# Distributed Systems

**Loretta Christey, Noah Harvey, Jack Pendlebury, Luke Waller**

Tuesday 13th December, 2022

SOFT564 - Software Engineering for Distributed and Interactive Systems.

University of Plymouth

# Contents

# List of Tables



# List of Figures

# 1 Distributed System Specification

## 1.1 Abstract

The term 'distributed systems' refers to the practice of splitting a system's functions across multiple nodes. These nodes can be geographically separated, or different aspects of one more cohesive system. Distributed systems can be made up of hardware and software nodes and see pervasive use throughout a variety of applications from consumer goods to military systems. Some examples are internet-of-things devices; cloud computing software; driver navigation systems; and aircraft fire control systems. This report will detail the process of designing a home-monitoring system using off-the-shelf components, with both hardware and software aspects. The system will monitor alarms, provide weather information, and track the interior temperature and humidity. The home system will also communicate with third party cloud-based systems for off-site data storage, providing security and reliability, as well as the ability to access the system's data from any location. An open-source Android app is also provided.

KEYWORDS: Arduino, Raspberry Pi, Distributed Systems, Bluetooth, Cloud Storage

## 1.2 Smart Home System

This report describes a smart home monitoring and control system with software security. It has IP connectivity through Wi-Fi for accessing devices and nodes, along with various other forms of communication. The system operates autonomously, while receiving user inputs to perform specific tasks, utilising various 'Smart Home' capabilities.

The system will monitor environmental conditions, store the information on multiple platforms and frequently check the networks operation. Various security features have been incorporated into the design, taking a novel approach to the application of standard sensors and peripherals. The network setup is distributed across a set of two branches, one with four nodes, the other with three. Multiple aspects of the Smart Home System are scalable, with the development of the system being driven by ease of replication and implementation of additional nodes.

## 1.3 Network Topology

Environmental nodes will monitor rooms, these nodes form part of a scalable subsystem that can be expanded to cover all rooms within a building. In theory we could add up to fifty additional environmental nodes but with hardware constraints and testing we envision the system would be limited to the region of twenty additional nodes. We have proven this capability by advancing a network branch to run four layers deep. If time allowed, we would also look to add additional buildings to the system by replicating the entirety of the project.

No single, central node has the responsibility of monitoring the entire network. The systems topology, shown in Appendix B means that each node can test its own neighbour, working its way up to the base node. This approach also allows for easier implementation

of new platforms [1] and their respective nodes. For testing purposes, each root node can echo down its respective network path to test responsiveness with low complexity. The short routing distance allows for ease of fault diagnosis, without effecting the entire system, and minimal latency.

# 2 Overview and Description of The System

## 2.1 Node Summaries

The table in Appendix C outlines the hardware connected to each node, and its forms of communication. The hardware within each platform was chosen through group mind-mapping meetings, alongside the capabilities of each microcontroller. The ESP8266 was the only board capable of Wi-Fi connectivity, and so the role for that board was always going to involve obtaining the time, GPS and weather data. It was then developed to upload information to a Google Sheet, shown in Appendix G.

## 2.2 Microntroller Optioneering

| Specification | NANO | UNO | MEGA | ESP8266 | Pi Pico |
|---|---|---|---|---|---|
| Flash Memory | 30KB | 32KB | 256KB | 4MB | 2MB |
| RAM | 2KB | 2KB | 8KB | 64MB | 264KB |
| Digital I/O Pins | 14 | 14 | 54 | 17 | 26 |
| Analog Input Pins | 8 | 6 | 16 | 1 | 3 |
| I2C | YES | YES | YES | YES | YES |
| SPI | YES | YES | YES | YES | YES |
| Bluetooth | NO | NO | NO | YES | NO |
| Wi-Fi | NO | NO | NO | YES | NO |
| Output Voltages | 3.3/5V | 3.3/5V | 3.3/5V | 3.3V | 3.3V |
| Clock Speed | 16MHz | 16MHz | 16MHz | 80/160MHz | 133MHz |

Figure 1: Microcontroller Optioneering

The Raspberry Pi Pico was deemed an appropriate board to use for basic functionality of LEDs, buttons and the sound detector due to the limited documentation available but high number of inputs that could be used for standard peripherals.

The Arduinos were used for more complex functionalities and computational tasks as they have extensive support and hardware tutorials. In addition to this, the team had more experience with the C programming language than Python/Micro Python used on the Pico.

The ESP was used as the Wi-Fi module, as it was the only microcontroller with the ability to connected to the NTP server and Weather API. Due to the limited number of GPIO pins, the ESP was used to display data, rather than acquire it through various sensors.

## 2.3 Communication Methods

Communication across the system will be monitored by the main base node, with other nodes reporting back to their branch leader. The branch leader will sit between the controller/extension nodes and the main base node, shown in Appendix B. They are a link layer between the main node and the controller nodes. This allows the main node to check the branch leaders more regularly, whilst the branch leaders each have their own nodes to check. This eleviates strain on the system by allowing each node to spread the load more evenly. The Wi-Fi node will be the branch 1 leader; the Back Door node is the branch 2 leader. All branch leaders that communicate using NRF modules can become the base node, taking over responsibilities of the node above it within the network tree.

The communication signals from the base node and branch leaders will control data transmissions. Each node within a branch will communicate when contacted by its own leader. For example, if node 21 on branch 2 wishes to contact a node in branch 1, the message will be echoed up to it's leader and passed along to the node 2 leader. Once this has occurred, the message will be relayed from the branch leader to the appropriate node. The figure below shows the receiving switch case statement that has been implemented throughout the system. The figure in Appendix H below shows the receiving switch case statement that has been implemented throughout the system.

The network uses four modes of interaction: parallel wired communication, Bluetooth, Radio, and Wi-Fi communications. A mobile phone app has been implemented to send commands to the system via the Bluetooth node. A Google Sheet is incorporated for storage of data. This allows for multiple concurrent users to manage and adjust the data without causing significant strain on the system. LEDs, and other small scale output devices, are used as proof of concept that the system could be used to control standard fixtures within the home.

## 2.4 Modes of Interaction

The system can be controlled by an IR remote, a phone app and buttons. The LCDs display the information that is most valuable to the user automatically. If the user interfaces request alternative data, or there is a security alert, the LCDs will display those messages instead. The 'Smart Home System' data is backed on the Google Sheet, which can be accessed by users via their own account. The entire breakdown of the modes of interaction are listed in Appendix C.

# 3 Overview of Code

## 3.1 Flowcharts

Flowcharts are referenced through the code, the full system overview is in Appendix J.

## 3.2 Network Management Code

Similarly to the TCP/IP the internal networking structure of the distributed system uses an address for both the transmitter and receiver [2]. The TCP/IP protocol is a good example networking protocol as it is universal across all platform architecture and can run on any compatible hardware. Although using a condensed version of the TCP/IP protocols, the distributed systems communications protocol has similar advantages and disadvantages.

### 3.2.1 Advantages and Disadvantages

**Advantages**

- Networking works over multiple system architectures
- Supports multiple routing protocols
- uses client-server architectures that are highly scalable
- Systems can operate independently
- System is lightweight and doesn't place unnecessary strain on the network

**Disadvantages**

- Networking can be difficult to set up and manage
- This system does not guarantee delivery of packets
- Not easy to replace the protocol

Despite these system wide limitations, this lightweight version of TCP/IP was utilised for the final network management solution. It provided flexible functionality to be hosted on a range of architectures and systems. The highly scalable nature of this protocol meant it was a suitable for the distributed system.

This was developed during week 5 of the project, which can be seen in §4.2.

### 3.3 Data Scrambling

For simple data security we decided on XOR encryption. XOR encryption uses a predefined private key to scramble the data being sent.

The code and output in Appendix E shows how XORing a value is a reversible process, so the same key can be used for encrypting and decrypting data. Therefore, each of the nodes would only need to have a local copy of the private key to be able to decrypt the incoming data and to encrypt the outgoing data. This method makes the transmitted data unreadable to anyone without the private key. XORing the data works on both numbers and text with only minor tweaking being required to allow universal functionality.

## 4 Planning and Organisation

As a group, progress and reflection meetings were carried out at the start of each lab session. The team discussed the lecture material and application of previously unconsidered items that arose from the lecture. The team discussed current design issues and challenges and planned the week ahead. The 'lab session breakdown' program formed the basis of the weekly development tasks, aligning them with the module criteria. Summaries of weekly goals, group work and contributions are documented in Appendix A, key points are discussed in the following weekly breakdown section.

### 4.1 Skill Identification

During the first weeks of development, the team established an understanding of skills amongst the group, allocating certain tasks to the appropriate owner.

Jack focused on Pico and sound sensor, speaker, sensors and alarm system. Luke focused on communications, looking at Wi-Fi, Radio, Bluetooth, GPS weather and the Web server. Noah worked on RFID tags, the IMU, LCD, network topology and various environmental sensors. Loretta looked at the IR sensor and remote. Over time the roles evolved. Jack focused on additional forms of communication from the Pico to the network, Luke focused on the creating user friendly platforms, Noah focused on the NRFs communication protocol and Loretta concentrated on the weekly focus for the group and reporting.

### 4.2 Weekly Plans

An overview of the weekly progress is detailed below, with more detailed breakdowns in Appendix A.

**WEEK 1**

The team collected the kit of components and wrote down an initial list of possible distributed system projects that would gain maximum marks according to the marking rubric. This was focused around creating a minimum of 5 interconnected platforms with various functions and peripherals. The lab work for the first week involved testing hardware individually for proof of operation, gathering libraries and datasheets.

The second task was exploring the methods that could be used to communicate between the various microcontrollers in the kit. This related to the way in which the NRFs would communicate and how other interfaces for users could control the system. The first brainstorm included the discussion of additional features, such as parallel, Wi-Fi and Bluetooth communication.

**WEEK 2**

A list of possible distributed system projects was formed, and through analysis and discussion, the less viable options were discarded. The systems which met the requirements detailed in the brief and could be completed within the module timeframe were explored further.

The peripherals used influenced the decision, and an environmental monitoring system was sketched out. The idea was based on around proving that the system could distribute tasks, record data, and carry out computational methods concurrently.

**WEEK 3**

The team began to develop a 'Smart Home' project with various capabilities ranging from security features to creative methods of human interaction using interactive components. The system was beginning to complete a variety of functions using various peripherals. Complications with using certain devices on the same board as the NRFs began to arise, details of which have been documented in the logbook in Appendix A. Ultimately, these situations helped to formalise roles and begin a network wide development plan.

Project Proposal: A 'Smart Home System' with a security platform, intruder sensing and scalable nodes to monitor the environmental readings in various rooms.

**WEEK 4**

A focus on building the platforms, setting milestones, and beginning to populate the skeleton of the report with what the team had experienced so far. Week 5 required a presentation of the project, so the team began to identify the areas of the project that relate to the key principles of a distributed system, the breakdown of how the project met various principles is in Appendix A.

Design and Build:

- Finalise design, platforms, above details and proposed system.
- Focus on resource sharing on LCDs and web page development.
- Discuss how the team will carry out quality assurance through operating each other's code, or asking other people to use the platform as they deem appropriate.
- Create testing schemas, saving in Appendix A.

**WEEK 5**

The team presented the 'Security, Smart and Environmental Monitoring Home System' project to Dr Toby Whitley and James Rodgers. Feedback was implemented into the next iteration of the design; the development plan is listed below:

Adding new nodes to the system would be achieved by using an IR remote to set the node ID. As this function could be carried out by the user, the element of transparency would still be withheld. Adding extra environmental nodes into the system would be achieved during a 'test network' function. A system wide ping would be dispersed to

every NRF device on the same channel as the home system. The data obtained by the additional platform would be used to derive its purpose.

If the control panel goes down, other platforms can take over the network maintenance functionality by taking on the base ID of '00'. When the other platforms receive a transmission that begins with this ID, they acknowledge that module as the leader and respond accordingly.

**WEEK 6**

The team felt the design and plan for development was on track, and the work on prototyping and communications development continued. The weekly discussions tended to focus on specific requirements of a distributed system while the challenges were often related directly to one of the principles.

Technical Issues:

The data needed to be encrypted, and the application of the system within the user's home and the security and protection of sensitive data was discussed.

The NRFs were temperamental, and the radio communication was often unreliable. The diagnosis routine for solving the intermittent issues is detailed more in test schemes in Appendix A.

The power consumption of each platform was discussed. The implementation of motion sensors and sound sensors to 'wake' the displays were discussed but noted as a future development. The operation of the peripherals and the feasible data acquisition rates were adjusted to reflect how often a user needs to measure the temperature, and how interrupts/timers could be used instead of blocking delays.

**WEEK 7**

The testing of individual nodes and the ability to request data from neighbouring nodes began. This started with using a mobile phone to communicate with the Bluetooth node and relay a message onto the main base node. It was at this point that the testing schema began to test extreme conditions to evidence concurrency.

The system was designed to hide the way in which each node is accessed, concealing the background data acquisition and storage on the Google Doc. The system was adjusted for application scenarios, where multiple houses could access their own data but not their neighbours.

**WEEK 8**

Dr Toby Whitley observed the 'Smart System' developments to date, and highlighted suggestions related to the following:

Scalability tests involved adding the environmental node at various points throughout the network and sharing the data acquired. By conforming that replicated nodes could operate concurrently with the entire system, with all of the readings shared to the base node and various neighbours, the node evidence scalability.

Intermittent or permanent redundancy of nodes was adjusted to monitor on a timer, 'pinging' all the network's nodes and acknowledging failures to respond. The system continues to function if a node goes down. The user is only notified if a platform fails to respond three time in a row, indicating a serious issue.

The pervasive components, smart phones, and Google docs, can discover new nodes and self-configure. Processes and communications execute when required and independently of other nodes.

**WEEK 9**
Evidence of tests were documented, with recordings placed in the logbook in Appendix A. A plan, setup, test and review approach was used, with the stages defined in the Gantt chart in Appendix F to prove the system was capable of the following:

Design and Build:

- Splitting data across multiple devices to reduce the amount of network I/O for certain algorithms, using timer, clocks, and events ordering/interrupts to prevent blocking/spinning
- Replication: Sharing information to ensure consistency between redundant services. Each node periodically chooses a random set of neighbours with which to exchange information. Information propagates through the system quickly, using the gossip approach
- Agreement: every correct process must agree on the same value and Integrity: every correct process decides at most one proposed value

**WEEK 10**
The team discussed final stages of prototyping as all the nodes were in full operation, although a few minor adaptations were required. It was during this week that the testing schemas were used extensively.

Tha alarm conditions were defined, data was stored permanently on the Google Sheet and various inputs were mapped to nodes throughout the system to prove both concurrency and scalability. It was possible to add the environmental sensor to various locations within the network.

**WEEK 11**
The report and relevant diagrams were updated to reflect the route the project ended up taking, as some changes were made to the network topology. The team were able to get together and discuss future developments, and things that would be done differently next time. Boxes were 3D printed for the finalised platforms.

The critical reflection section was completed in more detail to evidence how well the 'Smart Home System' performs as a distributed system.

**WEEK 12**
Additional resources were added to Appendix J of the report to illustrate the final flow, showing how a user input is processed and how far the data can travel throughout the network. Parts of the report were used to create the final presentation.

# 5 Distributed System Requirements

Early in the project it was decided a 'Smart Home and Security' would be the created system. This would include a range of 'nice to have' additions if the timescale allowed. Initially, the design requirements were considered. Multiple platforms, distributed communications, and real-world sensors were all considered whilst evaluating further developments such as voice recognition, Bluetooth connectivity to a smart phone app and GPS for the acquiring local weather data.

## 5.1 Summary Reflection of a Smart Home System with Security

The proposed distributed system incorporates a scalable subset of environmental room nodes with communication over a single system middleware, with characteristics stated in the descriptive table in Appendix K. Each system will be scalable to a street or larger area of buildings. To be classed as a distributed system the system must demonstrate ability to share data and resources, flexibility, openness, scalability, communications, and redundancy. The cost of the overall system must also be considered throughout development.

The initial proposal outlined a set of five distributed nodes able to communicate across a range of media. These included radio communications across all the nodes, Bluetooth, parallel, and Wi-Fi. These communication methods were outlined to fully utilise the OSI protocol stack. While not all nodes will make use of the full stack, the whole stack is used when taking a holistic view of the system. The OSI protocol stack can be seen in the table in Appendix L. Many of the architectural decisions of the design were based around the OSI stack along with the brief set out for the project. Ensuring that the nodes communicated effectively using secure transmissions where possible was paramount, security being a vital trait of a successful distributed system; this was prioritised by the development team. Radio modules were used as the primary communication method, allowing for long-range and stable communication between the nodes.

The front door node is the primary node, commanding all communication between the other nodes. This node performs setup routines for the other nodes, allocating device specific IDs. Once these various devices have been set up, they can operate independently without need for the main node to continue interaction for most standard communications. This node includes an LCD to display various information to the user. It has a remote control that is used to interact with the system, allowing for the addition of other nodes as well as other functionality such as disabling alarms. The main node also incorporates an inertial measurement device that allows the system detect tamper attempts. Excessive movement or vibration, such as that that would be caused by attempts to access the system without authorisation, will trigger the alarm. Once the alarm is set it can be reset using either the Bluetooth application or the infrared remote control. When in alarm mode the LCD communicates this to the user. When the alarm is disabled the LCD continues to function as normal, displaying current system status and any notifications.

The back door system utilises both radio communication and Wi-Fi connectivity. The back door uses the ESP 8266 board along with an RF module and an LCD screen. It allows the users to store data within a Google Sheet which can be accessed from any device with a web browser. The ESP node also retrieves weather data through the OpenWeather API, and the current date/time from an network time protocol server. This weather and time data is displayed on the LCD. The GPS coordinates used to collect the weather data are obtained via Bluetooth using the android phone application. If no new GPS data has been obtained when the system is power cycled it uses the last collected data which is stored on the device's EEPROM, reducing the rate that the GPS coordinates need to be updated. As a failsafe option, the GPS coordinates can be inputted manually using the infrared remote control. This can be used if the user has no access to an Android phone, or the app is no longer distributed. The Google Sheet used to store all the distributed system's data is an example of a highly scalable system. There is no real limit to the

number of users that can access the Google sheet at any given time. Atomicity is handled server-side, allowing multiple devices to both read and write to the storage. The third-party server hosting allows the system to be utilised across multiple geographic locations and be expanded upon with relative ease.

The environmental node utilises a temperature, pressure and humidity monitoring system that communicated with the system using the 2.4GHz radio signal. These boards are simple to create and implement into the system. They only store their own data and host no responsibilities for effective communication between the other nodes. This allows the environmental node to be a highly scalable piece of hardware for use within the system. Keeping this model affordable was important to allow the user to upgrade and expand upon the systems without significant cost. This is one of the driving points of distributed systems; to allow users to upgrade their system slowly over time whilst not losing out on functionality. This allows one core system to be utilised in every situation and for more subsystems to be added depending on specific user's use case.

The Pi Pico acts as an additional add-on board to the environmental monitor. It allows the user use of a novel clap-based interface to interact with their system. The Pico communicates using a parallel bus to the environmental sensor, allowing installation on already existing environmental monitoring boards. The Pico board was chosen for this purpose because it had limited support for wireless communications but good hardware support for on-board processing of sound inputs. Given the low cost and ease of installation of this module it is possible, much like the environmental sensor, to be added to any room desired within the distributed system's scope.

The final piece of discrete hardware is the Bluetooth android app connection node. This allows the user to interact with the system using their smart phone. Unlike the Google Sheets interface, the smart phone app can only be utilised within the range of the Bluetooth module. This means that the app can be used for more sensitive data analysis and for features that warranted elevated security privileges. The Bluetooth interfacing node utilises a radio module to communicate with the rest of the system. The actual Bluetooth connection is done using an HC-05 Bluetooth module. Given the low cost of the Bluetooth module, multiple nodes could be placed around the environment to allow multiple access points for the app. Multiple users could log into the app and be able to access differing features depending on account status.

The final system was able to communicate across a variety of nodes and range of media. These nodes shared data amongst themselves for alerting the user and storing important information in various formats such as on Google Sheets. This allowed the user to access this information anywhere that it is was required, providing the user had access to a device capable of displaying webpages. Given that major institutions such as the NHS used Microsoft Excel to store patient data [3], Google Sheets would prove to be a useful method for storing and accessing the systems data. To reduce the load on all the nodes and the perceived latency from the system, the nodes will share resources to allow multiple nodes to respond to requests of data, this allows the main node to remain free in case of a system error instead of acting like a router for the rest of the system.

The system is flexible and allows for outages of individual nodes to be handled effectively by the system. However, if many nodes become unresponsive to the system, function will be compromised. If the main node goes down, there will be a backup node

that can take its place. This ensures some level of redundancy within the system. Despite this, the system is not infallible, if the backup node also goes down the system will cease to function.

Throughout the projects, the costs have been kept as low as possible. This is in effort to try and make the system as affordable as possible. It was found throughout this project that is viable to use a multitude of less powerful microcontrollers as opposed to larger more expensive boards. This helps keep the design modular and cost effective.

The scalability of the system is clearly demonstrated using Google Sheets for data storage and monitoring. The spreadsheet used can be viewed by anyone that has access to the sheet and a device that load a web browser. The biggest limitation of Google Sheets is that only 100 users can edit a sheet at any one time [4]. This is not a realistic limitation for a small-scale house or even street-based system. Another example of scalability is the ability to register and install new environmental nodes into the system. This would allow for a change of layout to the system and expansion of the system without requiring a full reconstruction of the system.

The communications between the nodes are vital to the system functioning. Each node is capable of independent operation, within the bounds of its own self acquiring data. This system relies on effective communication to allow for complete operation to occur. Other features, such as the Bluetooth functionality, rely on communication to all the users to interface with the system. This system can transfer data around itself effectively and operates as expected under challenging conditions.

The redundancy is built into the system. If one of the environmental nodes goes down, the base node logs the error and continues normal operation. Once the node comes back online, this is logged by the base node and normal operation will continue. If the base node goes down, one of the second control nodes will attempt to take over control and allow the system to remain operational. If this fails, the system would be non-operational. Despite this there is still a suitable level of redundancy for a non-safety critical device.

# 6 Critical Reflection

Reflection has kept the project scope concise, helping the team members to relate the functionality of their platforms back to the criteria of a distributed system.

## 6.1 Planning and Organisation

The overview of Week 3 focused on assigning platforms and peripherals to those with the relevant skills. At the end of the project, the team felt that their skills were utilised and their contribution to the group involved carrying out tasks they enjoyed.

## 6.2 Roles and Contributions

Jack: 'Back Door Panel' with the Pi Pico detecting spikes in noise levels (such as intruders or clap to wake capability), LEDs/alarms for user observation and parallel com-

munication with a Nano. The Nano would be the scalable module that, if replicated with an NRF for radio communication, could be placed in various rooms around the home or outside within a case.

Luke: A user controlled hub capable of acquiring, displaying and storing data. The NTP was used to maintain the time throughout the whole network. GPS data was used to retrieve the current weather for the location, which was then displayed on the LCD. The Bluetooth chip formed a connection with a mobile phone, creating new user interface controls, with an additional element of security added due to most phones being password protected.

Noah: The MEGA was used in conjunction with an IMU and RFID scanner. The functionality included a 'tamper alarm' triggered by movement of the control panel module that these components formed. To disable the alarm, the fob key was presented to the scanner. An LCD was used to display the status of the alarm. Two NANOs were being used to test the communication between two NRF modules while the transmit and receive functionality was being adapted to operate with multiple nodes.

Loretta: Notes were taken throughout the development process, focusing on critical reflection, and preventing 'scope creep' in order to meet the various requires for distributed system within the timeframe. With so much hardware, and so many variations of nodes to create, keeping the system simple yet effective was vital. Achievements and challenges were discussed weekly, with individual members talking about their platform in depth.

The team worked well together with peer support, development and effort being equal on all sides. Individual members gave a different perspective and new ideas to the overall system. Early in the project the team agreed to use several components from their own collections to add to the system and later to add a second kit. On reflection, it would have been better to create a component list detailing all the components available to use in the system and then plan accordingly. The team's reflection involved discussing how the project would have benefited from a planned implementation, rather than the ad hoc implementation that took place.

Class lab time was limited to three hours each week and it was difficult to find time outside these hours due differences in timetables. The project would have benefit from more lab time, or team allocated time, written into the timetable. This would have allowed the team to discuss the needs of the project in more detail alongside working on the challenging parts of the project together.

## 6.3 Decision Making

As part of the group's regular reflection, plans were made according to what could achieve during the time allocated. If problems arose that would take too long to fix or implement new solutions for, and the team felt that there were other elements of the project that met the same criteria, the causation would be eliminated from the project. Such situations have been discussed in the future developments and challenges section. Items that were 'nice to have' additions, were included in a list that would be revisited if time allowed. Most of the development issues arose around the implementation of the communications structure, an important aspect of the project, and so more time was dedicated to making

the network reliable.

Decisions about installation of the system, wired or wireless was ongoing. On reflection, it would have been beneficial to review the environmental impact of using a wired system verses a wireless system. The considerations of battery components, power usage and cost of wiring the system, either as the building is built or retrofitting to an existing building could have been review and researched in more depth. Due to the nature of the project being a 'proof of concept' these details have been covered in the future development section.

## 6.4 Technical Implementation

Weekly team meetings provided a good basis for discussion, reflection, and updates. On reflection, it may have been better to make weekly meetings more formal by requesting a summary of progress from each team member. This information would have enabled the team to find challenging parts of the project in a timely fashion. The team felt that the initial goals set at the beginning of the project had been achieved, but that with more formal structure for reporting of progress, the team could have worked with more synchronicity and a greater understanding of the overall development of the system.

The way in which the network of nodes communicated was discussed weekly and were honed as the project progressed. With every iteration of the hardware design, the communications were redesigned to best fit the requirements and adapt to managing a larger network. In the time frame, the group felt that this was the best method for implementation and with reflection, still felt that this was the case. With more time to develop the project, the network would have been developed to incorporate other communication architectures, rather than using NRFs. The team felt that the NRFs and the maintenance of them was time consuming due to their unreliability. The radio modules had problems with sensitivity, power regulation and interference with other modules, such as the Bluetooth module.

## 6.5 Distributed System Achievements

**Resource Sharing**

Each branch can request data from its own nodes, or request data from the other branch's nodes to create a backup. This process can be triggered by the base node as a 'backup data' function, or by each branch as and when they require it. This data can be displayed on LCDs incorporated throughout the system, saved to the Google Sheets document, or simply passed along to the relevant node. The time and weather obtained through branch 1 is shared with the rest of the system, as well as displayed to the user when requested.

An IMU is housed within the control panel, concealing all wiring from intruders. The time and date of the trigger is stored across multiple platforms. Additionally, the environmental nodes store their data on various boards throughout the network. This is to enable the system to analyse and process the data elsewhere. Each board can request the functionality or processing power of not only its neighbour, but an entirely different branch if it needs to. For example, the control panel does not handle all of the network checking

routine by itself - it dissipates the task to branches, where they check the response of their nodes concurrently.

**Openness**

Additional phones can be added to the system to control the environment and alarm detection. New users can access the Google Sheets on any device with a web browser, if they have an account with permission to do so.

**Heterogeneity**

The Smart Home System contains an UNO, multiple NANOs and an ESP board programmed in C. There is an additional Pi PICO that was programmed in micro-python. The phone app, being on a new operating system, was written in Java. The base, leaders and attached nodes have different versions of communication protocol uploaded due to their roles within the system.

**Transparency**

Users will interact with the system via mobile phone app, IR remote and sound detection system directly. They can see information displayed on the LCDs, but the rest of the hardware will be encased in boxes. Testing, alongside other acknowledgements, have been sent via serial monitor during development. Environmental monitoring, control and transmissions will be hidden. Additionally, an IMU is encased in the control panel. The user cannot see this, an intentional design characteristic, so that if intruders attempt to disable the homes security system by tampering with the control panel, the owner is instantly notified.

**Fault Tolerance**

Jack's written test benches, Noah and Luke testing hardware testing schemas. This was to make sure that each piece of hardware worked as intended, before implanting it into the wider system. Network test functions were written to send a message to leader nodes, instructing them to check that all of the nodes within their own branch are active.

**Concurrency**

Nodes will work independently but communicate data as required. Additional devices and operating control apps can be added to the system and will work concurrently. This has been tested throughout the project, as detailed in the communications testing schema, by requesting tasks or data from different nodes in various orders and at the same time.

There is a possibility that several clients will attempt to access a shared resource, such as the Google Sheet and the mobile app, at the same time. Any object that represents a shared resource in a distributed system must be responsible for ensuring that it operates correctly in a concurrent environment. The network, in its current state, has been designed to only connect to one mobile phone at a time. The Google Sheet uses 'smart overwrite' features, alternatively, the append function where no data can be overwritten under any circumstances. Certain accounts with access to the Google Sheet can have different privileges, such as read only.

# 7 Testing

Week 9 began the testing phase of the project. The team met to decide the draft test plan and timings, creating a Gantt chart checklist. The team agreed to meet regularly to update each other on the progress and any issues arising from the testing procedures. Testing of nodes was completed individually on each node, and then network wide. 'Top down' testing will be completed across the system to show effectiveness and efficiency of the distributed communication system.

## 7.1 Testing Schemas

As part of the test plan, a Gantt chart in Appendix F was created and a list of tests that were expected induce an error were included in the plan. The aim of the plan was to test every aspect of the distributed system from a bottom-up perspective. Once the bottom-up plan was achieved the system would then be tested top down to ensure communications were effective across the system.

The testing schemas can be found in Appendix A, within the testing folder.

## 7.2 System Errors

When communicating with NRFs, it was important to manage the activity of the network as whole while receiving override commands from the users. For example, the system runs a 'test network' routine every ten seconds. Running this test too frequently fills the communications pipeline with unnecessary traffic. This often interrupted the commands coming from user submitting their own request. The solution to this problem was giving certain devices, and their respective messages, higher priority. This was handled by including codes at the start of a transmission so the receiving node could respond to codes in different manners, raising the appropriate flag for priorities. Testing was carried out by monitoring, figures in Appendix I when data packets were being transmitted and receiving, observing the NRF's behaviour.

## 7.3 User Inputs and Peripherals

User inputs, such as the mobile app and IR remote, were tested by being operated by different team members, and stand-ins for users who were unaffiliated with the development. Videos of the testing can be found in Appendix A, along with the outcomes and respective improvements in the testing schemas. The general performance of the remote was successful, with a low response latency; functionality of each button was assigned to short tasks or raising flags. Implementing the mobile app was more challenging due to having the intermediate form of communication, the Bluetooth chip, before passing signals along the NRF network. The NRFs would often interfere with the Bluetooth module, and so the hardware was relocated to a simpler module.

The LCDs worked during the first set up and have operated smoothly throughout the project. The designation of the LCDs to microcontrollers was relatively simple, given that

the microcontroller optioneering table in Appendix D outlined the necessary information. Given the ESP's GPIO limitations and Wi-Fi capabilities, the pins were used to transmit the information obtained from the NTP and weather API to the user via LCD.

The 'Control Panel' required an LCD to serve its capabilities of being the user facing node, with elements that were heavily focused on ease of use and instructing the system to carry out various tasks. The ability to run network wide checks and add new nodes was a requirement of the distributed system the team set out to create. During the project, the ability to run such checks was used for 'start up' cycles for when team members connected their nodes to begin other tests. Every test session was able to begin with each node testing their hardware through their own set up routine, followed by network wide 'ping' checks to ensure functionality before further tests were carried out.

The Bluetooth module was originally connected to the ESP but it often caused interference that prevented the entire module from operating. For this reason, the Bluetooth chip was placed onto a separate Nano as an additional platform, meaning that interference would only effect the Nano. It would not cause the ESP's other functionalities to cease, making the system more reliable.

## 7.4 Parallel Communications

To exhaustively test the parallel communications used from the Arduino Nano and the Pi Pico within the environmental sensor, a testbench was constructed using a Pi Pico to simulate an incoming parallel transmission.

Upon button press, the parallel testbench would mimic the actions taken by the transmitting side of the parallel communications bus. In the first version, the input is set using a set of slide switches, allowing for the tester to easily change the input without having to change any code. This initial implementation only uses two data lines, with four possible combinations of values, meaning the switches are fit for purpose. This would not be the case for larger numbers of data lines, with the numbers quickly becoming unfeasible to manually test.

To remedy this, software for loops can be used to automatically generate exhaustive ranges of input data. The software for loop visible in figure 2 shows code to generate an exhaustive set of inputs for an n-bit wide bus. The received transmission data is reported on the serial monitor connected to the Arduino. A correct result will show numbers from $0 \rightarrow 2^n$, where $n = data\ width$, in order on the serial monitor. This output makes obvious any transmission errors, as numbers will be out of order, or otherwise incorrect.

```
while data <= (2 ** data_width):
  data += 1
  print(data)
```

Figure 2: For loop used for data generation

This testbench could further developed by implementing automatic results verification. By having the Arduino echo results back to the Pi Pico, transmissions could be verified automatically and repeatedly, with no user input required.

## 7.5 Power Loss

Whilst loss of system power will cause the system to stop function, data is retained in non-volatile storage. Weather data is backed up to the Google Sheet, and future versions will also back up data regarding alarm trigger events and envionmental data.

GPS data retrieved from the Android app is saved into EEPROM, and if no location data can be found, will be retrieved from memory.

Partial power loss events will also effect system functionality. The diagram in appendix J.2 shows which aspects are critical to the operation of the system, and which aspects are safe to have disconnected.

## 7.6 Power Consumption

The reference material used within this subsection is included in Appendix A datasheet folder.

NRF testing revealed that the peak power consumption during a transmission of four bytes was 28.4mA. The datasheet in Appendix A states that the average power consumption is around 12mA, with a idle supply current of 900nA, making the NRF an ultra-low powered radio module.

The ESP-8266 datasheet states that, based on a 50% duty cycle, it consumes 70mA per hour, which meant that this platform had to be mains powered, due to the fact it would drain batteries within a day. The ESP has its low energy wireless data transport protocol, Message Queue Telemetry Transport (MQTT), that could not be utilised in this project due to only having one operation ESP.

The UNO, operating as a standalone board, consumes around 42mA. With the additional hardware connected, the consumption rose significantly. It consumed 20mA for the LCD with the backlight on, although later in the project the operating current was reduced to 1.2mA by switching off the backlight when idle. The IMU had a power consumption reading of 0.03mA.

The UNO consumed 65mA in total, the three NANOs consumed 40mA, the EPS consumed the most as it often hit 70mA and the Pi Pico also consumed 70mA. These measures of mA per hour resulted in a system wide power consumption of 0.36A.

## 7.7 Future Developments and Improvements

To save power, the sound sensor would have 'clap to wake' capability introduced. This would switch on displays, such as the LCD, and while nobody is home they can remain in an idle state. This would also provide additional functionalities, for switching on lights, or taking readings from all the environmental nodes within the system as and when the user requests it.

Backing up more data to Google Sheets, such as the time and date of alarm triggering events and the data acquired from the environmental sensors. The ability to request data from your home while you are away would also be incredibly beneficial to the user, especially if they can receive mobile alerts that a break in has occurred in their home.

When no Wi-Fi connection is available, the system would benefit from displaying the data acquired from the sensors placed throughout the home, and outside. Pressure could be used to predict rain and still present a weather forecast to the user, and so on. To allow for extra nodes to cover a larger distance using 'repeater stations' would be implemented to extend the reach to the distanced nodes.

To improve the reliability of the network, and introduce a larger number of nodes to test extremities. LoRaWAN [5] would be an excellent low power alternative that would also allow wireless integration with the internet at every node. The 'chirp' pulses are encoded, robust against external disturbances and perform well over larger distances.

# 8 Acknowledgements

The team would like to acknowledge Dr Toby Whitley, for information provided throughout the lectures and James Rogers, for assisting within the timetabled laboratory sessions.

# References


[1] D. K. Pradhan and S. M. Reddy, “A fault-tolerant communication architecture for distributed systems,” *IEEE transactions on Computers*, vol. 31, no. 09, pp. 863–870, 1982.

[2] M. E. Shacklett. (Jul 2021) What is tcp/ip? [Online]. Available: https://www.techtarget.com/searchnetworking/definition/TCP-IP

[3] BBC. (Oct 2020) Excel: Why using microsoft’s tool caused covid-19 results to be lost. [Online]. Available: https://www.bbc.co.uk/news/technology-54423988

[4] Google. (2022) Share and collaborate on a file with many people. [Online]. Available: https://www.bbc.co.uk/news/technology-54423988

[5] The Things Network. (2022) What are lora and lorawan. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/

[6] Amazon Web Services, Inc. (2022) What is restful api? [Online]. Available: https://aws.amazon.com/what-is/restful-api/

# A Project Links

OneDrive Folder - Testing Schemas, Pictures, Videos and Code
GitHub Repository - Code, Report and Flowcharts
SOFT564Z Notebook - Minutes, Plans and Development Notes

# B Network Topology

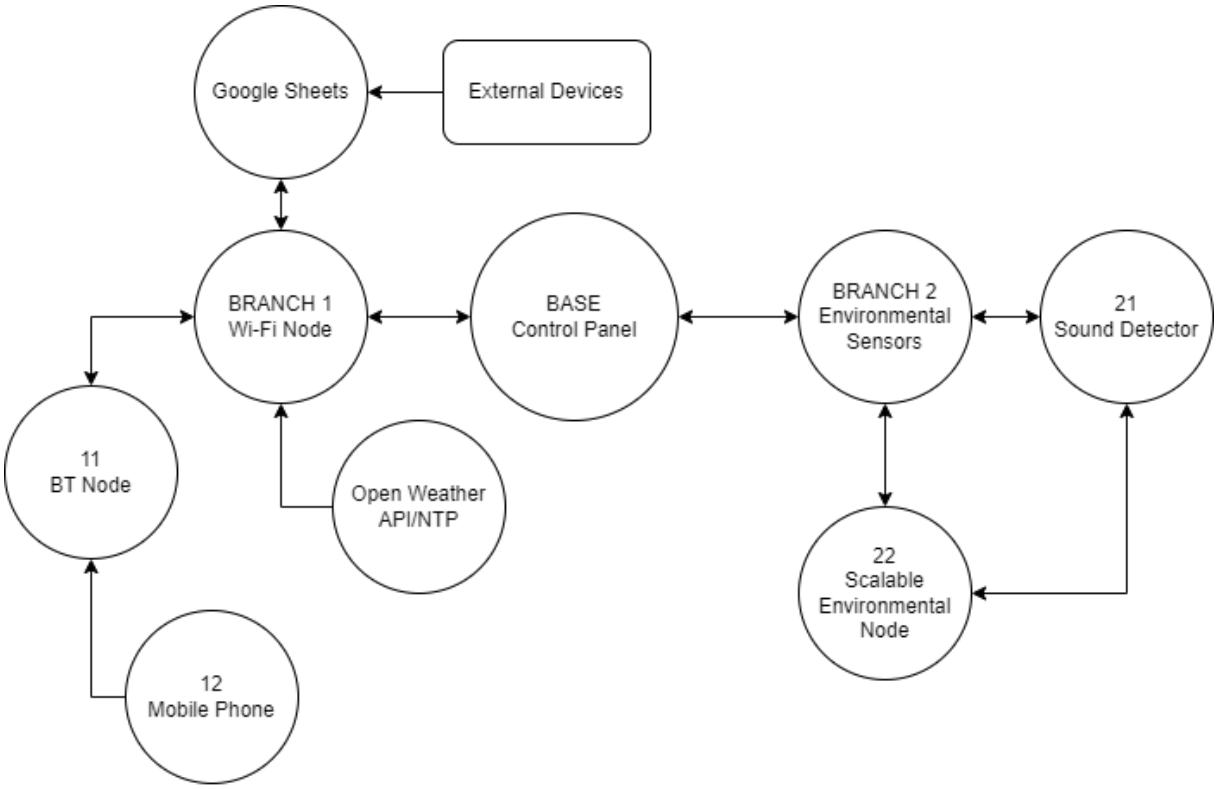



Figure 3: Network Topology

# C Node Summary

| NODE NAME | COMPONENTS | COMMUNICATION METHODS |
|---|---|---|
| Node 00: Control Panel | I. Arduino Uno and NRF<br>II. IMU<br>III. LCD<br>IV. IR remote | Radio<br>I2C<br>SPI |
| Branch 10 Leader: Wi-Fi Module | I. ESP8266 module with LCD<br>II. Motion sensor to detect motion and light up the LCD<br>III. Obtaining time, date, and weather forecast | Radio<br>Wi-Fi<br>I2C<br>SPI |
| Node 11: Bluetooth Module | I. Arduino Nano with BT module receiving from app<br>II. NRF | Radio<br>Bluetooth<br>SPI |
| Node 111: Bluetooth Phone App | I. Android Phone App | Bluetooth |
| Node 12: Google Doc | II. Highly scalable access point | Wi-Fi |
| Branch 20 Leader: Environmental | I. Arduino Nano<br>II. Temperature, humidity, and pressure sensor<br>III. NRF | Radio<br>SPI |
| Node 21: Sound Sensor | I. Raspberry Pi Pico<br>II. Buttons & LEDs<br>III. Sound detector | Parallel<br>SPI |
| Node 22: Scalable Proof of Concept | I. Arduino Nano<br>II. NRF Module | Radio<br>SPI |

Figure 4: Node Summary

# D Microcontroller Optioneering

| Specification | NANO | UNO | MEGA | ESP8266 | Pi Pico |
|---|---|---|---|---|---|
| Flash Memory | 30KB | 32KB | 256KB | 4MB | 2MB |
| RAM | 2KB | 2KB | 8KB | 64MB | 264KB |
| Digital I/O Pins | 14 | 14 | 54 | 17 | 26 |
| Analog Input Pins | 8 | 6 | 16 | 1 | 3 |
| I2C | YES | YES | YES | YES | YES |
| SPI | YES | YES | YES | YES | YES |
| Bluetooth | NO | NO | NO | YES | NO |
| Wi-Fi | NO | NO | NO | YES | NO |
| Output Voltages | 3.3/5V | 3.3/5V | 3.3/5V | 3.3V | 3.3V |
| Clock Speed | 16MHz | 16MHz | 16MHz | 80/160MHz | 133MHz |

Figure 5: Microcontroller Optioneering

# E Encryption and Decryption

```cpp
// Example XOR Key
#include <iostream>
#define xor_key 0b110011

int main(){
    int a = 0b101101;
    std::cout << "XOR Key : " << xor_key << std::endl;
    std::cout << "a : " << a << std::endl;
    int c = a ^ xor_key;
    std::cout << "c (a XOR xor_key): "<< c << std::endl;
    int b = c ^ xor_key;
    std::cout << "b (c XOR xor_key): "<< b << std::endl;
}
```

Figure 6: XOR Scramble - Transmission Encryption Code

```
XOR Key : 51
a : 45
c (a XOR xor_key): 30
b (c XOR xor_key): 45
```

Figure 7: XOR Scramble - Transmission Encryption Output

# F Gantt Chart

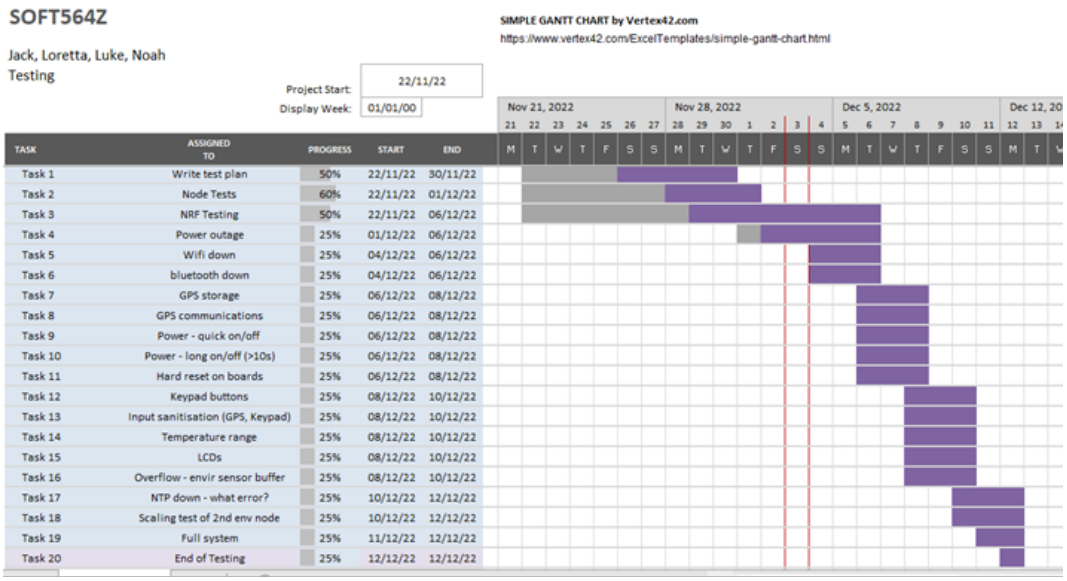



Figure 8: Testing Gantt Chart

# G Google Sheet

| Time | Weather | Temprature | Feels Like | Pressure | Humidity | Windspeed | Wind Direction | Location |
|---|---|---|---|---|---|---|---|---|
| 16:03:38 | "Clouds" | 275.72K | 273.94K | 1009mBar | 89% | 1.79m/s | 87degrees | "Plymouth" |
| 16:04:35 | "Clouds" | 275.72K | 273.94K | 1009mBar | 89% | 1.79m/s | 87degrees | "Plymouth" |
| 16:05:33 | "Clouds" | 275.72K | 273.94K | 1009mBar | 89% | 1.79m/s | 87degrees | "Plymouth" |
| 16:06:30 | "Clouds" | 275.72K | 273.94K | 1009mBar | 89% | 1.79m/s | 87degrees | "Plymouth" |
| 16:07:27 | "Clouds" | 275.72K | 273.94K | 1009mBar | 89% | 1.79m/s | 87degrees | "Plymouth" |
| 16:08:26 | "Clouds" | 275.72K | 273.94K | 1009mBar | 89% | 1.79m/s | 87degrees | "Plymouth" |
| 16:09:23 | "Clouds" | 275.72K | 273.94K | 1009mBar | 89% | 1.79m/s | 87degrees | "Plymouth" |
| 16:10:21 | "Clouds" | 275.72K | 273.94K | 1009mBar | 89% | 1.79m/s | 87degrees | "Plymouth" |
| 16:11:19 | "Clouds" | 275.72K | 273.94K | 1009mBar | 89% | 1.79m/s | 87degrees | "Plymouth" |
| 16:12:16 | "Clouds" | 275.72K | 273.94K | 1009mBar | 89% | 1.79m/s | 87degrees | "Plymouth" |
| 16:13:14 | "Clouds" | 275.72K | 273.94K | 1009mBar | 89% | 1.79m/s | 87degrees | "Plymouth" |
| 16:14:11 | "Clouds" | 275.72K | 273.94K | 1009mBar | 89% | 1.79m/s | 87degrees | "Plymouth" |
| 16:15:09 | "Clouds" | 275.72K | 273.94K | 1009mBar | 89% | 1.79m/s | 87degrees | "Plymouth" |
| 16:16:07 | "Clouds" | 275.72K | 273.94K | 1009mBar | 89% | 1.79m/s | 87degrees | "Plymouth" |
| 16:17:05 | "Clouds" | 275.72K | 273.94K | 1009mBar | 89% | 1.79m/s | 87degrees | "Plymouth" |
| 16:18:01 | "Clouds" | 275.72K | 273.94K | 1009mBar | 89% | 1.79m/s | 87degrees | "Plymouth" |
| 16:19:00 | "Clouds" | 275.72K | 273.94K | 1009mBar | 89% | 1.79m/s | 87degrees | "Plymouth" |
| 16:19:56 | "Clouds" | 275.72K | 273.94K | 1009mBar | 89% | 1.79m/s | 87degrees | "Plymouth" |

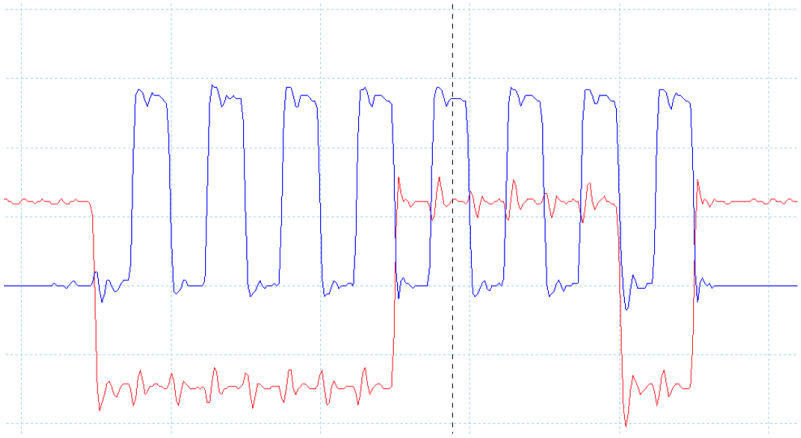
Figure 9: Google Sheet

# H Switch Case

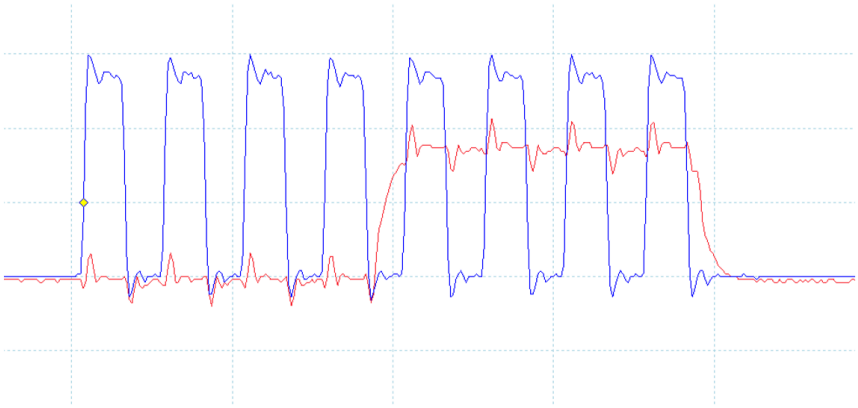
```
#define nodeNum 0

while (network.available()) {
  *incomingValue = rx();

  switch(incomingValue[0]){
    case 'nodeNum':{ //if value is for your node
      switch(incomingValue[1]){
        case '0':{ // transmission for main node controller
          break;
        }
        case '1':{ // transmission for slave device 1
          break;
        }
      }
      break;
    }
    default:{ // if value is not equal to your node value,
              //send on to base node
      tx(incomingValue, base_node);
      break;
    }
  }
}

```

Figure 10: Skeleton Code for RX Switch Case

# I Data Transmission



Data Transmission: 0000 1110 Shift Out Command on the MOSI line.



Data Receiving: 0000 1111 Shift In Command on the MISO line.

Figure 11: Data Transmission and Receive

# J Flowcharts

## J.1 System Diagram

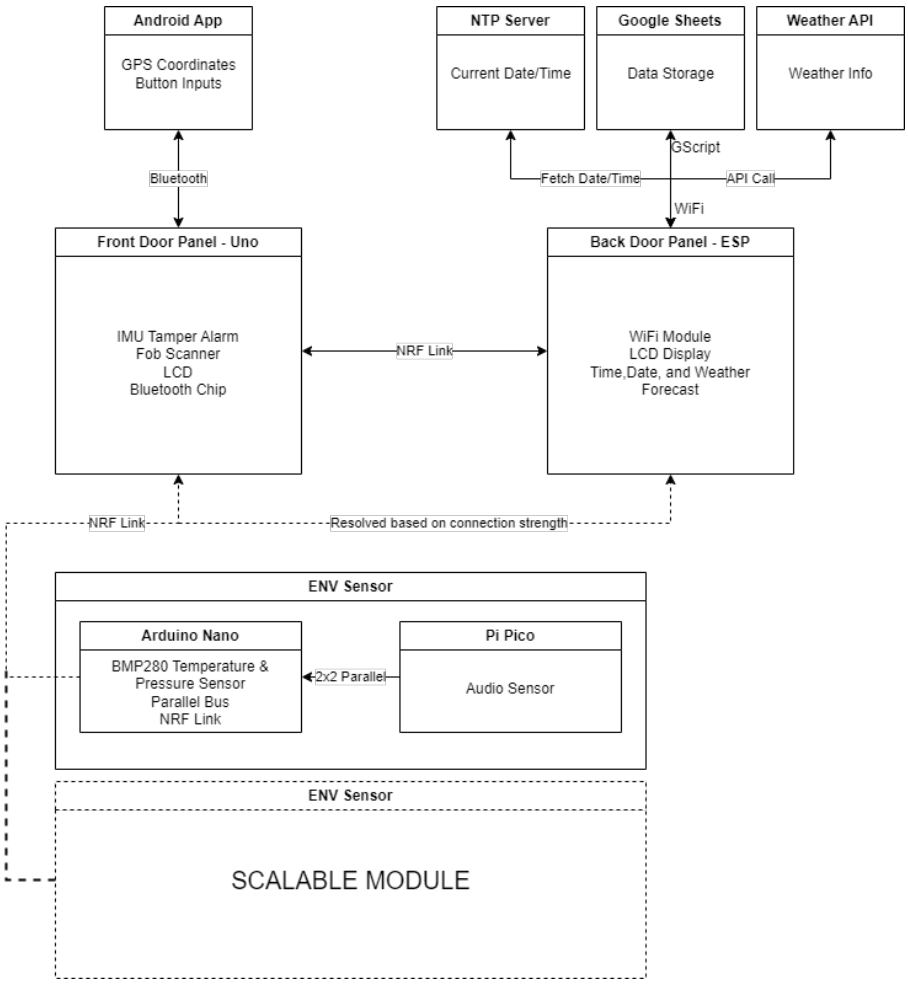



Figure 12: System Diagram

## J.2 Failiure Diagram

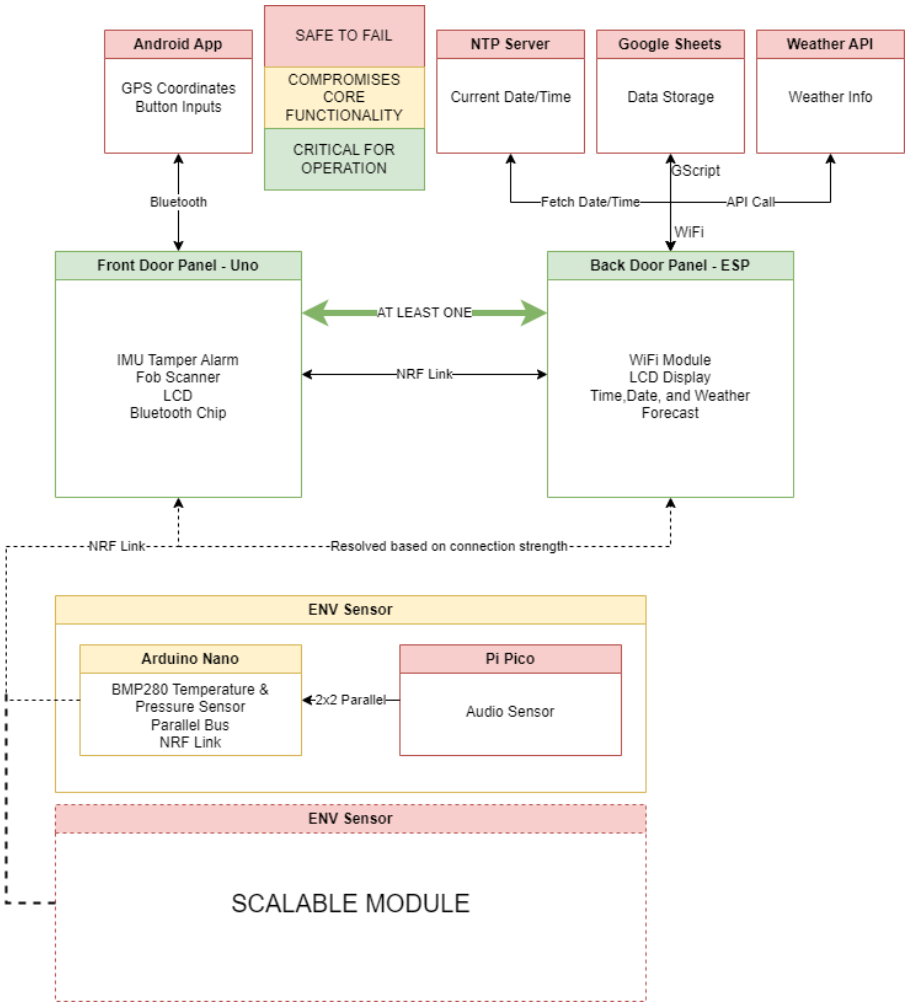



Figure 13: System Diagram Highlighting Critical Components

# K Elements of a Distributed System

| Aspect | Description |
|---|---|
| Data sharing | Users can be added to the system to allow access to control, alarm, or smart systems as required |
| Resource Sharing | Nodes will share data acquired from the various sensors throughout the system |
| Flexibility | Nodes can perform a multitude of roles throughout the system |
| Openness | How will additional nodes be added to the system. Must seem to the user that this is one singular system. Adding additional nodes should not affect the speed and reliability of the other nodes already in the system. |
| Cost | To remain commercially viable, the system costs will need to be minimised. This will be done by splitting up computationally expensive tasks and split them between multiple nodes. The balance between having multiple nodes to split up computational workload and the added cost of including more nodes will need to be balanced carefully. |
| Scalability | How are more nodes added to the system? How can the system be scaled to include increasingly larger environments. |
| Communications | Nodes within the system will work independently but transfer data when required. Additional interfaces and nodes can be added after setup and will work concurrently. The speed of the system will be determined largely by the speed of the communications between the nodes. |
| Redundancy | The system will remain viable and able to function effectively in the result of a node failure. The system can remain functional until a time when the node comes back online. Once this has occurred, the node will simultaneously re-join the system. If the controlled goes down, it may be possible to mirror the functionality on a secondary node to allow the system to continue to function. |
| Security | For simple data security we decided on XOR encryption. XOR encryption uses a predefined private key to scramble the data being sent. For node hardware security, the team decided on designing 3D printed tamperproof boxes. A motion sensor would be added to the control box to alarm if someone tries to remove it from the wall or forcibly open it. Concealing such features, although an IMU may not be recognisable, is essential to the systems integrity. The control panel would have batteries for back up power, so that the system is able to send alert messages across the platform. |

Table 1: Elements of a Distributed System(1)

| | |
|---|---|
| Concurrency | Communication between nodes is complex and the team decided on a time delay system where the main node sent out a signal and waited for a response. A maximum response time was set, in order to prevent the system from slowing down. The main node would move on if it did not receive a timely response, counting the number of times a node failed to respond. After three failures, the node was announced inactive and a fault reported to the user on the LCD. |
| Transparency | The team discussed how failure of a node would be discovered by the system but hidden from the user. This led to considerations of what might happen if the main node failed, and an adjustment of the topology to branches as oppose to a centralised system. The team also talked about data storage/website usage and how this was hidden from the users. Internode communication is also hidden from the user. |
| Heterogeneity | The Smart Home System contains an UNO, multiple NANOs and an ESP board programmed in C. There is an additional Pi PICO that was programmed in micro-python. The phone app, being on a new operating system, was written in Java. The base, leaders and attached nodes have different versions of communication protocol uploaded due to their roles within the system. |

Table 2: Elements of a Distributed System(2)

# L OSI Protocol Stack

| Layer | Protocols/Description | Utilisation |
|---|---|---|
| Application Layer | High-level protocols such as for resource sharing, RESTful [6] | Used resource sharing via Google Sheets to allow for remote access to stored data. |
| Session Layer | Translation of data between a networking service and an application | Used to encode and decode data sent for the API calls to the system |
| Transport Layer | Managing Communication sessions | Allowing for continuous data transfer back and forth between at least two nodes. Used with the NRFs to allow interconnected communication between all the leader nodes and their slave modules. |
| Network Layer | Structuring and managing a multi-node network (Wi-Fi, 2.5GHz Radio) | Connecting the ESP board to Wi-Fi for the various API calls used. Setting up a suitable mesh network utilising the NRF24L01 boards along with traffic control and separate addressing. |
| Data Link Layer | Transmission of data between two nodes physically connected (parallel interface) | Utilised a parallel interface to connect the Pi-Pico to an Arduino Nano controller board. |
| Physical Layer | Transmission or raw bit streams over a physical medium (Wires) | Connection of the modules to the main microcontroller boards. |

Table 3: OSI Protocol Stack