Utilizing LoRaWAN and Embedded Technologies for Low-Cost IoT-Based Meteorological Systems

Norbert Dajnowski [a], Aminu Bello Usman [b], Andy Guest [a] Abdulrazaq Abba [c]
[a] Department of Computer Science, York St John University norbert.dajnowski@yorksj.ac.uk, a.guest@yorksj.ac.uk,
[b] School of Computer Science, University of Sunderland, aminu.usman@sunderland .ac.uk
[c] School of Architecture, Computing and Engineering, University of East London, a.abba@uel.ac.uk

ABSTRACT
The advancement of meteorological stations is continuously supported by the advanced technologies and solutions toward achieving the most accurate and reliable data while maintaining cost efficiency. Governments and businesses are leveraging the potentials of Internet of Things (IoT) platforms to provide hyperlocal and highly sophisticated compact meteorological stations for analysing real-time weather conditions and forecasts with unprecedented accuracy. In this research, we developed a stable, low-cost meteorological system capable of recording weather parameters, powered with a low-cost and long-range data transfer technology to provide multiple nodes access to the internet through the LoRaWAN network server. Additionally, we developed a GUI application for visualising meteorological systems' data on a mobile app to the users in the form of numbers, charts, and graphs.

KEYWORDS
LoRaWAN, LPWAN, Meteorological Station, Embedded Technologies

1. Introduction

The world of environmental monitoring applications such as weather monitoring, smart city, and smart agriculture is continuously evolving with advance technologies and solutions to achieve the most accurate data readings, using cost-effective methods. A meteorological station is a facility hosted either on land or sea, containing instruments and equipment used for measuring the atmospheric and environmental parameters. Advanced climatological analyses help in developing a deeper understanding of hydro-meteorological systems and reveal trends over time that justified environmental change (Zare-Shehneh, 2023).

Fundamentally, the composition of a meteorological station comprises of machines, sensors, data storage, wireless communications, batteries, solar collectors, cellular or satellite communication technology. This hardware facilitates recording and transmission of meteorological data including air, wind speed, temperature, rainfall, atmospheric pressure, wind speed/direction, humidity; cloud height, light levels and visibility. The design of such system vary from expensive satellite imagery to local low-cost IoT-based systems. Regardless of the system design, the main expectation from a meteorological station is collecting data on different atmospheric conditions. In most common low-cost Internet of Things meteorological stations, the Arduino MCU or Raspberry Pi performs as the backbone. These IoT devices are fundamentally different in some way - the Arduino MCU only performs the specific programmed tasks, while the Raspberry Pi has the

multifunctional capabilities of a minicomputer. Additional advantage of using an Arduino MCU or a Raspberry Pi is the sensors availability and compatibility. The extensive work accomplished with the implementation of these devices indicates that they have had multiple updates and refactors, proving them to be low-cost and reliable for a system collecting meteorological data (León, 2017) . The meteorological system is sought after by specific industries, such as agriculture and astronomy; to provide them with a guide on yield or work quality improvements, accurate and better weather forecasting. Other application areas for a meteorological system include tourism and transport, atmospheric disturbance detection, weather phenomenon forecasting by providing a forecast for evacuation planning or adaptation (Chen, 2023).

To obtain the most accurate and real-time meteorological data, we deploy local IoT stations, which is relatively new concept, where sensors and hardware is further interconnected with the digital world (Leon, 2016). By assembling micro-controllers, sensors, and networking technologies, IoT systems can be developed for industrial use and smart cities. Networking technology is essential in IoT systems and should be selected accordingly to the communication requirements. When building meteorological systems, there is a wide range of connectivity options: WiFi, Bluetooth, Zigbee, SigFox, NB-IoT, and LoRaWAN. LPWAN networking outperforms the higher frequency WiFi or Bluetooth, especially in the context of a low-cost IoT based meteorological station, where power consumption must be maintained at low levels. A recent attractive option is LoRa, an acronym for long-range, it operates power-efficiently on a radio frequency range, specifically designed for low-power systems which require long-range connectivity while maintaining a low-cost status (Devalal, 2018). (Murdyantoro, 2019) LoRa operates on the open ISM spectrum, capable of establishing connection up to 10km; its low power consumption allows for the battery to last for up to 10 years, while sustaining noise rates of under 20dB during demodulation (Murdyantoro, 2019). These specifications classify LoRa as an attractive IoT networking solution. Although, LoRa is not free from any disadvantages - long-range applications utilising LoRa can experience packet loss (Wang, 2017).

Our contributions in this paper are as follows:

- We developed a stable low-cost meteorological system capable of recording weather parameters. The developed system comprises of MCUs, communication units and sensors, which record the temperature, humidity, pressure, wind data and air quality.
- We powered the developed meteorological system with a low cost and long-range data transfer technology to provide multiple nodes access to the internet through the LoRaWAN network server - The Things Network (TTN).
- We utilized ThingSpeak IoT analytics to collect, organise and analyse the sensors' data, by applying custom algorithms.
- We developed a GUI web application for visualising the collected data from the developed meteorological systems (in a simple format of numbers, charts and graphs).

2. Related Work

The meteorological system uses weather sensors, power supplies, data collectors, and cloud platforms to measure, transmit, record, and analyse weather data such as wind, rain, snow, sun radiation, rain, and so on. The recorded weather is analysed to produce accurate results and forecasts, which will benefit and empower multiple industries by providing current,

past, and future meteorological conditions, allowing them to prepare and adapt accordingly. One of the industries most affected by micro-scale meteorology systems is agriculture, where balance must be maintained between variables like temperature or soil moisture to produce the optimum crop yield. Similarly, astronomers rely on sky transparency forecasts to select suitable times for observation. An application of this sort requires a meteorological system capable of monitoring at a larger spatial scale compared to agricultural systems (Muller, 2013), (Zare-Shehneh, 2023). One of the most affected industries by microscale meteorology systems is agriculture, where balance must be up-kept between variables like temperature or soil moisture, to produce the optimum crop yield. Similarly, astronomers rely on sky transparency forecasts to select suitable times for observation. An application of this sort requires a meteorological system capable of monitoring at a larger spatial scale (Muller, 2013) .

2.1. *Meteorological Stations*

The weather nature can be observed using sensors like the BME280 or DHT-11/22 (Adi, 2020), which collect data on temperature, pressure, humidity, wind speed, and more; achieving accurate and tailored local results. Some more advanced systems collect data on wind speed and rainfall rate, using anemometers and rain gauges. The low energy Arduino MCU based on an Atmel 8-bit AVR microcontroller is capable of operating as the base of a meteorological station (Louis, 2016) which commands the multiple connected sensors. Sensors are efficient devices used for data collection by monitoring physical conditions, often installed on modules or a HATs board (Krishnamurthi, 2015). As an alternative, a cost-effective device like the SparkFun weather shield can create competition in the sensory market (Mathur, 2021)); the developer might also favour an open-source atmospheric model, which uses satellite imagery to provide comparable results at a lower cost.

Confusion of scales can be experienced during the early project stages, where the implementation and network methods are mismatched with the application objectives. This poses a risk of over-paying for redundant monitoring coverage, or the enhancement of an under-performing system. Agriculture's requirements for atmospheric observation are considerably more local compared to other industries, to the point where network-less solutions requiring manual data extraction can be sufficient (Muller, 2013). An Internet of Things (IoT) solution for agro-meteorological observation is proposed, it's formed of a Wireless Sensor Network (WSN) consisting of base stations and child nodes. The node structure includes a Raspberry Pi with a set of connected meteorological sensors, using WiFi for communication with the base stations (Sawant, 2017).

Another study on IoT environment monitoring in the Antarctic takes a different approach, concentrating on the wireless communication range and power efficiency. This is achieved by deploying a Raspberry Pi module with the implementation of LoRa. Functioning on the 434/868 MHz channels, the Sub-GHz networking technology can take advantage of Antarctica's flatness and maximise its range with the open field of view. (Gaelens, 2017) LoRa's properties like dynamic payload lengths, convenient data/packet rates, and its low-cost availability eliminate other options of Sub-GHz technology in this study, like Sigfox or LTE-M. A remote outdoor IoT system requires an increase of sensors and range while coexisting with an implementation design that would maintain low power consumption and hardware costs (Chandu, 2021). The experiment produced a valuable low-cost networking

solution for environmental condition observation but noted that it is not suitable for long-term activity due to the limitations of Raspberry Pi, possible malfunctioning in Antarctic's extreme cold conditions of -40 degrees centigrade.

2.1.1. Wind & Rain Monitoring

Rainfall gauges are cost-effective tools for rain data observation and recording, or just functioning to provide secondary results. Certain users who may not have access to reliable satellite imagery, or just prefer to acquire more accurate local data, utilise rainfall gauges in their meteorological systems. Research indicates that rain gauges are the most common devices used to measure the rainfall rate in specific locations (Pardo-Igúzquiza, 1998). A tipping bucket with a sensor is another local and digital option for rain data collection. The bucket fills up with rainwater and eventually tips, while the sensor captures the bucket tip timings (Warnakulasooriya, 2018). Using the acquired timings, and processing it through an equation, gives us an output of the average rainfall rate. Unfortunately, the random nature of rainfall reveals the algorithm's flaws, where it misses the low and high phases of rainfall (D'Amico, 2013), due to heavy or light rainfall. Occasionally, other systematic errors occur with rainfall meters, including incorrect gauge calibration, wind-induced under-catch, or wetting–evaporation losses (Ciach, 2003). New algorithms are developed and proposed regularly to improve the readings and eliminate flaws in the classic algorithm.

2.2. *Internet of Things & LPWAN*

The focus on cost mitigation in IoT meteorological stations must be maintained without major loss in networking performance. This has influenced the adaptation of Low Power Wide Area Network (LPWAN) solutions, which offer communication using Sub-GHz; a communication technology that operates on the unlicensed ISM bands under 1 Ghz (eg. 315, 433, 902, and 928Mhz); these are frequently used in related research in the fields of science, medicine, and manufacturing. Naturally these channels provide increased network area coverage due to their lower frequencies, compared to the wireless standard of 2.4Ghz (Frenzel 2013). The demand for developing IoT systems featuring low cost, low power and long-range connectivity influences the implementation of lower ISM band devices such as Lora, SigFox, or BLE (Sanchez-Iborra, 2016).

2.2.1. LoRaWAN

LoRa alliance has provided a solution to meet the industrial demand for scalable IoT connectivity (Alliance, 2019). It is an organisation focused on the development of LoRaWAN, made up of multiple institutions like Semtech, IBM, Actility and others. Operating on Semtech's chirp spread spectrum (CSS) (Sanchez-Iborra, 2016).. LoRaWAN converts the ordinary LoRa mesh network into a star network with the implementation of a central gateway. This network architecture allows for the connection of an extended number of nodes, while providing sophisticated authorisation.

LoRaWAN functions on the unlicensed Sub-GHz ISM bands, which are different depending on the geographical location, for example in Europe or USA it operates respectively on the frequencies of 868MHz and 915MHz. The low Sub-GHz frequency provides a receptiveness of up to 10km, conditional on the quality of the antenna; meanwhile it maintains configurable data rates of up to 300kbps (Di Serio, 2017). Additionally,

LoRaWAN's demodulation and multiple channels offer the option to scale the system in the future (Murdyantoro, 2019). Supported with the extra security and organisation of The Things Network and Thingspeak that provide a complete IoT network solution (Maureira, 2011).

2.2.2. LoRaWAN Security

LoRaWAN packets are AES128 encrypted, include a frame counter, and use two unique keys to establish a secure connection with The Things Network (TTN). *AppSKey* and *NwksKey* ensure that only authorised devices can communicate with the server. The *AppSKey* is used to encrypt the whole payload, including the frame counter. Whilst the *DevAddr* is signed by the *NwkSkey*, providing verification of integrity when validated by the network's gateway (Blenn, 2017).

3. Systems Components and Design

Certain requirements must be maintained when constructing an outdoor meteorological station, to ensure that it can capture the targeted parameters. First, the station must be suitable for all weather conditions, this can be achieved with an outdoor electronics enclosure consisting of a see-through plastic front, enabling observation of the light intensity. Second, for proper insulation, the antenna and power supply must be wired from outside of the box, with cable glands strategically placed over the drilled holes in the frame.

3.1. *Meteorological Sensors*

3.1.1. Temperature, Humidity, and Pressure

The DHT-11/22 is a popular temperature sensor for low-cost meteorological systems. It observes and records the ambient temperature change in a decimal value, which is then further converted into Degree Celsius or Fahrenheit by the MCU. We used the DHT-11/22 sensor to collect Temperature, Humidity, and Pressure data. The sensor's ability to capture real-time temperature and relative humidity in the surroundings (Pasha, 2016), provides the necessary ratios and variables to calculate the dew point using equation 1, here *Td* is dew point temperature (in degrees Celsius), *T* is the observed temperature (in degrees Celsius), and *RH* is relative humidity (in percent). This indicates the amount of water vapor needed to achieve an increase in saturation and temperature.

$$Td = T - ((100 - RH)/5 \tag{1}$$

To approximate the heat index in degrees Celsius, within ± 1.3 °F (0.7 °C), we applied equation (2) (Anderson, 2013)

$$HI = HI_1 + HI_2 \tag{2}$$

Where *HI* = heat index (in degrees Celsius)

$$HI_1 = c_1 + c_2T + c_3R + c_4TR + c_5T^2 + c_6R^2 \tag{3}$$

$$HI_2 = c_7T_2R_2 + c_8TR_2 + c_9T_2R_2 \quad (4)$$

T = ambient dry-bulb temperature (in degrees Celsius)
R = relative humidity (percentage value between 0 and 100).

Figure 1. Meteorological Sensors

The following coefficients determine the heat index, which tells us the level of heat sensitivity on human skin in the surroundings.

Table 1. Heat Index coefficients

Coefficients	Values
c_1	-8.78469475556
c_2	1.61139411
c_2	1.61139411
c_3	2.33854883889
c_4	-0.14611605
c_5	-0.012308094
c_6	-0.0164248277778
c_7	0.002211732
c_8	-0.00072546
c_9	-0.000003582

An alternative to the DHT temperature sensor is the BME280 sensor, which is similar to the DHT-11/22, performs multi-sensory functions; it can collect data on humidity, temperature, in addition to recording atmospheric pressure (Adi, 2020). The BME280 sensor provides additional data for more advanced weather predictions and calculations, while preserving cost efficiency and not requiring any extra sensory hardware.

3.1.2. Wind Speed and Rainfall Rate Sensors

Wind speed is a fundamental atmospheric parameter caused by air moving from high to low pressure, usually as a result of changes in temperature. The wind affects weather forecasting, maritime operations, construction projects, metabolism rate of many plant species, and countless other implications. We used Anemometers to record the wind speed, which was constructed in a fan-like shape to capture wind flow velocity. Usually, cups or lightweight objects are attached to the fan to increase the surface area. The increase in surface area on the blades provides more accurate readings of wind speed and rainfall rate (Warnakulasooriya, 2018).

$$Wind\_V = \frac{(voltage - 0.4)}{1.6 \times 32.4} \tag{5}$$

A rotor is placed onto a heavy steel base to ensure that the device remains static, wings containing cups are installed onto the rotor. The cups are installed to ensure that all of the air from the wind is captured. Inside the device, a black and white pattern is embedded, with an optical sensor connected over it. While the device spins due to wind gusts, the optical sensor will decide whether the anemometer is at a black or white position. Voltage returned to the Arduino MCU is dependent on the reflected shade. Recording the changes between white and black can tell us the speed of the fan; the more frequently the voltage switches, the faster the fan is spinning (Shaout, 2014).

3.1.3. MQ-135 Sensor

For collecting air quality data, an MQ-135 sensor is installed using an analog pin on the Arduino MCU. It belongs to the MQ series sensor group, which consists of the most inexpensive and common gas sensors available. MQ-135 is designed for air quality measurements, and is capable of collecting data on CO2, CO, NH3, NOx, Alcohol, Benzene, and Smoke levels in the air (Components 101 2021).

In the function *setup()*, the MQ-135 sensor is initialised and calibrated using the *calibrate()* function. The readings are updated at every cycle with the *update()* function, and added to the *mydata* packet for TTN. During calibration, the sensor resistance in fresh air is calculated and stored as *Ro*. All of the following sensor resistance *Rs* readings are calculated using the equation 6.

$$Rs = \frac{Vc}{VRL - 1} \times RL \tag{6}$$

The ratio of the acquired values *Rs/Ro* is compared to the figure 2, which gives a PPM result for each gas at that particular resistance ratio.

3.2. *Systems' Electronics Components*

3.2.1. SparkFun Weather Tools

The majority of the sensors are contained on a single weather shield made by Sparkfun (Mathur, 2021). This shield is a viable solution for the collection of standard weather readings. Equipped with its built-in Si7021 temperature and humidity sensor, MPL3115A2 barometric pressure sensors, and ALS-PT19 light sensor, it satisfies the standard requirements for our weather station's data collection (Mathur, 2021) . Additionally, via the RJ-11 connectors, a Sparkfun weather meter kit has been installed. The kit includes an anemometer, wind vane, and a rain gauge (Kaewwongsri, 2020). The anemometer sends a signal to the MCU after each revolution, which calculates the wind velocity. Whereas the wind vane reports its angle to the MCU, which is indicated by the device's resistance. The final meteorological component is an electronic rain gauge, which gathers water until 0.011 inches full, it then tips and transmits a signal to the MCU.

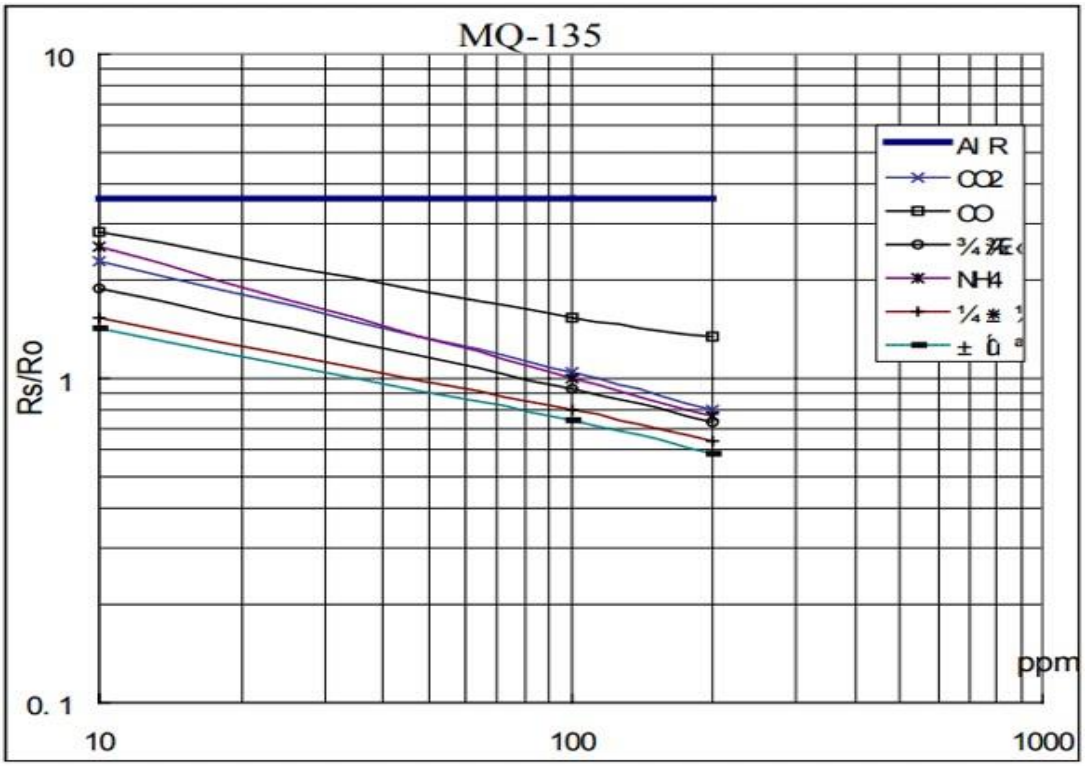

Figure 2. MQ-135 Typical Sensitivity Graph (Components 101 2021)

Figure 3. Installed SparkFun weather shield

3.2.2. Dragino LoRaWAN Gateway

We set up the Dragino LoRaWAN LPS8 (Dumitru, 2023) , an 8-channel gateway that enables communication between LoRa and the internet, specifically to the TTN application (The Things Network). To prevent exponential signal loss, we installed the LoRa devices so that a line of sight is provided without buildings obstruction. This is necessary to keep LoRa's long distance communication stable.

Figure 4. Dragino LoRaWAN LPS8 Gateway & Dragino LoRa Shield v1.4

3.2.3. Dragino LoRa Shield

To enable LoRa communication, we install a Dragino LoRa shield (Dinev, 2023) onto the Arduino MCU, as depicted on the right side of Figure 4. This LoRa shield is responsible for the transmission of sensor data to the gateway. Before any data transmission, The Thing's Network (TTN) establish authorisation between the gateway and the end-node, by utilising a series of keys. Each unique LoRa device receives a EUI and App Key. These two values, together with the application EUI are used for authorisation between the meteorological station and TTN.

3.3. *System Design*

To give an overview of the system, a Sparkfun weather shield is slotted onto a Arduino MCU and paired with an air quality sensor. Each meteorological station contains all the sensors required for the collection of standard weather data.

Regarding networking, a LoRaWAN gateway is deployed, and the meteorological nodes are equipped with LoRa HATs. This enables communication between end nodes and the gateway, while taking advantage of a star shaped network topology. The collected data is forwarded to the TTN (The Things Network) which integrates Thingspeak. Finally, the API is utilised in a flask web application to obtain real time data updates from the cloud (Aslam, Mohammed, & Lokhande, 2015).

A system design is illustrated in Figure 5, using LoRa the end devices transmit payloads to the authorised gateway. This LoRaWAN gateway receives the packet from the meteorological station and forwards it to the LoRaWAN network server (LNS).

3.3.1. The Things Network

The Things Network (TTN) is a crowd funded LoRaWAN network service, enabling users to connect their low energy devices for free. This IoT network consists of a mass collection of gateways, operating as central entities in their private star networks. TTN provides a secure and sustainable long-term connection and preserves the data feed's status (Barro, 2019). UK's flood network system, containing multiple sensors, utilises this technology to connect and communicate with the cloud (Blenn, 2017). Connecting the Dragino LPS8 gateway with TTN, provides a real-time display of the uplink and downlink payloads. Each meteorological station is separated into its own application which connects our single LPS8 gateway. The station's data stream is sorted into its own application on the cloud side of the system, integrating Thingspeak to forward our processed data for further analysis and visualisation.

3.3.2. Thingspeak

Thingspeak is an open-source application for storing and retrieving data over HTTP and MQTT, including an implementation of MATLAB for analytics and visualisations. We adopt Thingspeak's API to remotely retrieve our stored sensor data in real-time (Maureira, 2011). The communication method to the API is based on HTTP requests and authorisation keys (read/write). Data is stored inside *fields*, and up to 8 of these are available inside a channel. Read and write requests directly communicate with Thingspeak channels, which are identifiable by their unique channel ID. Additional metadata like the description,

latitude, longitude, and elevation is stored inside channels (Maureira, 2011). Once a channel is setup, MATLAB processes and visualises the data by default, but allows for further customisation (Pasha, 2016). We applied ThingSpeak's MATLAB functions for data visualisation and computation. MATLAB is by default integrated with Thingspeak, providing a visualisation using standard algorithms applied on field's data. This is a preferred back-end environment by engineers, due to its simple and interactive system which includes numeric computations, scientific visualisations and symbolic calculations (Valentine, 2022). providing a comprehensive look into our gathered readings.

3.4. *Software*

The Arduino IDE is used to write and upload C/C++ programs to the Arduino Uno MCU. Additionally, the application can upload programs to cross-platform devices; therefore, not limiting itself to the Arduino itself. Software responsible for calibrating and gathering sensor data, as well as establishing LoRa connectivity is written and compiled onto the end devices in Figure 5. Due to the program's scale, memory issues with the Arduino are possible. To ensure the program's memory requirements are satisfied, the Arduino Nano MCU or other smaller memory models of the Arduino should be avoided (unless they contain a minimum of 2KB SRAM).

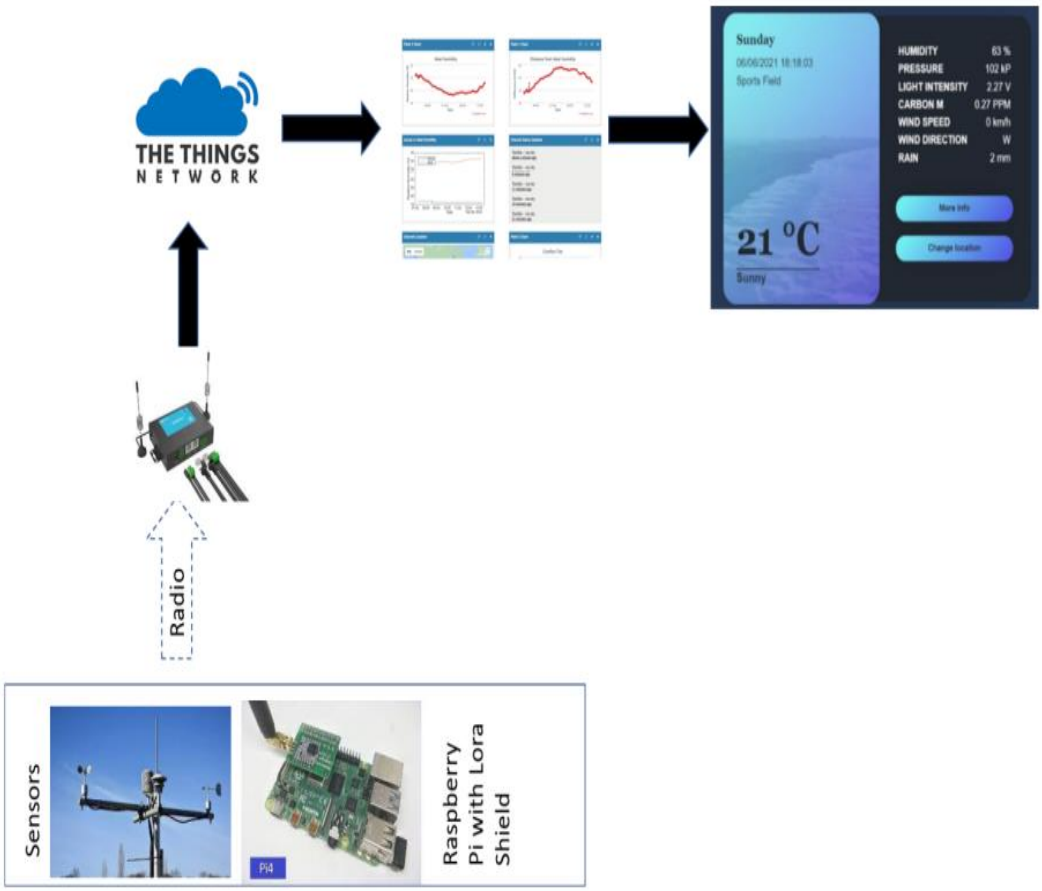

Figure 5. System Design

The subsequent content of this section contains references to dependencies involved in the [2] meteorological station programs, responsible for handling events like sensor management or calibration, which can be viewed in the top section of the meteorological station code.

3.4.1. Meteorological Sensors

The library *SoftwareSerial.h* enables serial communication through the MCU's additional digital pins. Since Arduino only supports serial communication via digital pin 0 and pin 1, enabling extra serial communication helps to support LoRa requirements. To improve the read/write with I2C devices, the *Wire.h* library is included. The next three libraries, *MQUnifiedsensor.h, SparkFunMPL3115A2.h,* and *SparkFunSi7021BreakoutLibrary.h* are responsible for interacting with the weather shield and air quality sensors (MQ-135). *SPI.h* allows for quick communication with Arduino's peripheral devices. Finally, the library lmic.*h*, developed by IBM, empowers the Dragino LoRa with the ability to connect between LoRaWAN networks, using either ABP or OTAA authorisation.

On every cycle of the program, the meteorological data is updated by calling the function *dataUpdate()* in program [2]. This function gathers new readings from each sensor and organises it into an array, called *mydata*, ready to be uploaded via LoRa to TTN. Two objects are declared at the beginning of the program, called *MPL3115A2* and *Weather*; they are responsible for the communication and collection of readings from the pressure, temperature, humidity, and light sensors.

An algorithm for recording wind and rain data takes averages readings from the Sparkfun anemmometer, wind vane, and rain gauge sensors. The natural volatility of this data creates anomalies and inaccurate results. As a resolution, readings are constantly recorded, and only the mean value over time is transmitted to TTN. The sensor interacts with the Arduino MCU by sending interrupts (*rainIRQ()* and *wspeedIRQ()*), which are triggered by the anemometer making a certain number of revolutions. Whereas the rain gauge interrupt is triggered by the magnet registering a tip in the bucket. Two separate loops are engaged in the program, one collects sensor data, while the other controls the LoRa module. Unfortunately, these loops can't be merged due to strict LoRa timings, which strains the MCU's memory efficiency.

3.4.2. LoRa Technology

Dragino LoRa shields are implemented into our system, enabling LoRa communication. With the networking hardware assembled and installed, the program provides instructions for handling the LoRa module. The *lmic.h* library is used to configure and transmit LoRa data. The configuration begins inside the *setup()* function, where *os-init()* is declared. Following that, the next lines of code attempt to create a session using unique TTN keys (DEVADDR, nwkskey, appskey). As previously mentioned, these three variables are critical for authorising the OTAA connection. They are declared in the code's header as static constants, which stores the keys in hexadecimal format. With the session now active, the next section begins the TTN gateway channel configuration. By default, the gateway will not take advantage of all 8 channels, which can affect the quality of our connection. Using the procedure *LMIC-setupChannel* the existing frequency range is expanded to use all of the available channels simultaneously. Following the configuration, the *do-send(&sendjob)* function is declared. This initiates a single repetition of the loop, collecting sensor data and

attempting to transmit it to TTN. Next the *onEvent(ev-t ev)* function is initiated, which schedules the *do-send()* function whenever it receives *EV-TXCOMPLETE* status. The code is also capable of handling multiple other status responses.

3.4.3. The Things Network

TTN hosts a collection of LoRaWAN gateways by the public, which enables smart devices, like meteorological stations, to create a low powered connection to the internet (Blenn, 2017) . For TTN to understand the received data from our stations, the program [2] is written. The *decoder()* function is responsible for receiving and converting the data into appropriate data types. It then further filters it into specific field variables and forwards them to our Thingspeak application.

3.4.4. Web Application

To present the data feed in a graphical user interface (GUI), we developed a web application, as shown in Figure 6. The back end of the application was written in Flask (Python) to support scaling and provide the necessary security between the application and Thingspeak (Aslam, Mohammed, & Lokhande, 2015). The client side has timed requests for the collection of data readings, coded in JQuery, allowing for dynamic weather data updates. Standard HTML, CSS and Javascript was used to develop the front end. We deployed the web application by hosting it on our York St John computer science server, which provides internet accessibility.

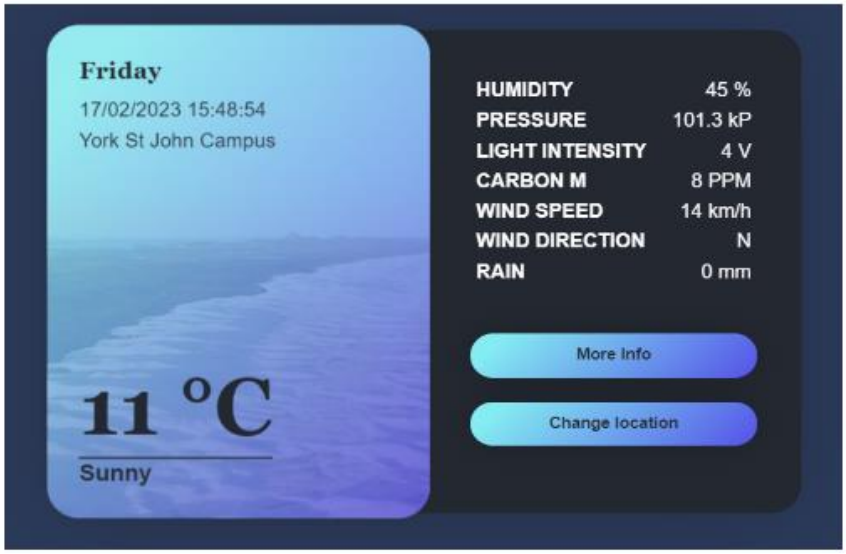

Figure 6. Web Application Front Page

- The application's back end is a layer responsible for the handling various website functionalities. Using function *start()* the upload of *index.html* is initialised, which produces the view of our application in Figure 6.
- The real-time upload of data from the Thingspeak API is handled by *"/data-update"*, which utilises the weather class to return our parameters and meta data back to the

client. Apart from the standard readings, *getWeatherCondition()* is used to compute the current general weather condition, which in Figure 6 is *Sunny*.

- On the client side, a jQuery *updateData.js* script is written to communicate with the */data-update* route. Additionally, the code is responsible for handling front-end data and executing simple logic/animations.

4. System Testing & Results

4.1. *Testing Overview*

Meteorological stations are deployed locally in two locations around York, Lord Mayors Walk University Campus, and the Sports Field. General tests are designed and performed, where the stations remain active for a couple of hours to collect data. Testing continues until enough data is gathered for sufficient results, through which we can evaluate the data's complete life cycle. Beginning at the sensors, then the Arduino MCU, LPS8 Gateway, TTN, Thingspeak, and finally ending its cycle at the web application with the user. The accuracy and reliability of the collected data is an important element of focus during testing, in addition to the LoRaWAN network performance quality, which can affect the original meteorological station results.

The first test involves the observation of active LEDs on the electronics and sensors, when the Arduino MCU is connected to the batteries. This will indicate if the meteorological station is receiving power. Thereafter, the MCU should begin calibrating the sensors and initialising a LoRa connection between the gateways. To inspect if a successful connection has been established, we open the TTN application and check if the status appears as online.

Gateway connection status can be confirmed by checking the TTN Gateway page, or alternatively by viewing the logs on the gateway's dashboard. The gateway is assumed fully functional in our system if data is being received simultaneously from both the meteorological stations. Every payload received by TTN should be decoded and forwarded to Thingspeak, and their timestamps can be used to compare and confirm that the data has been registered appropriately. Meanwhile, the payload values should be visualised using the default MATLAB generated graphs, with x and y axis values at a suitable accuracy.

Data updates should occur every 10 seconds and display from the correct Thingspeak channel. The testing of this can be accomplished by comparing the web application variables to the Thingspeak entries. The systems have been launched and left running for a couple of hours. A stable feed of readings has been generated by both the stations, and the LoRaWAN connection remained active during the testing phase. Generally, all the hardware and applications were successful in producing viable results.

4.1.1. Power & LoRa Connection Results

On power connection, Arduino MCU LEDs switch on, indicating that the device has launched. To further showcase the status, we provide a screenshot from TTN application, which displays our station's online status through the device web page. You can find this screenshot in. Additionally, TTN shows that the device is successfully sending payloads approximately every 30 seconds. Further analysis of the received payloads reveals the decoding script results has correctly sorted the data into a Thingspeak format.

The web application is launched in Google chrome, displaying the *index* page to the user. During the web application observation, we recognise the consistent time updates every 10 seconds, while concurrently including an update to all our atmospheric variables. This concludes that all the expected results are displayed and updated according to our requirements. The displayed atmospheric variables include the temperature, humidity, pressure, light intensity, carbon monoxide, wind speed, wind direction, and rain.

5. Discussion

The findings of this research reveal that LoRa networking is suitable for hosting IoT meteorological stations, particularly with an implementation of a LoRaWAN gateway, offering a scalable environmental observation system solution. Our test results have highlighted the pros and cons of constructing a meteorological station utilising low power MCUs and a LPWAN networking approach. Successful results on Arduino's power connection, and LoRa to TTN communication quality were collected. This indicate that the batteries can power the IoT weather station. However, in terms of sustainability, this may not be the best solution, since the batteries eventually lose charge causing the system to go offline. An improved method should be considered for power delivery to this system, for example the solar panel solution could be explored. It would include an implementation of a charging module, solar panels, and a lithium battery upgrade. We can also conclude that LoRa connectivity has been correctly established successfully, since the status appeared to be online post-system launch.

A further example of captured payloads from the weather station is displayed in our results. The challenge encountered here was the decimal/float numbers were not passing correctly to TTN. To adjust this, the values were converted to whole numbers by scaling them larger, and later returning them to their correct scale inside the web application. With a successful connection of the LoRa shield to TTN, the LPS8 gateway appears fully functional. The LoRaWAN gateway activity has contributed to the majority of testing results that we have achieved. Throughout the testing phase, screenshots have been captured and provided, highlighting the connected status, and displaying live data traffic from the meteorological stations.

Regarding statistics and data organisation, most of it was handled by the Thingspeak application. MATLAB algorithms were applied on incoming data entries, producing customisable graph visualisation. During testing, the entry timestamps in Thingspeak are compared to TTN payloads, confirming that every packet is transferred. It is worth highlighting that the timestamps are correct to the unit second, meaning that the delay at this stage is minimal. The chart lines gradually rise or drop to account for different weather changes, especially in the light intensity chart, where the sunset's effect is displayed by the reduction in the captured voltage level over time. Looking at the results, the Thingspeak charts often skip to a whole number, as they are automatically rounded to the nearest whole figure. This affects the graph accuracy, but the variables are later converted back into their original decimal values. Our web application delivers a simple and modern interface for live data visualisation. The index page provides a live display of every observed variable by the meteorological stations. The front end appears to give a good perspective of the collected data from the meteorological stations, with the only adjustment that transpired post-testing

being the publication of the Thingspeak channel links onto the web application, to provide a graphical history of our observed values to the user.

6. Conclusion

We examined the results throughout all the production stages, including the meteorological station assembly and development, IoT platform configuration and web application construction; with a general focus on LoRa's system performance. We developed a system capable of collecting and analysing all the mandatory weather data readings, as specified in the design stage. These include the temperature, humidity, pressure, and wind speed. Also, the project has been upgraded to include sensors capable of reading the light intensity, air quality, wind direction and rain rate. TTN has performed as expected by providing cloud connectivity for the LoRa data and connecting it with the Thingspeak platform. This enabled further handling, analysis, and visualisation of the collected sensor data using MATLAB. To establish a reliable and constant transmission of the meteorological data to TTN, a Dragino LoRaWAN gateway was deployed in our network. Employing its multi-channel mechanism and additional authorisation methods improve the performance and security of our system.

During the development phase, we encountered some basic performance issues with LoRa, where we had to adjust the meteorological stations' placement to ensure a line of sight is maintained between the nodes. LoRa's Sub-GHz frequency makes the communication noticeably sensitive to any obstruction on the line of sight. Additionally, the chosen locations posed inaccessibility issues for a wired power design, therefore we determined to power the system using batteries. Other important factors that we noted during the station construction was its waterproofing, air flow, condensation, transparency and more. Ensuring that all the parameters like light intensity or air quality, could be captured within our outdoor electronic enclosure.

This work has shown that LoRa technology is a capable networking solution for deployment in scalable projects involving multiple end nodes. Furthermore, we have displayed how LoRa can provide reliable and efficient live data transmission in an IoT system. While WiFi, Bluetooth and other technologies were all viable options, LoRa incorporates most sought-after networking features, such as long-range connectivity, low power, and reduce dependence on external infrastructure. This makes it a notably desirable IoT networking technology for both small and industrial scale applications.

Bibliography

Adi, P. &. (2020). A performance of radio frequency and signal strength of LoRa with BME280 sensor. *Telkomnika*, 18, 649-660.

Alliance, L. (2019). *Wi-Fi & LoRaWAN® Deployment Synergies.*

Anderson, G. B. (2013). Methods to calculate the heat index as an exposure metric in environmental health research. *Environmental health perspectives*, 121, no. 10 (2013): 1111-1119.

Aslam, F. A., Mohammed, H. N., & Lokhande, P. S. (2015). Efficient way of web development using python and flask. *International Journal of Advanced*

Research in Computer Science, , Vol. 6 Issue 2, p54-57. 4p. Retrieved from https://web.p.ebscohost.com/abstract?direct=true&profile=ehost&scope=site&authtype=crawler&jrnl=09765697&AN=102304622&h=joVEdFpDXSoL4r8j%2fv7qSrlMjd6vqE5ul2p2jxXNDhHoQLLHqKQdGzbHCVQF4orW0zigfu%2fts9YqHzPa%2bKRfXg%3d%3d&crl=c&resultNs=AdminWebAuth&resultLo

Barro, P. A. (2019). TLTN–The local things network: on the design of a LoRaWAN gateway with autonomous servers for disconnected communities. *In 2019 Wireless Days (WD),. IEEE*, pp. 1-4.

Blenn, N. a. (2017). LoRaWAN in the wild: Measurements from the things network. *arXiv preprint arXiv:1706.03086*.

Chandu, K. R. (2021). Performance analysis of Sub-GHz system for IoT applications. *International Journal of Electrical and Electronic Engineering & Telecommunications*, 10, no. 2 (2021): 125-132.

Chen, Y. K. (2023). Impact and Challenges of Intelligent IoT in Meteorological Science. *IEEE Internet of Things Magazine*, 6, no. 2 (2023): 58-63.

Ciach, G. J. (2003). Local random errors in tipping-bucket rain gauge measurements. *Journal of Atmospheric and Oceanic Technology 20*, 752-759.

D'Amico, M. S. (2013). Tipping bucket data processing for propagation application. *Electronics letters 49, no. 8*, 569-571.

Devalal, S. a. (2018). LoRa technology-an overview. *Second international conference on electronics, communication and aerospace technology (ICECA), IEEE*, pp. 284-290.

Di Serio, A. J. (2017). Potential of sub-GHz wireless for future IoT wearables and design of compact 915 MHz antenna. *Sensors* , 18, no. 1 (2017): 22.

Dinev, D. A. (2023). Analysis of LoRa RSSI Data Using Simulations and Real Devices. *18th Conference on Electrical Machines, Drives and Power Systems (ELMA)*, 1-4.

Dumitru, M.-C. R. (2023). LoRaWAN as Open Scalable IoT Ecosystem. *13th International Symposium on Advanced Topics in Electrical Engineering (ATEE), IEEE*, 1-6.

Gaelens, J. P. (2017). LoRa mobile-to-base-station channel characterization in the Antarctic. *Sensors* , 17, no. 8 (2017): 1903.

Kaewwongsri, K. a. (2020). Design and implement of a weather monitoring station using CoAP on NB-IoT network. *17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), IEEE*, pp. 230-233.

Krishnamurthi, K. S. (2015). Arduino based weather monitoring system. *International Journal of Engineering Research and General Science 3, no. 2*, 452-458.

León, J. R.-G.-P. (2017). Raspberry pi and arduino uno working together as a basic meteorological station. *arXiv preprint arXiv:1711.09750*.

Leon, M. C. (2016). *LoRa Weather Station.* Theses, Arizon State University.

Louis, L. (2016). Working principle of Arduino and using it. *International Journal of Control, Automation, Communication and Systems (IJCACS) 1, no. 2*, 21-29.

Mathur, V. Y. (2021). Weather station using raspberry pi. *Sixth International Conference on Image Information Processing (ICIIP). IEEE*, vol. 6, pp. 279-283.

Maureira, M. A. (2011). ThingSpeak–an API and Web Service for the Internet of Things. *World Wide Web 25*, 1-4.

Muller, C. L. (2013). Sensors and the city: a review of urban meteorological networks. *International Journal of Climatology*, 33, no. 7 (2013): 1585-1600.

Murdyantoro, E. R. (2019). Prototype weather station uses LoRa wireless connectivity infrastructure. *Journal of Physics: Conference Series*, vol. 1367, no. 1, p. 012089.

Pardo-Igúzquiza, E. (1998). Optimal selection of number and location of rainfall gauges for areal rainfall estimation using geostatistics and simulated annealing. *Journal of hydrology 210, no. 1-4* , 206-220.

Pasha, S. (2016). ThingSpeak based sensing and monitoring system for IoT with Matlab Analysis. *International Journal of New Technology and Research (IJNTR) 2, no. 6*, 19-23.

Sanchez-Iborra, R. a.-D. (2016). State of the art in LP-WAN solutions for industrial IoT services. *Sensors 16, no. 5*, 708.

Sawant, S. S. (2017). Interoperable agro-meteorological observation and analysis platform for precision agriculture: A case study in citrus crop water requirement estimation. *Computers and electronics in agriculture*, 138 (2017): 175-187.

Shaout, A. Y. (2014). Low cost embedded weather station with intelligent system. *10th International Computer Engineering Conference (ICENCO)*, 100-106.

Valentine, D. T. (2022). *Essential MATLAB for engineers and scientists.* Academic Press.

Wang, S.-Y. Y.-R.-Y.-H.-H.-C.-B. (2017). Performance of LoRa-based IoT applications on campus. *IEEE 86th vehicular technology conference (VTC-Fall)*, 1-6.

Warnakulasooriya, K. Y. (2018). Generic IoT framework for environmental sensing researches: Portable IoT enabled weather station. *International Conference on System Science and Engineering (ICSSE)*, 1-5.

Zare-Shehneh, N. F. (2023). Recent advances in carbon nanostructure-based electrochemical biosensors for environmental monitoring. *Critical Reviews in Analytical Chemistry*, 53, no. 3 (2023), 520-536.