International Journal of Communication Systems*, *26*(4), 413–448. https://doi.org/10.1002/dac.1348

Department of Environment, Food and Agriculture. (2024a). *Isle of Man Agricultural Strategy 2024*. http://tynwald.org.im/index.php/spfile?file=/business/opqp/sittings/20212026/2024-GD-0113.pdf

Department of Environment, Food and Agriculture. (2024b). *Isle of Man Food Security Plan 2024*. http://tynwald.org.im/index.php/spfile?file=/business/opqp/sittings/20212026/2024-GD-0114.pdf

DFRobot. (n.d.). *Waterproof DS18B20 Digital Temperature Sensor For Arduino - DFRobot*. Www.dfrobot.com. https://www.dfrobot.com/product-689.html

dukelec. (2019). *mbrush/doc/win_driver at master · dukelec/mbrush*. GitHub. https://github.com/dukelec/mbrush/tree/master/doc/win_driver

European Telecommunications Standards Institute. (2024, March). *Short Range Devices (SRD) operating in the frequency range 25 MHz to 1 000 MHz with power levels ranging up to 500 mW e.r.p.; Part 2: Harmonised Standard for access to radio spectrum for non specific radio equipment*. https://www.etsi.org/deliver/etsi_en/300200_300299/30022002/03.02.02_20/en_30022002v030202ev.pdf

Franklin, R. (2022, November 14). *AES vs. RSA Encryption: What Are the Differences?* Precisely. https://www.precisely.com/blog/data-security/aes-vs-rsa-encryption-differences

GIPdA. (2016). *GitHub - GIPdA/ZetaRF: Arduino Library for SiLabs Low Current Sub-GHz Wireless Transceiver: Si4455 (ZETA modules), Si446x (DRF4463F modules)*. GitHub. https://github.com/GIPdA/ZetaRF

Golden, T. (2019, November 1). *board: Standard Board mechanism for Dojo tasks*. PyPI. https://pypi.org/project/board/

HopeRF Electronic. (n.d.). *RFM69HCW ISM TRANSCEIVER MODULE v1.1 GENERAL DESCRIPTION TAL ND*. https://cdn.sparkfun.com/datasheets/Wireless/General/RFM69HCW-V1.1.pdf

Hossein, N. (2011). Frequency Hopping Spread Spectrum: An Effective Way to Improve Wireless Communication Performance. *Advanced Trends in Wireless Communications*. https://doi.org/10.5772/15482

Hu, Z., Chen, F., Shen, R., & Li, S. (2022). Design and implementation of forest fire monitoring system based on internet of things technology. *ITM Web of Conferences*, *45*, 01063. https://doi.org/10.1051/itmconf/20224501063

Hübschmann, I. (2021, January 24). *A Developers Guide to IoT Encryption Algorithms*. Nabto. https://www.nabto.com/iot-data-encryption-algorithm-guide/

Hum, S. (n.d.). *Ionospheric Propagation Ionospheric Propagation*. https://www.waves.utoronto.ca/prof/svhum/ece422/notes/20c-ionosphere.pdf

Institute of Electrical and Electronics Engineers. (2019). *IEEE Standard for Definitions of Terms for Antennas*. https://doi.org/10.1109/ieeestd.2014.6758443

ITU. (2019). *ITU: Committed to connecting the world*. Itu.int. https://www.itu.int/

Kiwi Electronics. (n.d.). *Kiwi Electronics*. Kiwi-Electronics.com. https://www.kiwi-electronics.com/en/raspberry-pi-zero-version-1-3-2840

Krawczyk, H. (1994). LFSR-based Hashing and Authentication. *Springer EBooks*, 129–139. https://doi.org/10.1007/3-540-48658-5_15

Kumar, Y. Anil., Kumar, A. Pavan., Gopinath, A., Aruna Jeyanthy, P., & M, S. K. (2024). Smart Solar PV Charge Controller System for Off Grid Applications. *Smart Solar PV Charge Controller System for off Grid Applications*, *19*, 1883–1886. https://doi.org/10.1109/iccpct61902.2024.10673056

Loxley, R. (2022). Detail Analysis of Radio Waves in Applied Physics. *Research & Reviews: Journal of Pure and Applied Physics*, *10*(6), 3–4. https://doi.org/10.4172/2320-2459.10.6.002

Met Office. (2016, April 14). *Weather stations*. Met Office. https://weather.metoffice.gov.uk/learn-about/how-forecasts-are-made/observations/weather-stations

Monolithic Power Systems. (2024). *Analog Signals vs. Digital Signals*. Monolithicpower.com. https://www.monolithicpower.com/en/learning/resources/analog-vs-digital-signal

Moon, K. (2024, October). *Digital and analogue signals - IGCSE Physics Revision Notes*. Save My Exams. https://www.savemyexams.com/igcse/physics/cie/23/revision-notes/3-waves/3-3-electromagnetic-spectrum/3-3-5-digital-and-analogue-signals/

National Geographic. (2022). *Atmospheric Pressure | National Geographic Society*. Education.nationalgeographic.org. https://education.nationalgeographic.org/resource/atmospheric-pressure/

National Oceanic and Atmospheric Administration. (2019, November 20). *Measuring Winds to Help Predict the Weather*. Www.goes-R.gov. https://www.goes-r.gov/featureStories/measuringWindsToHelpPredictTheWeather.html

OECD. (2014). New Approaches to Spectrum Management. *OECD Digital Economy Papers*. https://doi.org/10.1787/5jz44fnq066c-en

Ofcom. (2021). *Guidance on EMF Compliance and Enforcement*. https://www.ofcom.org.uk/siteassets/resources/documents/spectrum/emf/guidance-emf-compliance-enforcement.pdf?v=326016

Ofcom. (2024). *Ofcom*. Www.ofcom.org.uk. https://www.ofcom.org.uk/spectrum/

Pohl, J. (2022, May 24). *jopohl/urh*. GitHub. https://github.com/jopohl/urh

Prodanović, R., Rančić, D., Vulić, I., Zorić, N., Bogićević, D., Ostojić, G., Sarang, S., & Stankovski, S. (2020). Wireless Sensor Network in Agriculture: Model of Cyber Security. *Sensors*, *20*(23), 6747. https://doi.org/10.3390/s20236747

QuarkTwin. (2024, June 14). *Understanding Analog and Digital Signal: Definition,Types, Advantages and Disadvantages*. Quarktwin Electronic; QuarkTwin. https://www.quarktwin.com/blogs/other/understanding-analog-and-digital-signal-definition-types-advantages-and-disadvantages/509

Raspberry Pi Foundation. (2023). *Build your own weather station*. Raspberrypi.org. https://projects.raspberrypi.org/en/projects/build-your-own-weather-station/5

RATEL. (n.d.). *PROPAGATION AND SIGNAL STRENGHT PREDICTION | Ratel*. Mapepokrivenosti.ratel.rs. https://mapepokrivenosti.ratel.rs/eng/prediction

Rembor, K. (2025). *Adafruit Sensirion SHT40, SHT41 & SHT45 Temperature & Humidity Sensors*. https://cdn-learn.adafruit.com/downloads/pdf/adafruit-sht40-temperature-humidity-sensor.pdf

RF Solutions. (n.d.). *Zeta FM Transceiver Module Available in 433, 868, 915MHZ*. RF Solutions. https://www.rfsolutions.co.uk/radio-modules/zeta-fm-transceiver-module-available-in-433-868-915mhz/

RF Spectrum Allocation | Cadence. (2024, February). *RF Spectrum Allocation*. Cadence.com. https://resources.pcb.cadence.com/blog/2024-rf-spectrum-allocation

Ristić, V., Todorović, B., & Stojanović, N. (2022). Frequency hopping spread spectrum: History, principles and applications. *Vojnotehnicki Glasnik*, *70*(4), 856–876. https://doi.org/10.5937/vojtehg70-38342

Rouma, A. (2022, November 22). *GitHub - AlexandreRouma/SDRPlusPlus: Cross-Platform SDR Software*. GitHub. https://github.com/AlexandreRouma/SDRPlusPlus

Saakian, A. (2020). *Radio Wave Propagation Fundamentals, Second Edition*. Artech House.

Sabri, N. S. A., Zakaria, Z., Mohamad, S. E., Jaafar, A. B., & Hara, H. (2018). Importance of Soil Temperature for the Growth of Temperate Crops under a Tropical Climate and Functional Role of Soil Microbial Diversity. *Microbes and Environments*, *33*(2), 144–150. https://doi.org/10.1264/jsme2.me17181

Salanitro, S. (2024, April 23). *How does a Yagi antenna work?* Wimo.com. https://www.wimo.com/en/faq/post/how-does-a-yagi-antenna-work

SRFS Teleinfra. (2024, January 30). *Omni-Directional vs. Directional Antenna - SRFS Teleinfra*. SRFS Teleinfra. https://www.srfsteleinfra.in/omni-directional-vs-directional-antenna/

Standage, T. (2009). *The Victorian Internet*. Bloomsbury Publishing USA.

Sunil Bhooshan. (2021). *Fundamentals of Analogue and Digital Communication Systems*. Springer Nature.

Tamir, T. (2003). On radio-wave propagation in forest environments. *IEEE Transactions on Antennas and Propagation*, *15*(6), 806–817. https://doi.org/10.1109/tap.1967.1139054

Tewari, R. K., Swarup, S., & Roy, M. N. (1990). Radio wave propagation through rain forests of India. *IEEE Transactions on Antennas and Propagation*, *38*(4), 433–449. https://doi.org/10.1109/8.52261

The National Archives. (2022). *Wireless Telegraphy Act 2006*. Legislation.gov.uk. https://www.legislation.gov.uk/ukpga/2006/36/contents

The National Archives. (2025). *Telecommunications Act 1984*. Legislation.gov.uk. https://www.legislation.gov.uk/ukpga/1984/12/part/II/crossheading/standards-of-performance

The Pi Hut. (2015). *The Pi Hut*. The Pi Hut. https://thepihut.com/blogs/raspberry-pi-tutorials/18025084-sensors-pressure-temperature-and-altitude-with-the-bmp180

Trombly, E. (2021, August 27). *Python RFM69 library for raspberrypi*. RFM69. https://github.com/etrombly/RFM69

Tuononen, M. (2023, October 20). *Why continuous humidity profiling matters*. Vaisala. https://www.vaisala.com/en/blog/2025-01/why-continuous-humidity-profiling-matters

U, D., NS, C., GS, B., A Rao, A., & Bhagyashree. (2022). IOT BASED WEATHER MONITORING SYSTEM FOR SMART AGRICULTURE. *International Journal of Engineering Applied Sciences and Technology*, *7*(1), 298–301. https://doi.org/10.33564/ijeast.2022.v07i01.045

Wald, B. (2019, March 25). *Waterfall vs Agile: Which is Right for IoT Development?* Very. https://www.verytechnology.com/insights/agile-vs-waterfall-which-is-right-for-iot-development

World Economic Forum. (2024, January 10). *Global Risks Report 2024*. World Economic Forum; World Economic Forum. https://www.weforum.org/publications/global-risks-report-2024/digest/

Yoder, M. (2014, June 17). *Four Weather Factors for Plant Growth – North Carolina State Climate Office*. North Carolina State Climate Office. https://climate.ncsu.edu/blog/2014/06/four-weather-factors-for-plant-growth/

Zaborowska, Ł. (2025, January 17). *Dipole Calculator | Antenna Length Calculator*. Www.omnicalculator.com. https://www.omnicalculator.com/physics/dipole

Zhao, J., Xie, X., Jiang, Y., Li, J., Fu, Q., Qiu, Y., Fu, X., Yao, Z., Dai, Z., Qiu, Y., & Chen, H. (2024). Effects of simulated warming on soil microbial community diversity and composition across diverse ecosystems. *Science of the Total Environment*, *911*, 168793–168793. https://doi.org/10.1016/j.scitotenv.2023.168793

Zidar, S. (2024, August 20). *Air Temperature: From Ancient Observations to Modern Measurement Techniques*. Severe Weather Europe. https://www.severe-

weather.eu/learnweather/weather-instruments-theory/measuring-air-temperature-sz/

Appendices

Appendix 1 – Glossary of Key Terms

AES (Advanced Encryption Standard) – a symmetric encryption algorithm widely used for securing data due to its strong security and algorithmic efficiency, commonly used in IoT devices.

Antenna – a device used for transmitting or receiving electromagnetic waves.

Anemometer – a device used for calculating wind speeds.

Attenuation – the reduction in the strength of a transmitted signal as it travels through the atmosphere, through obstacles such as vegetation and air itself.

Bandwidth – the range of frequencies available for the transmission of data. The higher range of frequencies in a transmission means the more simultaneous data can be sent. Higher frequencies allow more bandwidth but less range, with lower frequencies allowing for further range with less bandwidth.

Dipole Antenna – an omnidirectional antenna with two conductive elements.

Effective Radiated Power (e.r.p) – the total power radiated by a radio transmitter,

Encryption – the process of encoding data to prevent access without a key.

Frequency Hopping Spread Spectrum (FHSS) – a method of reducing interference and eavesdropping via rapidly alternating the signal across a range of frequencies.

Frequency – the number of oscillations a wave produces per second, measured in Hertz. High frequency waves have shorter wavelengths, resulting in lower ranges, while the opposite is true of low frequency waves which allows them to penetrate obstacles better.

GPIO (General Purpose Input/Output) - the pins available on Raspberry Pi devices that can be used for interfacing with other hardware.

Hashing – the process of converting data into a fixed-size string (known as a hash value) which is unique to the input data, useful for comparing against for data-integrity checks.

Internet of Things (IoT) – interconnected networks of devices comprising of sensors, data processing and software for the collection, processing and exchange of data.

Omnidirectional Antenna – an antenna that equally radiates or receives signals across a 360° radius.

Propagation – the movement of waves through a medium.

RF (Radio Frequency) – electromagnetic waves produced by an antenna for wireless communication.

Signal-to-Noise Ratio (SNR) – a measurement of signal strength relative to background noise. A higher SNR typically means a stronger, clearer signal.

Star topology – a network topology where each node communicates directly to a central node.

Appendix 2 - Extra images

Appendix 2A – Weather station deployed in-field without radiation shield

Appendix 2B – Receiver using a spring antenna

Appendix 2C – Weather station using a spring antenna (encircled green)

Appendix 2D – Receiver using a wire antenna

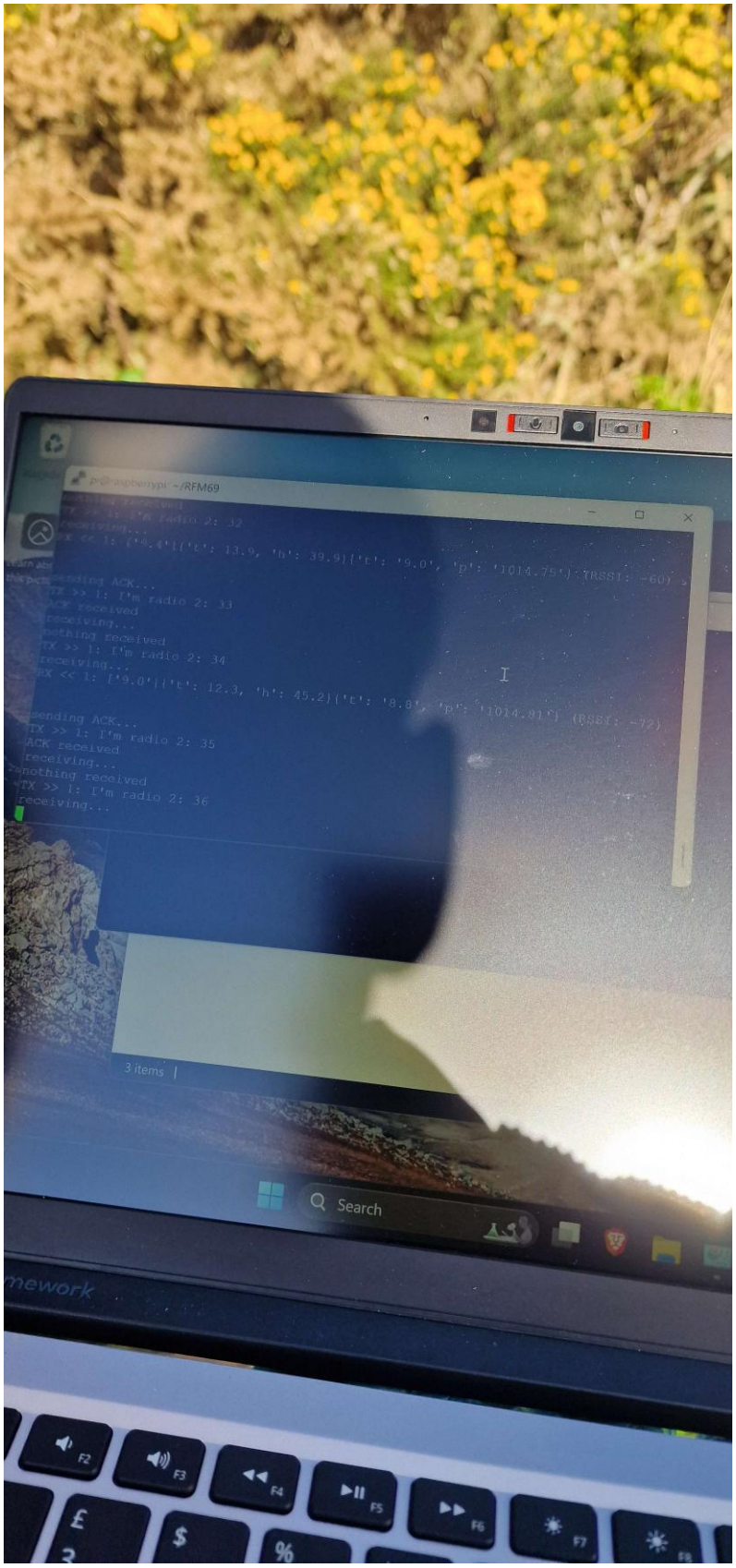
Appendix 2E – Weather station using a wire antenna (encircled green)

Appendix 2F – Receiver using a dipole antenna

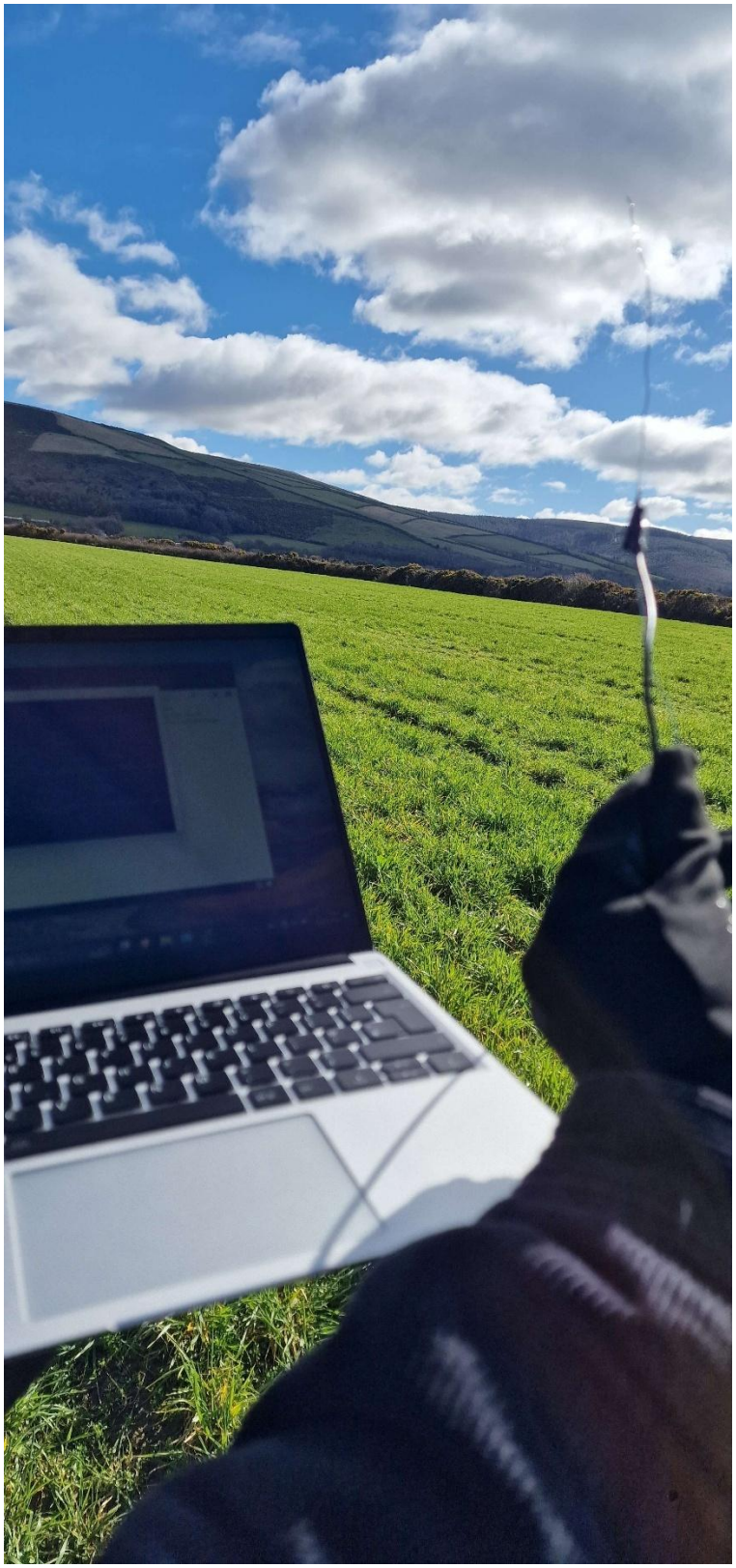
Appendix 2G – Weather station using a dipole antenna

Appendix 2H – Dipole antenna in line of sight (LOS) with the weather station (encircled red)

Appendix 2I – Live demonstration of the weather station output



Appendix 2J – Live demonstration of the weather station output with antenna



Appendix 4 – Test results


Test 3
0
-20
-40
-60
-80
-100
-120
RSSI (dBm)
0m
25m
50m
75m
100m
125m
150m
175m
200m
225m
250m
-44
-38
-46
-44
-54
-67
-53
-61
-100
-61
-75
-100
-70
-83
-100
-76
-100
-84
-100
-77
-80
-80
-82
Distance (metres)
Spring
Wire
Dipole

Appendix 5 – Adherence to compliance

Appendix 5A – ZetaRF

UK CA

UK Declaration of Conformity

Product Details

Product:	ZETA 433MHz
Model:	ZETA-433-SO, ZETA-433-D
Product Description:	Miniature RF transceiver module
Specification:	Operating frequency 433MHz

Manufacturer

Name:	RF Solutions Limited
Address:	William Alexander House, William Way, Burgess Hill, West Sussex, RH15 9AG, United Kingdom
Web site:	www.rfsolutions.co.uk
Email:	support@rfsolutions.co.uk

UK Authorised Representative

Name:	RF Solutions Limited
Address:	William Alexander House, William Way, Burgess Hill, West Sussex, RH15 9AG, United Kingdom
Web site:	www.rfsolutions.co.uk
Email:	support@rfsolutions.co.uk

This declaration is issued under the sole responsibility of the manufacturer.

The object of this declaration is in conformity with the relevant UK Statutory Instruments (and their amendments).

Statutory Instrument	Title
2017 No. 1206	The Radio Equipment Regulations 2017
2012 No. 3032	The Restriction of the Use of Certain Hazardous Substances in Electrical and Electronic Equipment Regulations 2012 (RoHS)

Designated Standard		Title
Safety (2016 No. 1101)	EN 62368-1:2020	Audio/video, information and communication technology equipment - Part 1: Safety requirements (IEC 62368-1:2014, modified)
	EN 62311:2020	Assessment of electronic and electrical equipment related to human exposure restrictions for electromagnetic fields (0 Hz - 300 GHz)
EMC (2016 No. 1091)	ETSI EN 301 489-1 V2.2.3 (2019-11)	ElectroMagnetic Compatibility (EMC) standard for radio equipment and services; Part 1: Common technical requirements; Harmonised Standard for ElectroMagnetic Compatibility
	In accordance with the specific requirements of:	
	ETSI EN 301 489-3 V2.3.2 (2023-01)	ElectroMagnetic Compatibility (EMC) standard for radio equipment and services; Part 3: Specific conditions for Short Range Devices (SRD) operating on frequencies between 9 kHz and 246 GHz; Harmonised Standard for ElectroMagnetic Compatibility
	EN 55032:2015/AC:2016 (Class B)	Electromagnetic compatibility of multimedia equipment - Emission Requirements
Radio (2017 No. 1206)	ETSI EN 300 220-2 V3.2.1 (2018-06)	Short Range Devices (SRD) operating in the frequency range 25 MHz to 1 000 MHz; Part 2: Harmonised Standard for access to radio spectrum for non specific radio equipment
RoHS (2012 No. 3032)	EN 63000:2018	Technical documentation for the assessment of electrical and electronic products with respect to the restriction of hazardous substances

Signed for and on behalf of:	RF Solutions Limited	
Place of issue:	West Sussex, England	
Date of issue:	30th March 2023	Phil Stevens, Compliance Manager

TCFs809 Issue 1.1 Page 1 of 1

RFM69HCW ISM TRANSCEIVER MODULE V1.1

GENERAL DESCRIPTION

The RFM69HCW is a transceiver module capable of operation over a wide frequency range, including the 315,433,868 and 915MHz license-free ISM (Industry Scientific and Medical) frequency bands. All major RF communication parameters are programmable and most of them can be dynamically set. The RFM69HCW offers the unique advantage of programmable narrow-band and wide- band communication modes. The RFM69HCW is optimized for low power consumption while offering high RF output power and channelized operation. Compliance ETSI and FCC regulations.

In order to better use RFM69HCW modules, this specification also involves a large number of the parameters and functions of its core chip RF69H's,including those IC pins which are not leaded out. All of these can help customers gain a better understanding of the performance of RFM69HCW modules, and enhance the application skills.

RFM69HCW

KEY PRODUCT FEATURES

- +20 dBm - 100 mW Power Output Capability
- High Sensitivity: down to -120 dBm at 1.2 kbps
- High Selectivity: 16-tap FIR Channel Filter
- Bullet-proof front end: IIP3 = -18 dBm, IIP2 = +35 dBm,80 dB Blocking Immunity, no Image Frequency response
- Low current: Rx = 16 mA, 100nA register retention
- Programmable Pout: -18 to +20 dBm in 1dB steps
- Constant RF performance over voltage range of module
- FSK Bit rates up to 300 kb/s
- Fully integrated synthesizer with a resolution of 61 Hz
- FSK, GFSK, MSK, GMSK and OOK modulations
- Built-in Bit Synchronizer performing Clock Recovery
- Incoming Sync Word Recognition
- 115 dB+ Dynamic Range RSSI
- Automatic RF Sense with ultra-fast AFC
- Packet engine with CRC-16, AES-128, 66-byte FIFO Built-in temperature sensor
- Module Size:16X16mm

APPLICATIONS

- Automated Meter Reading
- Wireless Sensor Networks
- Home and Building Automation
- Wireless Alarm and Security Systems
- Industrial Monitoring and Control
- Wireless M-BUS

Tel: +86-755-82973805 Fax: +86-755-82973550 E-mail: sales@hoperf.com http://www.hoperf.com