Enviro-Sense: Environmental Monitoring System for Raptor Nesting Sites

Prepared by:
Clement Malakalaka (MLKCLE001)
David Young (YNGDAV005)
Tinstwalo Macebele (MCBTIN001)
Dylan Hellig (HLLDYL001)

Prepared for:
EEE4113F
Department of Electrical Engineering
University of Cape Town

May 21, 2023

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

May 21, 2023

Clement Malakalaka

David Young

Tinstwalo Macebele

Dylan Hellig

Date

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Environmental monitoring is a vital aspect of wildlife research and conservation. Collecting data in natural environments is critical for assessing changes in environmental conditions, detecting pollutants, monitoring habitat health and drawing correlations between the environment and animal behaviour. Wireless Sensor Networks (WSNs) are the leading technology when it comes to implementing environmental monitoring systems. These networks deploy distributed sensor nodes that collect environmental data and support wireless transmission and communication. WSNs are inherently, flexible, scalable, and efficient, making them a perfect technology to be employed in remote areas with unpredictable weather conditions.

1.2 Problem Statement

We interviewed Dr. Chris Vennum, a wildlife movement ecologist, specialising in the research of large African raptors. Dr. Vennum wishes to monitor the environmental conditions at various raptor nesting sites in order to further understand how different nesting sites affect the environmental conditions around the site. Additionally, careful monitoring of such conditions gives Dr. Vennum further insight into the behaviour and well-being of the raptors.

The core challenges in conducting such research is that the various nesting sites are kilometers apart from one another, in remote areas without internet access. Additionally the nesting sites are in difficult to reach areas that cannot be frequently visited due to safety concerns as well as to not disturb the raptors. Dr. Vennum needs a way to monitor the environmental conditions at nesting sites and be able to access the data from the sites remotely and simultaneously.

To solve this problem, we propose Enviro-Sense, an environmental monitoring system that implements a wireless sensor network (WSN). This system contains sensor nodes that monitor environmental conditions at the nesting sites and streams their collected data to a nearby local rendezvous node where the data is packaged and streamed over long distance wireless communication to an off-site base station. The base station stores the data in a hybrid local-cloud storage database. We have developed an API for the database and an intuitive front-end user interface to allow researchers to access the data remotely.

1.3 Scope & Limitations

The system we have developed acts as a prototype for a potential real-world implementation. Due to budget constraints we have simplified the system to simulate the desired functionality with components that would likely need to be upgraded in the future. In our WSN prototype, we have chosen to use Wi-Fi as the communication technology instead of long-distance radio communication. This decision was primarily driven by budget constraints, as Wi-Fi components are more cost effective and readily available compared to specialized long-distance radio communication modules. By using Wi-Fi we can effectively simulate the desired communication functionality. However, it is important to note that Wi-Fi generally has a much shorter communication range than long-distance radio communication. As a result, the prototype's communication is confined to a small area around the WSN nodes, and hence is not suitable for large-scale deployments without upgrading to a long-distance radio communication module in the future.

1.4 Report Outline

This report is organized into the following sections:

1. Introduction:
    - This section provides the background information, problem statement, and scope and limitations of the report.
2. Literature Review:
    - In this section, we review the existing literature and research related to environmental monitoring systems, wireless sensor networks, and their applications in wildlife research and conservation. We also discuss relevant hardware and software used in such applications.
3. Efficient Communication & Seamless User Interface:
    - In this section, we provide detailed descriptions of the implementation of communication and the user interface for Enviro-Sense, as well as various tests used to verify the operation of these subsystems within Enviro-Sense.
4. Hybrid Local-Cloud Database:
    - In this section, we provide detailed descriptions of the implementation database architecture and corresponding API used with the local and cloud databases. We also provide various tests used to verify the operation of this subsystem of Enviro-Sense.
5. Reliable Sensor Node, Data Acquisition & Transmission:
    - In this section, we provide detailed descriptions of the implementation of the sensors used in Enviro-Sense, validation of sensor data, and sensor node power optimizations. We also provide various tests used to verify the operation of this subsystem of Enviro-Sense.
6. Power Supply:

- In this section, we provide detailed descriptions of the implementation of the method of providing power to the nodes in Enviro-Sense, as well as descriptions of the various components used. We also provide various tests used to verify the operation of this subsystem of Enviro-Sense.

7. Conclusions:

   - This section summarizes the findings and conclusions from the research and implementation of the Enviro-Sense system.

8. Recommendations:

   - Here, recommendations are provided for future enhancements, improvements, and potential applications of the Enviro-Sense system.

Chapter 2

Literature Review

2.1 Environmental Monitoring Technology

2.1.1 Importance of Environmental Monitoring

Environmental monitoring plays an important role in wildlife research. Gathering environmental data aids researchers in understanding wildlife behaviour patterns, and the quality of habitats and contributes to the furtherment of conservation efforts. Numerous sensors are available for collecting essential environmental data. Some examples of environmental sensors are thermistors to measure temperature, hygrometers to assess moisture content in the air and soil, and air quality sensors to evaluate airborne pollutants [4]. To meet the needs of researchers, environmental monitoring systems must be able to gather data from multiple sensors located at various sites, as well as store and allow remote access to the collected data.

2.1.2 Wireless Sensor Networks (WSN)

A wireless sensor network (WSN) consists of a collection of interconnected sensor nodes containing wireless devices that can collect and process data [5]. The nodes transmit their data over a wireless channel to a central storage facility and can often communicate amongst each other as well [4]. WSNs have become a go-to approach for environmental monitoring systems thanks to the continuous advancement of big data storage, computational capability, miniature electronics, and ubiquitous internet connectivity [6].

2.1.3 Environmental Sensor Networks (ESN)

In this paper, we define an environmental sensor network (ESN) as an implementation of a WSN specific to environmental monitoring. An ESN is a system of interconnected sensors that are designed to monitor various environmental parameters such as temperature, humidity, air quality, water quality, and other factors. These sensors are typically deployed over a wide area, they collect data on the environment and transmit it wirelessly to a central server or data management system [6]. Fig 2.1 shows the general architecture of an ESN, each sensor node contains one or more embedded sensors. The sensor data is streamed to an on-site base station. The base station collects the sensor data and transmits it to an off-site server. An end user can then access the data on the server remotely [1].

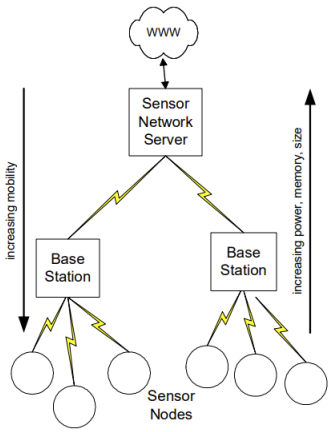


Figure 2.1: General ESN Architecture. Image source: [1]

2.1.4 Design Challenges for ESNs

There are many hurdles to overcome when designing an ESN, coming from challenges intrinsic to WSNs and the environmental context. Matin and Islam [7], as cited by Ukhurebor et al. [8], outline the following design challenges:

Fault Tolerance: Environmental sensors will most likely be deployed in an outdoor environment with harsh weather conditions. Consequentially, nodes will be more susceptible to hardware failures when compared to a controlled environment. This needs to be accounted for when designing sensor nodes for an ESN.

Scalability: As the number of sensor nodes of the system increases so does the complexity, computational load, and power consumption. If these factors are not accounted for in the initial system design, the system may not be appropriately scalable for the desired application.

Power Consumption: One of the most difficult challenges when designing a WSN is power consumption. Balancing the power requirements between sensor nodes, control units, and radio transmitters leads to constraints on the magnitude of the power supply. Careful consideration is needed to ensure a system utilises its power supply efficiently.

Transmission: Communication and data transmission is typically performed over ISM radio bands. Radio communication is intrinsically prone to interference and signal noise which can be worsened further by poor weather conditions. One needs to ensure that valuable sensor data is not lost during transmission.

2.1.5 ESN Requirements

When using a WSN to develop an environmental monitoring system it is imperative to consider what is required of the system to ensure its practical viability. G Barrenetxea et al. [9] outline 4 requirements for developing an environmental monitoring system:

Autonomy: The system must be powered throughout its operation.

Reliability: The system must perform as expected without continuous maintenance. End users must be able to use the system without technical knowledge of how the system works.

Robustness: The system must be able to withstand the environment in which it is deployed. It must be ensured that poor connectivity in remote locations or harsh weather conditions does not hinder the system's operation.

Flexibility: The system must be able to be altered to suit the needs of the end user. For example, if a user wishes to collect data at a different location.

2.2 Sensor Nodes

The sensor node (SN) is a major part of the Wireless Sensor Network (WSN) system [2]. It is responsible for sensing real-time environmental data collected by the sensors such as ambient temperature, and atmospheric pressure. Advances in wireless communications, digital electronics, and micro-electro-mechanical systems (MEMS) technology over the recent years allow for the creation of small, low-power, low-cost, multifunctional Sensor Nodes [10]. However, these sensor node networks are not perfect in terms of long-term environmental monitoring and network data security. Therefore, more power base stations and nodes must be employed. Many micro-sensor nodes can be used by WSN to collect high-precision data [10].

The use of SN networks is more advantageous over conventional sensing methods [10]. These advantages include increased accuracy, greater fault tolerance, broad coverage area, and extraction of localized characteristics. Wireless sensor nodes have unique features over traditional single-hop networks. Some of these features [10] are:

- All SNs can work as a router and communicate with each other.
- All SNs can automatically create a private network based on the communication protocol.
- Nodes can enter or leave the network equally.

However, SNs consume a lot of power for sensing, communicating, and data processing. This means that there must always be a continuous supply of power to the system. [11] suggest the use of switch-mode power supplies but careful consideration must be taken into account for the SN power system due to the risks of losses and reliability.

2.2.1 Sensor Node System Architecture

The main components of an SN are a microcontroller, transceiver unit, external memory, power source, and one or more sensors [2]. Fig. 2.2 shows the general SN architecture showing the components of the SN and their interfaces. These components allow the SN for data processing, sensing, and communications. The processing unit connects distributed SN via a communication protocol. This unit controls the functionality of the SN and processes data [11]. Most existing research uses WiFi and Bluetooth for wireless communication technologies [10]. However, this technology has limited remote access and a shorter transmission range. A solution to this problem is the Xbee communication

protocol module which can transmit data over a long range of 10-100m, and a data rate of 250 Kbit/s [2].

Microcontrollers are used as base stations in SN embedded systems over controllers such as Field Programmable Gate Arrays (FPGAs), Digital Signal Processing (DSP), and other general-purpose microprocessors. The advantages include low power consumption, ease of programming, low cost, and easy interface with other devices [2]. SNs store a lot of environmental data which limits the microcontroller storage space. A solution to this is using Flash memory due to greater storage capacity and lower cost [2]. SN communication units use the Industrial-Specific-Medical (ISM) band, which provides free radio, spectrum allocation, and global availability. However, the ISM band is susceptible to interference as outlined in section 2.4. The most relevant wireless communication used in many SN applications is Radio frequency (RF)-based communication [11].

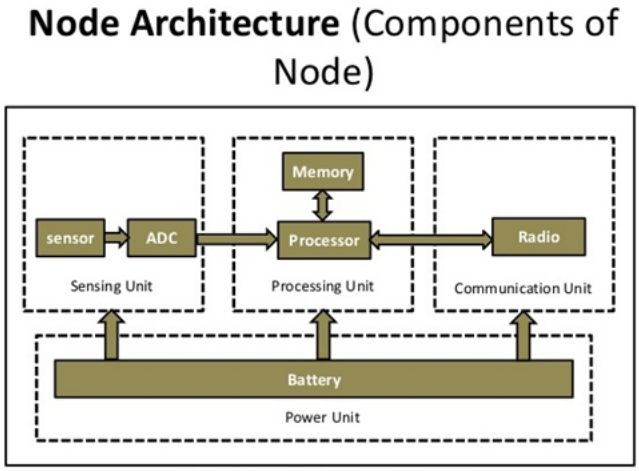


Figure 2.2: Sensor Node Architecture. Image source: [2]

2.3 Power

The Wireless Sensor Network (WSN) can be powered in many ways. This includes battery power, solar power, energy harvesting, and Power over Ethernet (PoE). This section covers possible and efficient ways that WSNs can be powered without compromising the reliability of the sensor. In general, different loads require a different power supply. The focus is mainly on supplying power to devices such as sensor nodes, antennas, and base stations. All come with different challenges and constraints.

2.3.1 Low-Power WSN

A wireless sensor network is made up of sensor nodes which are low-powered distributed devices that make up the sensor network. These nodes' function is to detect changes in the physical and environmental conditions, such as temperature and pressure, and to transfer the data to a base station or sink point where the user can access it. These networks have a seemingly limitless number of uses, however, the focus is on monitoring environmental changes [12].

Most of the time the network will be used in remote locations, and it might be challenging to recharge and replace power supply units. It is crucial to reduce the overall power consumption of the sensor

network [12].

2.3.2 Source of Energy Use

Analysing the power dissipation traits of wireless sensor nodes is the first stage in designing a low-power wireless sensor network. The network has different components that have a wide range of options, and picking the proper device will have an impact on energy consumption.

2.3.3 Power Supply

A key system component for wireless sensor networks is the power supply. In addition to taking the network's consumption characteristics into account, a low-power network must also take the supply side into account. Most of the time, a sensor node's power source is a battery. The battery is essential in deciding how long a sensor node will last because it powers the entire device. Batteries are intricate systems whose performance is influenced by a variety of elements, such as the battery's size, the material used for the electrodes, and the pace at which the active ingredients diffuse into the electrolyte [13].

The rate capacity impact is one of the most crucial elements that a designer must consider. It is among the most crucial elements that determine battery lifetime. The amount of current drawn from the battery or the discharge rate has an impact on this effect. Every battery has a manufacturer-specified rated current capacity. Battery life is significantly shortened when a device draws more current than its rated value. This is because a high current draw from the battery causes the active chemicals' rate of diffusion through the electrolyte to lag behind their rate of consumption at the electrodes.

Long-term high discharge rates exhaust the active components in the electrodes, which leads to battery failure.

2.3.4 Energy Reserves

Today, the main method of powering wireless devices is energy storage, namely electrochemical energy that is kept in a battery. Other types of energy storage, however, might be helpful for wireless sensor nodes. The fixed amount of energy stored on the device will determine the lifetime of the node regardless of the type of energy storage used. Usable energy per unit volume (J/cm3) will be the main metric of interest for all types of energy storage. Another problem is that an energy reservoir's capacity to produce electricity instantly is typically based on its size. The maximum power density (W/cm3) is thus a problem for energy reservoirs in specific circumstances, such as micro-batteries [14].

2.3.5 Power Distribution

A wireless node can store electricity on it, but in some cases, it can also receive power from a nearby energy-rich source. Because the power received at the node typically depends more on how much power is transferred than on the size of the power receiver at the node, it is challenging to evaluate the effectiveness of power distribution methods using power or energy density as parameters. However, an effort is made to characterize the effectiveness of power distribution systems as they pertain to wireless sensor networks [14].

2.4 Communication Methodology

Communication and transmission of data in WSNs is typically implemented using radio. These signals are transmitted on popular Industrial-Scientific-Medical (ISM) radio frequency (RF) bands. However, in some implementations of WSNs, they will use optical or infrared communication instead of RF [7].

2.4.1 Overview of ISM Communications

The ISM frequency bands are scattered over the frequency range of 6.78 KHz to 245 GHz in the radio spectrum. These bands were originally designed for use in microwave, medical equipment, and electrode-less lamps. However, over the years, wireless communication devices have been developed which also operate in these bands without causing interference with existing ISM applications [15]. A significant portion of these devices utilize short-range communication, such as wireless phones, Bluetooth devices, near field communication (NFC) devices, and wireless computer networks.

A framework for the use of these frequency bands is provided by the international telecommunication union (ITU), which is a specialized agency of the UN. Each member country of the UN has its own jurisdiction to follow the regulation framework. Some of the key regulations specified by the ITU are [16]

- Devices operating with the ISM bands are subject to maximum power limits which are intended to prevent interference with other radio services.
- Many ISM devices use frequency hopping to rapidly switch between different frequencies. The ITU specified rules for frequency hopping to ensure that devices do not interfere with other radio services.
- ISM devices are also subject to regulations governing their harmonic emissions. Harmonic emissions are unwanted radio signals that can interfere with other radio services. The ITU has established limits on the amount of harmonic emissions devices can produce.
- ISM devices cannot cause harmful interference with other radio services, and operators of ISM devices must take all reasonable measures to prevent interference.

While the ITU's policies carry weight and are typically followed by member states, they are not necessarily legally binding and may be considered more as guidelines than strict regulations. Therefore, countries have the liberty to adhere to the ITU's framework for the allocation and use of radio frequency spectrum. However, there have been cases where countries have deviated from these regulations. When it comes to the design of WSNs which utilize the ISM frequency bands, the exact frequencies to be integrated into the nodes will be determined by the country in which the WSN is to be deployed.

2.4.2 Overview of Optical Communications

Most existing research into WSNs have been focused on the use of radio frequency (RF) for communications. However, RF communications can severely limit WSNs lifetime and battery resources [17]. A solution to this issue is proposed by utilizing optical communications which offer distinct advantages over RF, such as a high data rate of transmission and lower power requirements ([18, 19, 20] as cited in [21]).

In the realm of optical communications, several recognized techniques of transmitting information via light are available. Namely, fiber-optic, free-space, infrared, and visible light. However, for the purpose of this paper, given that WSNs are intended to connect to wireless network, fiber-optic communication will be ignored because it requires laying of physical cables for communication.

Visible Light Communication

Visible Light Communication (VLC) is a type of optical communication which uses the frequencies from 430 THz to 790 THz to transmit data at speeds up to 10 Gbits/s [22]. Due to the nature of VLC, ambient light sources will degrade the performance of the VLC system. However, there are some binary coding techniques that were designed to reduce background noise, such as Hadamard coding and Manchester coding [22]. While this method of communication has its benefits, the biggest limitation is the limited range of transmission of 100 meters [23]. The light sources would have to be very high power LEDs or a similar light source [23].

Free-Space Optics Communication

Free-Space Optics Communication (FSOC) operates by transmitting invisible light beams that are safe for the eyes, using a laser to focus light onto a photon detector equipped with a telescopic lens that acts as the receiver [24]. Basic FSOC systems have been designed to transmit data at up to 155 MBps over a distance of 5.6km, and for shorter distances of <150 m, at up to 10GBps [24]. FSO systems have the benefit of ease of installation, high bandwidth, and no licensing required. However, it should be noted that the power consumption is higher than regular RF, the communication requires clear atmospheric conditions, and the cost of equipment ranges between $3000 and $10,000 [24]. This could prove difficult, and potentially impossible, to implement in an WSN due to power supply limitations, physical obstructions, and environmental conditions.

Infrared Communication

Wireless infrared communication uses infrared light to transmit data between two devices. Infrared is generally more suited for short-range communication scenarios, however, when line of sight (LOS) can be guaranteed, the range can be drastically improved to provide longer links [25]. If LOS cannot be guaranteed, then the issue of atmospheric path loss, which is a combination of clean-air absorption and scattering due to particles in the air, will cause the effective range of communication to degrade significantly [25]. However, infrared communication does offer low cost, light weight, and moderate data rates when LOS is provided [25].

2.5 Data Management

WSNs generally have limited resources including processing capabilities, memory, and energy/battery [3]. The goal of a WSN is typically to collect environmental data, such as temperature, humidity, light, or sound, and send it to users who need it for analysis or decision-making.

The data collected by WSN sensors needs to be transmitted to the users via sink node(s) [3]. A sink node is a powerful node that acts as a gateway between the WSN and an external system such as

the Internet. It collects the data from the WSN and forwards it to the external system for further processing or analysis [3].

2.5.1 Overview of Methods

There are currently three methods by which the WSN data could reach the sink node(s) as detailed in [3]:

- **Local Storage:** A data management method where data is stored in the node that produced it, and the sink node sends a query to the node to collect the data. However, this method has two main disadvantages; it can quickly exhaust the resources of nodes that store a large amount of data, and sinks may not know in advance which node is storing the data, leading to a communication overhead problem where queries are flooded to all nodes.
- **External Storage:** A data management method in which data is sent to the sink as soon as it is produced using routing protocols. However, this method has several disadvantages. Firstly, produced data cannot be aggregated. Secondly, if multiple sinks are deployed, the node duplicates the data and sends it to each sink. Additionally, under a mobile scenario, the sink may miss the data. Therefore, the external storage approach may not be feasible in applications where the WSN has an intermittent connection with a mobile sink.
- **In-network Storage:** This storage scheme addresses the drawbacks of other data management methods by replicating data in a set of nodes in the WSN. In this scheme, rendezvous nodes are used for data storage, and when a node detects an event, it sends data to the rendezvous nodes. Sinks interested in the data query the rendezvous nodes to retrieve the information, using the minimal number of hops to reduce complexity and energy consumption for sink queries, and thus increasing the lifetime of nodes [26].

Since in-network storage has been shown to be the preferred data management method, the rest of this section will discuss the basic concept, requirements, and challenges of in-network storage.

The concept of in-network storage and its benefits during sink querying are illustrated in Figure 2.2(a) and (b) respectively. Rendezvous nodes are illustrated by gray dots and the data producer by the black dot [3].

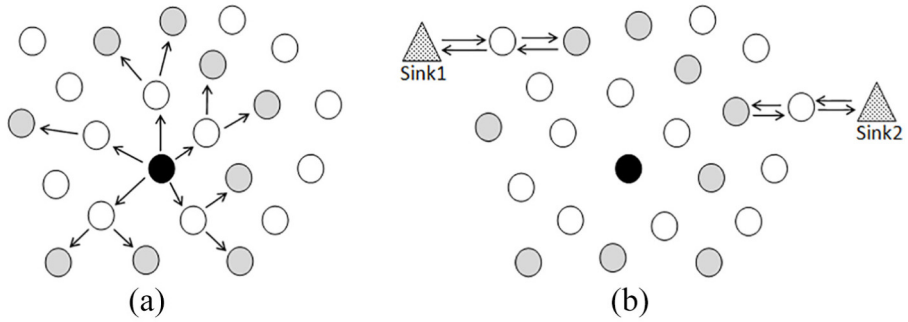


Figure 2.3: In-network storage in WSN and its advantages: (a) in-network storage and (b) external sink request. Image source: [3]

2.5.2 Requirements of In-Network Storage

There are three main requirements for sufficient quality of service for in-network storage [3]:

- **Reliability:** High reliability is a major requirement for successful in-network storage process as it ensures that each data message can reach any node in the network without errors, preventing network inconsistency and data loss.
- **Efficiency:** The two major efficiency requirements for in-network storage are time and energy. Time efficiency involves reducing the period of time that the wireless channel is occupied by a high number of messages to make the process as fast as possible [27], while energy efficiency involves minimizing power consumption during the process due to limited energy resources of sensor nodes, including flash memory operation, radio communication, and idle listening.
- **Scalability:** The in-network storage should be able to successfully operate on any scale of WSNs.

2.5.3 Challenges of In-Network Storage

The various key challenges of in-network storage, detailed in [3], are as follows:

- **Reliability:** Achieving high reliability in in-network storage is challenging due to the dynamic nature of WSNs, which can cause network connectivity to change over time, and in-network storage must ensure that all nodes and sinks can receive data despite potential disconnections and reconnections.
- **Communication overhead:** Communication overhead resulting from excessive redundant transmission due to the *broadcast storm problem* [28] can lead to high energy consumption in WSNs, making it important to avoid for reliable and energy-efficient in-network storage.
- **Hotspot problem:** The *hotspot problem*, which occurs when nodes near rendezvous locations expend their energy resources earlier due to multi-hop route intersection and traffic concentration, must be avoided to prevent the rendezvous location from becoming inaccessible for data source nodes [29].
- **Memory overhead:** The memory capacity of sensor nodes limits the storage capacity of in-network storage, and using the same storage nodes for a long time can lead to memory overhead, so different nodes should be selected as storage nodes at different times to solve this problem.
- **Energy efficiency:** To achieve energy-efficient storage in WSNs, the wireless communication and radio-on time must be minimized to avoid excessive redundant transmissions and high energy consumption.

'In-network data storage protocols for wireless sensor networks' by K. Mekki et al., [3], goes into much further depth on the inner-workings of in-network storage for WSNs, which will not be covered here due to scope.

2.6 Implementations

WSNs are employed for diverse environmental monitoring applications, such as agriculture, wildlife, climate, and forest monitoring [30]. This chapter provides an overview of some research papers that address WSN-based environmental monitoring solutions.

The paper [2], proposes a system, that can monitor gases in the air, in real time by utilising wireless sensor networks and IoT technologies. The sensor nodes use an Arduino Mega board connected to an MQ2 and MQ7 gas sensor. A Raspberry Pi acts as the base station and connects to the sensor nodes via the Zigbee communication protocol. An XBee transmission module is used for wireless linking between the sensor nodes and the base station. The base station analyses and stores the sensor data on a local and cloud database. The test results for the system indicated that it was highly accurate in collecting data, with a correlation coefficient greater than 0.95. Additionally, the system demonstrated a high transmission success rate of 99% and consumed a low amount of energy, with a total power consumption of 200mW. Some limitations of the system included potential transmission range obstacles over large distances and it was noted that the system's performance would likely decrease with the addition of more sensor nodes.

[31] developed a similar system to [2] for the purpose of greenhouse automation and monitoring. Their design also made use of a Raspberry Pi for the base station and collects temperature, humidity, and soil moisture data. The system proved to be extremely effective in improving the greenhouse environmental control as well as water and energy efficiency. However, they experienced issues with sensor accuracy due to the environmental conditions in the greenhouse.

An interesting habitat monitoring WSN application was developed by [32]. Their research uses a WSN for the purpose of counting birds within a habitat, using sensor nodes containing microphones. The sensors record audio samples, which are used to recognize bird species via a classification process. The data is stored at a central base where an algorithm is used to estimate the number of singing birds in the habitat.

2.7 Conclusion

Wireless sensor networks offer a promising solution for environmental monitoring due to their low cost compared to traditional monitoring systems, flexibility, real-time monitoring, data integration, and scalability. However, the successful deployment of such a system requires careful consideration of several challenges. The most significant challenges being power management, communication reliability, data storage, and the ability to withstand harsh environmental conditions. Achieving an effective environmental sensing system is dependent on finding a balance between reliability and performance, which can be accomplished by an attentive and resourceful design process.

Chapter 3

Efficient Communication & Seamless User Interface (YNGDAV005)

3.1 Introduction

In the realm of Wild African Raptor research, the acquisition and analysis of environmental data from nesting sites hold significant importance. Such data provides researchers with crucial insights into the raptors' behavior and health. Within our wireless sensor network (WSN) implementation, ensuring convenient and remote access to environmental data from these nesting sites is a key objective, as it directly impacts the effectiveness and efficiency of the research.

This chapter presents the design and operation of two of our specialized subsystems, dedicated to efficient communication and a seamless user interface. Efficient communication plays a vital role in the success of the WSN by enabling near real-time data transmission and minimizing delays between data collection and analysis. It ensures that researchers have access to up-to-date information, allowing them to respond promptly to changes in the environment or the raptors' behavior. Additionally, seamless user interface design enhances the usability of the WSN, making it intuitive and easy for researchers to interact with the collected data and gain valuable insights.

With the primary focus on optimizing the collection, transmission, and in-network storage of temperature and humidity data, these subsystems employ innovative approaches to ensure data integrity and streamline the user experience with our WSN. By implementing efficient communication protocols and designing a user-friendly interface, we aim to provide researchers with a reliable and convenient tool to monitor and analyze the environmental conditions at the raptors' nesting sites.

To achieve our objectives, we have made specific design choices to address the unique challenges posed by our project. Considering the limited internet access at the remote nesting sites, we have established a self-contained communication infrastructure utilizing sensor nodes, rendezvous nodes, and a centralized sink node. This design allows us to overcome the limitations of internet connectivity by creating a network within the nesting sites and ensures that we can collect and transmit environmental data efficiently.

In addition to communication, we have dedicated efforts to create a seamless user interface that facilitates easy access and interaction with the environmental data collected by the WSN. Our user interface design prioritizes intuitive navigation, data visualization, and user-friendly controls. By ensuring a seamless user experience, researchers and stakeholders can conveniently monitor and analyze

the environmental conditions, gaining valuable insights into the raptors' habitat and behavior.

Efficient communication and a seamless user interface offer numerous benefits to our WSN solution. By optimizing data transmission and ensuring easy access, we enhance the efficiency of environmental data collection and analysis. These optimizations can not only contribute to the success of our Wild African Raptor research but also enhance the scalability and usability of our WSN design, allowing for potential applications in various environmental monitoring projects. The implementation of efficient communication and a seamless user interface represents an advancement for the field of wildlife research and monitoring, empowering researchers with the tools they need to make informed decisions and drive positive change.

3.2 System Design

3.2.1 Requirements

Table 3.1: Requirements Table for Communication and UI Subsystems

Requirement ID	Description	Specification ID
R1	The communication subsystem must establish reliable and timely data transmission between sensor nodes and the sink node.	S1
R2	The subsystem must minimize bandwidth usage to conserve power and optimize data transmission efficiency.	S2
R3	The subsystem must operate effectively under limited internet connectivity or intermittent network access.	S3
R4	The user interface shall be accessible remotely, allowing researchers to monitor environmental conditions from any location with internet connectivity	S4
R5	The user interface shall display historical data trends and allow users to select specific time intervals for analysis.	S5
R6	The user interface shall be compatible with popular web browsers and mobile devices for enhanced accessibility.	S6

3.2.2 Specifications

Table 3.2: Specifications Table for Communication and UI Subsystems

Specification ID	Description	ATP ID
S1	The communication subsystem can transmits data from sensor node to sink node.	ATP1
S2	The subsystem has low power and bandwidth usage.	ATP2
S3	The rendezvous node and sink node create their own Wi-Fi network in the network and has external internet connection at sink node.	ATP3

S4	The user interface is accessible remotely, leveraging internet connectivity to enable researchers to monitor environmental conditions from any location.	ATP4
S5	The user interface provides a visual representation of historical data trends to enable users to analyze and understand long-term environmental patterns.	ATP5
S6	The user interface is compatible with popular web browsers on iOS, Android, MacOS, Windows, and Linux.	ATP6

3.2.3 Acceptance Test Procedures (ATPs)

Table 3.3: ATP Table for Communication and UI Subsystems

ATP ID	Description	Expected Outcome
ATP1	Verify that transmission of data from sensor node to sink node occurs.	Sensor data should be stored intermittently at the rendezvous node and at the sink node.
ATP2	Verify that the subsystem has low power and low bandwidth usage.	The communication between nodes should only occur at predetermined intervals, otherwise nodes should be inactive.
ATP3	Verify that the nodes collectively create their own Wi-Fi network and has external internet connection at sink node.	Rendezvous nodes should create their own network and sink node should have global internet connection.
ATP4	Verify that the user interface can be accessed remotely.	The user interface can be accessed outside the local network.
ATP5	Verify that the user interface can display historical data visually.	The user interface can generate charts or diagrams for the user.
ATP6	Verify that the user interface is compatible with popular web browsers.	The user interface operates identically across all platforms.

3.3 Design Choices

In this section, we outline the design choices we made for the communication and user interface subsystem, including the establishment of a self-contained communication infrastructure and the selection of communication protocols, to address the challenges of limited internet access, power constraints, and the need for a seamless user experience.

3.3.1 Self-Contained Communication Infrastructure

The self-contained communication infrastructure was a crucial design choice aimed at overcoming the challenge of limited internet access at the remote Wild African Raptor nesting sites. In order to establish this infrastructure, we strategically designed rendezvous nodes, which act as Wi-Fi access points (AP) [33], creating a localized communication network for the sensor nodes. By connecting to the AP of the rendezvous nodes, the sensor nodes can efficiently transmit data without relying on

external internet connectivity. This self-contained approach was chosen due to its ability to provide a reliable means of communication, ensuring data collection even in areas with no or intermittent internet access.

The role of the centralized sink node is pivotal within the self-contained communication infrastructure, as it serves as both an AP [34] for the rendezvous nodes and a connection to the external internet. Acting as the data aggregation point, the sink node receives data from the rendezvous nodes, facilitating further processing and analysis of the collected data. Moreover, the sink node acts as the gateway to the external internet, enabling remote access and interaction with the WSN. This feature provides researchers with convenient and remote access to the environmental data from the nesting sites, enabling them to monitor and analyze the data without physically being present at the locations.

3.3.2 Communication Protocols

To address limited internet access and budget constraints, we initially wanted to implement radio communication for long-distance transmission within our WSN. However, due to budget limitations, we opted to utilize short-distance Wi-Fi communication as the communication protocol. This choice enabled reliable and high-speed data transmission between sensor nodes and rendezvous nodes while leveraging existing Wi-Fi hardware on the microcontrollers. Additionally, we incorporated sleep modes where the communication between nodes is halted to minimize power consumption of the rendezvous nodes and sensor nodes.

3.3.3 Bandwidth Optimization Strategies

To optimize bandwidth usage within our wireless sensor network (WSN) and strike a balance between data accuracy and transmission efficiency, we implemented several strategies. Considering the relatively slow changes in temperature and humidity over short periods of time, we made the decision to record sensor data on the sensor node every two minutes. This allowed us to capture the necessary data while reducing the amount of data to be transmitted.

To further minimize transmission overhead, we introduced a data collection scheme where the rendezvous node would request the recorded data from the sensor node every 10 minutes. This approach significantly reduced the frequency of data transmission. Subsequently, the rendezvous node aggregated the collected data and stored it in a Comma-Separated File (CSV). To ensure long-term data storage and minimize additional data requests, the sink node would request the aggregated data from the rendezvous node once every 24 hours. Upon receiving the data, the sink node would store it in its local database for further analysis and processing.

While this approach introduces a potential delay of up to 24 hours in data availability for users, it was a necessary compromise to optimize bandwidth usage. By implementing this data collection and transmission scheme, we aimed to strike a balance between ensuring data accuracy and minimizing the overall transmission overhead within our WSN. This optimized approach not only conserves bandwidth resources but also enables efficient data transmission, making our WSN suitable for remote and low-bandwidth environments without sacrificing essential data for Wild African Raptor research.

3.3.4 User Interface

In designing the user interface (UI), the Bootstrap framework was used. Bootstrap is a popular front-end framework known for its responsive grid system, pre-designed UI components, and customization options. By leveraging Bootstrap, we ensured that the user interface adapts seamlessly to various screen sizes and devices, while also saving development time. The framework's extensive documentation and active community support provided valuable resources for learning and implementing Bootstrap effectively, resulting in an efficient, visually appealing, and user-friendly interface design.

3.3.5 UI: Visual Representation of Data

To enable the visual representation of data in our user interface, we chose to integrate Flot Charts, a widely-used JavaScript charting library. We selected Flot Charts for its extensive features, flexibility, and compatibility with modern web browsers. By incorporating Flot Charts into our design, we were able to create interactive and visually engaging graphs and charts that effectively communicate historical data trends to users. The library's comprehensive documentation and community support facilitated our implementation, ensuring smooth integration and customization according to our specific data visualization needs.

3.3.6 UI: Remote Accessibility and Internet Connectivity

To enable remote accessibility and internet connectivity for our user interface, we implemented a combination of technologies, including Cloudflare. By utilizing Cloudflare's content delivery network (CDN) and security features, we ensured fast and reliable access to the user interface from various locations. Furthermore, we leveraged the Sink Node of our wireless sensor network (WSN) to provide the website, enabling researchers to access the user interface directly through the WSN infrastructure. This approach eliminated the need for separate hosting services and streamlined the accessibility of the user interface for monitoring environmental conditions.

To make the UI remotely accessible, a Dynamic Domain Name System (DDNS) [35] must be set up. This was done using Cloudflare and a simple bash Cloudflare DDNS updater script [36]. This allowed the website to be accessed via a Universal Resource Locator (URL) from anywhere in the world.

3.4 Prototype Design

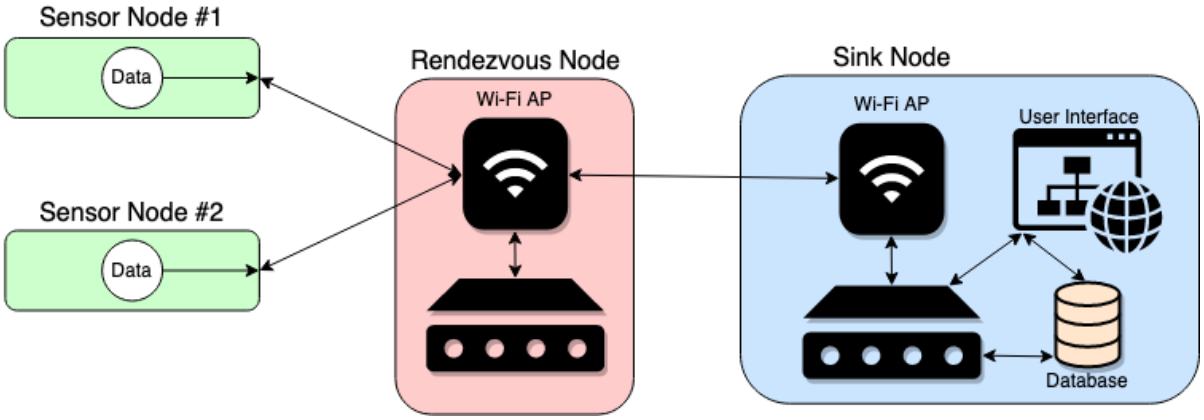


Figure 3.1: Prototype Communication and User Interface Design

Figure 3.1 illustrates the overall prototype design of the communication and user interface subsystems in our prototype WSN. For establishing connectivity, both the rendezvous node and the sink node create their own Wi-Fi access points (AP). The sensor nodes connect to the rendezvous node's AP [33], utilizing the ESP32-PICO-D4's Wi-Fi module. The rendezvous node establishes a connection with the sink node's AP [34] using the Wi-Fi module on the ESP32-CAM and the Raspberry Pi Zero respectively.

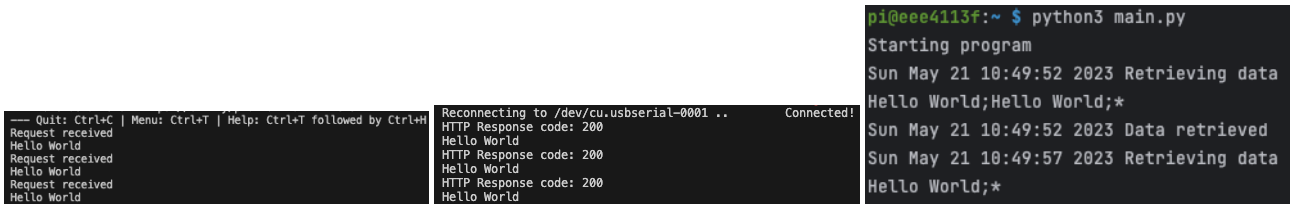
The design utilizes a various optimizations to alleviate network pressure and ensure efficient data transmission. Data collection occurs at a frequency of two minutes on the sensor nodes, while the rendezvous node requests the data from the sensor nodes every 10 minutes. Furthermore, the rendezvous node stores the collected sensor data for a 24-hour period before forwarding it to the sink node, ensuring ample time for data consolidation and optimization. At the sink node, the user interface is provided, leveraging web technologies and software framework such as Node.js and React.js, so that no external hardware resources are needed to host the website.

By incorporating specific microcontrollers, Wi-Fi modules, and software frameworks, the prototype achieves the desired functionality of the communication and user interface subsystems in the WSN.

3.5 Testing and Results

3.5.1 ATP 1: Data Transmission

To verify that data is able to be transmitted from sensor node to sink node successfully, a simple string can be generated at a sensor node and forwarded along the network to the sink node. The string can be outputted to terminal or over a serial connection at each of the devices in the network to verify that the string is successfully received at each node. It is expected that the string will be unaltered and be printed at each node.



(a) Sensor node. (b) Rendezvous node. (c) Sink node.

Figure 3.2: Terminal and serial outputs showing 'Hello World' propagate through the nodes in the network

As can be seen from figure 3.2, the 'Hello World' string is forwarded from the sensor to the rendezvous node, and finally to the sink node. The sink node displays the 'Hello World' string twice in the first request in (c) because the programs started executing at slightly different times, hence the first two requests to the sensor node got placed in the array on the rendezvous node before the sink node made its request and the array was cleared.

3.5.2 ATP 2: Low Network Traffic

To verify that the nodes only transmit data between nodes at predetermined intervals, we only need to understand at the code responsible. On the sensor node, transmission to the rendezvous node is only executed upon the sensor node receiving a HTTP GET request. The figure 3.3 below shows the code responsible for this behaviour.

```
server.on("/data", HTTP_GET, [](AsyncWebServerRequest *request){
  Serial.println("Request received");
  request->send_P(200, "text/plain", readData().c_str());
});
```

Figure 3.3: C++ code for HTTP GET request on the sensor node

The figure 3.4 below shows the code responsible for requesting the data from the sensor node. By simply adjusting the *interval* variable, the time between requests to the sensor node can be adjusted.

```
unsigned long previousMillis = 0;
const long interval = 5000;

void loop(){
  unsigned long currentMillis = millis();

  if(currentMillis - previousMillis >= interval) {
    // Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED ){
      data = httpGETRequest(serverNameData);
      Serial.println(data);
      // save the last HTTP GET Request
      myStrings.push_back(data.c_str());

      previousMillis = currentMillis;
    } else {
      Serial.println("WiFi Disconnected");
    }
  }
}
```

Figure 3.4: C++ code for HTTP GET request on the rendezvous node

Lastly, since the sink node was coded using Python, to adjust the time between sink node requests to the rendezvous node, a *time.sleep(#)* function was used, where # represents the seconds between requests.

3.5.3 ATP 3: Wi-Fi Network Creation

To verify that the rendezvous node and sink node are creating their own Wi-Fi access points (AP), we can use a smartphone to scan available networks.

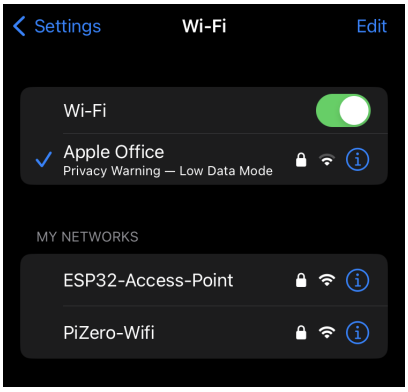


Figure 3.5: Network scan on iOS showing node APs

Figure 3.5 shows two networks being created. ESP32-Access-Point and PiZero-Wifi being created by the rendezvous node and sink node respectively. To verify if the sink node has internet connectivity, we can execute a ping command on the sink node.

```
pi@eee4113f:~ $ ping google.com -c4
PING google.com (172.217.170.78) 56(84) bytes of data.
64 bytes from jnb02s04-in-f14.1e100.net (172.217.170.78): icmp_seq=1 ttl=108 time=24.1 ms
64 bytes from jnb02s04-in-f14.1e100.net (172.217.170.78): icmp_seq=2 ttl=108 time=27.2 ms
64 bytes from jnb02s04-in-f14.1e100.net (172.217.170.78): icmp_seq=3 ttl=108 time=28.4 ms
64 bytes from jnb02s04-in-f14.1e100.net (172.217.170.78): icmp_seq=4 ttl=108 time=28.0 ms

--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 24.092/26.929/28.386/1.691 ms
```

Figure 3.6: Ping of google.com on sink node

Figure 3.6 shows that the sink node is able to successfully ping google.com, hence the sink node has internet connectivity.

3.5.4 ATP 4: UI Remote Access

To verify if the UI can be accessed remotely, we only need to go to the designated URL and port number in a browser. These were yu0n9f4m11y.co.za and port 3000.

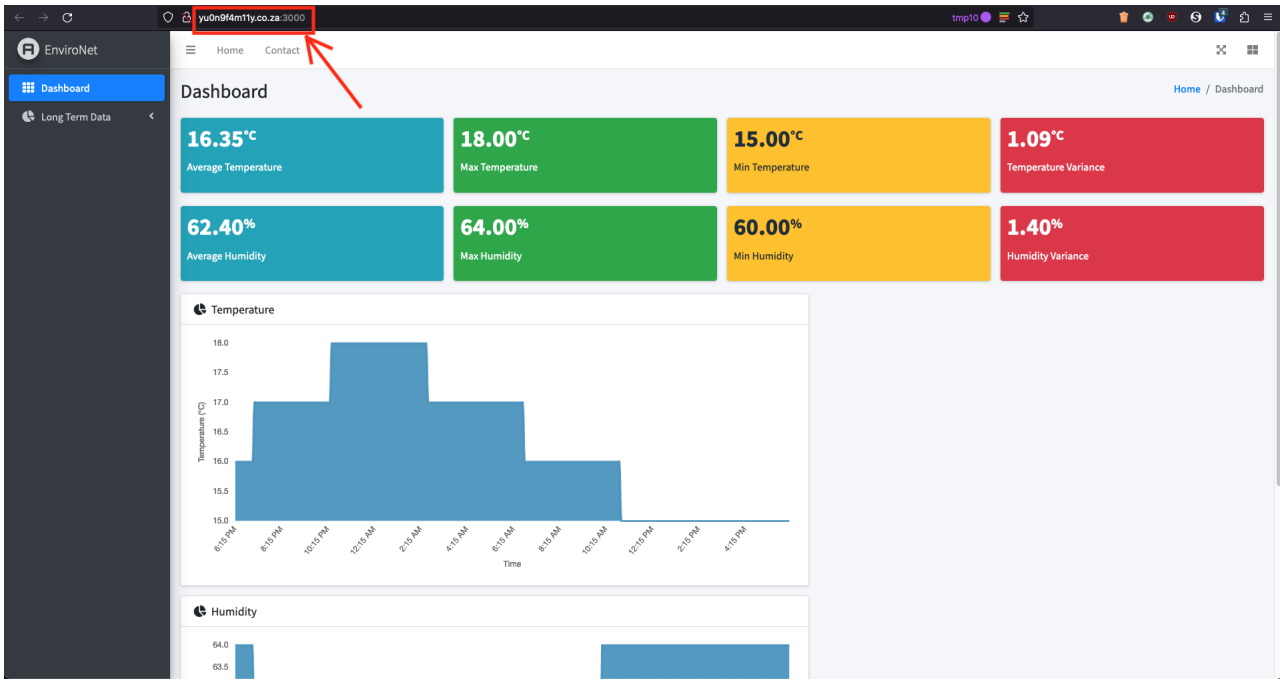


Figure 3.7: Homepage of user interface accessed remotely

Figure 3.7 shows that the UI could be accessed remotely via the URL http://y0n9f4m11y.co.za:3000/. In a proper deployment however the URL would be changed to a more relevant URL like https://envirosense.com/ or https://envirosense.co.za/. However, for illustration of functionality this suffices.

3.5.5 ATP 5: Displaying Data on UI

To verify that the UI can display graphs of data we can view the homepage of the UI.

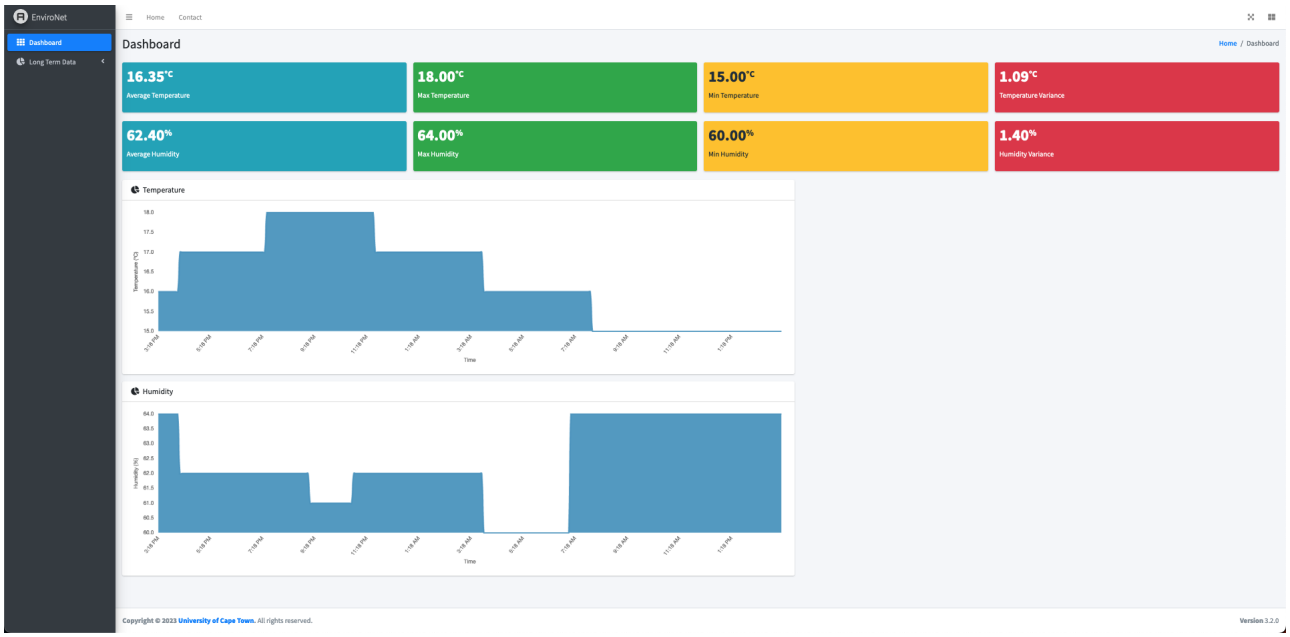


Figure 3.8: Homepage of user interface

Figure 3.8 shows that the UI can generate graphs of data and perform some basic calculations on the data. Unfortunately, due inexperience with JavaScript and web design, implementing graphs showing historical data based on a selected date range was unsuccessful. However, this feature can be implemented in the future iterations of the design.

3.5.6 ATP 6: UI Cross-Platform Compatibility

To verify if the UI is cross-platform compatible, we can use the same testing process as with ATP 4. ATP 4 was tested on the latest version of FireFox browser. The figures below show the same web page on various popular browsers.

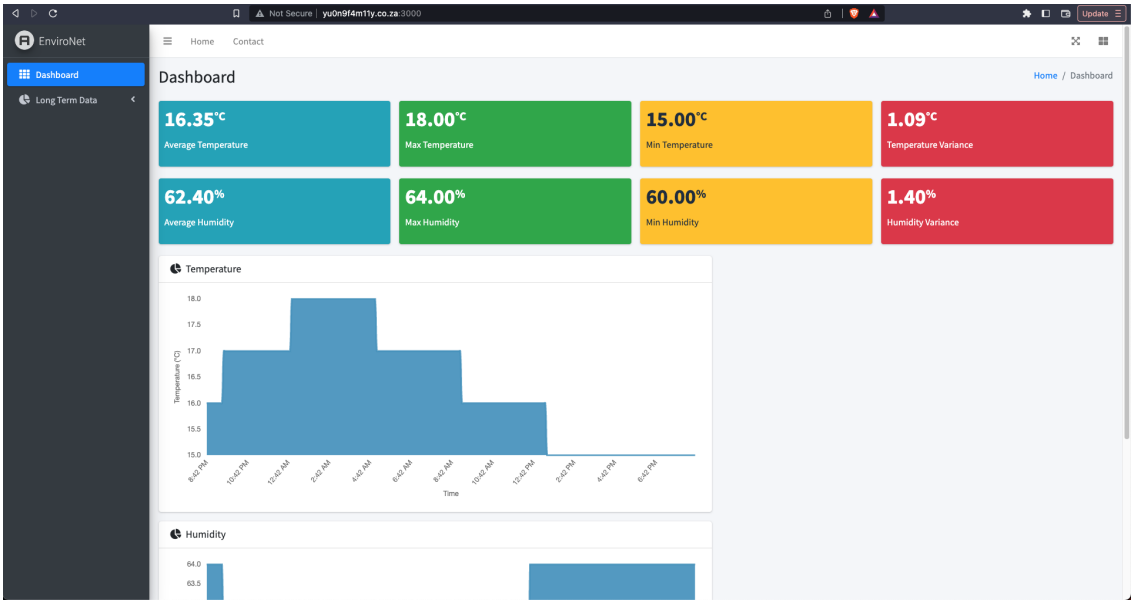


Figure 3.9: UI accessed via a Chromium-based browser on MacOS

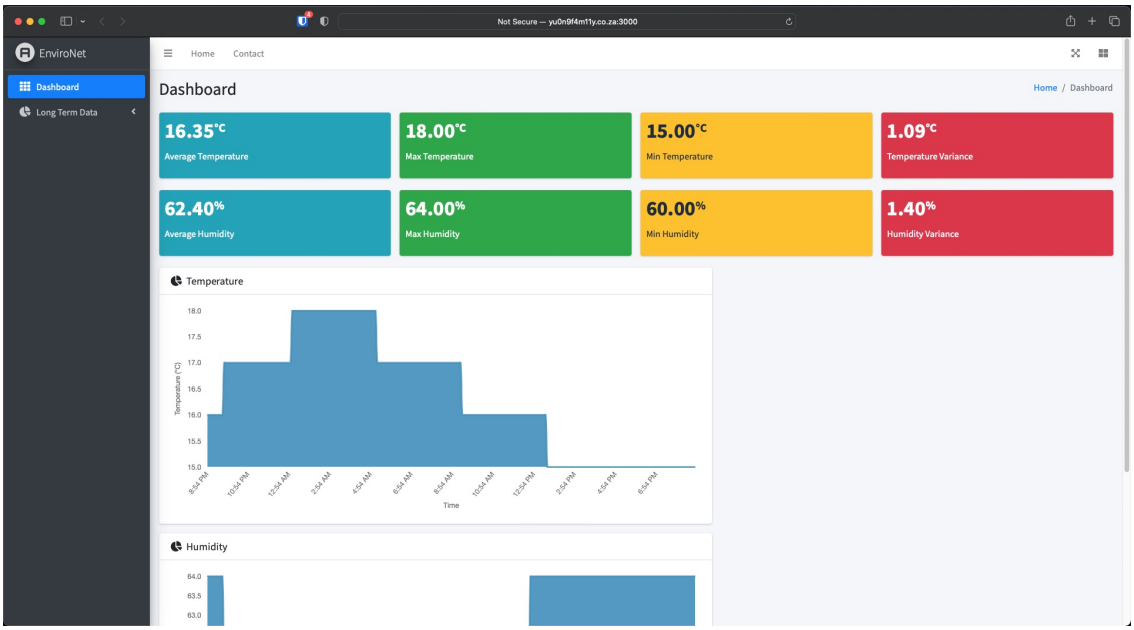
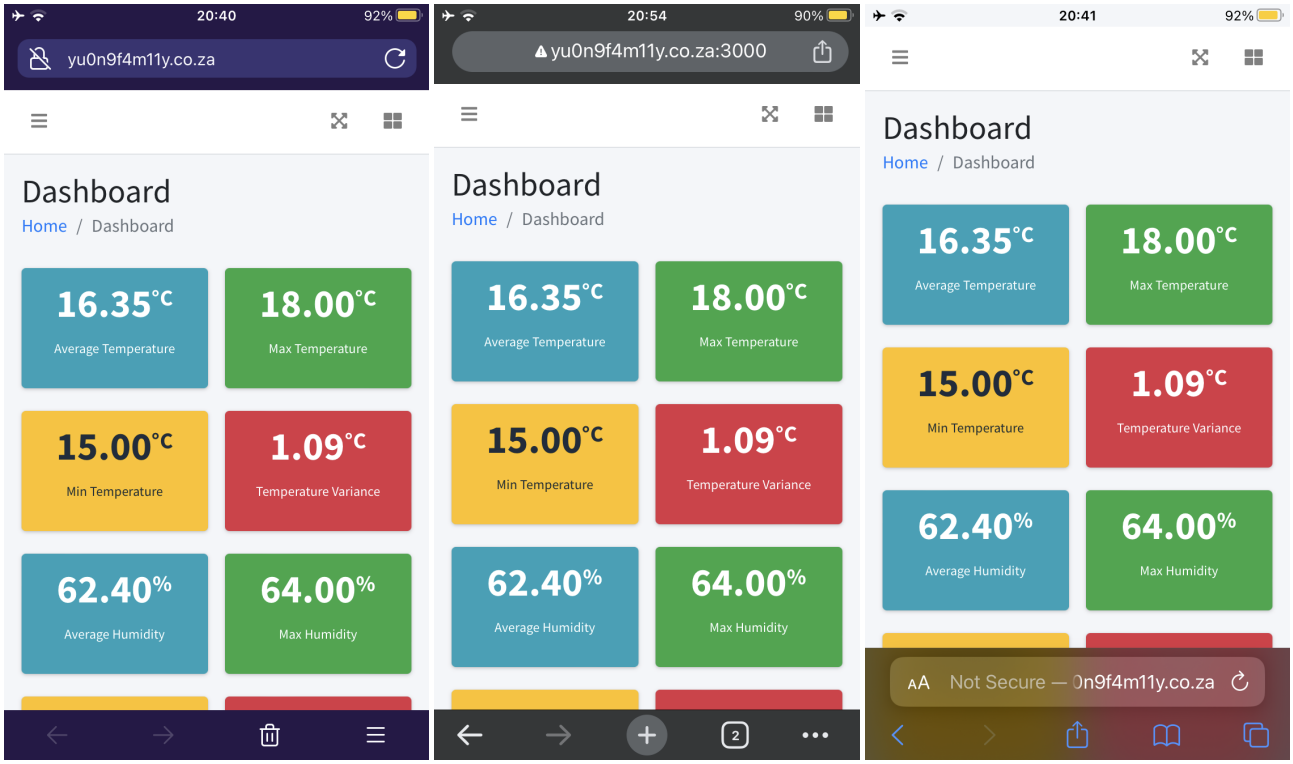


Figure 3.10: UI accessed via the Safari browser on MacOS



(a) iOS FireFox (b) iOS Chrome (c) iOS Safari

Figure 3.11: UI accessed via (a) FireFox on iOS, (b) Chrome on iOS, and (c) Safari on iOS

The figures (3.9, 3.10, 3.11) above show that the UI has been compiled by the browsers on various platforms identically. The tested browsers shown above account for approximately 84% of the browser market share on desktop devices [37] and approximately 90% on mobile devices [38].

3.6 Conclusion

In conclusion, this chapter presented the design and operation of specialized subsystems dedicated to efficient communication and a seamless user interface in the context of a wireless sensor network (WSN) for Wild African Raptor research. The chapter emphasized the importance of convenient and remote access to environmental data from nesting sites and outlined the objectives of optimizing data collection, transmission, and storage.

To address the challenges posed by limited internet access and the need for a user-friendly interface, specific design choices were made. The chapter highlighted the establishment of a self-contained communication infrastructure using sensor nodes, rendezvous nodes, and a centralized sink node. This design allowed for reliable and timely data transmission, even in areas with limited or intermittent internet connectivity. The user interface design focused on intuitive navigation, data visualization, and compatibility with popular web browsers and mobile devices.

The implementation of these design choices involved various strategies, such as utilizing short-distance Wi-Fi communication, implementing sleep modes to minimize power consumption, and optimizing bandwidth usage by aggregating and filtering data. The integration of the Bootstrap framework and Flot Charts library facilitated the development of a responsive and visually appealing user interface, capable of displaying historical data trends and enabling remote accessibility for researchers.

Overall, the chapter demonstrated the successful implementation of efficient communication and a seamless user interface in the prototype WSN. By optimizing data transmission and ensuring easy access to environmental data, the subsystems enhance the efficiency of data collection and analysis, contributing to the scalability and usability of the WSN design. These advancements have the potential to benefit not only Wild African Raptor research but also various other environmental monitoring projects.

Chapter 4

Hybrid Local-Cloud Database (HLLDYL001)

4.1 Introduction

In the field of wild African raptor conservation and research, the ability to gather and analyse environmental data from various nesting sites is of paramount importance. Access to such data allows researchers to monitor the relationship between this data and the behaviour and health of the raptors. In turn, facilitating the implementation of effective conservation strategies. An important aspect of our wireless sensor network (WSN) implementation is to ensure researchers can access environmental data from the nesting sites conveniently and remotely.

This chapter presents the design and operation of a specialised hybrid local-cloud database system, constructed specifically to handle the environmental data from sensor nodes. With the core aim of collecting and managing temperature and humidity data, this system utilises SQLite [39], an efficient, lightweight database engine for local data storage at the off-site base station of the WSN. The system is designed to retain three days' worth of data locally. Additionally, it transfers data older than three days to a cloud-based storage solution, Amazon Web Services (AWS) S3 bucket [40].

This hybrid local-cloud approach ensures that valuable data is preserved and accessible, regardless of the connectivity status at the nesting sites. Integral to the design of this system is a backend API. This feature allows users to interact with the data, allowing them to retrieve data within a specific time range, view statistical metrics, and generate plots for visual interpretation of the data.

The ensuing sections detail the requirements, specifications and acceptance test procedures (ATPs) for the database design, a section detailing the reasoning behind the choices made in the design process, an overview of the final design, and finally testing and results for the ATPs.

4.2 System Design

4.2.1 Requirements

Requirement ID	Description	Specification ID
R1	System should be able to collect temperature and humidity data from the WSN.	S1
R2	System should store up to 3 days of data locally.	S2
R3	System should transfer data older than 3 days to cloud storage.	S3
R4	System should provide an API for remote data access.	S4
R5	API should allow users to retrieve data within a specified range.	S5
R6	API should provide statistical metrics of the data.	S6
R7	API should provide data visualization features.	S7

Table 4.1: Requirements Table for Hybrid Local-Cloud Storage System

4.2.2 Specifications

Specification ID	Description	ATP ID
S1	Data Collection: Collects temperature and humidity data from WSN every 10 minutes.	ATP1
S2	Local Storage: Uses SQLite database on Raspberry Pi Zero for local storage.	ATP2
S3	Cloud Storage: Transfers data to AWS S3 bucket every 3 days.	ATP3
S4	API: REST API using FastAPI Python Framework	ATP4
S5	API Retrieval: Fetch selected variables within a specified time-range.	ATP5
S6	API Statistics: Provide minimum, maximum, mean, and standard deviation for selected variables within a specified time-range.	ATP6
S7	API Visualisation: Provide scatter plots and histograms for selected variables within a specified time-range.	ATP7

Table 4.2: Requirements Table for Hybrid Local-Cloud Storage System

4.2.3 Acceptance Test Procedures (ATPs)

ATP ID	Description	Expected Outcome
ATP1	Verify that the system collects temperature and humidity data.	System should successfully collect and store data from WSN.
ATP2	Verify that the system can store up to 3 days of data locally.	System should always contain the last 3 days of data on local storage.
ATP3	Verify that the system transfers data to AWS S3 after 3 days.	System data older than 3 days should appear in the AWS S3 bucket.
ATP4	Verify API follows a REST architecture.	API endpoints should always follow REST architecture principles.
ATP5	Verify that the API allows retrieval of data for selected variables within a specified time range.	API should always return all data for selected variables and time range.
ATP6	Verify that the API returns statistical metrics of data for selected variables within a specified time range.	API should always return the correct statistical metrics for selected variables and time range.
ATP7	Verify that the API returns visualisations of data for selected variables within a specified time range.	API should always return the correct visualisations for selected variables and time range.

Table 4.3: ATP Table for Hybrid Local-Cloud Storage System

4.3 Design Choices

The following section details the reasoning behind the various choices of architectures, frameworks, services and design implementations for the hybrid local-cloud storage system as outlined in the System Design section.

4.3.1 Hybrid Local-Cloud Storage System

A hybrid local-cloud storage system was chosen due to local storage limitations and to ensure data safety. Due to the limited local storage available on the Raspberry Pi Zero, storing three days' worth of data locally ensures that the system can function and collect data without overloading the system memory. After this period, data is transferred to an AWS S3 bucket [40], which technically provides unlimited storage space, ensuring that no data is lost due to local storage limitations.

Additionally, the AWS S3 bucket provides redundancy and data safety. In the event of hardware failure at the base station, the data stored on the cloud remains safe and accessible. Cloud storage also allows for easy scaling as the data volume grows over time.

4.3.2 SQLite Database

The choice of SQLite as the local storage solution was influenced by the local storage limitations. Firstly, the hardware employed at the base station, a Raspberry Pi Zero, has limited resources in terms of storage and processing power. SQLite, being a serverless, self-contained, and zero-configuration database engine, is an ideal fit for such a lightweight system [39]. It doesn't require a separate server process or system resources to operate, making it efficient for small devices like a Raspberry Pi Zero. Moreover, SQLite offers a full-featured SQL interface, providing a robust platform for managing and querying sensor data locally.

4.3.3 API

The integration of an API into the system is a critical component for enabling remote data access and interaction. Given that the raptor nesting sites are in remote locations without internet connectivity, it's impractical for researchers to manually retrieve the data on-site. The API provides a means for users to access the data conveniently and remotely, regardless of their location.

4.3.4 REST API Architecture

Representational State Transfer (REST) is a software design standard that defines the manner in which an API's endpoints are defined. Incorporating a RESTful API for the system enhances remote data accessibility and interaction. A Rest architecture leverages, scalability, simplicity, and independence [41]. The choice of a REST architecture introduces the possibility for the database to be implemented on multiple platforms in the future. FastAPI was chosen to implement the REST API for its execution speed, ease of use, robustness and built-in serialisation [42].

4.3.5 API Features

The API offers features that aim to improve the value and usability of the sensor data. A user can select which variables they would like to retrieve and a specific time range, ensuring researchers can extract data for a focused study. Statistical metrics and visualisations can be retrieved to provide insights into the trends in the data. These features aid in analysis and interpretation, enabling researchers to draw meaningful conclusions from the data.

4.4 Final Design

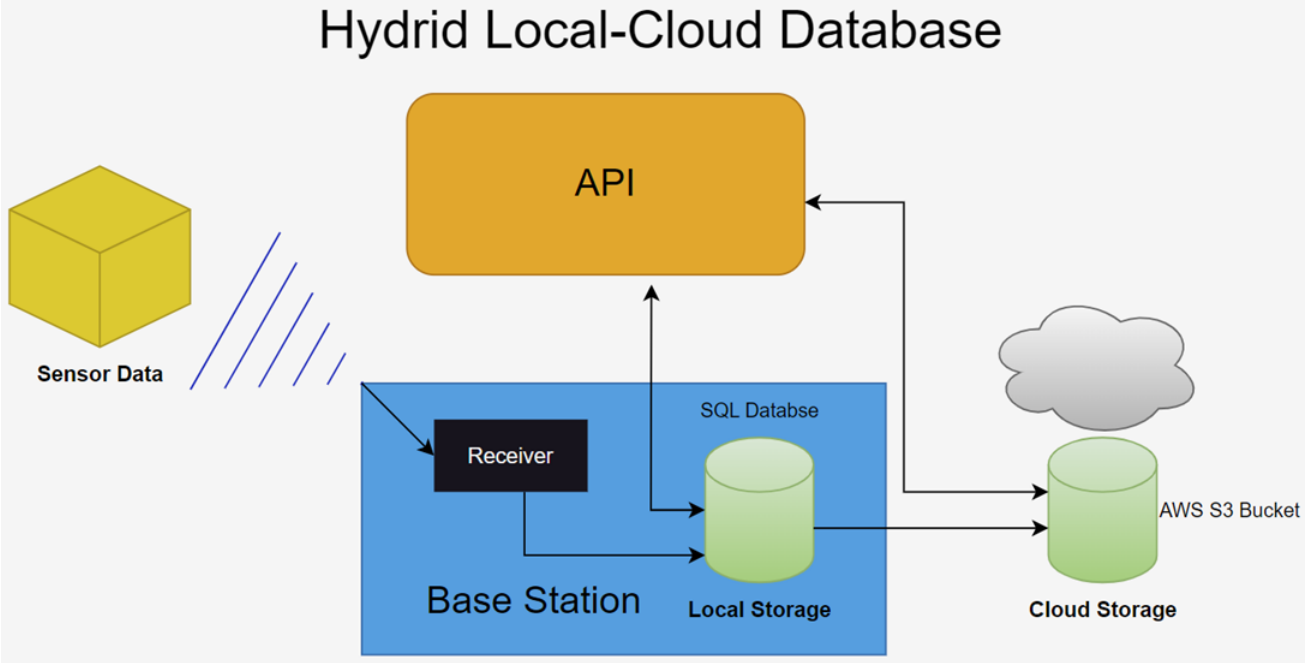


Figure 4.1: Final Hybrid Local-Cloud Database Design

Figure 6.9 depicts the final design for the database. Incoming sensor data is transmitted to the base station. To collect the incoming sensor data, a Cron job [43], which simply executes a script at a certain time interval, runs every 10 minutes. The script reads the data at the receiver and adds it to the local SQL database. Every 3 days another Cron job runs that executes a script to transfer the data in the local database to the AWS S3 bucket and clear the local database. The REST API makes queries to both the local and cloud database to fetch the sensor data. Depending on the user's preference the API will fetch certain variables within a specified time range. For any given range, the user may also choose to generate certain statistical metrics and data visualisations.

4.5 Testing and Results

4.5.1 ATP 1: Data collection

To verify that the sensor data is being added to the database every 10 minutes incoming data at the base station was monitored in the terminal for 30 minutes. After 30 minutes the number of entries in the local database was queried and the last 3 entries in the database were inspected. The expected outcome of this experiment is that the number of entries in the database increased by 3 and that the last 3 entries are equivalent to the 3 entries monitored in the terminal.

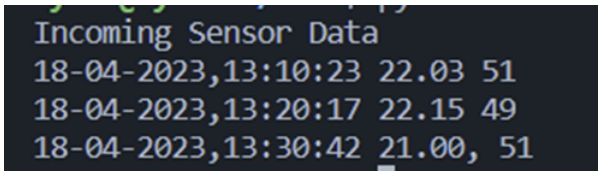


Figure 4.2: Terminal Output Showing Incoming Sensor Data Over 30 Minutes

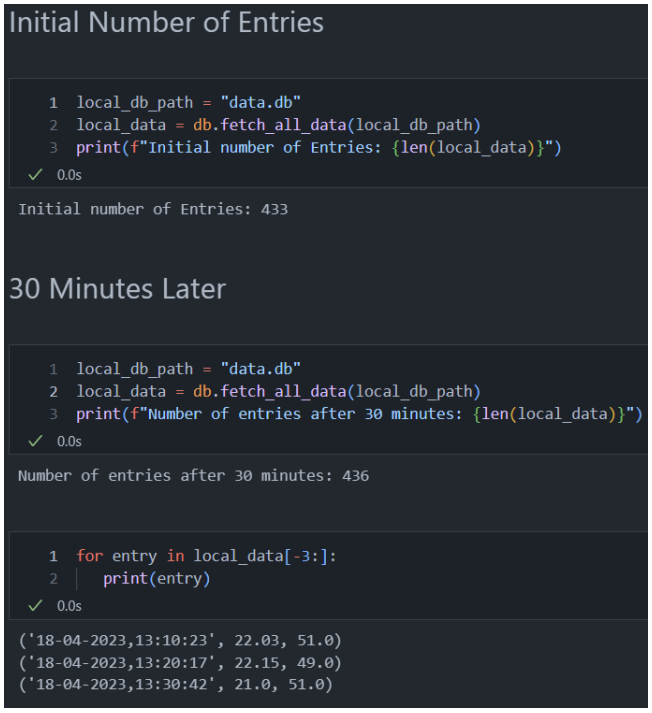


Figure 4.3: Test Output Showing State of Local Database Before and After 30 Minutes

Figure 4.2 depicts the terminal output on the base station showing incoming sensor data. Figure 4.3 shows the results from checking the number of entries and the last 3 entries in the database before and after 30 minutes. These results confirm that the number of entries in the database increased from 433 to 436 after 30 minutes. Additionally, the last 3 entries in the database are equal to the the 3 entries monitored on the base station terminal. These results confirm that the database can collect and store temperature and humidity data from the sensor network.

4.5.2 ATP 2: Local Storage

To verify that the local database always contains the last 3 days of data, a tester function was created that queries the database and returns the oldest and most recent dates of recordings contained in the local database. The test was executed on 18-05-2023. The expected output of this experiment is that the tester function should return the oldest date as 16-05-2023 and the most recent date as 18-05-2023.

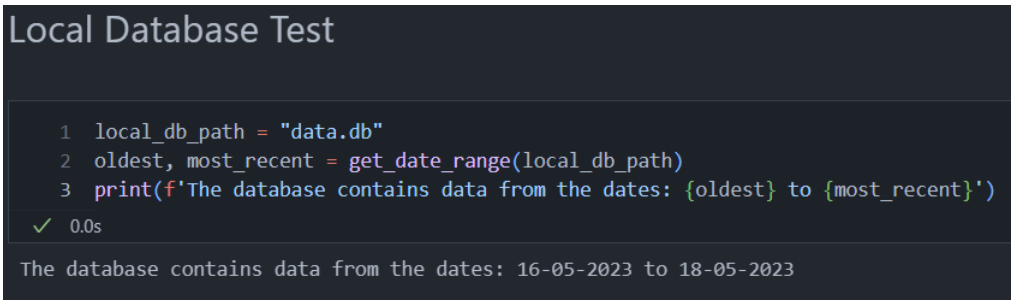


Figure 4.4: Output of Tester Function To Verify the Local Database Only Stores the Most Recent 3 Days of Data

Figure 4.4 depicts the output of the tester function that returns the oldest and most recent dates of data contained in the local database. The output shows the oldest date to be 16-05-2023 and the most recent date as 18-05-2023. These results are as expected and thus verify that the local database always contains only the last 3 days of sensor data.

4.5.3 ATP 3: Cloud Storage

To verify that data older than 3 days always appears in the AWS S3 bucket, a tester function was created that queries the most recent date of data in the cloud database. The most recent entry date was first queried on 16-05-2023 and then again on 19-05-2023. The expected output of this experiment should show the most recent entry to always be 3 days prior to the date on which the query was made. Meaning on 16-05-2023 the output should be 13-05-2023 and on 19-05-2023 the output should be 16-05-2023.

Figure 4.5: Output of Tester Function To Verify Data Older Than 3 Days Appears in the AWS S3 Bucket

Figure 4.5 shows the output of querying the most recent data in the cloud database on two dates separated by 3 days. The output on 16-05-2023 was 13-05-2023 and the output on 19-05-2023 was

16-05-2023. These results are as expected, confirming that data older than 3 days is always transferred to the AWS S3 bucket.

4.5.4 ATP 4: Rest Architecture

All of the API features simply query the database and return representations of the data. According to the REST architecture principles, an API request that retrieves and returns representations of data from a database but does not update or change any resources must be formatted as a GET request [41]. Therefore, to ensure the API follows a REST architecture all of the endpoints of the API, in its current state, must be GET requests. This has been executed in the design to verify the API follows a REST architecture. Figure 4.6 shows an example of one of the GET API endpoints used to fetch all of the data from the local and cloud databases.

```python
@app.get('/fetch_all')
def fetch_all():
    all_data = utils.get_all_data(s3_db_path, local_db_path)
    df = pd.DataFrame(all_data, columns=headers)
    df.to_csv('data.csv', index=False)
    return FileResponse('data.csv', media_type='text/csv', filename='data.csv')
```

Figure 4.6: API GET Endpoint to Fetch All Data

4.5.5 ATP 5: Data Retrieval

To verify that the API can fetch data from the database within a specified time range the database was manually queried between the timestamps 09-05-2023 at 10 am and 12-05-2023 at 6 pm and then stored in an array. Next, the same dates were queried using the API request to fetch a range of data from the database and stored it in an array. The expected output of this experiment is that the two arrays will contain the same data and thus be equal to each other.

```sql
SELECT *
FROM temperature_data
WHERE timestamp >= '2023-05-09 10:00:00'
AND timestamp <= '2023-05-12 18:00:00';
```

Figure 4.7: SQL Code For Manual Query

```python
@app.get('/fetch_range')
def fetch_range(start: str, end: str):
    data = utils.get_all_data(s3_db_path, local_db_path)
    data = utils.get_data_range(start, end, data)
    df = pd.DataFrame(data, columns=headers)
    df.to_csv('data_range.csv', index=False)
    return FileResponse('data_range.csv', media_type='text/csv', filename='data_range.csv')
```

Figure 4.8: API Endpoint to Fetch Range of Data

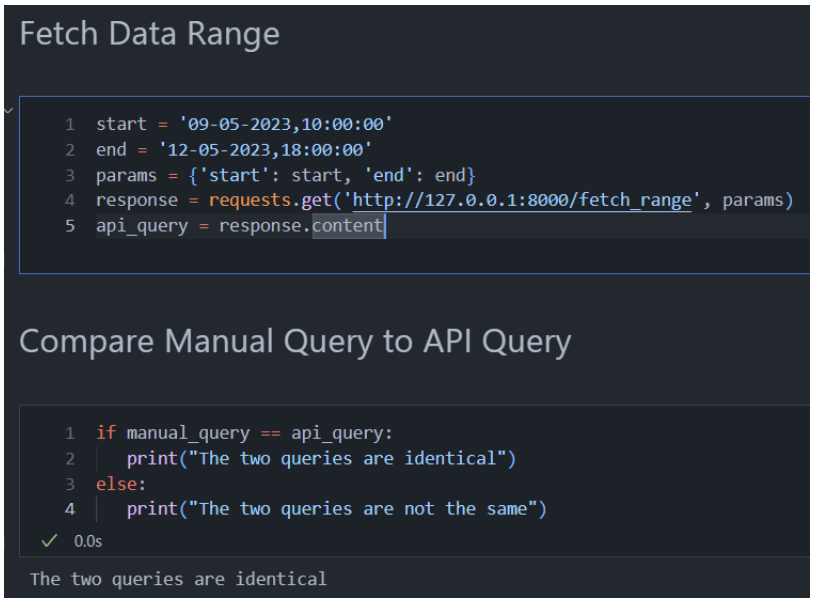


Figure 4.9: API Request to Fetch Ranged Data and Test Output Comparing API and Manual Query

Figure 4.7 shows the SQL to fetch the data manually, and figure 4.8 shows the API endpoint to fetch data within a specified time range. Figure 4.9 shows the results from making a request to the API endpoint and comparing the data to the manual query. These results show that the data contained in the manual query is equivalent to that produced by the API endpoint, verifying ATP 5.

4.5.6 ATP 6: Statistical Metrics

The same manual and API queries from ATP5 were used to verify that the API produces the correct statistical metrics for a specified range of data. The mean, median, maximum, minimum and standard deviation were calculated for the manual query and compared to those produced by submitting a request to the statistical metrics endpoint of the API. The expected output of this experiment is that the statistical metrics for both the manual query and those produced by the API request should be identical.

	Manual Query	**API Query**
Maximum Temp	30.96	30.96
Minimum Temp	13.25	13.25
Mean Temp	22.01	22.01
Median Temp	21.51	21.51
Standard Deviation Temp	5.70	5.70

Table 4.4: Table Comparing Manually Calculated Statistical Metrics and Those Returned from the API

Table 4.4 shows a comparison between the manually calculated statistical metrics and those returned by the API endpoint. As is depicted in the table the statistics are the same for both queries. These

results are as expected, verifying that the API can return the correct statistical metrics for data within a specified time range.

4.5.7 ATP 7: Data Visualisation

As for ATP 6, the same manual and API queries from ATP5 were used to verify that the API produces the correct data visualisations for a specified range of data. A scatter plot of temperature vs time was generated from the manual query and compared to the plot produced by the scatter plot endpoint of the API. The expected output of this experiment is that the plots from the manual query and the API endpoint are identical.

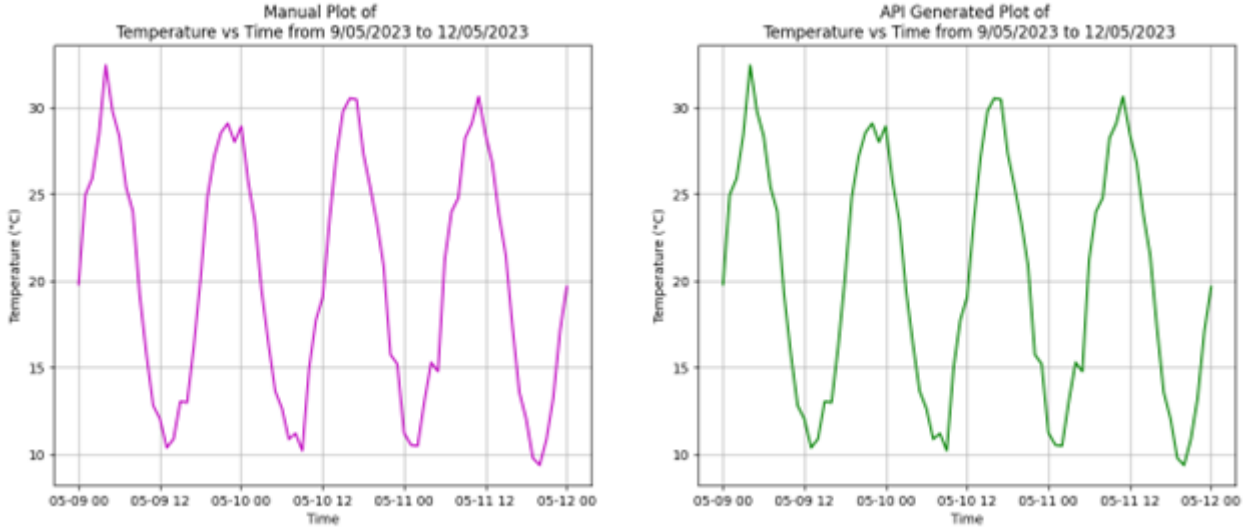


Figure 4.10: Comparison Between Manually Generated Visualisation and the Visualisation Returned from the API

Figure 4.10 shows a comparison between the plot generated from the manual query and the plot returned by the API endpoint. The two plots appear to be identical thus verifying that the API can correctly return a visualisation of data within a specified time range.

4.6 Conclusion

The implementation and testing of this system have solidified the effectiveness of the overall system design. The use of a lightweight SQLite database on the Raspberry Pi Zero has proven to be a resource-efficient solution, successfully collecting and storing sensor data while operating within the constraints of the hardware. The transfer of data older than three days to an AWS S3 bucket manages local storage space effectively and secures long-term data safety and redundancy.

The RESTful API, with its features such as selective data retrieval, statistical metrics, and data visualization, amplifies the accessibility, interpretability, and usability of the data. Its adherence to the REST design principles lays a solid foundation for future scalability, offering the ability to cater to increasing data and user demands without compromising system performance.

Overall, the testing conducted on the system reaffirms the intended design. Each test confirmed the system's alignment with the defined requirements and specifications.

Chapter 5

Reliable Sensor Node (SN) Data Acquisition & Transmission (MCBTIN001)

5.1 Introduction

In the context of the wireless sensor network implementation , the ability to collect and transmit environmental data at small spatial scales (i.e territory sizes) which are relevant to the raptors nesting sites is crucial for effective monitoring and research in the field of Wild African raptor conservation. The acquisition and analysis of such data provides the researchers with crucial information required to analyze the behaviour and health of the raptors.

This chapter focuses on the design and implementation of the sensor node data acquisition and transmission system, which serves as the primary access point for sensor data for the hybrid local-cloud database and the rendezvous node. This system is responsible for collecting data at predetermined intervals from the temperature and humidity sensor and streaming it wirelessly to the rendezvous node, which acts as central hub for data aggregation and communication. Additionally, the system takes into consideration factors such data accuracy and reliability, efficient transmission and compatibility with the overall system architecture.

The system is designed to collect data every 2 seconds from the sensor and transmit it every 10 minutes when the sensor node is connected to the rendezvous node over wireless connection such as WiFi. The use of the ESP32 PICO as the base station allows the system to utilize low-power optimization for 10 minutes during idle periods. This feature allows the system to have a longer lasting battery life.

To fulfill the requirements of this subsystem, the following sections explore the requirements, specification, Acceptable Test Protocols (ATP's) and specific design choices which led to the prototype design implementation. Finally, the chapter provides the testing and results of the system.

5.2 System Design

5.2.1 Context for Design

The sensor nodes play an important role in the overall system because they act as an access point for sensor data needed by the hybrid local-cloud database and the front-end subsystem. The design must consider factors such as sensor data accuracy, efficient transmission and compatibility with the overall system architecture. This subsystem comprises components that enable the system to meet both the user and functional requirements.

The sensor nodes serve as a primary means of gathering real-time environmental data in Wild African Raptor nesting sites, specifically temperature and humidity levels. It must also ensure reliable data acquisition at regular intervals for continuous monitoring of environmental changes. The gathered data must be wirelessly streamed to the rendezvous node for local and cloud storage and to be able to be displayed on the User Interface.

5.2.2 Requirements

Table 5.1: Requirements Table for Sensor node Data Acquisition and Transmission

Requirement ID	Description	Specification ID
R1	The subsystem should be able to collect temperature and humidity readings from the sensor.	S1
R2	The subsystem must provide real-time monitoring of sensor data.	S2
R3	The system must be able to communicate with the rendezvous node to transmit sensor data.	S3
R4	The system must have low-power mode optimisation for long-term usage.	S4
R5	The system must be easily integrated onto the WSN as a whole.	S5
R6	The system must ensure reliable and accurate data acquisition without any inherent biases or errors.	S6

5.2.3 Specifications

Table 5.2: Specifications Table for Sensor node Data Acquisition and Transmission System

Specification ID	Description	ATP ID
S1	The sensor nodes successfully collects temperature and humidity data every 2 seconds from the sensor.	ATP1
S2	The subsystem uses DS3231 RTC to provide accurate timely updates of sensor data.	ATP2
S3	System has built-in WiFi capabilities for wireless data transmission.	ATP3

S4	System utilizes deep sleep mode on ESP32 PICO D4 for 10 minutes after transmitting data every 10 minutes to the rendezvous node.	ATP4
S5	The system is easily integrated with other subsystems of the WSN.	ATP5
S6	The system create range checking methods to validate sensor data.	ATP6

5.2.4 Acceptance Test Procedures (ATPs)

Table 5.3: ATP Table for Sensor node Data Acquisition and Transmission System

ATP ID	Description	Expected Outcome
ATP1	Verify temperature and humidity data acquisition from the sensor.	System should successfully collects data from the sensor every 2 seconds .
ATP2	Verify that system can provide real time temperature and humidity monitoring.	System should successfully transmit sensor data with accurate timestamps to the rendezvous node every 10 minutes.
ATP3	Verify the subsystem can establish a stable wireless connection with the rendezvous node.	The sensor node should transmit data over WiFi to the rendezvous node.
ATP4	Verify the system goes into sleep mode for 10 minutes by comparing power consumption in different mode of operations.	The system should consume less power when not acquiring or transmitting data.
ATP5	Verify the system is easily integrated into the whole system on our WSN.	System must ensure seamless data transfer and compatibility with other nodes and power supply.
ATP6	Verify that the correctness and reliability of the collected data.	System should acquire data close to the reference measurements values or accurate sensors.

5.3 Design Choices

In the design of the SN data acquisition and transmission subsystem, several design choices were made to ensure reliable data acquisition and transmission of temperature and humidity. These choices are outlined below which include component selection, data acquisition techniques, Data validation and calibration techniques and low-power mode strategies.

5.3.1 Component Selection

Selecting the right components for the sensor nodes is a vital part of the design process which allow for ease of integration, compatibility with the overall system and cost-effectiveness.

Sensor Nodes Microntroller

Board	Version	Processor	Memory	WiFi	Deep sleep mode	Supported Integrated Development Environment (IDE)	Cost (Rands)
Arduino Uno	R3	ATmega328P 16 MHz	2KB SRAM, 32KB FLASH, 1KB EEPROM	none	No	Arduino IDE	R250@RoboFactorySA
STM32	F051R8	48MHz ARM Cortex-MO	6 to 64 KB FLASH 8 KB SRAM	none	Yes	STM32Cube	R173.22@Digi-Key SA
ESP32	PICO-D4	240MHz Tensilica Xtensa	448 KB SRAM 4MB FLASH	802.11 b/g/n	Yes	Arduino IDE ESP-IDF	R181.17@Electromaker
NodeMCU-32S	V3	LX6 240MHz Tensilica Xtensa LX6	512 KB SRAM	802.11b/g/n	Yes	Arduino IDE	R139@Eiferer

Table 5.4: Comparison of microcontroller board types for the sensor nodes

The microcontroller handling the data acquisition and transmission of temperature and humidity in the sensor nodes was carefully chosen based on specifications listed in Table 5.4. The subsystem requires data to be streamed wirelessly to the rendezvous node which eliminates the Arduino Uno R3 and the STM32 boards. Keeping in mind that the sensor nodes will be placed at remote nesting sites where there is limited internet access, but due to budget constraints the sensor nodes must have a local base station with built-in WiFi capabilities to transmit data.

This leaves the NodeMCU-32S and the ESP32-PICO-D4 which have similar capabilities. Both boards are based on the ESP-32 System-On-Chip (SoC) and they offer built-in WiFi connectivity which allow for seamless integration with wireless networks. However the ESP32-PICO-D4 is much smaller in size compared to the NodeMCU-32S allowing for a more compact design. The ESP32 PICO has a dual-core processor which allows for higher processing of sensor data. It also has advanced low-power modes which can extend the battery life of the sensor nodes. Even though the NodeMCU-32S is R49 cheaper, we have chosen this board as the local base station for data acquisition.

Chosen board: ESP32-PICO-D4

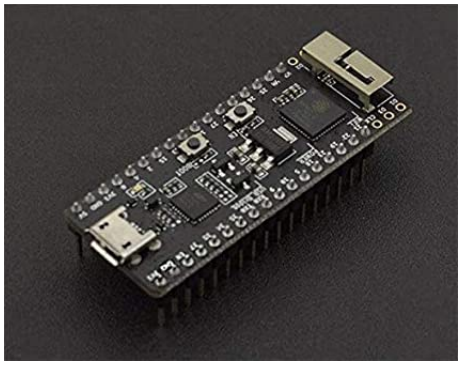


Figure 5.1: ESP32-PICO-D4 Board.

Temperature and Humidity Sensor

Sensor Type	Supply Voltage (Typ.)	Temperature range	Humidity range	Accuracy	Communication Protocol	Long term stability
DHT11 Temperature and humidity Sensor Module	5V	0 to 50 ℃	20 to 90%	±2ºC ±5RH	one-wire	±1%RH/year
DHT22 Temperature and Humidity Sensor Module	5V	-40 to 80ºC	0 to 100%	±0.5ºC ±2%RH	One-wire	±0.5%RH/year
BME280 Temperature and humidity sensor module	5V	-40 to 85ºC	0 to 100%	±0.5ºC ±2%RH	I2C SPI	±0.5%RH/year

Table 5.5: Comparison of temperature and humidity sensor modules for data acquisition

All the above sensor modules shown in Table 5.5 meet the requirements of the sensor node data acquisition and transmission subsystem. However the main objective of our wireless sensor network was to create a cost-effective solution but still able to achieve the user and functional requirements. Both the DHT22 and BME280 have similar specifications and a much better performance compared to the DHT11 but they are very expensive.

For the sensor nodes, we have chosen the most affordable option considering the budget constraints of the project. The DHT11 sensor module can achieve accurate temperature and humidity measurements and can be easily integrated into the system due to its one-wire communication protocol. The temperature range of 0 to 50℃ is good enough for monitoring the Wild African Raptor nesting sites conditions. The sensor does not require any external hardware for operation such an analog-to-digital (ADC) converter because it can be interfaced directly to a digital I/O pin. The other reason for selecting this module was that it is always in-stock. It is also widely used making it easy to find ample documentation, sample code and support on how to integrate it into IoT projects.

Chosen Sensor Module: DHT11 Temperature and Humidity Sensor Module

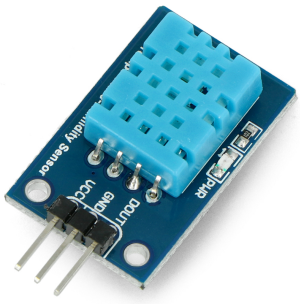


Figure 5.2: DHT11 Temperature and Humidity Sensor Module.

Real-Time Clock (RTC)

The DS3231 RTC module was chosen to ensure real-time monitoring of temperature and humidity by providing accurate timestamps of sensor data. This module was chosen because of its ease of integration with the ESP32-PICO board as it makes use of I2C interface. It has features such as high accuracy, battery back-up allowing it to maintain accurate timestamps when the power supply is removed. Since the sensor node utilizes the sleep mode when not transmitting data, the RTC can trigger the system to wake-up and sleep which saves battery life of the nodes.

5.3.2 Data Acquisition techniques

To implement the data acquisition process we have chosen the arduino programming language to develop firmware for the ESP32 PICO D4. This was chosen for its simplicity and utilization of ESP32 functions and libraries to interface with the DHT11 temperature and humidity sensor. To ensure continuous data collection the microcontroller periodically trigger the sensor and waits for the sensor to respond with the temperature and humidity measurements.The acquired data is streamed wirelessly to the rendezvous node which can allow multiple sensor nodes deployed at different nesting sites to transmit data as packets on our WSN. However for the real implementation, long-range transmission will be achieved using radio frequency (RF) communication.

5.3.3 Data Validation techniques

To ensure reliability and correctness of the collected data, we have chosen the range checking algorithm. Range checking is a technique used for data validation by making sure that the measured sensor data falls within the range or predefined thresholds specified in the datasheets. Range checks allows us to identify out of range data points. Through this algorithm we can maintain data integrity and accurate data transmission.

5.3.4 Low-power mode strategies

To ensure power optimisation of the sensor nodes during idle periods, we have utilized the low-power sleep mode supported by the ESP32-PICO-D4. This technique was chosen because it effectively reduces the total power consumption of the SN when it's not actively acquiring or transmitting data. This was achieved by programming the microcontroller in the Arduino IDE to wake up or sleep periodically in response to trigger events provided by the RTC module.

5.4 Prototype Design

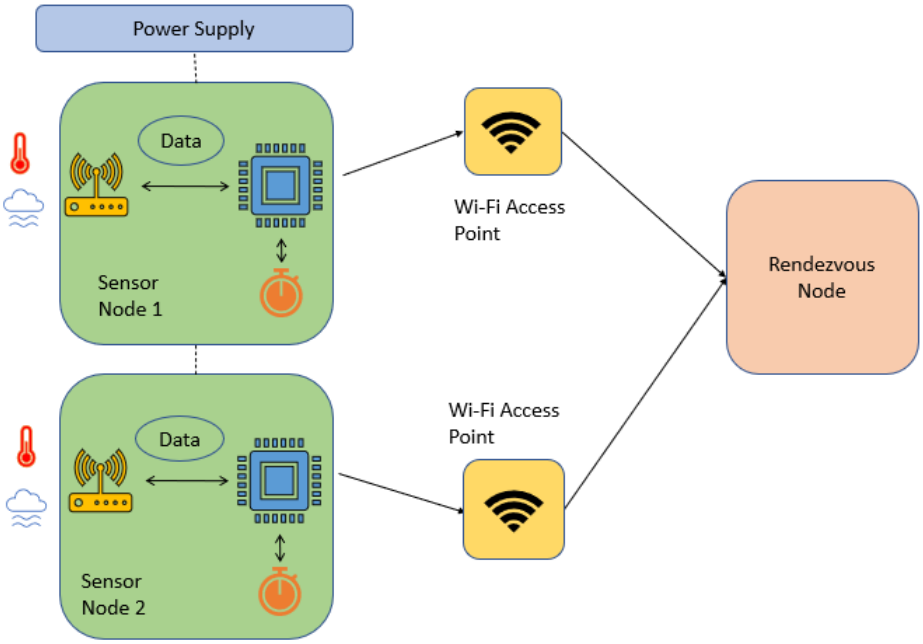


Figure 5.3: Prototype Design of the Sensor Node Data Acquisition and Transmission System

The prototype design of the sensor node data acquisition and transmission subsystem is shown in Figure 5.3. This design consists of a compact sensor node that integrates the ESP32 PICO, DHT11 Sensor and the DS3231 RTC. The ESP32 PICO is used as the SN base station for handling data acquisition, and wireless transmission with the rendezvous node. The DHT11 sensor was securely connected to the ESP32's GPIO pins, allowing it to measure timestamped temperature and humidity accurately with the use of the RTC.

The prototype design featured a small form factor, making it suitable for deployment in different nesting sites. To ensure portable and uninterrupted operation, the sensor node was powered by a battery. The design utilizes a wide range of features such as data validation and low-power modes techniques that allow for efficient and accurate real-time data collection and transfer.

This arrangement allows data to be sent immediately when connected to the rendezvous node Wi-Fi AP due to memory size limitations. The design is capable of collecting one reading per second (1Hz sampling rate). Through the data validation techniques, the system performs data smoothing, filtering and range checking to minimize sensor noise or fluctuations in the transmitted data. The system is able to send a string of data every 10 minutes to the rendezvous node upon request.

5.5 Testing and Results

5.5.1 Setup for Testing

Figure 5.4 shows the setup for the sensing system. The system uses an ESP32-PICO for temperature and humidity data acquisition from the connected DHT11 sensor module. This microcontroller also

serves as an access point for sensor data to be transmitted wirelessly to the rendezvous node. The DS3231 RTC is connected to the system to ensure real-time data collection and transmission. A series of tests were conducted to evaluated the performance and reliability of the SN data acquisition and transmission subsystem. The firmware development for testing was implemented in Arduino IDE.

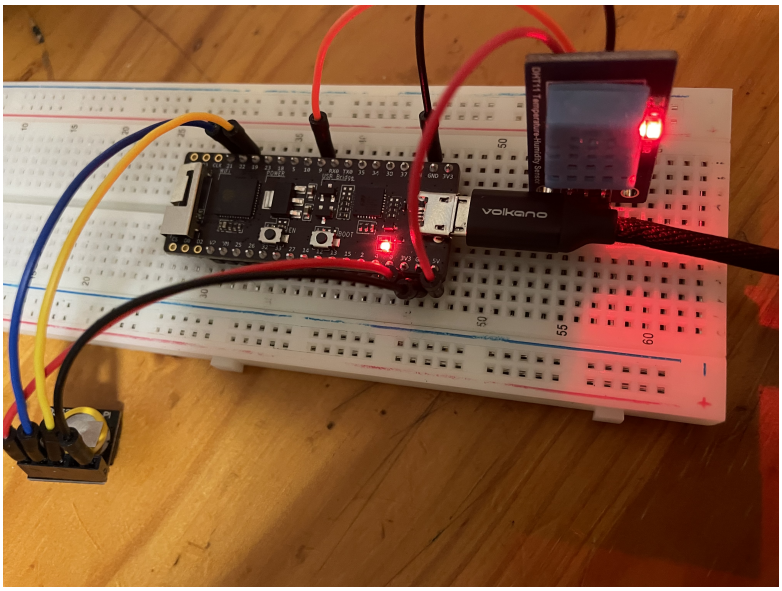


Figure 5.4: Sensor Node setup for Testing

5.5.2 ATP 1: Data acquisition

To verify that the system is able to successfully read and collect temperature and humidity data every 2 seconds. The ESP32 PICO sends a read command to the sensor and wait for it to respond with the sensor data readings. Figure 5.5 shows the code that ensures that the data collected does not have errors. This ensures that the DHT11 sensor is properly connected to the system.

Figure 5.6 shows the terminal output with successfully acquired temperature and humidity data on the sensor node base station for every 2 seconds when the request is made to read the DHT11 sensor. These results confirms that the subsystem is able to monitor and collect sensor data continuously without any inconsistencies in the readings.

```
float temperature = dht.readTemperature();
float humidity = dht.readHumidity();
if (isnan(temperature) || isnan(humidity)) {
  Serial.println("Failed to read data from DHT sensor!");
  return;
}
else{
  Serial.println("Successfully read DHT11 sensor!");
}
```

Figure 5.5: C++ code for Data Acquisition and Connection Error Checking

```
Successfully read DHT11 sensor!
21-5-2023,14:50:17 23.00 46.00
Successfully read DHT11 sensor!
21-5-2023,14:50:19 23.00 46.00
Successfully read DHT11 sensor!
21-5-2023,14:50:21 23.00 46.00
```

Figure 5.6: Terminal Output for Data Acquisition

5.5.3 ATP 2: Real-time Data Transmission

The verify that the system is able to transmit data with accurate timestamps to the rendezvous node, a tester code was implemented and Figure 5.7 shows that data is successfully transmitted to the node. This also verifies ATP 3 which requires the sensor node to have a stable wireless connection for data transfer. The expected outcome is the sensor node must transmit data to the rendezvous node every 10 minutes during the normal mode of operation. The terminal output shows that this was successfully achieved.

```
Data transmission successful!
21-5-2023,15:39:12 23.00 40.00
Data transmission successful!
21-5-2023,15:49:14 23.00 37.00
```

Figure 5.7: Terminal Output to Verify wireless data transmission to the rendezvous node

5.5.4 ATP 3: Wireless connection reliability for data transmission

This was refined by ATP 2. Figure 5.7 above shows that a reliable WiFi connection was established between the sensor nodes and the rendezvous node to allow for seamless data transmission with the accurate timely updates after every 10 minutes. The achieved range of transmission was over 50 meters which can be easily extended by using long-range transmission systems but due to budget limitations, that was not implemented on our WSN.

5.5.5 ATP 4: Low-power mode optimisation

The ESP32 PICO utilises low-power sleep mode features that allow the system to consume less power by simply triggering it to sleep or wake up for a certain period using a timer as shown in Figure 5.8. To verify that the system goes to sleep after data transmission, the functions in Figure 5.8 have been executed in the design of the sensor nodes for ATP verification.

```
Serial.println("Data transmission successful!");
esp_sleep_enable_timer_wakeup(10 * 60 * 1000000); // Wake up every 10 minutes
(in microseconds)
// Put ESP32 into sleep mode
Serial.println("Going to sleep...");
esp_deep_sleep_start();
```

Figure 5.8: C++ code for sensor node deep sleep mode

5.5.6 ATP 5: System Integration

To verify that the system can be integrated with the other subsystems. The system was able to be powered by the power subsystem. The API utilised in the design of the hybrid local-cloud database system was able to fetch sensor data with accurate timestamps coming from the sensor nodes. The communication and User Interface (UI) was able to fetch data over WiFi from the sensor nodes as discussed on their ATP's for the systems.

5.5.7 ATP 6: Data Validation

A simple range checking algorithm was implemented to verify that the sensor node collect and transmit data that is within the expected range of the sensor and any data outside that range was considered invalid thus requiring further handling and investigation. However the DHT11 sensor was found to produce accurate and reliable data that was collected and sent to the rendezvous node. The data was also compared to reference measurements and it was found to be close to the expected values outputted on the terminal as mentioned in the datasheet. Therefore no additional calibrations were needed.

```
// Data Validation using Range Checking Algorithm
//Define the temperature and humidity range limits
TEMPERATURE_RANGE = (0, 50)  # in degrees Celsius
HUMIDITY_RANGE = (20, 90)  # in percentage

//Acquire temperature and humidity readings from the DHT11 sensor
temperature = get_temperature_reading()
humidity = get_humidity_reading()

//Perform range checking
if not (TEMPERATURE_RANGE[0] <= temperature <= TEMPERATURE_RANGE[1]):
    print("Temperature reading is outside the acceptable range.")

if not (HUMIDITY_RANGE[0] <= humidity <= HUMIDITY_RANGE[1]):
    print("Humidity reading is outside the acceptable range.")
```

Figure 5.9: Data Validation using Range Checking Method

5.6 Conclusion

Overall, the sensor node data acquisition and transmission system is designed to meet the requirements for collecting and transmitting temperature and humidity data. This is achieved by the use of the ESP32 PICO, DHT11 sensor and DS3231 RTC. This made the design to be compact, portable and suitable for deployment in different nesting sites. The system proved to provide an effective access point for sensor data and thus contributing to the overall success of the wireless sensor network implementation.

The subsystem was able to maintain data accuracy, reliability, efficient transmission, and data correctness due to the data validation techniques and low-power mode optimization features of the ESP32 PICO. The design of the subsystem involved several components and design choices. This ensured that the system is cost-effective while providing the required performance. However the system posed limitations

for long -range transmission of data > 50m to the rendezvous node due to its utilization of Wi-Fi built-in capabilities.

The system was tested to evaluate its performance and reliability, including verifying data acquisition, real-time monitoring, wireless communication and system integration. This confirmed the capabilities of the system to meet the overall requirements and specifications of the WSN.

Chapter 6

Power Supply (MLKCLE001)

6.1 Introduction

This power supply was selected since our system will be located in areas where access to the power grid is limited. This power supply is required to supply power to the sensor node and the rendezvous node with the power specifications of 1A@5V and 6mA@5V respectively. The power supply makes use of the solar panels that generates power to charge the battery. It has a power management module that regulates the power from the solar panels and that ensures the optimal charging of the battery, with LEDs indicating when the battery is charging, done charging, and warnings should there be anything wrong with the solar panel. The battery has a battery health monitor that notifies the user of the battery capacity status. This power supply has an adjustable power output since the two nodes that need to be powered have different power specifications.

6.2 System Design

6.2.1 Requirements

Requirement ID	**Description**	**Specification ID**
R1	The system should be able to supply power to the sensor node and rendezvous node.	S1
R2	The system should be able to recharge the battery at any given location selected.	S2
R3	The system should be able to monitor the battery health.	S3
R4	The system should be able to protect itself and the load from overvoltage and overcurrent.	S4

Table 6.1: Requirements Table for Power Supply

6.2.2 Specifications

Specification ID	Description	ATP ID
S1	An adjustable output of current and voltage with the range of 500mA to 1.2mA and 1.2V to 5V .	ATP1
S2	A mini solar panel to generates power to charge the battery.	ATP2
S3	A voltage divider circuit that uses micro controller to calculate and display the battery capacity status.	ATP3
S4	Protection circuitry for overvoltage and overcurrent.	ATP4

Table 6.2: Specifications Table for power supply

6.2.3 Acceptance Test Procedures (ATPs)

ATP ID	Description	Outcome
ATP1	Measure the output of the power supply system.	The system should show the expected voltage and current.
ATP2	Measure the output of the solar panel.	The solar panel should produce a voltage that can charge the battery.
ATP3	Measure the voltage across the battery..	The battery percentage should displace on the LCD.
ATP4	Increase in voltage and current from the source.	The circuit should be able to detect any overvoltage and overcurrent.

Table 6.3: ATP Table for Power Supply

6.3 Design Process

This section explains the process of selecting the suitable components that will meet the requirements. Different modules are compared as part of the selection process.

Power Source	Specifications	Use and Brief
Mini Solar Panels	9V,220mA	It is portable which makes it easy to use at any location. It is low-power solar which is safe for our system. Low cost.
Power Grid	AC from Eskom.	Sufficient power to supply any kind of system. High cost.
Battery	3.7V Lithium battery.	It is non-chargeable but replaceable after use. Cost more as you replace it every now and then.

Table 6.4: Power sources table

Comparing the three power sources above, the mini solar panels is more suitable for our power supply following reasons:

- Our system will placed at areas where there is limited access to power grid therefore the solar is much better than the power grid.
- Solar panel generates its own power unlike with non-recheable that needs to be replaced after use.

Module/Component	Specification	Use and Brief
Power management module	• 6V to 24V solar panel • 3.7V Li rechargeable battery • Provide 5V/1A output.	It regulates the power from the solar panel.
Charge controller	• 6-60V • Max output: 30A .	It controls the charging of the battery.

Table 6.5: Battery charging components

Comparing the two modules above, the power management module is more suitable than the charge controller. As much as both are affordable, the power management module gives more than the charge controller. The power management module is not only able to charge the battery but also regulates the power, it has a protection circuit which is the critical part of the whole power supply sub-module.

An adjustable buck converter is more suitable for use than a fixed buck converter. As two nodes are expected to be powered, designing the power supply that has adjustable output is more convenient.

6.4 Final Design

This section covers the choices taken to complete the design that will meet the design requirements. The final design will consist of the following components/modules:

1. Mini Solar Panel
2. Rechargeable 3.7V Lithium Ion Battery
3. Power Management Module

4. Adjustable Buck Converter
5. Battery health monitor

6.4.1 Mini Solar Panel

This mini solar panel is 9V 220mA that is able to generate power to charge the battery. In this design, it is connected to the power management module which in turn regulates this power from the solar panel. The solar panel is left connected at all times.

Figure 6.1: Mini solar panel

6.4.2 Rechargeable 3.7V Lithium Ion Battery

The battery that has been chosen for the design is the 3.7V Lithium ion rechargeable battery. This battery is able to carry a maximum of 4.5V to the minimum of 2.1V.

Figure 6.2: Rechargeable battery

6.4.3 Power Management Module

The power management module was selected as the suitable charging component. It plays an important role in the entire power supply sub-system. It takes power from the solar panel and regulates it. It then charges the battery to the suitable level. It has LEDs that indicates when the battery is charging, done charging and any warnings should there be anything wrong with the solar panel. It has the protection circuitry for any over voltage, over current.

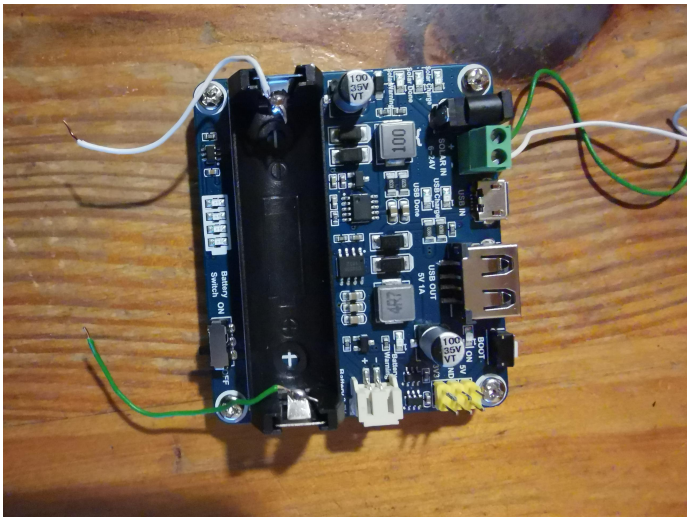


Figure 6.3: Power management module

6.4.4 Adjustable Buck Converter

An adjustable buck converter was selected for this design. The power supply is expected to supply different nodes with different power specifications. Therefore it is suitable to use an adjustable buck converter that takes the output from the power management module and converts it to a suitable power specification.

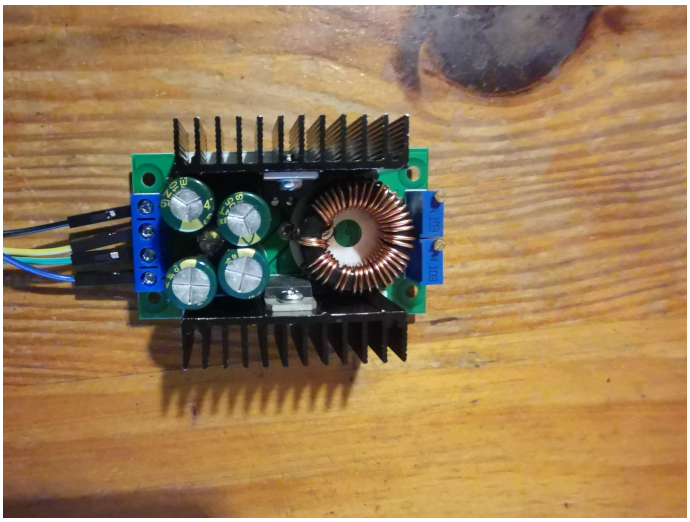


Figure 6.4: Adjustable buck converter

6.4.5 Battery health monitor

The battery health is designed in order to keep track of the battery level. A voltage divider circuit was built with the microcontroller. It displays the percentage of the battery capacity. Due to the budget constraints we were not able to add the temperature sensor of the battery on our battery health monitor.

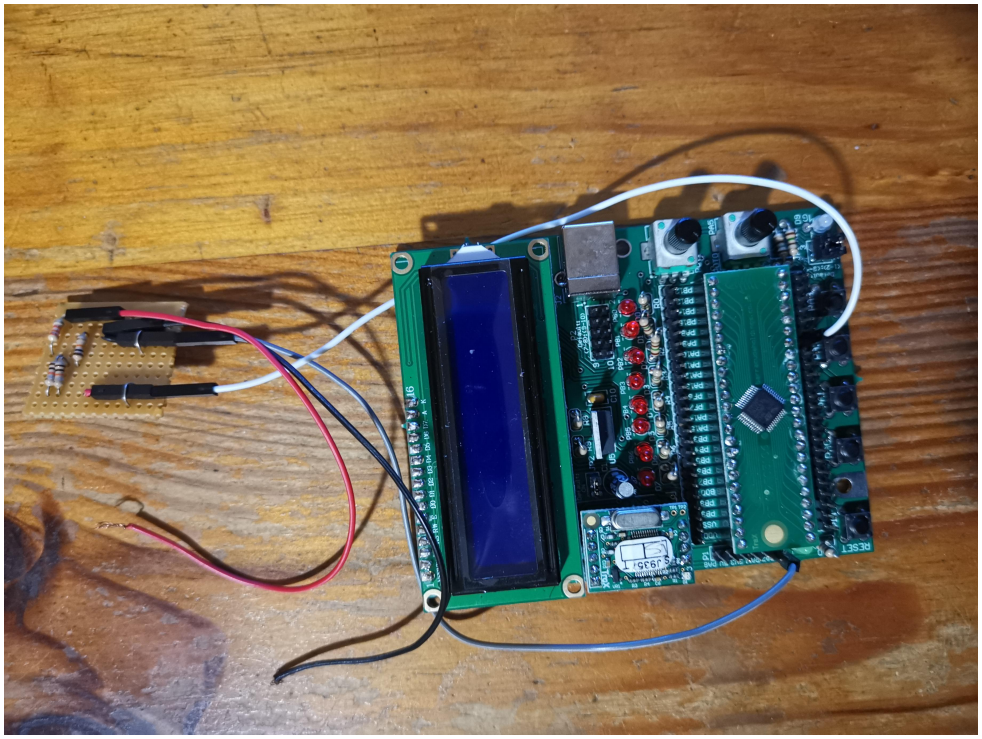


Figure 6.5: Battery health monitor

The figure below shows a prototype of the final design of the power supply sub-system. The solar generates power that is transferred to the power management module that in turn regulates the power

to charge the battery. The battery store energy for later use. The battery is connected to the battery health monitor that determines the capacity percentage and displays it. The buck converter takes the power from the power management module and steps it down to the desired values according to our nodes.

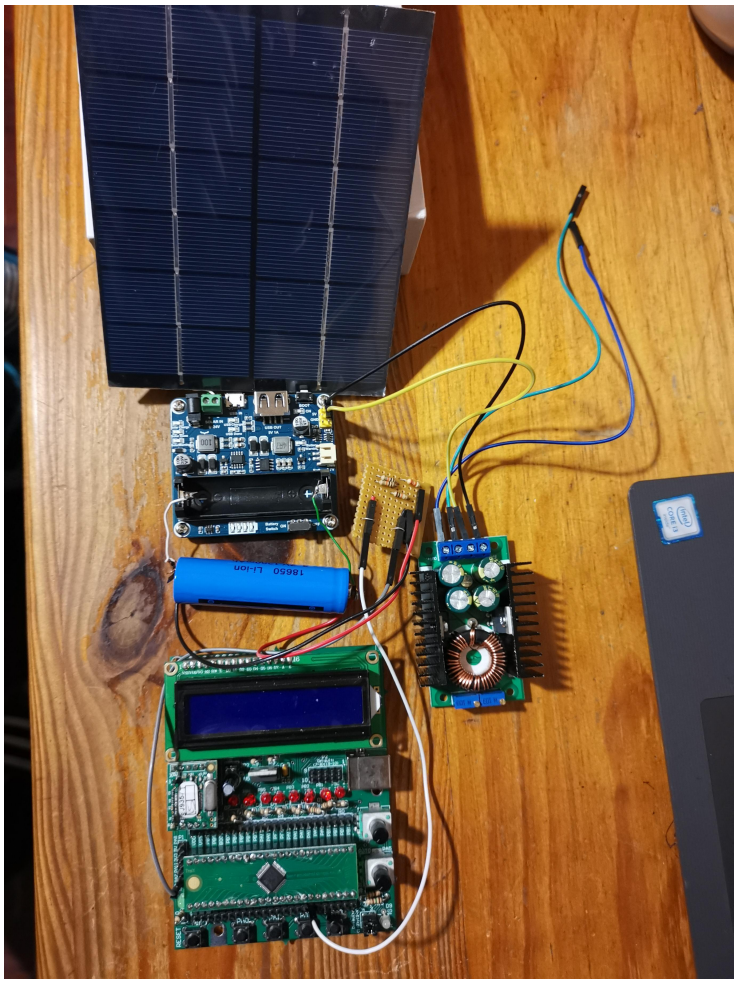


Figure 6.6: Final Design

6.5 Testing and Results

This section covers the testing and results of the prototype.

The aim was to ensure that our power supply is able to meet the specifications of the sensor node which are 5V and 1A. Also to ensure that it is indeed an adjustable power supply. After several test of the system, the figures below shows the results of the output voltage and current.

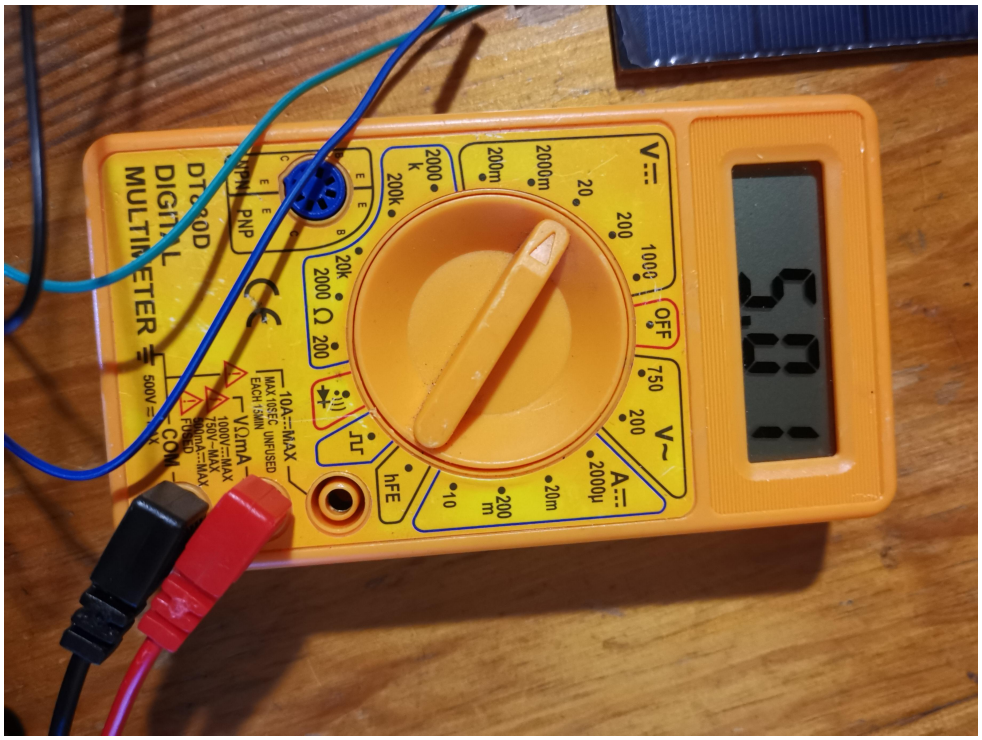


Figure 6.7: Voltage output

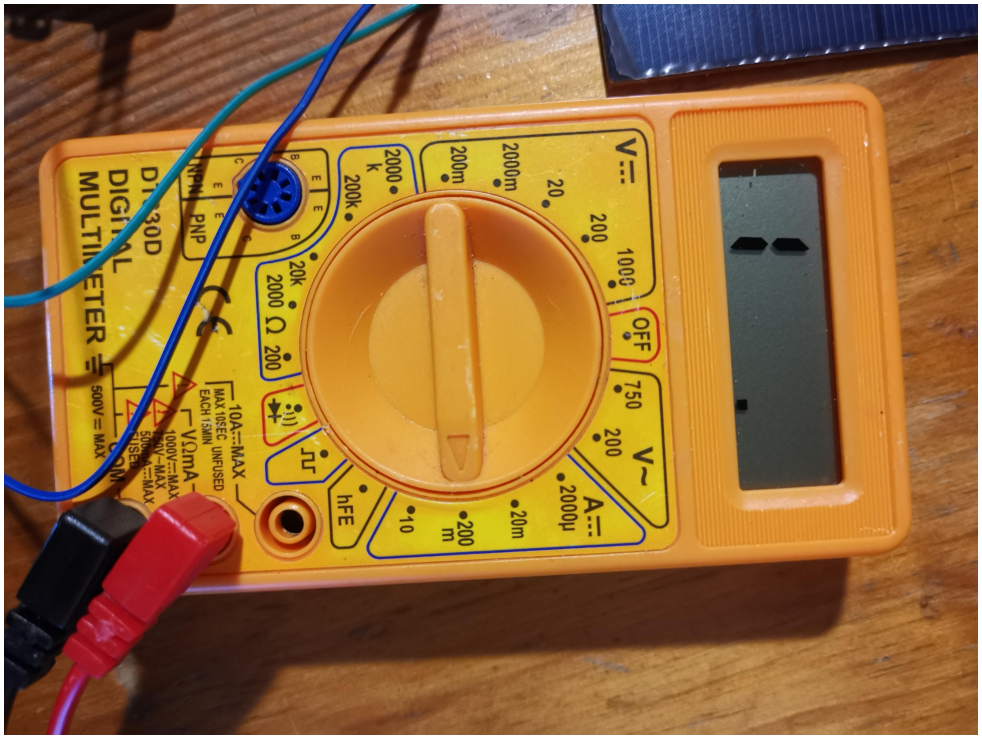


Figure 6.8: Current output

The following figure shows the results of the battery health monitor that displays the percentage of the battery capacity.

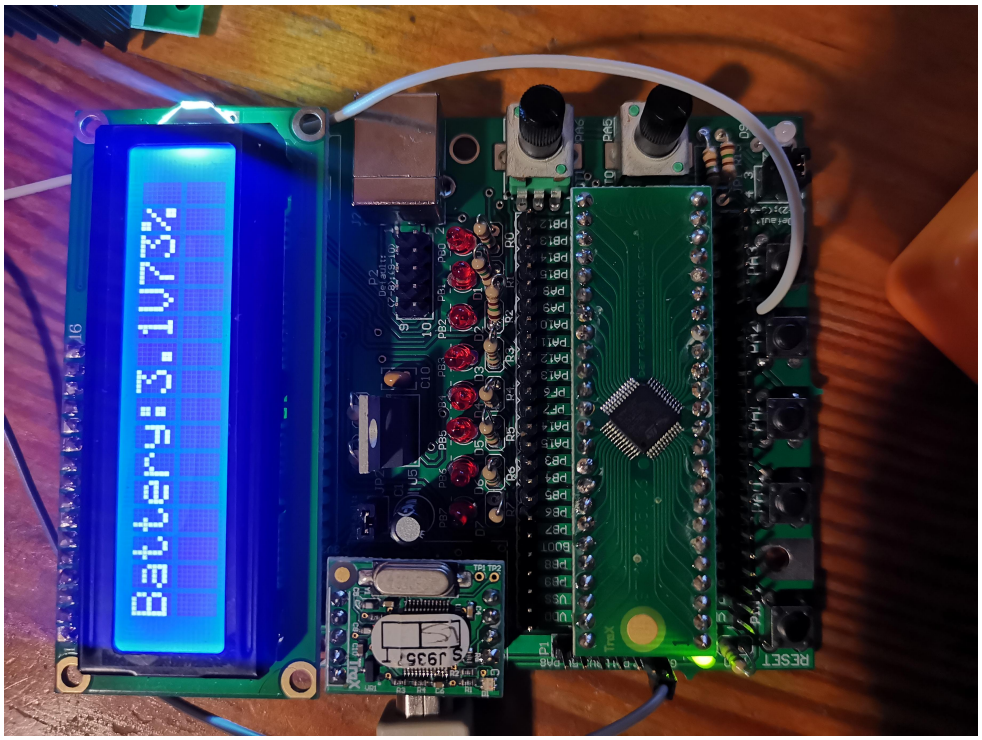


Figure 6.9: Battery health monitor results

6.6 Conclusion

In conclusion, a power supply system that incorporates a solar panel, rechargeable battery, power management module, buck converter, and battery health monitor is an excellent alternative to traditional sources of energy. This system provides a reliable and sustainable source of power for our wireless sensor network. The solar panel converts sunlight into electrical energy, which is then stored in the battery. The solar power management module ensures that the battery charges optimally to extend its lifespan. The buck converter regulates the voltage, making it suitable for nodes of our system, while the battery health monitor tracks the battery's performance and alerts the user if any issues arise. All these components work together to provide a smooth and efficient power supply system that is environmentally friendly and cost-effective in the long run.

Chapter 7

Conclusions

This research paper has presented the design and implementation of a wireless sensor network prototype intended for monitoring environmental conditions at wild African raptor nesting sites. Leveraging a system of sensor nodes, solar power supplies, a local rendezvous node, and a remote base station equipped with a database, API and intuitive user interface, our prototype provides a hopeful proof of concept as a technology to aid researchers in understanding and protecting wild African raptors.

Our prototype has proven to effectively gather temperature and humidity data from 2 sensor nodes and wirelessly transmit the data from the sensor nodes, to the local rendezvous and finally to the off-site base station without any loss of data. The wireless nature of the system ensures non-invasive data collection which benefits both the wildlife and the researchers. The power supply module ensures that the nodes within the network can stay powered throughout their operation thanks to solar-powered charging and can maintain power for multiple days even when there is little sunlight during the day. The hybrid local-cloud database, API and easy-to-use API bolsters the system to be highly accessible to researchers, eliminating the need to worry about the technical nuances of the system and focus on their important research.

It must be noted that the wireless sensor network we have implemented is only a prototype and as such there is still much do to before creating a system that can be deployed in the real world. Nonetheless, testing has shown positive results and the prototype operates as intended. We are hopeful that this prototype will act is a solid foundation for future development.

Chapter 8

Recommendations

Throughout the design of our prototype and the writing of this report we have some recommendations for future enhancements and improvements.

1. Node Protection: Given the harsh environmental conditions in the Kalahari, it is crucial to protect the sensor and rendezvous nodes from the elements. Designing and implementing protective cases or enclosures for the nodes to ensure their longevity and reliable performance in challenging conditions would be greatly beneficial.

2. Power Efficiency: Since there is no power grid nearby, optimizing power efficiency is vital for the longevity of the WSN. Optimizing the power consumption of the sensor nodes by selecting energy-efficient sensors and employing low-power communication protocols could also be an avenue to explore.

3. Rainfall Measurement: To measure rainfall accurately, we would like to incorporate appropriate rain gauge sensors into our sensor nodes. These sensors can help capture rainfall data along with other environmental factors, providing a comprehensive understanding of the ecosystem being monitored.

4. Reliable Long-Distance Communication: Ensuring that the long-distance wireless communication between the rendezvous node and the off-site base station is robust and reliable. We would like to implement long-distance radio communication.

5. Node Discovery and Scalability: Enhancing the communication subsystem to facilitate easy discovery and integration of new nodes into the network. Explore protocols like the Lightweight Directory Access Protocol (LDAP) or Service Discovery protocols (e.g., DNS-SD) to enable automatic discovery and integration of new nodes without manual configuration.

6. Real-time Data Updates in UI: Implement mechanisms in the web-based user interface to enable automatic updates with new data points and new nodes. Utilize technologies such as WebSocket or Server-Sent Events (SSE) to establish real-time data streaming from the sink node to the user interface. This will provide researchers with up-to-date and dynamic access to environmental data.

7. Redundancy and Data Resilience: Considering the importance of the collected environmental data, implementing appropriate redundancy mechanisms to ensure data resilience would be greatly beneficial. Explore techniques such as data replication, distributed storage, or data backup strategies to protect against data loss in case of node failures or network disruptions.

8. Security Considerations: As the WSN deals with environmental data collection and storage, it is crucial to address security concerns. Implementing secure communication protocols (e.g., SSL/TLS) to protect data transmission and consider data encryption techniques to ensure data privacy and integrity would be helpful.

Bibliography

[1] K. Martinez, J. K. Hart, and R. Ong, "Environmental sensor networks," *Computer*, vol. 37, no. 8, pp. 50–56, 2004.

[2] A. R. Jaladi, K. Khithani, P. Pawar, K. Malvi, and G. Sahoo, "Environmental monitoring using wireless sensor networks (wsn) based on iot," *Int. Res. J. Eng. Technol*, vol. 4, no. 1, pp. 1371–1378, 2017.

[3] K. Mekki, W. Derigent, E. Rondeau, and A. Thomas, "In-network data storage protocols for wireless sensor networks: A state-of-the-art survey," *International Journal of Distributed Sensor Networks*, vol. 15, no. 4, pp. 155 014 771 983 248–, 2019.

[4] J. J. Lahoz-Monfort and M. J. Magrath, "A comprehensive overview of technologies for species and habitat monitoring and conservation," *BioScience*, vol. 71, no. 10, pp. 1038–1062, 2021.

[5] F. Martincic and L. Schwiebert, "Introduction to wireless sensor networking," *Handbook of sensor networks: Algorithms and architectures*, pp. 1–40, 2005.

[6] P. W. Rundel, E. A. Graham, M. F. Allen, J. C. Fisher, and T. C. Harmon, "Environmental sensor networks in ecological research," *New Phytologist*, vol. 182, no. 3, pp. 589–607, 2009.

[7] M. A. Matin and M. Islam, "Overview of wireless sensor network," *Wireless sensor networks-technology and protocols*, vol. 1, no. 3, 2012.

[8] K. E. Ukhurebor, I. Odesanya, S. S. Tyokighir, R. G. Kerry, A. S. Olayinka, and A. O. Bobadoye, "Wireless sensor networks: Applications and challenges," in *Wireless Sensor Networks*, S. S. Yellampalli, Ed. Rijeka: IntechOpen, 2020, ch. 2. [Online]. Available: https://doi.org/10.5772/intechopen.93660

[9] G. Barrenetxea, F. Ingelrest, G. Schaefer, and M. Vetterli, "Wireless sensor networks for environmental monitoring: The sensorscope experience," in *2008 IEEE International Zurich Seminar on Communications*. IEEE, 2008, pp. 98–101.

[10] D. Ye, D. Gong, and W. Wang, "Application of wireless sensor networks in environmental monitoring," in *2009 2nd International Conference on Power Electronics and Intelligent Transportation System (PEITS)*, vol. 1, 2009, pp. 205–208.

[11] S. Lan, M. Qilong, and J. Du, "Architecture of wireless sensor networks for environmental monitoring," in *2008 International Workshop on Education Technology and Training & 2008 International Workshop on Geoscience and Remote Sensing*, vol. 1. IEEE, 2008, pp. 579–582.

[12] P. H. Chou and C. Park, "Energy-efficient platform designs for real-world wireless sensing applications," in *ICCAD-2005. IEEE/ACM International Conference on Computer-Aided Design, 2005.* IEEE, 2005, pp. 913–920.

[13] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava, "Energy-aware wireless microsensor networks," *IEEE Signal processing magazine*, vol. 19, no. 2, pp. 40–50, 2002.

[14] J. M. Rabaey, J. Ammer, T. Karalar, S. Li, B. Otis, M. Sheets, and T. Tuan, "Picoradios for wireless sensor networks: the next challenge in ultra-low power design," in *2002 IEEE International Solid-State Circuits Conference. Digest of Technical Papers (Cat. No. 02CH37315)*, vol. 1. IEEE, 2002, pp. 200–201.

[15] A. Kumbhar, "Overview of ism bands and software-defined radio experimentation," *Wireless personal communications*, vol. 97, no. 3, pp. 3743–3756, 2017.

[16] I. T. Union, *Radio Regulations*. Geneva, Switzerland: ITU, 2020, vol. 1.

[17] P. S. Lakshmi, M. G. Jibukumar, and V. S. Neenu, "Network lifetime enhancement of multi-hop wireless sensor network by rf energy harvesting," in *2018 International Conference on Information Networking (ICOIN)*, 2018, pp. 738–743.

[18] Y.-H. Wang, Y.-F. Chen, and J.-c. Tsai, "Tunable corner cube retroreflector (ccr) fabricated with 3d printing and origami," in *2017 International Conference on Optical MEMS and Nanophotonics (OMN)*, 2017, pp. 1–2.

[19] C. Quintana, Q. Wang, D. Jakonis, X. Piao, G. Erry, D. Platt, Y. Thueux, A. Gomez, G. Faulkner, H. Chun, M. Salter, and D. O'Brien, "High speed electro-absorption modulator for long range retroreflective free space optics," *IEEE Photonics Technology Letters*, vol. 29, no. 9, pp. 707–710, 2017.

[20] Z. Zhang, J. Wang, Z. Xu, J. Zhao, and Y. Wei, "A novel detection mode of mrr communication based on speckle technology," in *2017 16th International Conference on Optical Communications and Networks (ICOCN)*, 2017, pp. 1–3.

[21] Z. Altarawneh, S. Althunibat, and R. Mesleh, "Optical wireless sensor networks using tunable optical filters," *Physical communication*, vol. 52, pp. 101 625–, 2022.

[22] L. U. Khan, "Visible light communication: Applications, architecture, standardization and research challenges," *Digital Communications and Networks*, vol. 3, no. 2, pp. 78–88, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352864816300335

[23] R. Sagotra and R. Aggarwal, "Visible light communication," *International Journal of Engineering Trends and Technology*, vol. 4, no. 3, pp. 403–405, 2013.

[24] M. Garlinska, A. Pregowska, K. Masztalerz, and M. Osial, "From mirrors to free-space optical communication—historical aspects in data transmission," *Future internet*, vol. 12, no. 11, pp. 1–18, 2020.

[25] J. B. Carruthers, "Wireless infrared communications," 2002.

[26] Á. C. Rumín, M. U. Pascual, R. R. Ortega, and D. L. López, "Data centric storage technologies: Analysis and enhancement," *Sensors*, vol. 10, no. 4, pp. 3023–3056, 2010.

[27] X. Ma, J. Liang, R. Liu, W. Ni, Y. Li, R. Li, W. Ma, and C. Qi, "A survey on data storage and information discovery in the wsans-based edge computing systems," *Sensors*, vol. 18, no. 2, p. 546, 2018.

[28] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, 1999, pp. 151–162.

[29] R. Jaichandran, A. A. Irudhayaraj *et al.*, "Effective strategies and optimal solutions for hot spot problem in wireless sensor networks (wsn)," in *10th international conference on information science, signal processing and their applications (ISSPA 2010)*. IEEE, 2010, pp. 389–392.

[30] K. Vyas, S. Shukla, H. Dsouza, D. D'Souza, and V. Narnaware, "A survey on environment monitoring using sensor networks," *Asian Journal For Convergence In Technology (AJCT) ISSN-2350-1146*, vol. 6, no. 3, pp. 95–99, 2020.

[31] N. P. Shah and P. Bhatt, "Greenhouse automation and monitoring system design and implementation," *International journal of advanced research in computer science*, vol. 8, no. 9, pp. 468–471, 2017.

[32] E. Stattner, N. Vidot, P. Hunel, and M. Collard, "Wireless sensor network for habitat monitoring: A counting heuristic," in *37th Annual IEEE Conference on Local Computer Networks-Workshops*. IEEE, 2012, pp. 753–760.

[33] R. Santos, "Esp32 client-server wi-fi communication between two boards," Jan 2020. [Online]. Available: https://randomnerdtutorials.com/esp32-client-server-wi-fi/

[34] C. Johnston, "Raspberry pi 3 - wifi station+ap," Mar 2018. [Online]. Available: https://imti.co/iot-wifi/

[35] M. Pramatarov, "What is dynamic dns? how does it work and how to setup ddns?" Jan 2022. [Online]. Available: https://www.cloudns.net/blog/what-is-dynamic-dns/

[36] K0p1-Git, "K0p1-git/cloudflare-ddns-updater: Dynamic dns (ddns) service based on cloudflare! access your home network remotely via a custom domain name without a static ip! written in pure bash ," Apr 2022. [Online]. Available: https://github.com/K0p1-Git/cloudflare-ddns-updater

[37] Apr 2023. [Online]. Available: https://gs.statcounter.com/browser-market-share/desktop/worldwide

[38] Apr 2023. [Online]. Available: https://gs.statcounter.com/browser-market-share/mobile/worldwide

[39] R. D. Hipp, "SQLite," 2020. [Online]. Available: https://www.sqlite.org/index.html

[40] "Aws::s3::bucket - aws cloudformation." [Online]. Available: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-s3-bucket.html

[41] "What is a restful api?" [Online]. Available: https://aws.amazon.com/what-is/restful-api/

[42] "Fastapi." [Online]. Available: https://fastapi.tiangolo.com/lo/

[43] L. Reznick, "Using cron and crontab," *Sys Admin*, vol. 2, no. 4, pp. 29–32, 1993.