AUTONOMOUS DATA FERRYING FROM A SELF-ORGANIZING, SELF-HEALING WIRELESS SENSOR NETWORK IN DISCONNECTED ZONES

25-26J-010

Project Proposal Report

Dion Wickrama Arachchi

B.Sc. (Hons) in Information Technology Specialising in Computer Systems & Network Engineering

Department of Computer Systems Engineering

Faculty of Computing

Sri Lanka Institute of Information Technology

Sri Lanka

September 2025

AUTONOMOUS MULTI-SENSOR ENVIRONMENTAL MONITORING NODE

25-26J-010

Project Proposal Report

B.Sc. (Hons) in Information Technology Specialising in Computer Systems & Network Engineering

Department of Computer Systems Engineering

Faculty of Computing

Sri Lanka Institute of Information Technology

Sri Lanka

September 2025

DECLARATION OF THE CANDIDATES & SUPERVISOR

We declare that this is our own work, and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Name	Student ID	Signature
Dion Wickrama Arachchi	IT22360496	Dion

Date: 28-8-2025

The above candidates are carrying out research for the undergraduate Dissertation under my supervision.

Signature of the Supervisor: Dinithi Pandithage Date: 28/08/2025

Abstract

Remote environmental monitoring in disconnected zones poses challenges in data collection, power management, and network reliability. This project proposes an autonomous data ferrying wireless sensor network (WSN) with self-organizing and self-healing capabilities to address these challenges. The focus of this individual component is the design of robust sensor nodes equipped with a solar-recharged power unit, multi-sensor module, and adaptive power management. The sensor nodes will operate in three modes (normal sensing, low-power beacon, deep sleep) to prolong battery life while storing critical data locally. An autonomous mobile data ferry (e.g. unmanned aerial vehicle) will periodically collect compressed sensor data from the nodes, where this project introduces a comparative evaluation of two lightweight entropy-based algorithms – Huffman coding and LZ-family compression (miniz). The proposed methodology includes an energy-efficient hardware design using an ESP32-S3 microcontroller and environmental sensors (air chemistry, acoustics, geomagnetic, atmospheric data), combined with an on-node lightweight compression algorithm (autoencoder-based) to minimize transmission load. The system's feasibility will be evaluated through prototypes and simulations, anticipating results that demonstrate prolonged node lifetime (>12 months) and reliable data delivery despite sparse connectivity. Successful implementation would fill a crucial gap in WSN deployments for remote or off-grid areas, enabling long-term environmental sensing with minimal maintenance.

Keywords: Wireless Sensor Network, Data Mule, Low-Power Design, Solar Energy Harvesting, Internet of Things (IoT) Compression.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Modern wireless sensor networks (WSNs) have enabled pervasive environmental monitoring by combining sensing, data processing, and wireless communication in compact embedded devices [2, 3]. A typical sensor node consists of four main units: a sensing unit (for measuring environmental parameters), a processing unit (microcontroller for local computation), a communication interface (radio transceiver), and a power supply [3]. These nodes are often deployed in large numbers to gather data over wide areas – for example, to monitor wildlife habitats, agricultural fields, or remote ecosystems.

However, deploying conventional WSNs in disconnected or hard-to-reach zones presents significant challenges. In a traditional WSN, each node relays data hop-by-hop to a base station, requiring a connected network topology. In sparsely populated areas or wilderness, maintaining continuous multi-hop connectivity would require many relay nodes, which are often impractical and costly [4]. Researchers note that "the monitored area can be far away from the nearest access point, and deploying additional sensors for relaying data becomes too costly, making mobile data collection more efficient [4]. Furthermore, sensor nodes themselves are not trivial in cost – a basic node can cost on the order of $100 and runs on limited batteries, and manually replacing batteries or adding infrastructure at remote sites is labour-intensive [5]. These realities undermine the scalability of static WSN deployments for large, disconnected regions [4, 5].

As a result, there is growing interest in "data mule" or "message ferry" architectures, where mobile agents physically travel to sensor nodes to collect their data opportunistically [1, 4]. This approach essentially creates a delay-tolerant network: sensor nodes store data locally until a mobile collector (e.g. a drone or robot) comes within range to download it. Prior work has shown that using mobile data mules can greatly reduce the need for intermediate relay nodes and extend network lifetime, since nodes can transmit over short distances when the mule is nearby [1, 6]. For example, one study demonstrated that robots acting as data ferries eliminated the need for many static relays and significantly decreased energy consumption and data

loss in a sparse network [1]. In disconnected networks with no persistent links, mobile ferries may be “the only viable solution to ensure the network’s connectivity,” filling in connectivity gaps in the monitored area [1].

Another critical challenge in such scenarios is power management. Sensor nodes in remote locations often must operate on battery power for months or years. Renewable energy sources (like solar) can recharge nodes, but energy still must be carefully conserved. Research in low-power WSN design emphasizes dynamic power management and multiple operational modes to extend battery life [2, 7]. For instance, You et al. propose fine-grained power modes (five states) for sensor nodes, modulating energy use based on communication activity, and achieved a ~74% reduction in idle power consumption compared to conventional single-mode operation [2]. Many modern sensor platforms incorporate sleep modes, duty-cycling, and adaptive shutdown of sensors to save energy when data is not needed in real-time [2, 8]. These techniques align with the concept of self-healing, self-organizing networks – the system can autonomously adjust to conserve resources and maintain coverage even if individual nodes go offline temporarily due to low power or faults [9].

Within this broad domain, our project is situated at the intersection of off-grid Internet of Things (IoT) sensing, energy harvesting, and delay-tolerant networking. The overall aim is to build a self-organizing, self-healing WSN that can operate in disconnected zones (where there is no cellular or internet connectivity) by relying on autonomous data ferrying. The individual component addressed in this proposal is the hardware and power management design of the multi-sensor (MS) nodes that form the backbone of the network. This includes selecting and integrating environmental sensors, designing the embedded system (microcontroller and storage), implementing an intelligent Power Management Engine (PME) with multiple modes, and ensuring the node can interface with the mobile data ferry via short-range communication (e.g., Wireless Fidelity – Wi-Fi – or Bluetooth Low Energy).

1.1. Background & Literature Survey

The concept of using mobile collectors for WSN data gathering has been explored in various forms over the past two decades. Early work by Shah et al. introduced the idea of Data MULEs (Mobile Ubiquitous Local Explorers) as a three-tier architecture for sparse sensor networks, where mobile agents ferry data from sensors to access points, drastically reducing the need for long-range transmissions by sensor nodes [12]. Subsequent research refined this concept into practical frameworks: Zhao et al. coined the term "message ferrying" in disruption-tolerant networks, demonstrating that having mobile ferries as the only means of communication can greatly increase network lifetime in disconnected scenarios [14]. More recent studies, such as Tsilomitrou and Tzes (2022), have formulated path optimization problems for multiple mobile elements (MoEls) to ensure timely data pickup from sensors without buffer overflow [1]. These works collectively show that mobility can be leveraged to solve connectivity and scalability issues in WSNs, albeit with the trade-off of increased data latency (as data waits for collection).

In parallel, extensive literature exists on energy-efficient sensor node design. Because WSN nodes often run on batteries, researchers have developed both hardware and algorithmic strategies to maximize node lifetime [2, 7]. Key techniques include ultra-low-power hardware components, energy harvesting (solar, thermal, etc.), and dynamic power management (DPM). DPM involves transitioning the node into different power states (active, idle, sleep, deep sleep) depending on workload. For example, a node might sleep most of the time and wake up periodically to sense and transmit or wake on external triggers. Vieira et al. (2006) emphasize that a sensor node's design must balance performance with energy use across sensing, processing, and communication tasks [3]. Communication (radio transmission) is typically the most energy-intensive operation [10], so reducing communication frequency or range has a big impact on power consumption. This is another advantage of the data mule approach: since the mobile collector comes close to the node, high-power long-range transmissions can be avoided [1].

In terms of sensor modalities for environmental and biodiversity monitoring, prior studies guide our choices. Acoustic monitoring has emerged as a powerful, non-intrusive method to survey wildlife. Researchers deploy passive acoustic recorders to capture sounds of birds, frogs, bats, and insects, which can then be analysed to infer species presence and behaviour [11]. In fact, passive acoustic monitors have been used to collect large-scale data on wildlife populations, providing insights into biodiversity trends across temporal and spatial scales [11]. Similarly, measuring atmospheric carbon dioxide (CO2) and volatile organic compounds (VOCs) in the environment can serve as indicators of biological activity and human impacts. VOC sensors detect gases emitted from vegetation, soil microbes, or pollutants, and an increase in CO2 (especially in enclosed or underground spaces) often correlates with respiration from organisms or decomposition processes. Thus, combining bio-acoustic sensing with air chemistry sensing can give complementary views of an ecosystem's state – one through soundscape analysis, the other through chemical signatures.

Magnetic sensors (magnetometers) and weather sensors (pressure, temperature, humidity) are also common in ecological sensor networks. A magnetometer can pick up geomagnetic anomalies (useful for geological mapping) or serve as a compass to tag the orientation of measurements. Barometric pressure and temperature/humidity readings provide context to other sensor data; for example, gas sensor readings can be corrected for temperature/humidity influences, and pressure trends can indicate altitude changes or weather shifts that might affect sensor readings. Including these in each node creates a more self-contained, context-aware sensing unit.

A survey of related projects reveals a few systems with comparable goals, though none with the exact proposed feature set. For example, the NASA "Sensor Web" concept and some habitat monitoring projects have used UAVs (drones) to periodically download data from ground sensors, but these often rely on simpler power management and do not implement multi-modal energy saving at the node level [15]. On the other hand, there are long-lived WSN deployments for volcano monitoring or glacier monitoring that use solar panels and aggressive sleep scheduling, but they typically use fixed radio relays or satellite uplinks to gather data

instead of mobile ferries [16, 17]. Our approach seeks to integrate the best of both worlds: robust low-power sensor nodes and an autonomous data ferrying mechanism, to enable deployments in areas with zero communication infrastructure (no cell coverage, no reachable base station) and no regular human maintenance.

From a data management perspective, transmitting raw sensor data even infrequently could be inefficient, especially if bandwidth between node and data mule is limited (for instance, BLE beacons have very limited payload, and Wi-Fi transfers are power-expensive). This motivates the use of on-node data compression. Traditional compression algorithms (Huffman coding, LZ variants) work well for general data, but they may not be optimized for patterns in sensor data streams. Recent research has explored machine learning-based compression, such as using autoencoders to learn compact representations of sensor signals [13]. Shylashree et al. (2025) demonstrate a joint compression-encryption method using an autoencoder for linear sensor network data, significantly reducing data size with tolerable loss [13]. While their scheme assumes a base station with more computational power, the concept can be adapted in a lightweight form for our sensor nodes: a simple neural network or pattern-learning algorithm could compress recurring sensor patterns (daily temperature cycles, periodic sounds, etc.) before storage or transmission [13]. We aim to leverage an AI-based lossless compression algorithm – likely a microcontroller-friendly autoencoder combined with entropy coding – to minimize the amount of data that needs to be sent to the data mule, thereby saving both energy and time.

In summary, the literature provides building blocks and justification for each aspect of our proposed solution: mobile data ferries solve connectivity and scalability issues in remote WSNs [1, 4]; advanced power management and energy harvesting techniques address the longevity challenge [2, 7]; and multi-modal sensing combined with intelligent on-node processing (compression) maximizes the scientific value of the data collected while minimizing resource usage [11, 13]. Our task is to integrate these components into a coherent system tailored to the specific problem of disconnected-zone monitoring.

1.2. Research Gap

Despite the wealth of related work, there remains a clear gap in fully autonomous, long-term sensor networks for disconnected zones. Most existing WSN deployments assume either occasional human intervention (for replacing batteries or retrieving data loggers) or the presence of some communication backbone (such as a satellite uplink or a short-range multi-hop path to an internet-connected gateway). There is currently no widely adopted solution that allows a sensor network to self-sustain for extended periods (1+ years) in a completely isolated environment while automatically ferrying its data to a remote base.

In previous studies, the problems of mobility, power management, and data optimization have largely been addressed in isolation. For instance, path planning for data mules is often studied with the assumption that sensor nodes are simply always on and buffering data [1]. Conversely, ultra-low-power sensor designs exist, but they typically assume either a static network topology or offline data retrieval. The novel opportunity here is to combine aggressive power-saving hardware design with intelligent mobile data collection and on-node data compression – effectively bridging the gap between IoT devices and delay-tolerant networking. However, compression has mostly been treated abstractly or off-node, and no published work provides a head-to-head evaluation of entropy-based algorithms (Huffman vs LZ) directly on ESP32-class nodes with energy profiling. This creates a clear knowledge gap: determining the “best” algorithm for resource-constrained, solar-powered nodes in disconnected zones remains unresolved. Our project directly addresses this gap by benchmarking both algorithms on-device and linking the results to practical energy-autonomy outcomes.

Concretely, the knowledge gap can be stated as: How can we design a WSN node that is capable of autonomous, long-duration operation (through renewable energy and power management) and is seamlessly integrated into a self-organizing network that uses mobile ferries for data retrieval? The “self-healing” aspect also implies that nodes should gracefully handle energy depletion. While multi-mode operation is known in WSN research, our proposed three-tier power mode (normal, passive,

critical) with autonomous mode switching based on energy levels is a creative approach not seen in standard WSN platforms. This is designed to prevent total node failure: even at critical battery levels, a node might still wake occasionally to advertise its presence.

Another gap is in the system integration and validation: we are not only proposing a theoretical model but planning to build a working prototype of these sensor nodes and test them within a year's development cycle. Many of the cited works are simulation-based (especially for routing or scheduling algorithms) or focus either on hardware prototyping or algorithm design alone. Our project will contribute practical insights by implementing the full stack – hardware, firmware (embedded software), and network coordination – and evaluating it in a real-world scenario.

Furthermore, by focusing on an individual component (the sensor node subsystem), we can dive deeply into optimizing that component while ensuring it will fit into the larger team project. The expectation is that by the end of this project, we will have demonstrated a novel sensor node design that advances the state-of-the-art in terms of energy autonomy, on-node entropy-based compression, and integration with mobile data ferries. This will fill the research gap of how to make WSNs truly autonomous and self-sustaining in environments where previously one would assume it "too remote" or "too maintenance-heavy" to deploy long-term sensors.

1.3. Research Problem

Based on the above gap, the primary research problem can be formulated as follows:

How can we develop a self-organizing wireless sensor network node that reliably collects environmental data in disconnected zones for long durations, and what hardware/power management innovations are required to enable autonomous data ferrying and self-healing capabilities in such a node?

This broad question can be broken down into specific research questions/issues that need to be addressed during the project:

i. Power Autonomy Problem: How can an off-grid sensor node manage its energy usage and generation to operate continuously for a year or more without human intervention? This includes sub-problems of efficient solar energy harvesting, battery management, and dynamic power mode control under varying environmental conditions (sunlight availability, temperature effects on batteries, etc.).

ii. Disconnected Data Retrieval Problem: In the absence of real-time communication infrastructure, how can data be ferried periodically from sensor nodes to a base station reliably? What communication protocols and data handoff techniques are best suited for the node-to-mule interaction (e.g., BLE advertising vs Wi-Fi direct handshake)? How often should the data mule visit (considering node memory limits and data generation rates) to prevent data loss?

iii. Integration of Multi-Modal Sensing: How to process and compress multi-sensor data (acoustic, gas, magnetic, climate) on a constrained device so that critical information is preserved while redundant data is minimized? This raises problems of on-node signal processing (e.g., detecting events like a VOC spike or an animal call) and choosing compression algorithms that can run on the ESP32 microcontroller in real-time.

iv. Network Self-organization and Self-healing: What strategies will allow the network to reconfigure or adapt if some nodes go offline or if energy is low? If a node goes into deep sleep, how does it rejoin or signal the network upon revival? These questions relate to developing simple coordination protocols or fail-safes that make the network resilient to individual node issues.

v. System Validation: What are the criteria to evaluate the success of the proposed system, and how can we test them? This involves formulating metrics like node uptime, data yield (percentage of data successfully ferried to base), energy consumption per operation, etc., and designing experiments (perhaps a pilot deployment or a lab emulation of field conditions) to measure these.

By addressing these specific problems, we aim to answer the overarching research question and demonstrate a solution that pushes the boundary of what current sensor networks can do.

The significance of solving this research problem is high: it would effectively enable persistent monitoring of remote or extreme environments (rainforests, mountains, disaster zones, etc.) with minimal infrastructure. This is important to a variety of stakeholders – ecologists and conservationists who need data from wild areas, industrial companies monitoring pipelines or mines where connectivity is sparse, and government agencies handling disaster response or climate observations in remote regions. The next sections will outline the objectives and the methodology for tackling this research problem through my individual component of the project.

2. OBJECTIVES

2.1. Main Objective

Design and implement energy-autonomous, multi-sensor wireless nodes with adaptive power management and data compression capabilities, which can seamlessly integrate into a self-organizing, self-healing sensor network and support autonomous data ferrying in disconnected zones.

Explanation: This main objective encapsulates the end goal of the individual component: to create the hardware and firmware for a sensor node that meets the requirements of long-term autonomous operation and compatibility with the data mule system. It implies building the node (hardware assembly and PCB if needed), writing the embedded software that runs on it (to manage sensors, power, and compression), and ensuring the node can operate as part of the overall network. Success will be measured by whether the node can run continuously (with solar recharging) and deliver its data via the mobile collector as intended.

2.2. Specific Objectives

i. Develop the Sensor Hardware Platform: Select and integrate at most four environmental sensors – including an air chemistry sensor (for VOC/equivalent CO2), a digital Micro-Electro-Mechanical Systems (MEMS) microphone (for bio-acoustic data), a 3-axis magnetometer, and a barometric pressure/temperature/humidity sensor – with the ESP32-S3 microcontroller. This includes designing the wiring/interface – Inter-Integrated Circuit (I2C), Inter-Integrated Circuit Sound (I2S), or Serial Peripheral Interface (SPI) as needed) – on a prototype board, ensuring all sensors can be read reliably at required intervals. Measurable outcome: a fully assembled sensor node prototype with all sensors producing valid readings.

ii. Implement the Power Management Engine (PME): Develop a firmware module on the ESP32-S3 that monitors the node's battery level and controls

the node's operating mode (Normal sensing, Passive beacon-only, Critical deep-sleep). The PME will enforce rules such as: if battery drops below threshold X, switch to Passive mode (stop heavy sensing and only advertise via BLE); if below threshold Y, enter deep sleep and wake minimally. It will also manage the solar charging input via a power controller (e.g., reading current/voltage from the INA219 sensor to decide when to charge or when battery is full). Measurable outcome: demonstration that the node automatically transitions between modes under simulated power conditions.

iii. Integrate Solar Charging and Battery System: Design the power unit using a rechargeable Li-ion battery (e.g., 3.7V 18650 cell) and a solar charging module. Ensure proper regulation (using DC-DC converters if needed) to provide stable 3.3V to the electronics. The objective includes protecting the battery (overcharge/discharge protection) and verifying that under typical daylight conditions the solar panel can replenish enough energy for 24-hour operation. Measurable outcome: the node running continuously through a day-night cycle in a test, with battery levels recovering during sun periods.

iv. Develop On-Node Data Compression Algorithm: Implement and evaluate two lightweight entropy-based algorithms: Huffman coding (symbol-based, lossless, low computational cost). LZ-family compression (miniz) (dictionary-based, higher compression ratios but with added CPU/RAM demands). Measurable outcome: Comparative results for compression ratio, CPU cycles, RAM usage, and energy per byte saved. This delivers practical guidance on which algorithm is better suited for disconnected-zone WSN nodes.

v. Local Data Storage & Fault Tolerance: Set up a local storage mechanism (using the ESP32's flash or an external SD card if needed) to buffer sensor readings between collection times. This involves designing a simple local database or file format to queue sensor data (especially important for acoustic recordings or high-frequency data). Additionally, implement basic fault-tolerance measures such as data checksums or duplicate logging to avoid data

loss if a transfer is interrupted. Measurable outcome: the node can store at least e.g. one week's worth of data locally (exact amount depending on sensor sampling rates) without overflow and can recover or retain data after a reset or crash (showing persistence).

vi. Testing and Evaluation Plan: Although not a development task per se, a crucial objective is to formulate a thorough testing plan to evaluate the node's performance against requirements. This includes creating test cases (which we outline later in section 4.6) for functionality (sensors working, communications working) and for performance (battery longevity test, environmental stress test in varying temperatures/humidity to ensure robustness). By the end of the project timeline, we should conduct a field test where a prototype node is deployed outdoors and a drone (or a person with a mobile receiver) simulates the data ferry periodically to collect data. Measurable outcome: A report on test results, including metrics like how long the node ran on battery, success rate of data transfers, any observed failures, etc. These will indicate if the objectives have been met and where further improvements are needed.

3. METHODOLOGY

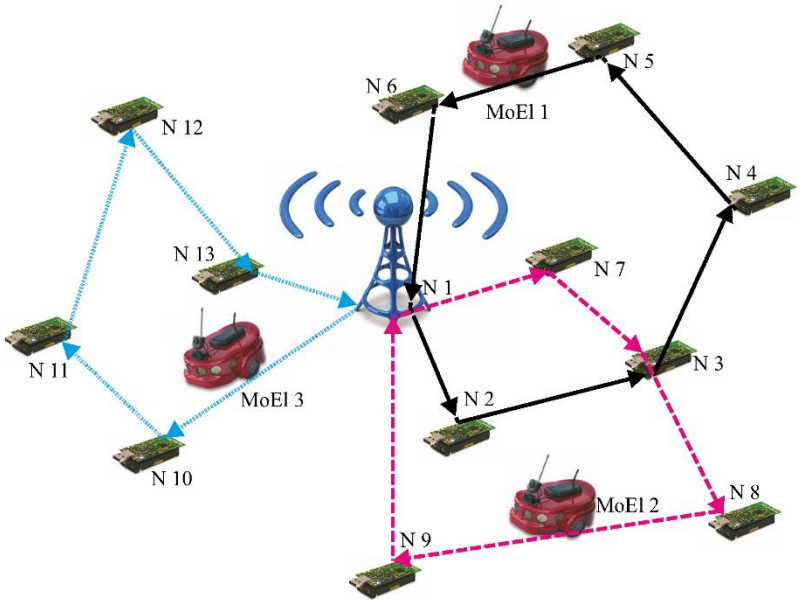

Figure 3.1: Data mule concept with multiple MoEls collecting data from static sensor nodes and delivering them to a base station.

Source: Mobile Data-Mule Optimal Path Planning for Wireless Sensor Networks [1]

This concept (adapted from Tsilomitrou & Tzes [1]) underpins our project's approach of using mobile agents to ferry data for a sparse sensor network. In our system, a single autonomous drone will serve as the data mule, periodically visiting each sensor node.

Overall System Architecture: The proposed system consists of several sensor nodes (the devices we are building) deployed across the target area, and one mobile data ferry (drone) that travels through the area to collect data from the nodes. The nodes are designed to be self-organizing – they automatically discover neighbours, form clusters, and elect cluster leaders without manual configuration of routes or infrastructure. They are also self-healing – if one node loses power or goes offline, it does not crash the network, and when it comes back (or recharges), it can continue operation seamlessly.

For clarity, Figure 2 illustrates the architecture of individual sensor nodes and its interaction with the data mule and the base station/cloud.

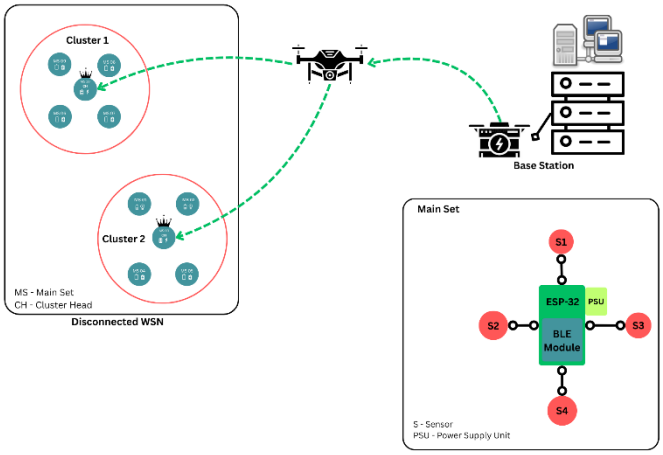

Figure 3.2: High-Level Architecture of the Proposed System

The sensor node contains: (a) the Sensor Suite (ENS160 gas sensor, INMP441 microphone, HMC5883L magnetometer, BME280 weather sensor), all connected to the microcontroller's Sensor I/O bus (I²C for most, I²S for microphone); (b) the Processing Module (ESP32-S3 microcontroller) which includes sub-blocks for Storage Manager (writing to flash memory), Power Management Engine (logic controlling power modes), and Compression Algorithm (for data reduction); (c) the Communication Module, which on the ESP32 provides both Wi-Fi and BLE radios (with an external antenna for range); (d) the Power Unit, comprising the lithium battery, a solar panel input through a charge controller (CN3065 based module), and an INA219 sensor to monitor current/voltage. The mobile data ferry (drone) is depicted as an entity that comes within wireless range of the node – at that time, the node's Communication Module either sends out data or responds to the mule's data request. The mule then carries that data to the Base Station, which could be a field laptop or a server that the drone connects to when it returns to a connectivity zone. The base station aggregates all data and could upload to a cloud for remote access.

System Operation Flow: Each sensor node follows a scheduled duty cycle. At defined intervals (e.g., every 5 minutes), the Processing Module (ESP32-S3) wakes up from deep sleep. The Sensor Suite (ENS160 gas sensor, INMP441 microphone, HMC5883L magnetometer, BME280 weather sensor) collects environmental readings. Readings are passed through the Sensor I/O bus (I²C for ENS160,

HMC5883L, BME280; I²S for INMP441). The Storage Manager writes raw or partially processed data into flash memory. Before long-term storage or transmission, the Compression Algorithm reduces data size to save both storage and communication energy. The Power Management Engine continuously evaluates energy status using feedback from the INA219 current/voltage monitor and solar charging via the CN3065 controller. In Normal Mode (battery sufficient, solar input available), all sensors operate as scheduled. In Passive Mode (low power or night conditions), high-demand operations (e.g., microphone recording) are suspended. The node conserves energy, advertising only a lightweight BLE beacon that broadcasts its ID and "alive" status. In Critical Mode (battery near depletion), the node spends most of its time in deep sleep (up to 90%) and wakes briefly for minimal beaconing. This mode preserves survival until solar input restores charge.

Development Tasks and Timeline: The implementation will follow a timeline that matches major project milestones. Below is the Work Breakdown Structure (WBS) of tasks for this individual component, along with estimated time frames and milestones:

i. Task 1: Requirement Analysis and Design Specification (July-August 2025) – In the initial phase, gather detailed requirements (scientific data needs, environmental conditions of deployment site) and finalize sensor selection. Outcome: a written specification and system diagram (Figure 2). Milestone: Proposal report submission (mid Aug 2025) with clear design plan.

ii. Task 2: Hardware Prototyping (September-October 2025) – Order and assemble the components: ESP32-S3 dev kit, sensors modules (ENS160+AHT21, INMP441, HMC5883L, BME280), power modules (battery, solar panel, charger, INA219). This involves soldering and wiring on a breadboard or PCB. During this period, test each sensor individually with simple code to ensure they work (e.g., read gas concentration from ENS160, capture audio from microphone and check noise levels, etc.). Milestone: By end of October, a preliminary hardware prototype with sensors returning readings on serial monitor.

iii. Task 3: Firmware Development – Phase 1 (November-December 2025) – Start developing the embedded software on the ESP32. Phase 1 focuses on core functionality: reading sensors on schedule and storing data and implementing power mode switching basics. For example, write a loop that reads all sensors, then goes to light sleep for X minutes. Implement an interrupt or timer for wake. Integrate the INA219 reading to gauge battery and logic: if battery < threshold, set a flag for low power. Also, implement BLE beacon advertising at this stage (using ESP-IDF or Arduino BLE libraries, create an advertisement payload with node ID). Milestone: Demo in December 2025: Node runs for several hours with dummy sensor readings, showing it can sleep and wake, and a BLE scanner app sees its beacon when in low-power mode.

iv. Task 4: Firmware Development – Phase 2 (January – February 2026) – Advanced firmware features now: integrate and benchmark Huffman and LZ/miniz compression algorithms, comparing their ratio, speed, memory, and energy cost. Parallel to this, incorporate the file system (ESP32's Serial Peripheral Interface Flash File System – SPIFFS – or LittleFS) to store readings persistently. Milestone: By end of February, lab test where the ESP32 connects to a laptop (as pretend drone) over Wi-Fi and successfully sends a batch of stored data that, when decompressed on the laptop, matches the original within acceptable error.

v. Task 5: Power Management Tuning and Testing (March 2026) – With all functionalities in place, refine the power management. This means measuring current draw in different modes and adjusting duty cycles. For example, find how often the node can beacon without draining too much, or determine if the microphone can be duty-cycled (maybe only turn on mic during certain hours or based on a lightweight wake trigger from an auxiliary detector by a sound above threshold, etc.). Conduct tests with the actual solar panel: put the node outside to charge and see how the charging module behaves (does it fully charge the battery by midday? does the system cut off at correct voltage?). Potentially iterate the hardware if needed (e.g., add a larger solar panel or

bigger battery if tests show insufficient energy). Milestone: Achieve a continuous 48-hour run in simulated real conditions by mid-March (covering day-night-day with some sensor activity and at least one data upload event).

vi. Task 6: Integration with Team & Field Trial (April 2026) – Work with the team member handling the drone/mobile unit to integrate protocols. This may involve fine-tuning timing – e.g., how long should the drone hover for data transfer, or implementing a retry mechanism if transfer fails. Possibly perform a small-scale field trial: place a node in an open area or forest patch, have the drone fly by (or a person walk by with a receiver) to collect data. Evaluate range (how close does the drone need to be). Make any last adjustments discovered during integration (like increasing beacon frequency if drone had trouble detecting, etc.). Milestone: End of April, have a demonstration video or live demo where the drone collects data from at least one prototype node deployed in a mock scenario.

vii. Task 7: Documentation and Final Refinements (May 2026) – Complete all documentation, including the final report, and research paper draft. Also prepare for final presentation and viva – which will involve presenting results, showing that our objectives (like compression achieved, battery life, etc.) have been met. Any remaining minor improvements (like code optimization or enclosure of the node in a weatherproof box) can be done in this phase. Milestone: Final project submission and presentation in May 2026, demonstrating an excellent proposal outcome.

Figure 3.3: Gantt Chart

Throughout these tasks, we will maintain a feedback loop – continuously testing each module as it's developed (unit testing sensors, then integration testing for power and communications). If any task slips or a certain approach fails, we have some float in the timeline (for instance, if one of the tested entropy-based algorithms proves infeasible due to memory or energy overhead, fallback to the alternative is possible, as both will be benchmarked early in the timeline).

Development Tools and Resources: We plan to use the ESP-IDF (Espressif IoT Development Framework) or Arduino-ESP32 core for firmware coding (C/C++). Testing will utilize tools like a multimeter and oscilloscope (to measure current consumption), and possibly a logic analyser (to debug I2C signals if needed). Version control (Git) will be used for managing code. For compression algorithm

development, Python and PC-based simulations will be used first, then converted to C code or use of TensorFlow Lite Micro if an ML model is used. The team will also use regular meetings to integrate components: e.g., ensuring the drone's software knows how to request data from the node – this might involve writing a small Python script on a companion computer of the drone to handle transfer, which is within our capability.

Anticipated Results and Conclusion of Methodology: By following the above methodology, the expected result is a working prototype system by the project end. The anticipated conclusion is that the individual component – the sensor node – can meet its requirements: it will run continuously via solar power, collect multi-modal environmental data, compress and store that data, and offload it via an autonomous data mule with minimal human intervention. We expect to recommend this approach for real-world deployments and possibly identify any limitations (for example, we might discover the need for bigger solar panels in dense canopy, or that acoustic data is too heavy to transmit frequently, etc.). These findings will be part of the final recommendations.

4. PROJECT REQUIREMENTS

For the successful implementation of the sensor node component, a set of requirements has been identified. These include functional requirements (what the system should do), non-functional requirements (quality attributes and constraints), user requirements (from the perspective of an end-user or stakeholder using the system), and system requirements (technical specifications of the hardware/software environment). Additionally, we outline tentative use cases, test cases, and wireframe considerations (where applicable). All requirements are drafted to guide development and will be refined during the project as needed.

4.1. Functional Requirements

i. F1: Data Sensing – The node shall continuously monitor environmental parameters using the integrated sensors. Specifically, it shall measure volatile organic compound (VOC) levels and equivalent CO2 (via ENS160), capture ambient sound for bio-acoustic monitoring (via INMP441 microphone), measure magnetic field intensity (via HMC5883L), and record temperature, humidity, and barometric pressure (via BME280). Each sensor reading should be time-stamped and saved for transmission.

ii. F2: Data Storage – The node shall locally store sensor readings in a memory buffer or file system. It must support storing at least one week's data (assuming a baseline of, say, 1 reading per minute for low-frequency sensors and a few short audio clips per day). Old data should be overwritten or intelligently managed if storage is full (circular buffer or FIFO queue of records).

iii. F3: Data Compression – The system shall compress sensor data before transmission. For example, numeric time-series (VOC, temperature, etc.) might be delta-encoded and Huffman compressed, and audio data might be compressed (using a lightweight codec or down sampling). This compression should be lossless for critical low-frequency data and can be near-lossless for

high-volume data like audio (i.e., some compression that does not significantly affect analysis outcomes).

iv. F4: BLE Communication – The node shall advertise using Bluetooth Low Energy.

v. F5: Power Mode Management – The node shall automatically manage its power modes: Normal Mode: Full functionality (all sensors active on schedule, higher communication readiness). Passive Mode: Low-power state (e.g., only essential sensors or infrequent sampling, maybe just VOC and periodic sound snippets, BLE advertising on). Critical Mode: Deep sleep with minimal wakeups (perhaps wakes hourly just to check battery). Transitions shall be based on battery level thresholds or other triggers (e.g., dusk/dawn detection from solar panel).

vi. F6: Solar Energy Harvesting – The system shall charge its battery from the solar panel when light is available. It must prevent overcharging and prevent deep discharge damage to the battery. The INA219 sensor readings can be used to implement MPPT (Maximum Power Point Tracking) in a simple way or at least to log how much current is coming in.

vii. F7: Self-Healing Networking – In case of a temporary node shutdown (e.g., battery dead at night), the node shall resume operation when power is sufficient (sun recharged) and rejoin the network without manual intervention.

viii. F8: Unique Identification and Security – Each node shall have a unique ID (possibly the Project ID plus node number, or MAC address of ESP32) so that the data mule can distinguish between nodes.

ix. F9: User Configurability – It shall be possible to configure certain parameters (sampling rate, thresholds, etc.) either by editing a config file on the node or via a command from the data mule. For example, a researcher might want to change the microphone schedule if too much data is collected. This could be done by sending a config update command when the drone connects.

4.2. Non-Functional Requirements

i. NF1: Low Power Consumption – The sensor node should consume extremely low power, especially in Passive and Critical modes. Target consumption in deep sleep is in the order of tens of microamperes, and in normal active sensing mode should ideally average under 10 mA (except bursts during ESP-NOW transmission to UAV or microphone recording). The overall energy usage should be such that the system can run indefinitely under average sun conditions (target: solar panel can supply average daily energy >= daily consumption).

ii. NF2: Durability and Environmental Resistance – The node (when enclosed properly) should withstand the environmental conditions of the deployment zone. This includes operating temperature range (e.g., -10°C to +50°C if in outdoors in temperate climates, or appropriate range for local environment), humidity (the enclosure should prevent damage from moisture), and ideally some waterproofing. The hardware chosen should be stable over time (for example, the Li-ion battery chosen must handle the expected temperature range safely).

iii. NF3: Data Reliability – No significant data loss should occur due to system failures. The use of flash storage means wear levelling is needed if heavy writing; the system should handle power cuts gracefully (e.g., using atomic writes or journaling to avoid corrupting the entire log). Also, when transmitting, data integrity should be checked (via checksums) to ensure the base station gets correct data. We target >99% data delivery of logged data over the network's operation.

iv. NF4: Scalability – The design should be scalable to at least dozens of nodes. Although this component is six nodes, the approach should not preclude having 20-50 nodes in one area. This implies that the data mule's schedule and the network protocols can handle multiple nodes. For example, BLE advertising by many nodes should use different intervals to avoid collision, or

the mule might poll nodes one by one. Our node design should allow setting different beacon intervals if needed to coordinate in a larger network.

v. NF5: Maintainability – It should be relatively easy to update firmware on the nodes in the field. Perhaps support Over-the-Air (OTA) updates when the drone is nearby (the drone could send a new firmware file to the node). At minimum, the code should be modular and well-documented so future developers can adjust it. Also, replacing components (like if a sensor fails) should be possible by modular design.

vi. NF6: Cost Effectiveness – The target cost per node should be kept low. We aim for each node's BOM (bill of materials) to be within a reasonable range. This will make the solution more practical for scaling up or commercialization. The components chosen (ESP32, common sensors) reflect this requirement.

vii. NF7: Data Privacy and Security – While our environmental data might not be highly sensitive, if the system were used in other contexts (e.g., security monitoring), we should ensure data is not easily intercepted or tampered with. Non-functional security measures include using encryption for data at rest on the node (maybe not needed for environment data) and encryption in transit (e.g., use BLE pairing with encryption). Also ensure that a rogue device cannot easily drain node's battery by triggering transmissions (maybe the node only responds to authenticated requests from our drone).

viii. NF8: Compliance and Safety – Use of radio frequencies should comply with local regulations. The power unit should be safe (proper handling of Lithium battery charging to avoid fire hazard). Also, the device should not harm wildlife (for example, the microphone's ultrasonic range should not inadvertently disturb animals; likely not an issue as it is passive listening).

4.3. User Requirements

Here the term “user” could refer to the end user of the system, e.g., a researcher or field technician deploying the network, rather than a consumer using a UI. We frame requirements in terms of what a user expects the system to do.

i. U1: Easy Deployment – A field researcher should be able to deploy each node with minimal setup. For instance, after assembly, turning on a node might be as simple as flipping a switch, and then the node self-configures (calibrates sensors, starts logging). The user should not need to manually pair the node with the drone – this should be pre-configured. Essentially “place it and let it run” functionality.

ii. U2: Status Monitoring – The user wants to know that the node is functioning correctly when deployed. Each node should provide some basic indication of status (could be an LED blink pattern or a status message via BLE). For example, an LED that blinks every 10 seconds could indicate the node is alive and in a certain mode (fast blink = normal mode, slow blink = low-power mode). Alternatively, the user could query the node via a mobile phone app over BLE to get status (battery, sensor health). This requirement ensures confidence during deployment and maintenance.

iii. U3: Data Retrieval and Viewing – The end user (e.g., an ecologist) eventually needs to see the collected data in a usable form. While the drone and base station portion will handle aggregation, from the node’s side the data should be well-organized and annotated (e.g., each record with timestamp and node ID). The data coming off the node should be in a documented format (Comma Separated Values – CSV, JavaScript Object Notation – JSON, or binary with a defined schema). This makes it straightforward to import into analysis software later. Users require that data is easily correlated with time and location (so the node’s data likely needs a clock; perhaps the drone can synchronize node RTC – real-time clock – periodically, or the node just resets time every pickup, etc.).

iv. U4: Longevity – Users (and funders) expect the nodes to last through the intended study duration without needing frequent manual intervention. If the goal is a 1-year deployment, the user requirement is that "the nodes should run for one year in the field on solar power, with maybe only seasonal check-ups". This implies robust build quality and that battery sizing is such to handle a few days of bad weather etc. This requirement translates to design choices like battery capacity and solar panel size that we have accounted for.

v. U5: Modularity and Extensibility – A user might want to customize or extend the sensor node for different expeditions (for example, add a different sensor like a camera or a soil moisture probe for another deployment). The system should thus be modular: it should be possible to add another sensor on the I/O bus with some firmware adjustments. Also, if the user doesn't need one of the default sensors, they could disable it to save power. Essentially, this means our design and documentation should allow others to adapt the node easily.

vi. U6: Safety in Handling – The user, when handling the node for deployment or retrieval, should not have to deal with complex or dangerous procedures. For instance, battery charging circuits should be safe (no exposed wires that can short, etc.), and turning the node off for transport should be straightforward. Also, any sharp edges or fragile parts should be minimized since a user might carry a bunch of these nodes in a backpack to deploy.

4.4. System Requirements

i. S1: Microcontroller Specifications – The ESP32-S3 microcontroller must have sufficient resources: at least 8 MB Flash and 512 KB (0.5 MB) of RAM on our development board, which it does (ESP32-S3-DevKitC-1U has 8MB flash, and ESP32-S3 typically has 512KB internal RAM). This is needed to run the compression algorithms and possibly buffering audio data in memory.

ii. S2: Power Hardware – Battery: 3x18650 Li-ion in parallel for ~3.5 Ah capacity at 3.7 V nominal. Solar Panel: 6V, 1W (or larger if testing shows

need). Charge controller: CN3065-based module which outputs 4.2V, 500mA charging. The system should include a voltage regulator (e.g., a buck converter) to get 3.3V for ESP32 and sensors from the battery. Also, appropriate resistors for voltage measurement if needed (though INA219 covers that).

iii. S3: Sensor Interfaces – The I2C bus on ESP32 will be used for ENS160 (gas), HMC5883L (magnetometer), BME280 (pressure). We must ensure each has unique I2C addresses or adjust if conflicts (it appears these have fixed addresses but likely different). The bus will run at standard 100kHz or 400kHz. I2S interface will be used for the INMP441 microphone: the ESP32-S3's I2S peripheral must be configured in the correct mode (typically I2S master, and the microphone as slave). We need pins for WS (word select), SCK (serial clock), and SD (serial data in). These pins should be chosen carefully not to conflict with other needed pins (like strapping pins, etc.). The system requires that the ESP32's I2S can handle at least 16 kHz mono audio (which it can).

iv. S4: Memory and Storage – The file system on the ESP32 (SPI Flash) should be partitioned to allocate ~2-4 megabytes (MB) for data storage (out of 8MB). If more is needed, consider adding an external Secure Digital (SD) card (through SPI interface). For initial prototype, using internal flash with SPIFFS is simpler. Also ensure the lifetime: flash memory has write/erase cycle limits (~104 to 105 cycles), but writing small logs daily is fine.

v. S5: Clock/Timing – The node should have a real-time clock or at least use ESP32's built-in RTC timer for timestamping data. It might not be very accurate over months, so ideally whenever the drone connects, it could send current time to sync the node's clock. System requirement: ability to keep time within say ±1 minute accuracy per day (the ESP32 RTC can drift, but periodic sync can correct). This is needed for data to be meaningful (especially if correlating with other data sources).

vi. S6: Software Environment – Development will use the Arduino framework or ESP-IDF (C/C++) and the code will be compiled using standard ESP32 toolchain. The node firmware should be small enough to fit in flash (which is easy, as 8MB is plenty for our code).

vii. S7: Use Case Throughput – If the node collects, say, a 10-second audio clip when triggered (at 16 kHz, 16-bit PCM, that's ~320 kB of data), compressing it might bring it to maybe 100 kB. The data mule connection at even 1 Mbps can transfer that in <1 second. So that's fine. If using BLE, BLE 5 can achieve perhaps 100-200 kbps in ideal conditions, so 100 kB might take a few seconds, still okay. The system should handle these data volumes within the limited time a drone might have near a node (maybe 10-20 seconds hovering). So, the requirement is that any single transfer should ideally complete within 10 seconds.

viii. S8: Interoperability – The node's software should be compatible with standard data formats: e.g., use Waveform Audio File Format for audio if storing, CSV for numerical logs, or if binary, provide a decoder. This ensures the data can be integrated into typical data analysis pipelines. Also, interoperability in terms of replacing sensors: the code should separate sensor drivers (so if ENS160 is out of stock, one could swap in a similar sensor with minimal changes).

4.5. Use Cases (Tentative)

We describe some representative use cases to illustrate how the system will be used and behaves. These are written somewhat informally as scenarios:

i. Use Case 1: Deploy Sensor Node in Field. Actors: Field Engineer (user deploying node). Pre-condition: Node has been assembled and programmed with firmware; battery is charged. Scenario: The engineer takes the sensor node to a location (e.g., deep in a forest) where it needs to be deployed. They place the node (for instance, mounted on a stake or hung in a tree to expose

the solar panel to light). They flip the power switch on. The node boots up, performs a quick self-test (e.g., blinks LED to indicate all sensors OK), then enters operation. The engineer checks an app or the LED to confirm it's working (maybe the node's BLE beacon can be seen on a phone app showing battery and sensor status). Satisfied, the engineer leaves. Post-condition: Node is now autonomously collecting data. The user might mark the GPS coordinates of the node for the drone's route planning.

ii. Use Case 2: Low-Power Self-Preservation. Actors: Sensor Node. Pre-condition: It's monsoon season, and for several days there's very low sunlight. Battery has been draining. Scenario: The node's PME monitors the battery voltage dropping below 20%. It switches from Normal to Passive mode: it stops power-hungry operations (maybe stops audio recording schedule) and only takes minimal sensor readings (maybe one measurement per hour of VOC and temperature to keep track). It also reduces communication attempts (only does minimal BLE beaconing). This extends the battery life. Another day of no sun and battery hits 5%. Now PME triggers Critical mode: the node goes to deep sleep and wakes briefly every 2 hours just to send one BLE ping and check battery. During this period, some data might not be collected (or maybe it queues but sensing mostly halted). The drone comes by for its routine collection – if the node is asleep, cluster head (and thus drone) might miss it unless the timing aligns. If missed, it's okay; data remains stored. Eventually, sunlight returns, solar panel charges the battery above, say, 50%. The node exits critical mode to Passive, then to Normal mode automatically. It resumes normal sensing and on the next drone visit, cluster head will send not only new data but also any backlog that was not sent due to low power. Post-condition: The node survives the power crisis without dying; no manual intervention was needed. Data integrity is maintained (though perhaps with some gaps during critical phase which is expected).

iii. Use Case 4: Node Maintenance and Update. Actors: Field Technician, Sensor Node. Pre-condition: After 6 months, project team wants to update the

compression algorithm for better performance. Scenario: A technician goes out with the drone or a laptop to where nodes are. Using the BLE interface, the technician puts the node in maintenance mode (could be done by a physical magnet or button that triggers a General-Purpose Input/Output as a signal). In maintenance mode, the node might accept an OTA update file. The technician uploads new firmware (this could also be done via the drone if within range). The node reboots into new firmware, retaining its data memory if possible. The technician also maybe replaces the battery if it's degraded (though within a year it should be fine). The node then continues operation. Post-condition: Node is updated with minimal downtime.

These use cases highlight how the system is expected to function in real scenarios, guiding the validation of our functional requirements.

4.6. Expected Test Cases (Tentative)

We will prepare a series of test cases to verify the system against requirements. Some key tests include:

i. Test 1: Sensor Reading Accuracy – For each sensor, test that the node can read valid data. For example, expose the ENS160 to a known source (like exhale near it to see CO2 rise), use a reference CO2 meter if available. For the magnetometer, move a magnet near it and see if readings change. For the microphone, play a test sound and see if it's detected (in code, maybe computing an amplitude). Expected result: The node captures changes in environment correctly and differentiates sensor signals.

ii. Test 2: Data Logging and Retrieval – Simulate a few hours of operation. Then connect a laptop to the node and retrieve the stored data (via Wi-Fi or physically removing memory). Check that the data is complete, timestamps make sense, and format is correct. Expected result: Data file is readable and contains all expected entries with no corruption.

iii. Test 3: Compression/Decompression – Take a block of sensor data and run it through the node's compression routine, then decompress on PC and compare with original. For audio, measure the waveform difference or at least ensure major features preserved. Expected result: Lossless reconstruction for non-audio data; audio perhaps with minor distortion but acceptable quality vs size trade-off.

iv. Test 4: BLE Beacon Range – Place the node and use a phone or receiver to determine at what distance the BLE advertisement can be seen. Test in open air and with some obstructions (like vegetation if relevant). Also measure how often the beacons come (should match our configured interval). Expected result: Beacon visible at least 30m in open conditions and as configured frequency.

v. Test 5: Power Consumption Profile – Using a multimeter or specialized equipment, measure current draw in active sensing mode, idle normal mode, deep sleep mode, and charging mode. This will verify our power model. Expected result: Deep sleep current in tens of μA, normal active in tens of mA, etc. Confirm that solar can provide enough current to charge given the consumption.

vi. Test 6: Solar Charging Efficacy – Put the node with solar panel under direct sun for a day (or use a solar simulator lamp). Start with battery half full, run node normally. Track battery voltage over time through the INA219 logs. Expected result: Battery should gain charge in sun, ideally reaching near full by end of day, and overnight drops but not below a safe threshold if days are consecutive sunny. If cloudy simulated, see how long it lasts.

vii. Test 7: Mode Switching – Artificially manipulate battery (e.g., use a variable power supply or partially cover solar panel to discharge battery) to see the thresholds trigger mode changes. Possibly use software commands to simulate as well. Expected result: At set threshold (like 20%), the system prints or indicates it went to Passive mode (could check a flag or LED behaviour), and at lower threshold goes to Critical (which we'll know

because it drastically reduces activity). Then raise battery (charge) and see it revert mode correctly.

viii. Test 8: End-to-End Field Test – This is a comprehensive test: Deploy 2-3 nodes in a small area (like around campus or a field). Use a drone or simply a person walking with a laptop configured as the mule to collect data. Do this for a few cycles (like pretend each hour is a day or so). Expected result: The collector successfully gathers data from all nodes with minimal loss. If a node was deliberately shut off or blocked to test self-healing, see that others still delivered, and when the node came back it resumed.

ix. Test 9: Robustness to Failures – Induce some failures: E.g., disconnect a sensor to see if system handles it (should not crash, maybe log an error and continue with others). Crash the ESP32 (maybe via intentional watchdog trigger) and ensure it reboots and continues (we should enable auto-reboot on crash). Simulate memory nearly full and see that old data gets overwritten or warning given. Expected result: System recovers gracefully from minor errors and continues running without manual reset.

5. DESCRIPTION OF PERSONNEL AND FACILITIES

In this section, we outline the individual composition and the available facilities/resources that will support the development of the individual component.

Personnel: Dion Wickrama Arachchi (IT22360496) – Undergraduate student (author of this proposal) responsible for the MS Hardware & Power Design component of the project. Role includes designing the sensor node hardware, writing embedded code, and collaborating to integrate the node with the data ferry system.

Facilities and Resources:

i. Hardware and Prototyping: Components such as the ESP32-S3 development kits, sensor modules (ENS160, INMP441, HMC5883L, BME280), and supporting power electronics (solar panel, CN3065 charger, INA219 current sensor) will be procured directly from local electronic distributors (international suppliers if needed). A personally purchased soldering iron kit, multimeter, breadboards, and jumper wires will be used for hardware assembly and testing. Oscilloscopes and advanced RF measurement gear may be borrowed or rented if critical for validation. A low-cost 3D printer (or outsourced printing services) will be used to create weatherproof enclosures with appropriate openings (mesh for microphone, vents for air-quality sensor, transparent cover for solar panel).

ii. Computing Resources: Development, simulation, and data analysis will be carried out on personally owned laptops and desktops equipped with Python, MATLAB/Octave, or equivalent open-source tools. If additional compute power is needed (e.g., for training compression autoencoders or handling acoustic datasets), additional services will be utilized. Cloud storage and collaboration will be managed using GitHub (for version control), Google Drive/Notion (for project management), and referencing tools such as Zotero or Mendeley.

iii. Software Tools: Arduino IDE, ESP-IDF, and Visual Studio Code will be installed on personal machines. Microsoft Word/LibreOffice (for reports),

LaTeX (for papers if required), and project tracking tools (Trello/Notion) will support planning and reporting.

iv. Testing and Measurement: A bench power supply with current monitoring (personally purchased) will be used for power testing. If deeper analysis is required, specialized equipment such as an Otii Arc can be rented or borrowed. Signal strength and range tests will be done using ESP32 BLE scanning tools, or handheld RF meters available commercially.

v. Logistics and Support: All logistics will be managed independently. Components not locally available will be ordered online. Any mechanical fabrication or UAV repair services can be outsourced to local workshops if necessary.

6. BUDGET AND BUDGET JUSTIFICATION

The project, being primarily a prototype development, requires a budget to procure hardware components and possibly some services (like PCB printing, if needed). Below is an itemized budget of the components for one sensor node and then extrapolated for multiple nodes if applicable. The budget is justified with reasoning for each cost.

i. UAV (Quadcopter with payload capacity for Raspberry Pi/long-range Wi-Fi module) – Qty: 1 at LKR 60,000 (~$200) – Justification: A drone is required to serve as the mobile data ferry, collecting buffered data from sensor nodes in disconnected areas and delivering it to the base station. The cost estimate covers a mid-range consumer/prototyping drone with adequate flight time (~20–25 minutes) and payload capacity for lightweight electronics. This UAV is critical to the system architecture, enabling periodic data collection.

ii. Budget for 1 Sensor Node:

   a. ESP32-S3-DevKitC-1U-N8R8 – Qty: 1 at LKR 1,850 (~$6) – Justification: This is the core microcontroller board with USB-UART, etc. We choose this development kit for ease of programming and its included antenna connector. It's affordable and readily available.

   b. SMA Antenna (2.4 GHz) + IPEX to SMA pigtail – Qty: 1 at LKR 690 (~$2) – Justification: To extend the wireless range, especially for BLE, an external antenna is needed. The price covers one small antenna and the adapter cable to the ESP32's IPEX connector.

   c. ENS160 + AHT21 Air Quality Sensor Module – Qty: 1 at LKR 1,850 (~$6) – Justification: This combo module provides VOC, eCO_2, temperature, humidity. It's a bit high-end hence the cost. Ensures accurate environmental readings (worth the cost for a research-grade output).

   d. INMP441 Digital Microphone Module – Qty: 1 at LKR 840 (~$3) – Justification: MEMS microphone for acoustic sensing. Fairly low cost.

Alternatively, could budget a higher cost if needed for possibly an amplifier or better mic, but this should suffice.

e. HMC5883L Magnetometer Module – Qty: 1 at LKR 450 (~$2) – Justification: 3-axis compass for geology context. It's a very cheap and common module (GY-271 board). We include it for completeness.

f. BME280 Pressure/Temp/Humidity Sensor – Qty: 1 at LKR 950 (~$3) – Justification: To get pressure (for altitude estimate) and cross-check humidity/temp with AHT21. BME280 modules are relatively inexpensive.

g. 18650 Li-ion Battery (3500 mAh) – Qty: 3 (in parallel) at LKR 780 each = LKR 2,340 (~$8) – Justification: Using 3 cells in parallel increases capacity and current sourcing, while keeping voltage 3.7V. We choose a reputable brand if possible (Samsung/LG), for safety and true capacity. 3 cells give ~10,500 mAh which might be overkill; we could do 2 cells to save cost, but we prefer extra due to cloudy days.

h. 3x18650 Battery Holder (1S parallel) – Qty: 1 at LKR 120 (~$0.4) – Justification: A holder to house the cells securely and wire them in parallel. Alternatively, we could solder directly but a holder is convenient for replacement.

i. INA219 Current/Voltage Sensor Module – Qty: 1 at LKR 350 (~$1) – Justification: Allows monitoring of battery and solar currents, important for power management algorithm. Low cost for the benefit it gives in data.

j. CN3065 Solar Charge Controller Module – Qty: 1 at LKR 600 (~$2) – Justification: Tiny module to handle charging from solar. 500mA max fits our 1W panel input (which is ~167mA at 6V ideally). Cheap and effective for single-cell Li-ion charging.

k. Solar Panel 6V 1W (110x60mm) – Qty: 1 at LKR 350 (~$1) – Justification: Chosen based on availability and form factor. Might consider two panels for more power if budget allows; but for now, one panel per node is the baseline.

If tests show more is needed, we might use two in parallel (doubling cost per node's solar to LKR 700).

l. PCB and Miscellaneous Components (wires, resistors, capacitors, enclosure) – Lump sum ~LKR 850 (~$3) – Justification: We may fabricate a custom PCB to mount everything or at least use prototyping boards. There's cost in PCB printing (if outsourced, maybe LKR ~2000 for a small batch), plus connectors, jumper wires, a switch, an LED, screws, etc. Also, possibly a small plastic waterproof box or acrylic to mount it (which might be a few hundred LKR). This miscellaneous category covers those.

Subtotal per Node: Approximately LKR 11,240 (roughly $37). For a group project, if we build say 6 nodes (to demonstrate multi-node network): Total for 6 nodes: ~LKR 67,440 (approx. $223).

Total budget (6 nodes + UAV): ~LKR 127,440 (~$423)

This budget is reasonable for a final year project. We have purposely chosen many parts available through local supplier to minimize shipping delays or import costs. The budget factors in some buffer in the "miscellaneous" category, which can also cover any unexpected component replacements or upgrades (for instance, if we decide to double solar panels or if a sensor gets damaged and we need a new one).

There is little to no expenditure on software (since all development tools are free) – the main costs are hardware. No licensing fees or cloud service fees are anticipated during development; if we were to use a cloud database for final demo, there are free tiers or institute servers we can utilize.

Finally, we note that our budget has an element of scalability: if future work wanted to deploy 10+ nodes, the per-node cost is around $37 which is quite economical compared to commercial environmental sensing systems. Keeping the budget low also improves the potential for entrepreneurship because it means a product based on this could be affordable.

Any remaining budget after building prototypes could be used to refine the product (maybe printing better enclosures or adding a small solar charge indicator).

REFERENCE LIST

[1] O. Tsilomitrou and A. Tzes, “Mobile Data-Mule Optimal Path Planning for Wireless Sensor Networks,” Applied Sciences, vol. 12, no. 1, p. 247, Dec. 2021, doi: https://doi.org/10.3390/app12010247.

[2] S. You, J. K. Eshraghian, H. C. Iu, and K. Cho, “Low-Power Wireless Sensor Network Using Fine-Grain Control of Sensor Module Power Mode,” Sensors, vol. 21, no. 9, p. 3198, May 2021, doi: https://doi.org/10.3390/s21093198.

[3] Marcos, Adriano, and Silva, “Designing Wireless Sensor Nodes,” Lecture notes in computer science, pp. 99–108, Jan. 2006, doi: https://doi.org/10.1007/11796435_12.

[4] G. Anastasi, M. Conti, Emmanuele Monaldi, and A. Passarella, “An Adaptive Data-transfer Protocol for Sensor Networks with Data Mules,” World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a, pp. 1–8, Jul. 2007, doi: https://doi.org/10.1109/WOWMOM.2007.4351776.

[5] D. Bhadauria, O. Tekdas, and V. Isler, “Robotic data mules for collecting data over sparse sensor fields,” Journal of Field Robotics, vol. 28, no. 3, pp. 388–404, Feb. 2011, doi: https://doi.org/10.1002/rob.20384.

[6] Deepak Bhadauria and V. Isler, “Data gathering tours for mobile robots,” 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3868–3873, Oct. 2009, doi: https://doi.org/10.1109/iros.2009.5354343.

[7] Y. Zhang, L. Zhang, P. Han, and L. Zhang, “Overview of low-power strategies for wireless sensor nodes,” Fourth International Conference on Computer Technology, Information Engineering, and Electron Materials (CTIEEM 2024), p. 17, Apr. 2025, doi: https://doi.org/10.1117/12.3058486.

[8] “How to Greatly Improve Battery Power Efficiency for IoT Devices | Analog Devices,” Analog.com, 2024. https://www.analog.com/en/resources/technical-articles/greatly-improve-battery-power-efficiency-for-iot-devices.html

[9] C. Elliott and B. Heile, “Self-organizing, self-healing wireless networks,” pp. 355–362, Nov. 2002, doi: https://doi.org/10.1109/icpwc.2000.905836.

[10] “Wireless Sensor Network (WSN) Architecture And Applications,” ElProCus - Electronic Projects for Engineering Students, Oct. 05, 2016. https://www.elprocus.com/architecture-of-wireless-sensor-network-and-applications/

[11] M. Proctor and S. Webb, “Capturing Bird Calls and Other Wildlife Sounds With Bioacoustics – Noble Research Institute,” Noble Research Institute, Dec. 11, 2020. https://www.noble.org/regenerative-agriculture/wildlife/capturing-bird-calls-and-other-wildlife-sounds-with-bioacoustics/

[12] R. C. Shah, S. Roy, S. Jain, and W. Brunette, “Data MULEs: modeling a three-tier architecture for sparse sensor networks,” IEEE Xplore, May 01, 2003. https://ieeexplore.ieee.org/document/1203354.

[13] N Shylashree, S. Kumar, and H. Min, “Combined compression and encryption of linear wireless sensor network data using autoencoders,” Scientific Reports, vol. 15, no. 1, May 2025, doi: https://doi.org/10.1038/s41598-024-84017-8.

[14] W. Zhao and M. H. Ammar, “Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks,” Jun. 2004, doi: https://doi.org/10.1109/ftdcs.2003.1204352.

[15] S. Chien et al., “Using Autonomy Flight Software to Improve Science Return on Earth Observing One,” Journal of Aerospace Computing Information and Communication, vol. 2, no. 4, pp. 196–216, Apr. 2005, doi: https://doi.org/10.2514/1.12923.

[16] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh, “Monitoring volcanic eruptions with a wireless sensor network,” IEEE Xplore, Feb. 01, 2005. *https://ieeexplore.ieee.org/abstract/document/1462003*.

[17] I. Martin et al., “A High-Resolution Sensor Network for Monitoring Glacier Dynamics,” IEEE Sensors Journal, vol. 14, no. 11, pp. 3926–3931, Nov. 2014, doi: *https://doi.org/10.1109/jsen.2014.2348534*.