Process of developing a mesh network with Raspberry Pi in wooded areas

A Final year project Submitted Towards Consideration
for a Bachelor of Engineering

Author
Mistaherd

Supervisor
Philip Creevy

South East Technological University
Department Of Engineering Technology
School of Engineering
Ireland.
August 9, 2025

Contents

List of Figures

List of Tables

Listings

Glossary

APD Avalanche PhotoDiode
API Application Programming Interface
ASK Amplitude Shift Keying
AWG Agile Waveform Generator

B2B Back-2-Back
BBP Baseband Processor
BER Bit Error Ratio
BL Bandwidth-Length
BLAST Bell Labs LAyered Space Time
BT Time Bandwidth Product

CD Chromatic Dispersion
CDMA Code Division Multiple Access
CPM Continuous Phase Modulation
CSI Channel State Information

D Dispersion Coefficient
DD Direct Detection
DECT Digital Enhanced Cordless Telecommunications
DPO Digital Phosphorous Oscilloscope
DPM Digital Phase Modulation
DSP Digital Signal Processing

EDFA Eridium Doped Fiber Amplifier

FBMC Filter Bank Multi-Carrier
FDM Frequency Division Multiplex
FDMA Frequency Division Multiple Access
FEA Finite Element Analysis
FEC Forward Error Correction
FFT Fast Fourier Transform
FIR Finite Impulse Response
FRS Full Response Signalling
FTTx Fiber To The x

GASK Gaussian Amplitude Shift Keying
GFDM Generalised Frequency Division Multiplexing
GIPO General Purpose Input/Output
GLPF Gaussian Low-Pass Filter
GMSK Gaussian Minimum Shift Keying
GSM Global System for Mobile Communications
GVD Group Velocity Dispersion

IFFT Inverse Fast Fourier Transform
IIR Infinite Impulse Response
IMDD Intensity Modulation Direct Detection
ISI InterSymbol Interference
IVI Interchangeable Virtual Intruments

LAN Local Area Network
LD Dispersion Length
LD Laser Diode
LUT Look-Up Table

MC Multiple-Carrier
MIMO Multiple Input Multiple Output
MLSE Maximum Likelihood Sequence Estimation
MMF Multi Mode Fiber
MSK Minimum Shift Keying
MSO Mixed Signal Oscilloscope
MZI Mach-Zehnder Interferometer
MZM Mach-Zehnder Modulator
NGPON Next Generation Passive Optical Network
NLSE Non-Linear Schrödinger Equation
NRZ Non-Return to Zero

ODN Optical Distribution Network
OS operating system (OS)
OFDM Orthogonal Frequency Division Multiplexing
OOK On Off Keying
OSA Optical Spectrum Analyzer
OSNR Optical Signal to Noise Ratio

PAPR Peak to Average Power Ratio
PD Photo Diode
P-i-N P-doped Intrinsic N-doped Photodiode
PON Passive Optical Network
PRS Partial Response Signalling

QMDD Quadrature Modulation Direct Dectection

RF Radio Frequency
RIN Relative Intensity Noise

SCPI Standard Commands for Programmable Instruments
SISO Single Input Single Output
SMF Single Mode Fiber
SNR Signal to Noise Ratio
SOA Semiconductor Optical Amplifier
SPM Self Phase Modulation
SS Spread Spectrum
SSFM Split-Step Fourier Method
SSSFM Symmetricised Split Step Fourier Method

TCM Trellis Coded Modulation
TDM Time Division Multiplex
TDMA Time Division Multiple Access
TFM Tamed Frequency Modulation
TIA TransImpedance Amplifier
TDD Test Driven Develpoment
UFMC Universal Filtered Multiple Carrier
USB Universal Serial Bus

VISA Virtual Instrument Software Architecture

WDM Wave Division Multiplex

Abstract

In this project we aim to transmit data in a forest across a wireless channel, we will look at references to learn about the technology and why it is used, We are considering using the Pi to transmit sensor data by using a serial hat, this report focusing on the process of development in aiming to achieve the nature of a mesh network

Chapter 1

Introduction

1.1 Introduction

This project discusses the process of developing a mesh network from the sensor to serial level Communication, Communication is a key infrastructure in electronics and technology, as this can depict data between different devices, The nature of this is not a simple one due to several factors such as when an object obstructs the path of the signal, other signals interfering with the information that is being sent across the communication channel, There are multiple ways of avoiding this interaction one way on a high level is to use a mesh network like Lora, a mesh network is a type of network where no node in the network acts as a master meaning no device controls the each other. Imagine a group of friends playing a game of telephone. In a regular game, the message goes from one person to the next in a line. But what if, instead of a line, the friends formed a circle, and everyone could whisper the message to the person next to them? That's kind of how a mesh network works. this is possible thanks to the routing of each device. these will select what device talks to each other this project will look at the nature of this interaction

1.2 Motivation

The motives for looking at this topic are the following:

1. To familiarize with Linux os environment
2. To familiarize with bash scripting
3. To Showcase knowledge in the programming language python
4. To Showcase knowledge of embedded systems like the Raspberry Pi and Arduino Uno
5. To Showcase the process of selecting a sensor in the purposed area
6. To Familiarize with communication standards in the purposed area

Chapter 2

Literature Review

2.1 Introduction

The following literature review explores mesh networks in a wooded area, When communicating from two devices across a network there are many issues associated with this communication such as signal loss due to:

- Environmental conditions such as rain .lighting etc
- Whether the device's antenna is in line of sight with each other
- If the devices are in the line of sight with each other. We can still reflect from a multi-path environment
- Possibility of falling trees obstructing the path of the signal causing more attenuation in the signal strength

This project aims to explore mesh networks and transmit data across them, a mesh network is a type of network where no node in the network acts as a master. A node is a device that has a transceiver. As we look at the environment in which this project will be carried out, we can expect different phenomena to occur such as Attenuation According to ITU (1) "Attenuation due to vegetation varies widely due to the irregular Nature of the medium and the wide range of species, densities and water content obtained in practice" Transmitting any radio wave takes energy, Another factor to consider is whether wind will cause a delay in the signal. This report aims to show my findings and try to account for environmental conditions

2.1.1 Overview

The following section provides a brief overview of this project on mesh networks in a forest the following question is:

1. What frequencies can transmit in a forest
   - What are the disadvantages of transmitting at this range
   - What are the effects of the multi-path environment when there is a line of sight
   - What happens to bon-line of sight
2. What sensors /senor modules should be used
   - What sensors will give a good range in an Irish forest
   - What are the limitations on the board used
   - Is there any need for any additional hardware to accommodate a specific board

3. What microprocessor/hardware should be used?
    - The advantages/disadvantages of Arduino vs Raspberry Pi
    - What is the major factor in the choice
    - How are the sensors wired to the processor
    - How to read the data
    - What is the effective resolution needed for each application

2.1.2 Mesh network

A mesh network is a type of network that uses multiple devices to relay data between each other, making a decentralized network. The mesh to be used is a wireless mesh network which is created through the connection of wireless access point(WAP) nodes. Wireless mesh networks work through mesh nodes, mesh clients and gateways:

1. Mesh node

   nodes act as mesh routers and endpoints

2. Mesh clients

   these are end devices

3. Gateways

   Data passes through the gateway as it enters or exits a network

The following is a block diagram of a mesh network:

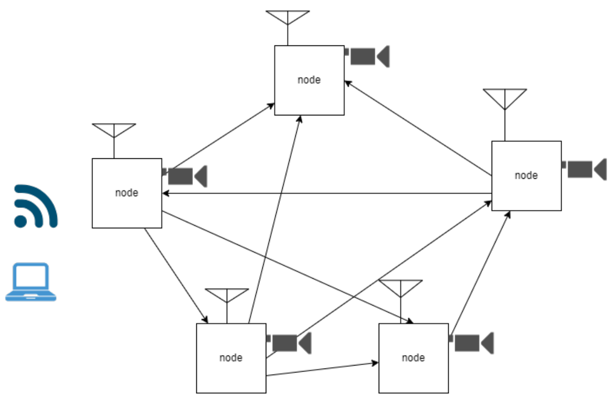


Figure 2.1: Basic block diagram of a mesh network

Each node will be attached to a tree, each having a transceiver

2.2 Hardware Consideration

In this project there needs to be data to transmit, The network needs to be able to:

1. Transmit data for example temperature, humidity, light and camera images.
2. Read data every hour and store it as a CSV file, The image file will depend on the module chosen
3. Detect any animal that passes the node with a motion sensor.

The following is a rough circuit diagram for the project:

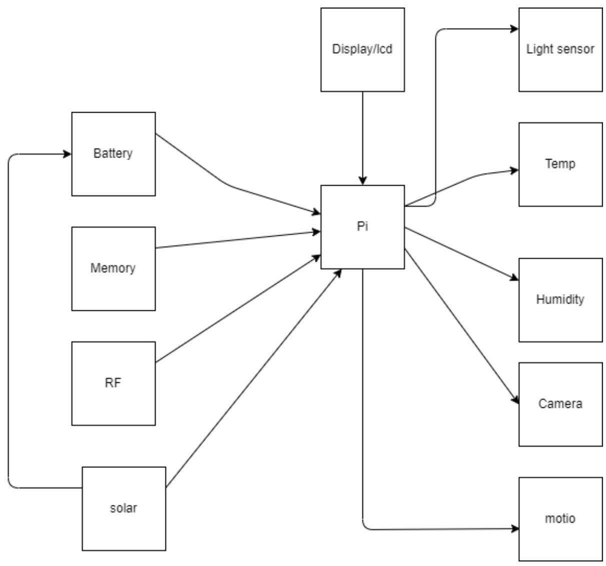


Figure 2.2: Rough circuit diagram for project

1. A PCB cannot be used due to the ordering process taking too long for the time frame of this project
2. Any type of board like wire wrap would take too long and would be outside of the goals of this project
3. A choice of either the Arduino or Raspberry Pi is left

This section discusses the following:

1. The sensors to be used in the project
2. The ADC we will have to will have to consider
3. The camera chosen will have to be considered for this project
4. The memory module considerations
5. The battery chosen
6. A choice made between Arduino or Raspberry PI

2.2.1 Sensor considerations

In this section, the process of considering each component of the sensors will be discussed. These components are:

1. Temperature
2. Humidity
3. Light
4. Motion

Temperature & Humidity sensor

The sensor needs to be able to work in the following conditions:

1. The mesh node will be outside
2. The device is in Ireland
3. The device is in a forest

Taking these requirements into consideration, the temperature in Ireland was researched.

The following table was obtained from Met Eireann (2). which shows the highest air temperature in a shaded area

Highest Shaded Air (˚C)	Station	Date
18.5˚C	Dublin (Glasnevin)	10th 1998
18.1˚C	Dublin (Phoenix Park)	23rd 1891
23.6˚C	Dublin (Trinity College)	28th 1965
25.8˚C	Donegal (Glenties)	26th 1984
28.4˚C	Kerry (Ardfert Liscahane)	31st 1997
33.3˚C	Kilkenny (Kilkenny Castle)	26th 1887
33.0˚C	Dublin (Phoenix Park)	18th 2022
31.7˚C	Carlow (Oak Park)	12th 2022
29.1˚C	Kildare (Clongowes Wood College)	1st 1906
25.2˚C	Kildare (Clongowes Wood College)	3rd 1908
20.1˚C	Kerry (Dooks)	1st 2015
18.1˚C	Dublin (Peamount)	2nd 1948

Table 2.1: Highest shader air Met Eireann(13^{th} June 2023)

According to the table, the highest temperature is 33.3. The other extreme of the Lowest temperature was then considered:

Lowest Shaded Air (°C)	Station	Date
-19.1°C	Sligo (Markree)	16th 1881
-17.8°C	Longford (Mostrim)	7th 1895
-17.2°C	Sligo (Markree)	3rd 1947
-7.7°C	Sligo (Markree)	15th 1892
-5.6°C	Donegal (Glenties)	4th 1979
-3.3°C	Offaly (Clonsast)	1st 1962
-0.3°C	Longford (Mostrim)	8th 1889
-2.7°C	Wicklow (Rathdrum)	30th 1964
-3.5°C	Offaly (Clonsast)	8th 1972
-8.3°C	Sligo (Markree)	31st 1926
-11.5°C	Wexford (Clonroche)	29th 2010
-17.5°C	Mayo (Straide)	25th 2010

Table 2.2: Lowest shader air Met Eireann(13^{th} June 2023)

According to the table above the lowest temp is -19.1 In consideration of where the project the condition is a range of -19.1°C to 33.3°C.

Humidity was also researched, Humidity refers to the amount of water vapor in the air. The following table was obtained from Met Eireann (3):

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Year
Mean at 0900UTC	87.0	86.4	84.0	79.5	76.9	76.7	78.5	81.0	83.4	85.5	88.5	88.0	83.0
Mean at 1500UTC	80.6	75.7	71.0	68.3	68.0	68.3	69.0	69.3	71.5	75.1	80.3	83.1	73.3

Table 2.3: Realtive Humidity(%) according to met eirrean

The ranges are from 68.3% to 88 % Taking these considerations into account. Here are the different components:

Components	Voltage Range	temperature range	Accuracy	Analogue /Digtial outputs	Current in	additional information
DHT22	3-6 volts	-40 to 80^o C	+/-0.5ºC	Digital	1.5mA	sample period 2 seconds
LM35D2	4 -30 Volts	-55 to 150	+/-0.5ºC (at 25ºC)	Analogue	10mA	None
TMP36	2.7 to 5.5 Volts	-40 to 125	+/-1ºC (at 25ºC)	Analogue	250 µA	NONE
LM75	3.0 to 5.5V	-55 to 125ºC	+/-2.0ºC (at -55 to 125ºC range))	Analogue	100 µA	NONE
DHT111	3-5.5V	0-50 °C	±2°C	Digital	1mA	sample period 1 second

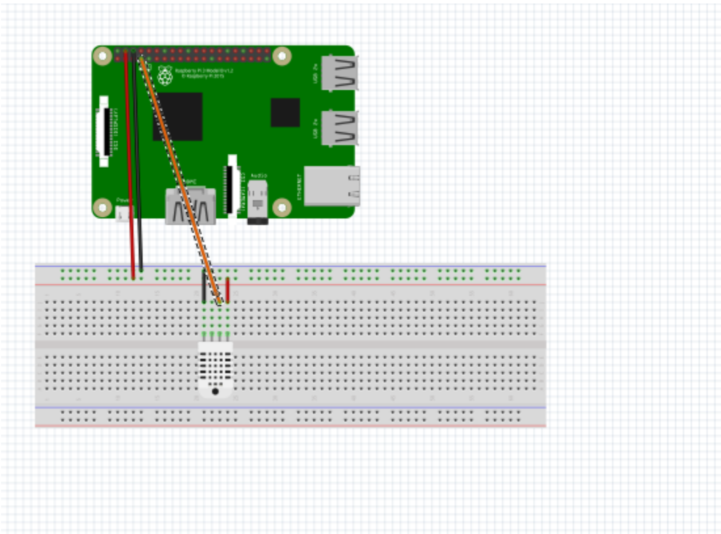
Table 2.4: Comparing of temperature sensors

After this, the choice between the two sensors was narrowed down to DHT22 and DHT11. The following are the advantages and disadvantages of the DHT22 and DHT11:

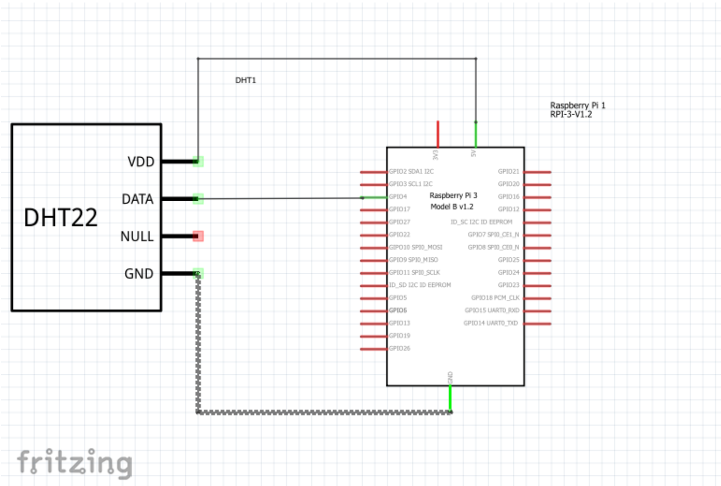
Device	Advantages	Disadvantages
DHT22	good accuracy has temp and humidity, falls in our temp range	sample period 2 seconds
DHT11	OK voltage,better sample period	draws a lot of current , and our of range

Table 2.5: Comparing DHT22 and DHT11

DHT22 was chosen. Which has a Digital output. See a wiring diagram next page: This will have an interface of the following:



(a) Interface for DHT22



(b) Schematic for DHT22

From above the schematic is seen. DHT22 connections are the following:

- VDD is connected to 5v of the Pi
- the Data pin is connected to GPIO 3
- Gnd pin of the Pi is connected to the ground of DHT22

(4) The following is the link to the datasheet of this module when reading from this component. There is a delay of 2 seconds due to the sampling period.

Light sensor

In this section, the following will be considered:

1. What region the project is in
2. What light levels are expected in this country
3. What sensor will accommodate this range

For this sensor, the outside aspect of the project must also be considered. The following table was found on (5):

Imminence	Example
0.002 lux	Moonless clear night sky
0.2 lux	Design minimum for emergency lighting (AS2293).
0.27 & 1 lux	Full moon on a clear night
3.4 lux	Dark limit of civil twilight under a clear sky
50 lux	Family living room
80 lux	Hallway/toilet
100 lux	Very dark overcast day
300 to 500 lux	Sunrise or Sunset on a clear day. Well-lit office area.
1,000 lux	Overcast day; typical TV studio lighting
10,000 to 25,000 lux	Full daylight (not direct sun)
32,000 to 130,000 lux	Direct sunlight

Table 2.6: Illuminates values

This table is the associated lux level indicating when the values are. From above the sensor should ideally be 0.002 to 25000 lux, Bearing this in mind these components were researched:

Modules	Voltage Range	Analogue /Digital Outputs	illumination range	Current rating
LM393 with GL5528	3.3v to 5v	Analogue	0 lux to 100lux	250nA
DFR0026	3.3v to 5v	Analogue	1 Lux to 6000 Lux	120uA
LM393 with n5ac501085	max 150V	Analogue	10 lux to 100lux	1mW
LM393 with NSL-06S53	max 100v	analogue	1 to 100	50mw

Table 2.7: table of light sensors

After research DFR0026 (6) is the option proposed for use as it is the best for this application, which will have an analog output.

To see the interface (see below):

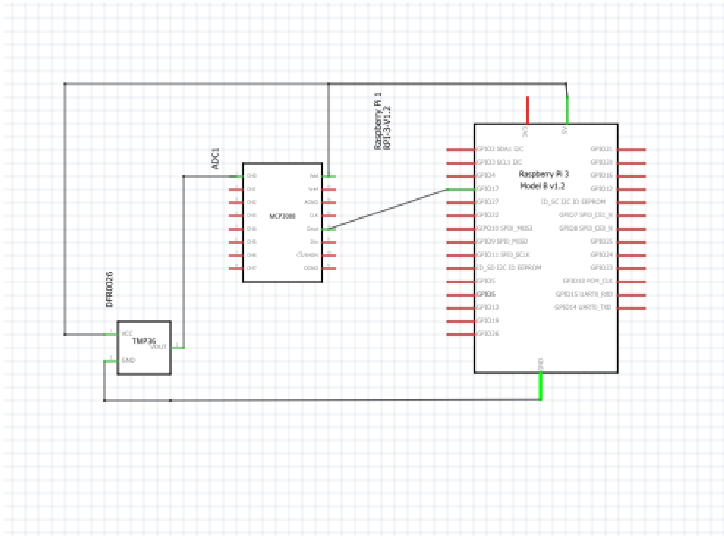


Figure 2.4: Interface for DFR0026

The following are the connections:

1. VCC pin is connected to 5v
2. Gnd of the sensor is connected to GND of the Pi
3. The output is connected to ch 0
4. the output ranges from 0 to 5 v

The component relies on the ADC which is on page11

Motion sensor

For this section, the following must be considered:

1. The range of the sensor
2. The degree of the sensor
3. How long of a delay is the sensor

The following are the components considered:

Modules	Voltage Range	Distance	Max angle	Analogue /Digital Outputs	Power
HC-SR501	5-20V	3 to 7m	110	Digital	50uA
AM312	4.5-20v	3m	130	Digital	60uA
AS312	-0.3 - 3.6V	12m	130	Digital	100mA

Table 2.8: Motion sensor components

The sensor chosen is AS312(7)(which has a delay time of 2 seconds) which is a digital interface to see the wiring, See below:

The following is the interface for the device:

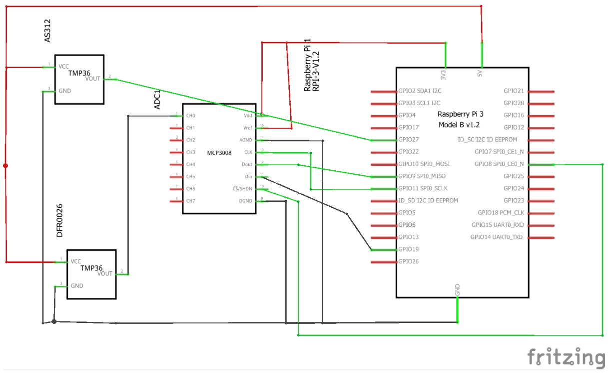
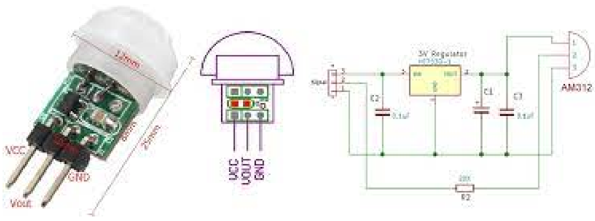


Figure 2.5: Interface for AS312

The connections are the following:



Pinout of AS312

1. VCC is connected to 5v pin of the Pi
2. GND is connected to the GND of the Pi
3. Vout is connected to GPIO 17

This component has the following:

1. Range of 12 meters
2. An angle of 65^o degree
3. A Delay of 15 μ Seconds

2.2.2 Radio Module

For this section, there are the following considerations:

- The devices are in a forest
- Meaning Gigahertz isn't a desirable frequency
- A module that has low-power
- A model that will have a high throughput

Through research, I found the following table:

Modules	Tx/RX Voltage	Frequency	Range	TX/RX power	Through put	Error detection	Rx sensitivity	Hopping channel
Gravity: 315MHZ RF Receiver Module	3.3v/5v	315Mhz	50 m	-10dbm -95dm	9.6kbps (max)	none	-108 dbi	no
MM2 Series 900 MHz OEM Radio	3.5 5.0	902 928 Mhz	32km	1175 ma tx rx 125ma	80 - 115.2 kbps	32-bit CRC	-108dbm @ 115.2kbps for BER 10^{-4} -103 dm @153.6 for BER 10^{-4}	50 to 112, user-selectable
RF 433MHz Transmitter/Receiver Module	5V 3-12v	433.92 MHz	20-200 meters	10 mW	2kbps	ASK modulation no error check	-105 db	no
Digi XBee-PRO 900HP RF Module	2.1 to 3.6v dc	902 to 928 Mhz	14km - 6.5km	24dm	10kps -200kps	NONE	-101dm	FHSS (Software Selectable Channels)

Table 2.9: Radio modules found in research

Out of these, the MM2 Series 900 MHz(8) was chosen. Note that the seller of this radio module has limited the documentation of this module. This makes it hard to draw an interface for this module.

2.2.3 ADC Considerations

For the ADC there are the following considerations:

1. Low power
2. High-bit resolution
3. Low number of channels
4. High sample rate

The two things needed for this is a high bit Resolution and a high sample rate

Device	Resolution	Sample rate	Input range	Power consumption
ADC pi Zero	17 bits	100KHz	0-5.06v	10mA
MCP3008	10 bits	200 ksps	2.7v- 5.5v	500uA
DFR0553	16 bits	1.7Mhz	0 5.0V	10mA

Above are the components to choose from for this project, MCP3008 was chosen due to its resolution and sample rate

The following is the schematic for the MCP3008(9)

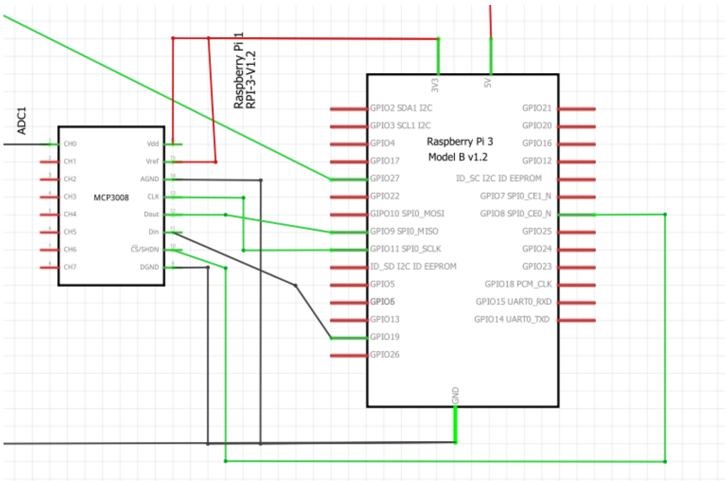


Figure 2.6: Schematic for MCP3008

The following are connections:

1. VDD is connected to 3v3 pin of the Pi
2. VRef is also connected to the 3v3 pin of the Pi
3. AGND is connected to the GND pin
4. CLK pin is connected to GPIO port 11
5. Dout pin is connected to GPIO port 9
6. Din pin is connected to GPIO port 19
7. CS pin is connected to GPIO port 8
8. DGND ping is connected to the GND pin

This component has the following:

1. A 10-bit resolution
2. Seen as the reference will be 3.3v
3. 200ksps, meaning the delay to read is 5μ seconds

2.2.4 Camera

For the camera, the following has to be considered:

1. Focal length
2. Resolution
3. Power
4. The lux values it operates at

Modules	Voltage range	lens size	Image Resolution	Video Resolution	Frame Rate	Type of Output	Preferred condition	Power
Raspberry Pi VR 220 Camera	3.3V ac	can change with lens	3280 X 2462	1920 x 1080	30 FPS	Need to research	Daytime	38mA
DIGILENT 410-358	3.6v	optical size 1/4 inches	2592 x 1944	?	?	Digital	?	200mA
The Raspberry Pi NoIR	3.3v	1/4 inches	3280 x 2464	1080 or 720	30 60 fps	need to research	house	38mA
OV7670 VGA	2.45 to 3.0v ac	1/6 inches	2.36mm x 3.6um	?	30 fps	analogue	need to research	60mW

Table 2.10: Camera module

The camera chosen is a Raspberry Pi VR 220(10) Camera To see how to connect look at the following link.

2.2.5 Memory module

For this section, we consider the following:

1. The file formatting of the sensor data
2. The file formatting of the camera data
3. What are the possible sizes of data
4. What is the memory size of the Raspberry Pi OS

In my project, the following is used:

1. The sensor data is stored in a CSV file with the following heading: (timestamp, heat, humidity, light level, anything detected), Which can be around 25KB
2. For the camera using 10 MB is the largest file size
3. For the Raspberry Pi download the Raspberry Imager, This has lots of options such as the following on page 17

After has been we must consider a mircoSD, The following are considerations:

Product Name	Capacity	Speed class	Read speed	W
SAMSUNG EVO Plus Class 10 microSDXC	256 GB	U3	up to 130MB/s	up
SANDISK Ultra Performance Class 10 microSDXC	128 GB	class 10 u1	up to 80 MB/s	up
Silicon Power 32GB 3D	32GB	class 10	Up to 100MB/s	Up
SanDisk 64GB Extreme PRO	64GB	UHS Speed Class 3	Up to 100MB/s	up

Table 2.11: Mirco SDs in consideration

The best here is the silicon power 32GB due to its temperature range and read and write times. we can consider extra memory:

1. DHTT22 has a sample period of 2 seconds
2. AS312 which has a delay time of 15 μ seconds
3. MCP3008 5 μ seconds

Through research, the following was discovered :

Brand	Product Name	Storage Capacity	USB Version	Data Transfer Sp	Read/Write Spee	Durability	Encryption	Form Factor	Compatibility	Price
SanDisk	Cruzer Glide 1GB USB 3.0 Flash Drive	1 GB	USB 3.0	100 MB/s	80 MB/s	Water and shock resistant	No	Standard	Windows, Mac, Linux	$5.99
PNY	Turbo 1GB USB 2.0 Flash Drive	1 GB	USB 2.0	480 Mbps	300 Mbps	Water and shock resistant	No	Standard	Windows, Mac, Linux	$3.99
Verbatim	Store 'n' Go 1GB USB 2.0 Flash Drive	1 GB	USB 2.0	480 Mbps	300 Mbps	Not specified	No	Standard	Windows, Mac, Linux	$2.99

Table 2.12: Memory usb to consider

The Raspberry Pi 4 supports USB 2 and USB 3. For this, the Turbo 1GB USB 2 Flash Drive will be chosen.

2.2.6 Battery

In this section, the following must be considered:

1. Enough power for all sensors and radio module
2. Storage of the battery
3. Discharge rate of the battery (how many operating hours can be got out of the battery)

Here are the following Devices I found :

Modules	Voltage	Interface	Power	Chemistry	Supply time
Li-polymer Battery HAT	5v	Micro USB	1.8A	lithium battery	5 hours

Table 2.13: battery considerations

The battery chosen is the li-polymer which has a micro USB How to charge:

- Step 1: Insert the Li-polymer battery into a 2.0mm battery socket
- Step 2: Connect the power adapter to a micro USB or Type-C interface by USB cable.

Aside: this component has the following:

1. A battery that is 3.7v $3000mAh$
2. Output voltage of 5 volts
3. an estimated Power supply time of 5 hours

2.2.7 Arduino vs PI Consideration

In this project, we will have to choose between what microprocessor we will use. There are 3 options:

1. PCB (printed circuit board) where the circuit is designed in a program like Fusion 360. The major issue is due to the current state of silicon chips which will slow down the progress of the implementation stage
2. Arduino
3. Raspberry Pi

The advantages and disadvantages of the Arduino and the Raspberry Pi are the following:

Arduino	Pi
Advantages	Advantages
1. Arduino has a 10-bit ADC	1. Pi can compile Python (easier to write)
Disadvantages	Disadvantages
1. Arduino has a supper set of C++ 2. Arduino only has 6 Analogue pins	1. Pi is a technically a small CPU 2. The Pi needs an ADC circuit to deal with inputs that are analog

Table 2.14: Advantages /Disadvantages of Arduino vs pi

However, the Arduino would be more efficient than the Raspberry Pi due to Raspberry Pi having an Operating System. The Pi was picked due to familiarity with Python and Linux. Linux can be used to handle the networking side of the project At the cost of some efficiency in power for an easier time writing the code for this project

Picking a Raspberry Pi

Now that a device was chosen to use the following needs to be defined:

1. The amount of GIPO PORTS we need
2. Nature of the output of the sensor
3. Speed of the clock

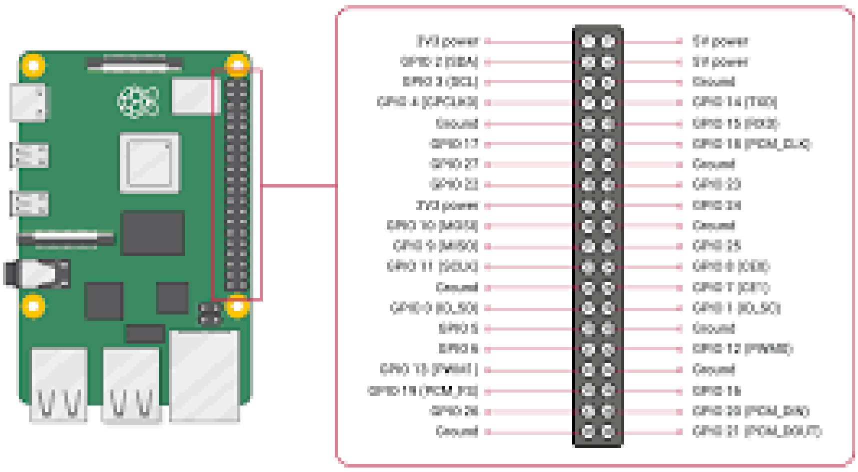
GPIO(General purpose input/output) is used to select the input/output. The Pi can take in digital signals only Seen as the components are chosen that require A GPIO port (temperature/ humidity, on page 7, Light on page 8, motion on page 9)

At least 3 GPIO ports to be available to us as the light sensor and the motion will need an adc as looking through the documentation First, let's look at the different models:

Raspberry Pi Model	Internal Clock Speed	Power (Watts)	GPIO Features	Type of Connectors	SRAM
Raspberry Pi 1 Model B+	700 MHz	5.5	26 GPIO pins	1 HDMI, 1 micro USB, 1 USB 2.0, 1 audio jack	512 MB
Raspberry Pi 2 Model B	900 MHz	7.5	40 GPIO pins	1 HDMI, 1 micro USB, 4 USB 2.0, 1 audio jack	1 GB
Raspberry Pi 3 Model B+	1.4 GHz	8	40 GPIO pins	1 HDMI, 1 micro USB, 4 USB 2.0, 1 audio jack, 1 Gigabit Ethernet, 1 PoE header	1 GB
Raspberry Pi 3 Model A+	1.4 GHz	5	26 GPIO pins	1 HDMI, 1 micro USB, 2 USB 2.0, 1 audio jack	512 MB
Raspberry Pi Zero	1 GHz	1.2	40 GPIO pins	1 mini HDMI, 1 micro USB, 1 micro-USB OTG	512 MB
Raspberry Pi Zero W	1 GHz	1.3	40 GPIO pins	1 mini HDMI, 1 micro USB, 1 micro-USB OTG, 1 Wi-Fi/Bluetooth module	512 MB
Raspberry Pi Zero 2 W	1 GHz	0.8	40 GPIO pins	1 mini HDMI, 1 micro USB, 1 micro-USB OTG, 1 Wi-Fi/Bluetooth module	512 MB
Raspberry Pi 4 Model B	1.5 GHz	7	40 GPIO pins	2 HDMI, 2 USB 3.0, 2 USB 2.0, 1 Gigabit Ethernet, 1 audio jack	1 GB, 2

Table 2.15: Table of Raspberry Pis

The above table displays the modules, As our radio module is 900Mhz we want 1.5GHZ which is the Raspberry Pi 4. This needs a USB c -charger and an HDMI mini cable. For wiring our Pi here is the GPIO pin layout:



Pinout for the pi

2.2.8 Conclusion

In this project, the hardware needed is the following components:

1. 1 x Raspberry Pi 4 Model B
2. 1 x HDMI cable
3. 1 x USBC cable
4. 1 x USB C charging head
5. 1 x DHT22
6. 1 x DFR0026
7. 1 x AS312
8. 1 x MM2 Series 900 MHz
9. 1 x MCP3008
10. 1 x Raspberry Pi VR 220 Camera
11. 1 x Li-polymer Battery HAT
12. 1 x Turbo 1GB
13. 1 x silicon power 32GB microSD

2.3 Software considerations

Having established the essential hardware needed for this project. Next, consider the following for the software of the project:

1. How to structure code
2. Linux set up of sever and nodes
3. How will data be sent
4. Will this be an OOP or functional approach?
5. How to program each device?

2.3.1 Raspberry Pi OS

In this section, it must be kept in mind that each OS is heavyweight the following needs to be considered:

1. If the SD is formatted the data on the SD is lost. Does it corrupt the card?
2. An os that is low in capacity
3. Is a desktop needed or can we use the terminal?
4. How does the OS respond to USB drives?

According to the (**?**) the imager will erase all the data while installing the os. From research, the suggestion of backing up the data is a good suggestion. for now, the recommended OS is used. and strip down as the project progresses. which will be discussed in the methodology section of this report.

2.3.2 Sensor code

In this section, the following will be discussed :

1. DHT22
2. AS312
3. MCP3008
4. DFR0026
5. Kuman for Raspberry Pi 3B+ TFT LCD Display
6. Raspberry Pi VR 220 Camera

the project code will mainly be object-oriented. so the goal is to first test it with my laptop and Create A bash file full of commands to install the libraries, making the code split up into different parts so that all that is needed is the libraries used and code that won't all have to be compiled in one file.

DHT22

In this section we have to consider the following:

1. The GPIO port as on page 7 This is connected to port 3
2. The type of output is digital so no extra hardware/code is needed

The following is a rough guide on how to read from the DHT22 from the following link. Firstly open the terminal in the Pi and type the following commands:

```
git clone https://github.com/adafruit/
    ↪ Adafruit_Python_DHT.git
cd Adafruit_Python_DHT
sudo apt-get update
sudo apt-get install build-essential python-dev
sudo python setup.py install
```

the code does the following:

1. firstly git clone will clone the repository onto to device
2. Then change directories a
3. update Linux
4. install dev kit for python
5. and install the setup

this will then lead to the following code:

Listing 2.1: Example code for DHT2

```
#Libraries
import Adafruit_DHT as DHT
from time import sleep
def setup_DHT22(Gpoiport:int):
humidity,temp=dht.read_retry(DHT.DHT22, Gpoiport)
    sleep(5)
    return humidity, temp
h,t=setup_DHT22(3)
print('Temp={0:0.1f}*C␣␣Humidity={1:0.1f}%'.format(t,h))
```

this code will do the following:

1. Import DHT from the Adafuit library
2. in the function which takes the GPIO port as an integer this will read the data on the pin and print it out

AS312

for this section i followed this link we also want to keep in mind the following:

1. This has a digital interface and is connected to GPIO 27

Here are the rough steps firstly type the following into the terminal

```
sudo apt-get install python-rpi.gpio
```

which will install a gpio python module Then type this into an IDE of your choosing

Listing 2.2: Example code for AS312

```
        import RPi.GPIO as GPIO
        import time

        pir_sensor = 27
        GPIO.setmode(GPIO.BOARD)

        GPIO.setup(pir_sensor, GPIO.IN)
        current_state = 0

        time.sleep(0.1)
        current_state = GPIO.input(pir_sensor)
        if current_state == 1:
            print("GPIO pin %s is %s" % (pir_sensor,
               current_state))
            # trigger camera
        # must look up this
        GPIO.cleanup()
```

this code does the following:

1. it will look at the pin for a pulse
2. Once it senses a pulse it will trigger the camera

DFR0026

from the last example, nothing has changed from the last component an example code for this can be found on page 19

2.3.3 MCP3008

for this section, we want to consider the following:

1. The MCP3008 data out is GIPO 9

This section follows this link firstly try the following in command in the terminal

```
sudo raspi-config nonint do_spi 0
```

Listing 2.3: ADC code

```
        from gpiozero import MCP3008
        from time import sleep
        DFR0026 = MCP3008(channel=0, device=0,port=9)

        print ('raw: {:.5f}'.format(DFR0026.value))
        sleep(0.1)
```

this code will select a channel and device, port and print the values of the ADC's

2.3.4 Raspberry Pi VR 220 Camera

to get started with this simply look at the following link here is an example of the code of this module :

Listing 2.4: example code for camera

```
from picamera import PiCamera
from time import sleep

camera = PiCamera()

camera.start_preview()
sleep(5)
camera.stop_preview()
```

this will take a photo of what is in front of the camera

2.3.5 MM2 Series 900 MHz

for this section, the seller of this module has no public documentation so it is hard to come up with an make interface for this section

2.3.6 code structure

The code structure for this will be an object-oriented program all the individual sensors and hardware for the pi will be as displayed above the code in this section will be formatted into objects for example I will have an object called proj_sensor and a method of this would be DHT22 while an attribute of this would be the sample rate the following is a rough breakdown of the structure of the code

- Sensor object
  - Temperature and humanity method
  - light method
  - Motion method which triggers the camera
  - Battery method which is a constructor method
  - Memory method which links with the radio
- radio object which reads from Memory and transmits the data

2.3.7 File structure

For the File structure, we want our sensor data to be stored every hour in a CSV file with the following column headings:

1. timestamp
2. Heat
3. Humidity
4. light level
5. motion detected (True/False)

for the writing to Date, we will use Pandas to write to the CSV file for file sorting, I will use the Python Library glob which I can use to look for files the following is an example of how to make a CSV file: firstly let's make a data frame:

Listing 2.5: sample code for turning sensor data into a data

```
        import pandas as PD
        import numpy as np
        from datetime import datetime
        cols_name=["Timestamp", "Temperature", "Hummidty", "
           Light_level", "Motion_dected"]

        #assume that being recorded now
        data=[]
        timestap=datetime.now()
        timestap=timestap.strftime("%d/%m/%Y %H:%M:%S")
        Current_state=1
        Heat=0.40
        Hummidty=1.0
        Light_level=0.23
        data=np.array([[timestap],[Heat],[Hummidty],[Light_level
           ],[Current_state]])
        data=data.T
        df= pd.DataFrame(data,columns=cols_name)
```

Next, use the.To_csv method from pandas another Libraries that could be useful is the Tkinter here is a sample of how to store where the file is gonna be:

Listing 2.6: example code for storing directory

```
        import tkinter as tk
        from Tkinter import filedialog
        import json
        import os

        root = tk.Tk()
        root.withdraw()
        selected_dir = filedialog.askdirectory()

        if not os. path.exists('selected_dir.json'):
            # Write the selected directory to a JSON file
            with open('selected_dir.json', 'w') as f:
                json.dump(selected_dir, f)
                print("Successfully saved selected directory to 
                   JSON file.")
        else:
            print("File 'selected_dir.json' already exists. Not 
               saving the directory.")

        root.quit()
```

Other useful Libraries allow you to select all .csv, png called glob for our TDD Section, we will have to use the following command:

```
# !/bin/bash

dir_name=$1

size=$(du -sh "$dir_name" | cut -f1)

echo "Directory size: $size"
```

This is a script that will look at a director this can be a home directory that will call the space 13K the ”— cut -f1” will only focus on the size string message and then print out the size. this is just a sample script

2.3.8 Test Driven development

In this project ill will be using Test Driven Development (TDD) is a software development approach where tests are written before the actual code the following are the advantages of TDD:

1. Advantages
   (a) TDD forces you to consider potential failure points and edge cases upfront, leading to earlier detection and resolution of bugs.
   (b) TDD encourages you to think about the desired behavior and interfaces of your code
   (c) TDD provides immediate feedback on whether your code works as intended,

2.4 Attenuation

Attenuation refers to a reduction in the strength of a signal. Attenuation occurs with any signal, whether digital or analogue. Seen the aim of making a network the first step is to look into what frequencies can be transmitted and received.
In the environment in which we want our project to take place, we want the following:

1. An antenna that a high so we can affect the data rate of the signal
2. A frequency range at which Attenuation is not present

Through research, I found the following plots:

1. First Plot The first plot I got for Savage e.t al pg. 7 (11)

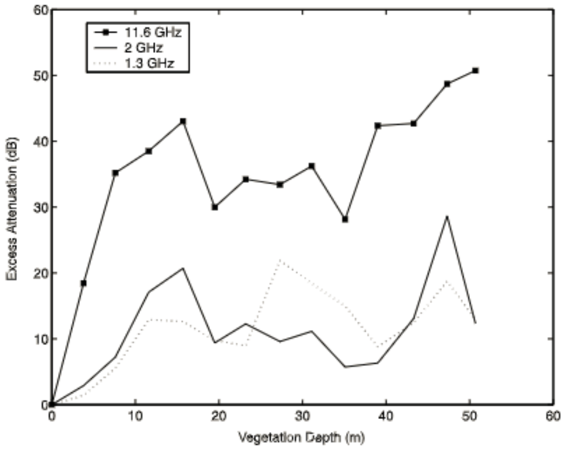


Figure 2.7: Silver Maple in-leaf excess attenuation for the line of trees geometry (receiver antenna height: 3.5 m, SAVAGE ET AL.pg.7

This graph displays as vegetation depth increases Attenuation rises. The problem with this graph is that it doesn't give an in-depth view of which attenuation occurs. This then led me to look up the International Telecommunication Union (1) recommendations for Attenuation in wooded areas

2. Second Plot V is the vertical polarization H is the horizontal polarization

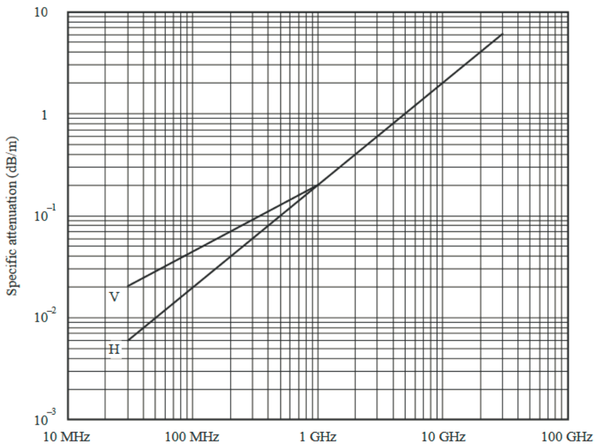


Figure 2.8: Specific attenuation due to woodland (Recommendation ITU-R P.833-7 (02/2012) Attenuation in vegetation pg.5

From this graph we can assume the following:

(a) From a frequency ≥15GHz we can assume Attenuation is more components
(b) Around the 1 GHz range we get low values of Attenuation
(c) in the MHz range we get the best response

from this, I selected the range which is $10^6 hz$

so now that we established our range let us consider what happens when it rains(12)

Frequency MHz	Attenuation dB/m
106	0.04
466	0.12
949	0.17
1852	0.3
2118	0.34

Figure 2.9: Predicted attenuation due to rain for the region, which is measured by using the ITU standards,(Source: Hindawi(2014))

Ideally, we want a low MHz but we want speed and this is dictated by what we choose let's further see how radio waves are affected by water/rain

2.4.1 Absorption of water

for this, I found this graph from Lunken Heimer (13)

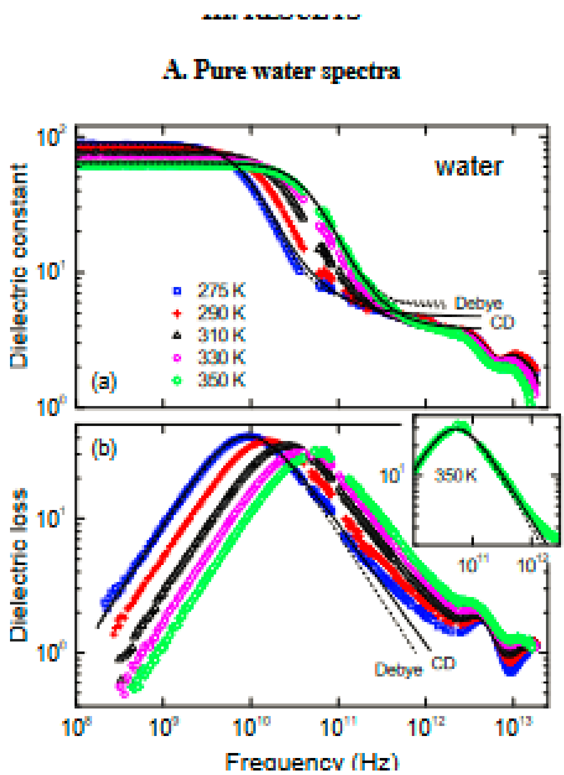


Figure 2.10: absorption of water

According to the graph, Water absorbs MHz frequencies which will affect the transmission in the transmission and in some cases, we might have to consider non-line-of-sight communication when it rains or we might also consider another node to route to receive the node.

2.5 mesh network considerations

For this section, we have to consider the following:

1. How are we setting up the network
2. What framework are we using to set this Up
3. What are the advantages/disadvantages

In my research I found two main frameworks that this project could use to achieve the mesh network these are the following:

1. LORA
2. Zigbee

According to Chen (2023)(14) "LoRa, as one of Low Power Wide Area Networks (LP-WANs) technologies, aims to enable IoT devices to perform long-range communications with lower power consumption [18]. LoRa makes use of the chirp spread spectrum (CSS) modulation to improve the transmission distance up to kilometres and also be resistant to multi-path effects."

According to Vlad(15), "ZigBee is an LP-WPAN (Low-Power-Wireless Personal Area Network) with short range and low power consumption, as mentioned before. The range for ZigBee devices is up to fifty meters and it is characterized by a low data rate, having a maximum value of 250 kbps. The protocol is suitable for sensors and IoT applications because of the low data rate and low power consumption"

the following are the differences between the two: from research, these are very similar but

LoRa		ZigBee	
Advantages	Disadvantages	Advantages	Disadvantag
Long transmission distance	Low transmission rate	Low power consumption	Low data ra
Low power consumption	Slow data transfer rate	Long range	Limited ran
Multi-channel information procession	Small payload	Scalability	Signal interfer
Strong anti-interface ability	Low bandwidth	–	High-sensitivity
High-sensitivity levels	Spectrum interference		

Table 2.16: Advantages and Disadvantages of LoRa and ZigBee

it seems if I plan on adding lots of Zigbee is the best for this challenge

2.6 Review key of research Papers

The following are the research papers I used

1. zhao

   In my research, I found multiple projects that are similar to mine In Zhao(2023)(16, zhao) used LORA to track light sensitivity, air pressure one of the challenges Zhao came across was Attenuation as stated above and also the author came across the problem of not having sufficient solar panels

2. Daniel

   Another paper I found in my research is by Daniel (17) In this, Daniel discusses modeling radio wave propagation in a forest environment which isn't in the scope of the project Daniel's work shows that a better approximation for transmission loss was a key read to under what happens on a more in-depth scale in my project

3. Anna

   (18) in Anna's paper she mainly used LORA where she compared line of sight and the non-line line of sight environments in urban and forested areas this paper aims to study the effects of signal propagation in different environments.

4. ITU

   (1) in ITU in most research papers I found it referred back to this document this document was very helpful in terms of understanding Attenuation and challenges that face

2.7 Summary

This report highlights the challenges at come from transmitting data in a wooded area these challenges are the following:

1. Attenuation
2. Absorption

In a wooded area, we established that Attenuation occurs due to the reflection, and penetration of radio through any type of medium. We established that our antenna will have to be in the Mhz range but will still have signal loss /errors due to Absorption of the signal received due to rain or water being in the signal path we have yet to consider the non-line of sight environment but this is to be discussed when prototyping, this report mainly focuses on the hardware where the focus is on sensors such as:

- Temperature
- Light
- Motion
- Humidity

The report focuses on how to read this data from a Software perspective the code will be an object-oriented program where the code will be separated into different blocks of code so the file size is minimized and leads to a faster compile time.

Chapter 3

Methodology

3.1 Introduction

In this Section the proposed methodology of this project will be discussed this will cover the following:

1. The Procedure of the project
2. The Additional research
3. The setup of the Raspberry Pi
4. The Software Model Development
5. The Data Analysis Methods
6. The validity and reliability
7. The Limitations and Delimitation
8. The timeline

3.2 Procedure

The following are the steps of this project:

1. Consider the environment in which we commutate across
2. Determine the desired range for the sensor to operate in.
3. Find the wide range of components available
4. Limit the base hardware based on the constraints of the project
5. Select hardware based on these constraints
6. State the software needed for the project
7. State how to set up the software
8. State the software needed to drive this sensor
9. Write unit test to develop the software
10. Discuss how to track the sensor data
11. Discuss the timeline of the project
12. Discuss the limitations

3.3 Setup of Raspberry Pi

Firstly once you have your Pi here is a quick guide to setting the pi are the following:

1. Unpack the pi be sure to connect the keyboard mouse and HDMI cable
2. Download the Raspberry Pi imager and select the 32-bit recommended os
3. Put the micro_SD card into the pi once the pi is set you can make subdirectories for this project type the following:

   ```
   git clone https://github.com/mistaherd/meshnetwork_in_forest.git
   ```

   This will download the essential environment for setting up the Pi Initial this will have to be built out through the process of the project look at the timeline Section
4. Next configure the legacy camera options

   ```
   sudo raspi-config
   ```

5. Download the following tools from Linux

   ```
   sudo apt update
   sudo apt install raspistill
   sudo apt install tmux
   ```

6. set working directory and activate the virtual environment

   ```
   cd <path/to/of/your/own>/meshnetwork_in_forest
   source env/bin/activate
   ```

3.4 Additional Research

This section will discuss any extra research done on the project. in this section we will discuss the following:

1. ADC
2. Radio module

3.4.1 ADC

The MCP3008 was not available when ordering parts, Another part for this was chosen which is the DFR0553 which has the following:

1. a supply voltages(VCC) of 3.3 to 5 v
2. Analog signal detection 0 to 5v
3. 4 analog channels
4. resolution of 16 bits
5. Operating current of 3mA

3.4.2 Radio module

for this section, we want to keep the following in mind :

1. We want a module that will send and receive data
2. we don't want an expensive solution due to wanting to have multiple nodes
3. must we pick a standard?
4. what module has an open source project on it
5. How do we set up a mesh network with this

Do we need a radio standard?

Let's assume we communicate with two pi via wires we know that an interference will occur when we commutation that is wireless we can have multiple cases where interference can occur these are the following:

1. the signal being reflected off objects such as trees
2. the signal can reach the receiver due to an object blocking the antenna
3. the signal isn't power to be picked up by the receiver

one essential part of this project is the ability to have our nodes have an address to set this up from a communication preceptive we could develop this when there is an open-source project that has sorted out the routing for you. the the only issue with this approach is if any issues come from the open-source project we will inherit the bugs with this in mind the following standards were found

1. LoRa

LoRa

In (19) Lora is used that organize sensor data from all nodes in the spanning tree toward the root(laptop /PC) this can be shown by the following: this proves it possible to make a mesh

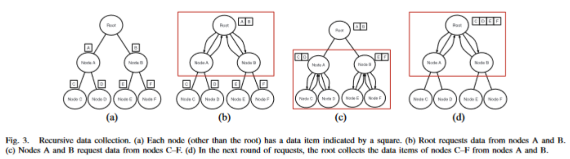


Figure 3.1: protocol Wu used(wu_lie et.al,2023:16705)

network using Lora.

from looking online Lora has more projects that are open source meaning we can use it. freely for example

Lora is uses spread spectrum modulation, In (20) spread spectrum is apparent in Shannon's theorem which states the channel capacity C the upper limit on the information rate of data that can be communicated at a lower error rate through the received signal power S:

$$C = B\log_2(1 + \frac{S}{N})$$

Where B is the is the bandwidth of the channel in hertz. Where the bandwidth is:

$$B = F_{max} - F_{min}$$

spread spectrum creates a pseudo-random code sequence that modulates the data signal which will determine how the signal is spread out.

To simulate the system we can use the following FIR response as an example in a given

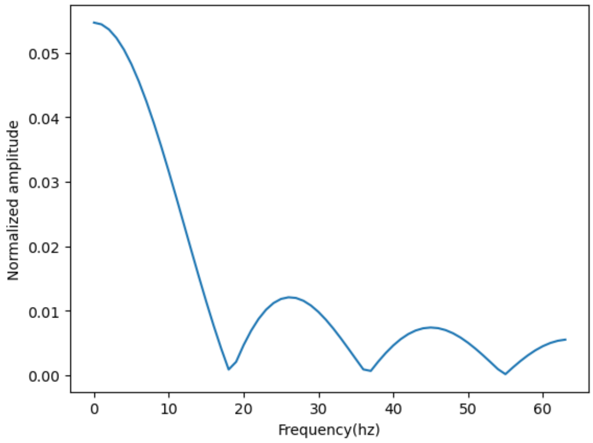


Figure 3.2: sample graph of a FIR response

medium of transmission, each bandwidth is the length of the sinc-roll-off which degrades depending on the impulse response in this given bandwidth channels are separated in the same fashion.

3.4.3 What is the difference between a port and a channel

In (21) "A port is a virtual point where network connections start and end. Ports are software-based and managed by a computer's operating system. Each port is associated with a specific process or service. Ports allow computers to easily differentiate between different kinds of traffic: emails go to a different port than webpages, for instance, even though both reach a computer over the same Internet connection."

3.4.4 Why the MM2 Series 900 MHz wasn't picked

When ordering the parts for this module issues were due to the company not selling the product to enterprise-level businesses so then two alternative radio modules were found:

1. SB Components LoRa HAT for Raspberry Pi
2. RPIZ SHD LORA433 Raspberry Pi Shield - LoRa, 433 MHz, SX1268

when we compare these we get the following table:

Modules	Tx/RX Voltage	Frequency	Range	TX/RX power	Through put	Error detection	Rx sensitivity	Hopping channel
SX1268 433M LoRa HAT	5v	410.125~493.125MHz or 850.125~930.125MHz	5KM(Sunny day; open area; Antenna: AUX 5dBi, Height 2.5m; Air Speed: 2.4kbps)	11ma /100ma	0.3Kbps	None	-147dBm@0.3Kbps (On air)	None
SB Components LoRa HAT for Raspberry Pi	5v	915/868/433 MHz	5km	22dBm	0.3Kbps	None	N/A	None

Table 3.1: Comparing New Radio modules

3.4.5 SB Components LoRa HAT

An E22-900T22S on the board has a throughput rate of 0.3kbps-62.5kbps so the maximum time it will take to get to a node will be around 16 seconds depending on the distance, the module has two ways of interacting with the board:

1. Pi
2. USB to Windows desktop In this configuration we can test a single node from our laptop this is just to test sending messages across the serial

This hat supports three frequencies:

1. 868 Mhz
2. 433 Mhz
3. 915 Mhz

E22-900T22S

E22-900T22S is a wireless serial port module (UART) based on SEMTECH's SX1262 RF chip. It has multiple transmission modes, working in the 850.125MHz 930.125MHz, (default 900.125MHz) which has the following functions:

1. LoRa spread spectrum
2. Listen before talk(LBT):

   The module will monitor the channel before transmitting data if the environment exceeds the threshold. it will be delayed.
3. RSSI(Received Signal Strength Indicator):

   It's a measurement of how strong a radio signal is when it's received by a device, This is used in tandem with the LBT function
4. Networking function:

   The module can implement multi-level repeater networking, as discussed above section when a single is sent over a long distance it gets weaker, and walls floors and other objects can block or distort this signal, A repeater receives these signals and simply amplifies and retransmits the data, Multi-level has secondary repeaters that will boost the signal further in order to avoid signal loss.
5. Ultra-low power consumption:
6. Broadcast monitoring:Set the module address to 0xFFFF, which can monitor the data transmission of the module

3.5 Software Module Development

This section is here to discuss the method we took for developing software for the following:

1. Sensors
2. ADC
3. Camera

4. Radio module
5. Memory management
6. TDD

3.5.1 Sensors

This Section will discuss the following:

1. DHT22
2. AS312
3. DFR0026

To see the light sensor look on page 33

DHT22

For this section, we used the following libraries:

```
#!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/lib/
    python3.11
import adafruit_dht
import board
import pandas as PD
```

This uses the library from this link

1. we define our class

```
    class DHT22:
    ##Set DATA pin to pin 4
        def __init__(self):
            "This will set up the  data pin  for  DHT2"
            # self.dhtDevice =adafruit_dht.DHT22(board.D4)
            self.dhtDevice =adafruit_dht.DHT11(board.D4)
            self.humidity=self.dhtDevice.humidity
            self.temperature=self.dhtDevice.temperature
```

In this class, we have defined our DhT device as 11 seeing as the DHT22 was broken so we set our GPIO pin 4 and set the variables that read the sensor data

2. Next we read the data from the following function.

```
        def Read_DHT22_data(self)-> tuple[float,float,str]:
        """ This  will set a DHT instance and  return the 
           data from the sensor"
        try:
            return self. temperature, self.humidity
        except RuntimeError as e:
            print(f"Error reading sensor: {e}")
            return None, None
```

this will return the temperature and humidity if the sensor is not connected this will return nothing. next use the following:

```
    if __name__ =="__main__":
        DHT22()
```

AS312

1. For this we import the following libraries:

```
#!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/lib/python3.11
import RPi.GPIO as GPIO
import time
```

2. Next we set up our variables in the class

```
class AS312:
def __init__(self):
    "connect the AS312 to pin 17"
    self.pin_number=17
    self.GPIO=GPIO
    self.GPIO.setmode(GPIO.BCM)
    self.GPIO.setup(self.pin_number,GPIO.IN)
    self.current_state=0
```

This sets the current state as 0

3. next we detect movement

```
    def read_state(self)->bool:
        time.sleep(0.1)
        self.current_state =bool(self.GPIO.input(self.pin_number))
        return self.current_state
```

DFR0026

From the repository DFRobot_ADS1115, the following is considered:

1. Import the libraries:

```
#!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/lib/python3.11
from DFRobot_ADS1115 import ADS1115
import time
```

2. Define our variables:

```
class DFR0026():
    def __init__(self):
        self.ADS1115_REG_CONFIG_PGA_6_144V = 0x00  # 6.144V range = Gain 2/3
        self.ADS1115_REG_CONFIG_PGA_4_096V = 0x02  # 4.096V range = Gain 1
```

```
        self.ADS1115_REG_CONFIG_PGA_2_048V = 0x04  # 2.048V
            range = Gain 2 (default)
        self.ADS1115_REG_CONFIG_PGA_1_024V = 0x06  # 1.024V
            range = Gain 4
        self.ADS1115_REG_CONFIG_PGA_0_512V = 0x08  # 0.512V
            range = Gain 8
        self.ADS1115_REG_CONFIG_PGA_0_256V = 0x0A  # 0.256V
            range = Gain 16
        self.ads1115 = ADS1115()
        self.ads1115.set_addr_ADS1115(0x48)
        self.ads1115.set_gain(self.
            ADS1115_REG_CONFIG_PGA_6_144V)
        self.adc_channel = 0
```

This configures all the pins and sets the associative gain.

3. Read the analog channel:

```
    def read_voltage(self):
        return self.ads1115.read_voltage(self.adc_channel)
```

3.5.2 Camera

Here are the steps for module development of the Camera:

1. installs the following libraries:

```
#!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/lib/python3.11
from picamera2 import Picamera2 ,Preview
from time import sleep
from datetime import datetime
```

2. we dine our class variables

```
class Raspberry_Pi_VR_220:
    def __init__(self):
        "setup an instance  for the  camera"
        self.timestamp=datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
        self.fname ='/home/mistaherd/Documents/Github/meshnetwork_in_forest/Images_camera/{}.png'.format(self.timestamp)
        self.camera=Picamera2()
        self.camera_config=self.camera.create_preview_configuration()
        self.timeamount=2
```

3. make the function for taking a picture

```
    def take_pic(self)-> str:
        "This will take  a picture from the camera"
        self.camera.configure(self.camera_config)
        self.camera.start_preview(Preview.QTGL)
        self. camera.start()
        sleep(self.timeamount)
        self.camera.capture_file(self.fname)
        return self. fname
```

3.5.3 Memory Management

For this, we want to read data and append and check the memory size. Here are the following steps:

1. import the following libraries:

```
#!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/lib/python3.11
import pandas as PD
from DHT22 import DHT22
from AS312 import AS312
from DFR0026 import DFR0026
import glob
import re
import subprocess
```

2. define our class senors

```
class sensor_data:
    def __init__(self):
        self.dht22 = DHT22()
        self. humidity, self.temperature=self.dht22.Read_DHT22_data()
        self.AS312=AS312(17)
        self.motion_detected =AS312.read_state()
        self.DF0026 =DFR0026()
        self.light_value=self.DF0026.Read_data()
        self.fname="sensor_data.csv"
```

3. We write and append our data to the CSV file

```
def write_append_csv(self):
data = { "Timestamp": self.timestamp,
    "Temperature(oc)": self.Temperature,
    "Humidity(%)": self.humidity,
    "Light(lux)": self.light_value,
    "Motion Detected": self.motion_detected
    }
df = pd.DataFrame(data)
if glob.glob(self.fname):
    df.to_csv(self.fname,mode='a' ,index=False,header=False)
else:
    df.to_csv(self.fname,mode='w' ,index=False)
```

4. Next we define our variables for testing memory

```
    class Memory_tester():
        def __init__(self):
            self.units={"K":10e3,"M": 10e6,"G":10e9}
            self.regex ="\d{4}\.\[0-9]{1,3}[K,M,G]"
            self.fname="../bash_scrpits/memorytest.sh"
            self.output_bash=subprocess.check_output(["bash",
               self.fname],universal_newlines=True)
```

5. Next we check our memory

```
        def check_memory(self):
            try:
                if re.search(self.regex,self.output_bash):
                    value,unit=match.group(0).split()
                    try:
                        return float(value)*self.units[unit]
                    except KeyError:
                        raise ValueError(f"unknown␣unit:␣{
                           unit}")

            except subprocess.CalledProcessError as e:
                raise ValueError(f"Error␣running␣script:{e.
                   output}")
```

6. we then make an error if it's using 20 percent memory

```
        def error_check(self):
            mem=self.check_memory()
            max=32*10e9
            if mem >= 0.2* max:
                raise MemoryError("memory␣on␣pi␣is␣about␣to␣␣
                   used␣up")
```

7. to make sure our class runs from another Python file

```
        if __name__=="__main__":
            sensor_data()
            Memory_tester()
```

3.5.4 Radio module

This section is based on the GitHub repository: `https://github.com/sbcshop/Lora-HAT-for-Raspberry-Pi` here are the following approaches for this module

1. First import the following libraries:

```
        #!/home/mistaherd/Documents/Github/
           meshnetwork_in_forest/env/lib/python3.11
        import time
        import serial
```

```
        import pandas as PD
        import numpy as np
        import threading
        import base64
        from memory_mangment import sensor_data
```

2. we define our class and its constants

```
        class Transciever:
            def __init__(self):
                self.transceive_ser=serial.Serial(port='/dev/
                   ttyS0',baudrate=9600,parity=serial.
                   PARITY_NONE,stopbits=serial.STOPBITS_ONE,
                   bytesize=serial.EIGHTBITS,timeout=1)
                self.message="Hello world!"
                self.chunk_size=240
                self.txt_fname="/home/mistaherd/Documents/
                   Github/meshnetwork_in_forest/Tests/
                   transmited_text.txt"
                self.png_fname="/home/mistaherd/Documents/
                   Github/meshnetwork_in_forest/Images_camera
                   /camera_output_2024-05-19_13_25_18.png"
                self.csv_fname=sensor_data().fname
                self.timelimit=time.time()+6
                self.recived=self.transceive_ser.in_waiting
                self.event=threading.Event()
```

Where "self.transceiver_ser" sets our serial port up which in Linux is '/dev/ttyS0' we can control timeout to be 6 seconds but for this section will keep it at 1, "self. event" it setting a triggering an event to occur in the code "

3. Setup an interrupt

```
        def serial_interrupt(self):
        if self.recived:
            self. event.set()
```

If there is any information to be sent on the wireless channel this will stop all operations

4. Make a process that will calculate the bytes before sending the data

```
        def cal_bytes(self)-> int:
            return len([bytes(self.data[i],'utf-8').hex() for
                i in range(len(self.data))])
```

5. Test sending and receiving Hello world!

```
        def transceive_test_message(self,transceive:bool):
            "send /receive a hello world"
            if transceive:
                # self.message
                #transmite
```

```
                self.transceive_ser.write(bytes(self.message , 'utf -8’))
                time.sleep(0.2)
            if not transceive:
                while time.time()< self.reive_timelimit:
                    self.transceive_ser.attachInterrupt(self.serial_interrupt)
                    if self. event.is_set():
                        data_read=self.transceive_ser.readline()
                        data=data_read.decode("utf -8")
                        print("message␣received:",data)
                        self. event.clear()
```

6. test send/receiving a txt file

```
        def transceive_test_txt_file(self ,transceive:bool):
            "send␣/revive␣a␣txt␣file"
            if transceive:
                with open(self.txt_fname ,’r’) as f:
                    data=f.read()

                self.transceive_ser.write(bytes(data ,’utf -8’))
                time.sleep(0.2)
            if not transceive:
                while time.time()< self.timelimit:
                    self.transceive_ser.attachInterrupt(self.serial_interrupt)
                    if self. event.is_set():
                        data_read=self.transceive_ser.readline()
                        data=data_read.decode("utf -8")
                        print("message␣received:",data)
                        self. event.clear()
                        return data
```

7. Test sending and revecing csv file

```
        def transceive_test_csv(self ,transceive:bool):
            if transceive:
                with open(’/home/mistaherd/Documents/Github/meshnetwork_in_forest/main/sensor_data.csv’,’r’) as f:
                    data=f.readlines()
                data=’’.join(data)
                lora.write(bytes(data ,’utf -8’))
                time.sleep(0.2)
            if not transceive:
                while time.time() <self.timelimit:
                    self.transceive_ser.attachInterrupt(self.serial_interrupt)
```

```
                    if self. event.is_set():
                        data=self.transceive_ser.readlines()
                        output=[data[i].decode()[:-1].split(",") for i in range(len(data))]
                        df=pd.DataFrame(output)
                        self. event.clear()
                        return df
```

8. Test sending and receiving an image file

```
        def Transcevie_png_file(self,transceive:bool):
            "Transmit a PNG file"
            if transceive:
                with open(self.png_fname, 'rb') as f:
                    self.data = f.read()
                if self.cal_bytes()>self.chunk_size:
                    chunks=[data[i:i+self.chunk_size] for i in range(0,len(self.data),self.chunk_size)]
                    for chunk in range(len(chunks)):
                        encoded_chunk=base64.b64encode(chunk)
                        self.transceive_ser.write(encoded_chunk)
                else:
                    raise ValueError("Image file must be corrupted")
            if not transceive:
                output=[]
                self.transceive_ser.attachInterrupt(self.serial_interrupt)
                if self. event.is_set():
                    while(self.transceive_ser.read() != b''):
                        data_read = self.transceive_ser.read()
                        print("bytes reviced %a"%data_read)
                        output.append(base64.b64decode(data_read))
                    output=b"".join(output)
                    self. event.clear()
                    return output
```

9. For the demo make sure to define the following:

```
        def transive_choice(self,arugement):
            """ run this for demo"
            if not self. event.is_set():
                #transmit something
                self.transmit=True
                choice ={
                    1:lambda :self.transceive_test_message(self.transmit),
```

```
                    2:lambda :self.transceive_test_txt_file(self.transmit),
                    3:lambda :self.transceive_test_csv(self.transmit),
                    4:lambda: self.Transcevie_png_file(self.transmit)}
                choice[arugement]()
            #revived somthing
            self.transmit=False
            choice[self.user_message]()
```

10. has a file as a module

```
        if __name__=='__main__':
            Transciever()
```

The following is the process of development:

- To make a test that will be there for the coding section of the project

this section will discuss the following for testing:

1. 1 x DHT22
2. 1 x DFR0026
3. 1 x AS312
4. 1 x SB Components LoRa HAT for Raspberry Pi
5. 1 x MCP3008
6. 1 x Raspberry Pi VR 220 Camera
7. 1 x Li-polymer Battery HAT

DHT22

According to the data sheet (4) seen the data is 8 bits and the range at which this operates at -40 to 80^{o}c for temperature meaning we have at least 7 bits in the exponent to represent the measured value. to represent the high end of this sensor I used the following calculation:

$$2^6 + 2^4 = 80$$

which means we have 2 bits dedicated to decimal place so the high temperature is 80.3^{o}c for the lowest temp, we have 6 bits to represent - 40 due to 2s complement so the lowest will be -40.3^oC so with that, that establish we must make a unit that will do the following:

1. Test if the output is a float
2. Test the high end of the temp sensor so it reads 80.3 as the highest
3. Test for the lowest temp around

be sure to follow the steps for folder setup and follow the instructions on page **??**. we get the following sample code:

Listing 3.1: sample test intial code

```
import unittest
from protest import Read_DHT22
class test_project_code(unittest.TestCase):
    def test_DHT_22_temp_output_type(self):
        self.assertIsInstance(Read_DHT22, float)
    def test_DHT22_temp_range(self):
        self.assertGreaterEqual(Read_DHT22,-30.3)
        self.assertLessEqual(Read_DHT22,80.3)
```

This code imports unit tests. the DHT22 is a Python file where we can install functions from other Python files this can be useful for testing purposes then we initialized a test class called Unittest. test as our first function of the class we check if the number of the output is a float or not this is for testing temperature the next function we test for is the range look at the datasheet online. This code is simply testing the limits of the DHT22 for humidity, the Datasheet ranges from 0 to 100 % we want to test for the following:

1. Test if the recorded output is a tuple
2. Test if the temperature recorded is a float or integer
3. Test if the temperature recorded is in the range from -30 to 80.3
4. Test if the humidity recorded is a float or integer
5. Test if the humidity recorded is between 0 and 100

this led to the following code

Listing 3.2: sample test for DHT22

```
import unittest
from DHT22 import DHT22
dht22_instance=DHT22()
class test_project_code(unittest.TestCase):
    hum,temp=Read_DHT22(2)
    def test_DHT22_output_type(self):
        self.assertIsInstance(dht22_instance.Read_DHT22_data,
           tuple)

    def test_DHT_22_temp_output_type(self):
        self.assertIsInstance(temp, (int,float) )

    def test_DHT22_temp_range(self):
        self.assertGreaterEqual(temp,-30.3)
        self.assertLessEqual(temp,80.3)
```

seen as we expect our sensor to print out humidity and temp values we set the output to a tuple to test for this we use "isInstacne" which will test if it is a tuple next, we test for the limits of the humidity

DFR0026

According to the data sheet (9) we must keep in mind that this component is connected to an ADC this will give me the following test conditions:

1. Test if the output is a dictionary with elements of a string and integer in it.
2. Test the range of this with the upper limit being 5v the analog voltage meaning:

$$\begin{aligned} \text{output} &= \frac{2^n \cdot \text{Analogue Input value}}{\text{Reference voltage}} \\ &= \frac{2^{16} \cdot 5}{5} = 65536 \end{aligned}$$

3. test the lover limit being 3.3v as the analogue$\frac{2^{16} \cdot 3.3}{5} = 43253$

Listing 3.3: unit test for DFR0026 and MCP3008

```
    import unittest
    from DFR0026 import DFR0026
    class test_project_code(unittest.TestCase):
    def test_DFR0026_out_type(self):
```

```
        self.assertIsInstance(DFR0026().read_voltage(),dict[str,
            int])
    def test_DFR0026_out_range(self):
        self.assertLessEqual(DFR0026().read_voltage(),65536)
        self.assertGreaterEqual(DFR0026().read_voltage(),43253)
```

AS312

for this section we want our tests to be the following:

1. test if output type is boolean

we can now add to the snipppet :

Listing 3.4: unit test for AS312

```
    import unittest
    from AS312 import AS312
    AS312_instance=AS312()
    class test_project_code(unittest.TestCase):
    def test_AS312_out_type(self):
        self.assertIsInstance(AS312_instance.read_state,booll)
```

seen as this is a motion sensor our output will be true or false.

Raspberry Pi VR 220 Camera

according to the datasheet (22) The resolution it uses is 1080p50 which is 1920x1080p so our tests will have to incorporate the following:

1. Test if the file can run

This would lead to the following code snippet.

Listing 3.5: camera unit test

```
    def test_Raspberry_Pi_VR220_out_shape(self):
        self.assertEqual(camera_obj.run.returncode, 0)
```

This function checks the pixel count or resolution

memory module

in this section will discuss the following:

1. silicon power 32GB For this use a bash script(see this on page 21) to test the size in a certain range for the silicon SD card
2. which will be 0B to 32GB

```
        def Test_memory_silicon_power_32GB(self):
            self.assertLessEqual(memorytest_obj.check_memory
                ,32e9)
            self.assertGreaterEqual(memorytest_obj.
                check_memory,0)
```

SB Components LoRa HAT

for this section, we want to test the following:

1. Test the serial connection
2. Test the serial interrupt
3. Test the sending/receiving of a message like "Hello world"
4. Test the sending/receiving of a text file
5. Test the sending/receiving of a CSV file
6. Test the sending/receiving of an image file

This will give the following code:

```
import unittest
from Radiomodule import Transciever
Transciever_instance=Transciever()
class test_project_code(unittest.TestCase):
    def test_serial_connection(self):
        self.assertIsInstance(Transciever_instance.
           transceive_ser,serial.Serial)
    def test_serial_interrupt(self):
        self.assertEqual(Transciever_instance.event.is_set()
           ,(False,True))
    def test_transceiver_test_message(self):
        message=Transciever_instance.message
        Transciever_instance.transceive_test_message(True)
        received_message=Transciever_instance.
           transceive_test_message(False)
        self.assertEqual(message,received_message)
    def test_transceiver_test_txt_file(self):
        txt_fname=Transciever_instance.txt_fname
        with open(txt_file,'r') as f:
            expected_txt=f.read()
        Transciever_instance.transceive_test_txt_file(True)
        received_txt_file=Transciever_instance.
           transceive_test_txt_file(False)
        self.assertEqual(expected_txt,received_txt_file)
    def test_transciver_test_csv(self):
        csv_fname=Transciever_instance.csv_fname
        expected_df=pd.read_csv(csv_fname)
        Transciever_instance.transceive_test_csv(True)
        reviced_df=Transciever_instance.transceive_test_csv(
           False)
        self.assertEqual(expected_df,reviced_df)
    def test_trancsive_img_file(self):
        img_fname=Transciever_instance.png_fname
        with open(img_fname,'rb')as f:
            expted_out=f.read()
        Transciever_instance.Transcevie_png_file(True)
```

```
            received_bin=Transciever_instance.Transcevie_png_file
               (False)
            self.assertEqual(expted_out,received_bin)
    if __name__ == '__main__':
        unittest.main()
```

Unit test iterations

This section will discuss the iterations of the unit test following are the iterations of our unit testing:

1. In the first iteration the following was worked on:
   - DHT22
   - DFR0026
   - AS312
   - Camera
   - turbo 1GB
   - silicon power 32Gb

   This was formed in the when the lit review was written

2. Most of the iterations were coming out of the code and testing each section

3.6 Data Analysis Methods

In this section we will discuss the tool we use to analyze the sensor data the following will be discussed:

1. Met Eireann weather forecast API
2. Matplotlib

3.6.1 Met Eireann weather forecast API

we can compare our humidity and temperature and compare this with the output of this API

3.6.2 Matplotlib

we can use this library to plot our changes in the data.

3.7 Timeline

The research phase will be conducted from 20/09/2023 to August 9, 2025. The prosed impanation phase is from January till August 9, 2025The project file timeline can be visualized as the following:

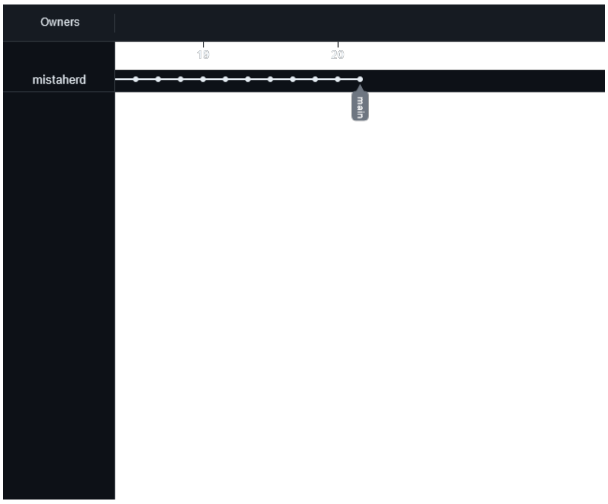


Figure 3.3: Visualized timeline of project files versions

This was gotten from Git Hub.
The project code can be visualized as the following:

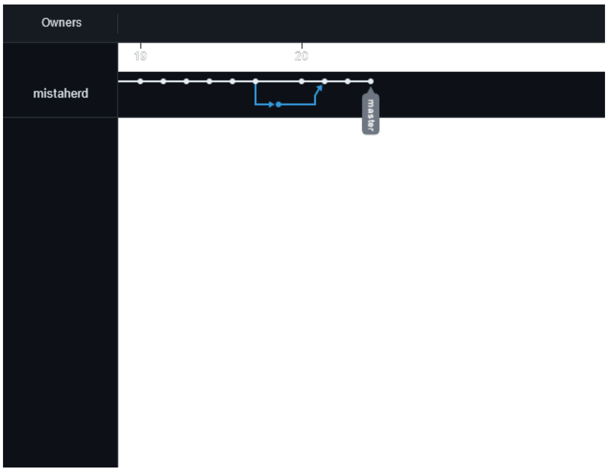


Figure 3.4: Visualized timeline of project programming timeline

Chapter 4

Results

In this section we will be showing results for different aspects of this project this will include the following:

1. Recorded data from sensors
2. Recorded data from the transceiver
3. Recorded data from testing the mesh network

4.1 Recorded data from sensors

This section will have tables from the following components:

1. DHT22 **heat and temp**
2. AS312 **Motion**
3. DFR0026 **Light**
4. Raspberry Pi VR 220 **Camera**

4.1.1 DHT22

Results during protypeing

date/time of record	Temperature	Humidity
2024-05-03_17-31-52	20	49
2024-05-03_17-43-54	20	49
2024-05-03_17-31-52	20	49
2024-05-03_17-43-54	20	49

Table 4.1: Recorded data from DHT22 on the $5^{th}$of march

If this is plotted the following is gotten: last we tested if our code satisfies our Python code after testing the unit test code we updated the following message

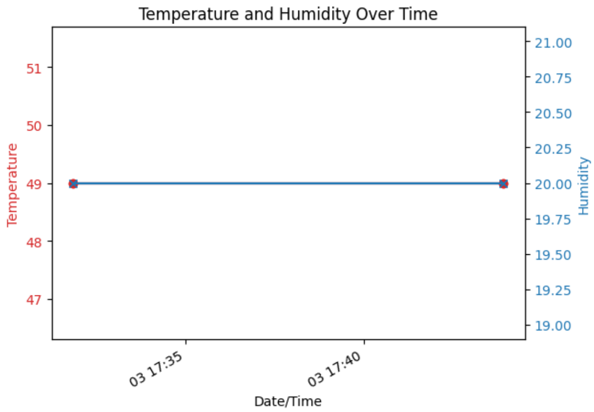


Figure 4.1: Temperature and Humidity plotted over

Figure 4.2: unit test message for DHT22 module

4.1.2 AS312

date/time of record	motion detected(yes/no)
2024-03-25_15-02-57	False
2024-03-25_15-04-37	True
2024-05-03_18-07-51	False
2024-05-03_18-18-37	True

Table 4.2: Recorded data from AS312 on the August 9, 2025

4.1.3 DFR0026

For the first test, we got the following table: If this is plotted we get the following:

Date/time of record	lux values(lux)
2024-03-25_15-02-57	940
2024-03-25_15-03-13	945
2024-03-25_15-04-37	4963
2024-05-03_18-54-57	1284

Table 4.3: Recorded data from DFR0026 on the 25th of march 2024

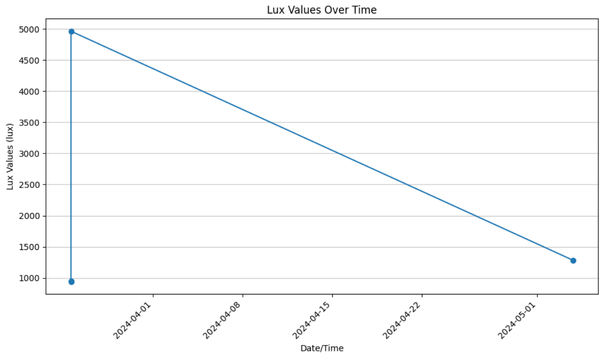


Figure 4.3: lux values overtime

4.1.4 Raspberry Pi VR 220

When testing the Raspberry Pi VR 220 the following picture was taken:

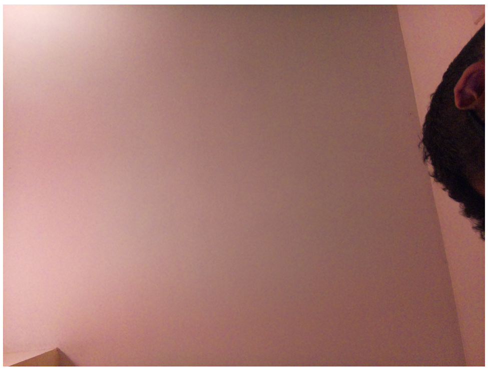


Figure 4.4: A photo from 25th of march 2024

4.2 Recorded data from transceiver

When testing the radio module the following was tested:

1. Sending a message across the serial
2. Sending a txt file across the serial
3. Sending a CSV file across the serial
4. Sending an image file across the serial

   no results were obtained due to factor on 52

Chapter 5

Discussion

In this section we will discuss the following:

1. Results of the sensor data
2. Results of the camera
3. Results of the Lora module

5.1 Discussion of results of Sensor data

in this section, the data gathered from the sensor will be discussed Note:(**all tests here were conducted indoors**):

1. DHT22 (Temperature & Humidity):

   On page **??** compared to the average room temperature which is 20^oc, The range of humidity in a room is from 30% to 60% in the plot on page **??** note in the file sensor_data.csv there entries that are 0, this is due to testing different components and make these values as 0. during the development stage of this project we ran unit tests this module as seen on page 49.

2. AS312 (Motion):

   As seen on page 49 this will output a table that is True when an object is detected and false when no object is detected

3. DFR0026 (Lux): As seen on page **??** is a table full of lux values according to this link we ideally want a lux value of 800 to 1700 lux which satisfies these conditions.

5.2 Discussion of Results of camera data

As seen on page **??** which shows an image that was taken by the camera attached to the pi

5.3 Discussion of Lora module

While updating the repository for this section the Pi's SD cards were corrupted due to a faulty binary file which was later fixed but this error led to the following error message:

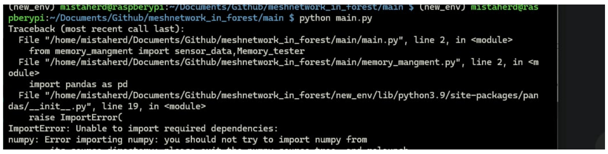


Figure 5.1: environment error message

Chapter 6

Conclusion & Future Work

6.1 Conclusion

In this final year project, We have explored the implementation and development of mesh networks within the challenging environment of a forest. Through:

1. The wireless commutation environment
2. The desired characteristics of the sensors
3. The available electronic devices for this limitation
4. The limited hardware available
5. The fundamental software needed
6. The Setup of the software
7. The examination of the programming software needed
8. The test before the writing of any code
9. The discussion of how to record data
10. The Limitations

6.1.1 Key Contributions

This project has contributed significantly to the understanding of the process of developing mesh networks in forest settings by:

- **Developing a Model:** In this we used serial communication to test the nature of which the network will send/ receive data from the network
- **Addressing Challenges:** in a line of sight environment the sending and receiving of images is a hard task due to how large the file is byte by byte
- **Performance Evaluation:** every method eventually worked via serial but image files had the most problems
- **Practical Implications:** Schedule the sending and receiving of data we can use this to extract the sensor data

- **Familiarity with different tools:** This project forces the project implementation to use the terminal in Linux, bash, In Linux the concept of Dev files are files where the port can be defined for example port 80 on the window would be /dev/tty80, also the addition of tools like ssh which allowed remote connection to the Pi board, batch files used to make these connections easier, A stronger familiarly with git was gained due to the keeping the project version controlled, A display of further knowledge of python was gain due to project such as using virtual environments and how to properly structure OPP program, An understanding of TDD for coding projects was gained where the implementor is forced to use these methods

6.2 Sources of Error

In this section, the following will be discussed:

1. Radio module
2. MCP3008
3. Time management
4. Lack of knowledge of Linux OS
5. Camera
6. Lack of hat for sensors

6.2.1 Radio Module

This section will discuss the problems noticed while working on this:

1. When trying to look for radio modules for the MM2 Series 900 MHz this part couldn't be ordered due to the vendor only accepting business customers this cost 4 weeks of research to look for another module

2. Once the appropriate modules were after looking and noticing that the documentation was very poor it did help as well.

3. One avenue not expected of this was enabling the serial. So once we enable via sudo raspi-config, and use the code from this link (use lora_receiver.py and lora_transmitter.py) when we get the following error:

   Figure 6.1: Issue when trying to write to serial

   This issue took a lot of time to fix a simple chmod rw operation will fix the issue which should be indicated in the documentation but there isn't.

4. After getting a working demo on all use cases of the module the image file caused the most issues this was due to the nature of how large the large data in an image file further research should have been put into how an image is sent across a communication channel.

5. The lack of documentation meant time was spent looking at the components on board and finding how they worked.

6.2.2 MCP3008

In this section the Errors that occurred with this module are the following:

1. When it came to ordering this part, There were issues with the venter so another ADC had to be chosen this caused another few weeks to go by.

6.2.3 Lack of knowledge of Linux os

This section will discuss the errors that occurred theses are the following:

1. Since the lack of experience with Linux is evident this led to time spent learning about certain commands that will help with the project concepts such as piping commands nmap where foreign before this project a lot of time was spent learning the best tools for this project

6.2.4 Camera

This Section will discuss the errors that occurred with this module these are the following:

1. When using the python file "camera.py" this wouldn't comply due to how the old library is no longer supported this costs around 2 weeks due to finding different forms and tracking the right library down eventually. the decision was made to make a bash command to run the camera

6.2.5 Lack of hat for sensors

This Section will discuss the issues of this which are the following:

1. When wiring up all the sensors it took several minutes to wire the devices up which led to miss wiring the temp sensor which blew leading to an order for a replacement to be made
2. This issue along with the could have led to the code being simpler in nature them what is present in the repository

6.3 Future Work

While this project hasn't achieved its core objectives, several avenues for future research and development remain open:

- **Sockets:** The project could have ventured into socket programming, A socket is an endpoint of a two-way communication link between two programs running on the network. we picked serial communication for testing but this is where the main server and client can defined.
- **Helpful Tools:** Linux has a wide range of networking tools such as:
  - **Nmap** is a network scanner designed to discover hosts and services on a computer network. It sends packets and analyzes the responses to gather information
  - **ifconfig** Which provides extensive control over interfaces, addresses and routes
  - **traceroute** Traces the route packets take to reach a destination
  - **route** Displays or manipulates the kernel's IP routing table
  - **arp** Displays and manages the Address Resolution Protocol (ARP) cache, which maps IP addresses to MAC addresses

  - **tcpdump** Captures and analyzes network traffic, useful for diagnosing network issues.
  - **iftop** Displays a real-time bandwidth monitor for network interfaces.

- **Energy Efficiency:** Investigate further what can be done to make our system more energy effective i.e piplineing the sensor data

- **Security:** Investigate how to make our data secure via RSA and different protocols

- **PCB board :** After testing the board one problem was wiring up the sensor a potential for making a custom Pi hat that will provide an easy way of connecting our sensor

6.3.1 Final Remarks

The deployment of mesh networks in forest environments holds immense promise. The work provided here doesn't outline how to achieve this but can be viewed as a method to achieving this nature

Chapter 7

Appendix

Bibliography

[1] ITU, “Recommendation itu-r p.833-4$''_{,R}ecommendationITU - RP.833 - 7_{.}, 2012.$

[2] m. Eirrean, “Weather extreme records for ireland.”

[3] m. eirrean, “Dublin 1920-2010 humidty aveages.”

[4] sparkfun.

[5] wiki, “Lux,” Oct 2023.

[6]

[7] micros.

[8] freewave.

[9] ada.

[10] RS.

[11] N. D. S. A. V. E. Savage, N. and J. Austin, “Radio wave propagation through vegetation: Factors influencing signal attenuation,,” $_{R}adioSci._{,}38, 1088, 2003.$

[12] R. Sabetahd, S. A. Mousavi Ghasemi, R. Vafaei Poursorkhabi, A. Mohammadzadeh, and Y. Zandi, “Response attenuation of a structure equipped with atmd under seismic excitations using methods of online simple adaptive controller and online adaptive type-2 neural-fuzzy controller,” *Computational Intelligence and Neuroscience*, vol. 2022, p. 1–25, 2022.

[13] P. Lunkenheimer, S. Emmert, R. Gulich, M. Köhler, M. Wolf, M. Schwab, and A.Loid, “Electromagnetic-radiation absorption of wate,” tech. rep., Experimental Physics V, Center for Electronic Correlations and Magnetism, University of Augsburg, 86159 Augsburg, German, ?

[14] H. Chen, F. Xing, Q. Yang, Y. Shu, Z. Shi, J. Chen, and Z. Tao, “A lightweight mobile-anchor-based multi-target outdoor localization scheme using lora communication,” *IEEE Transactions on Green Communications and Networking*, vol. 7, no. 4, pp. 1607–1619, 2023.

[15] V. Gavra, O. A. Pop, and I. Dobra, “A comprehensive analysis: Evaluating security characteristics of xbee devices against zigbee protocol,” *Sensors*, vol. 23, no. 21, p. 8736, 2023.

[16] M. Zhao, R.-J. Ye, S.-T. Chen, Y.-C. Chen, and Z.-Y. Chen, “Realization of forest internet of things using wireless network communication technology of low-power wide-area network,” *Sensors*, vol. 23, no. 10, 2023.

[17] Y. C. Daniel, “Modeling of radiowave propagation in a forested environment,” tech. rep., NAVAL POSTGRADUATESCHOOL MONTEREY, CALIFORNIA, 2014.

[18] a. E. Ferreira and F. Ortiz, “A study of the lora signal propagation in forest, urban, and suburban environment$''_{,}tech.rep., Institutnationalderechercheenscienceset$, 2020.

[19] D. Wu and J. Liebeherr, “A low-cost low-power lora mesh network for large-scale environmental sensing,” *IEEE Internet of Things Journal*, vol. 10, p. 16700–16714, Oct 2023.

[20] F. . 2003 and A. C. Information, “An introduction to spread-spectrum communications,” Jan 2023.

[21] c. flare, “What is a computer port? — ports in networking — cloudflare.”

[22]

Appendix A

Python Scripts

A.1 Python Scripts

A.1.1 DHT22

Listing A.1: DHT22code

```
#!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/lib/python3.11
import adafruit_dht
import board
import datetime
import pandas as pd
class DHT22:
##Set DATA pin to pin 4
    def __init__(self):
        # self.dhtDevice =adafruit_dht.DHT22(board.D4)
        self.dhtDevice =adafruit_dht.DHT11(board.D4)
    def Read_DHT22_data(self)-> tuple[float,float,str]:
        try:
            Humidity=self.dhtDevice.humidity
            Temperature=self.dhtDevice.temperature
            timestamp =datetime.datetime.now()
            timestamp = timestamp.strftime("%Y-%m-%d %H:%M:%S")
            return Temperature,Humidity,timestamp
        except RuntimeError as e:
            print(f"Error reading sensor: {e}")
            return None, None
    def write_to_csv(self,filename:str):
        temperature, humidity, timestamp = self.Read_DHT22_data()
        if temperature is not None and humidity is not None and timestamp is not None:
            data = [(temperature, humidity, timestamp)]
            df = pd.DataFrame(data, columns=['Temperature', 'Humidity', 'Timestamp'])
            df.to_csv(filename, index=False)
        else:
            print("Failed to retrieve data from sensor. Data not written to CSV.")
dht_sensor = DHT22()
```

```
dht_sensor.write_to_csv("sensor_data.csv")
```

A.1.2 AS312

Listing A.2: code for AS312

```python
#!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/lib/python3.11
import RPi.GPIO as GPIO
import time
import datetime
import pandas as pd
#pin 17
class AS312:
    def __init__(self,pin_number:int):
        self.pin_number=pin_number
        self.GPIO=GPIO
        self.GPIO.setmode(GPIO.BCM)
        self.GPIO.setup(self.pin_number,GPIO.IN)
        self.current_state=0
        self.timestamp=datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    def read_state(self)->int:
        self.current_state =self.GPIO.input(self.pin_number)
        return self.current_state
    def append_data(self):
        data={
            "Motion Dectected": [self.current_state],
            "Timestamp": [self.timestamp]
        }
        df =pd.DataFrame(data)
        df.to_csv('sensor_data.csv',mode='a' ,index=False,header=False)
pir_sensor = AS312(17)
try:
    time.sleep(0.1)
    current_state =pir_sensor.read_state()
    timestamp=pir_sensor.timestamp
    print("GPIO pin %s is %s" % (pir_sensor.pin_number,current_state))
    if current_state == 1:
         print("Motion dectected")
    pir_sensor.append_data()
except KeyboardInterrupt:
    pass
finally:
    GPIO.cleanup()
```

A.1.3 DFR0026

Listing A.3: Code for DFR00026

```
#!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/lib/python3.11
from DFRobot_ADS1115 import ADS1115
import time
class DFR0026():
    def __init__(self):
        self.ADS1115_REG_CONFIG_PGA_6_144V        = 0x00 # 6.144V range = Gain 2/3
        self.ADS1115_REG_CONFIG_PGA_4_096V        = 0x02 # 4.096V range = Gain 1
        self.ADS1115_REG_CONFIG_PGA_2_048V        = 0x04 # 2.048V range = Gain 2 (default)
        self.ADS1115_REG_CONFIG_PGA_1_024V        = 0x06 # 1.024V range = Gain 4
        self.ADS1115_REG_CONFIG_PGA_0_512V        = 0x08 # 0.512V range = Gain 8
        self.ADS1115_REG_CONFIG_PGA_0_256V        = 0x0A # 0.256V range = Gain 16
        self.ads1115 = ADS1115()
        self.ads1115.set_addr_ADS1115(0x48)
        self.ads1115.set_gain(self.ADS1115_REG_CONFIG_PGA_6_144V)
        self.adc_channel=0
    def read_voltage(self):
        return self.ads1115.read_voltage(self.adc_channel)
        #time.sleep(0.2) after read it
light_vaule=DFR0026()
print(light_vaule.read_voltage())
```

A.1.4 Camera

Listing A.4: Code for Camera

```
#!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/lib/python3.11
from picamera import PiCamera
from time import sleep
from datetime import datetime
class Raspberry_Pi_VR_220:
    def __init__(self):
        """setup␣an␣instan␣␣for␣the␣␣camera"""
        self.timestamp=datetime.now().strftime("%Y-%m-%d_%H:%M:%S")
        self.fname ='/home/mistaherd/Documents/Github/meshnetwork_in_forest/{}.png'.format(self.timestamp)
        self.camera=PiCamera()
        self.timeamount=2
    def take_pic(self)-> str:
        """this␣will␣take␣␣a␣picture␣from␣camera"""
        self.camera.start_preview()
        sleep(self.timeamount)
        self.camera.capture(self.fname)
        self.stop_preview()
        return self.fname
camera=Raspberry_Pi_VR_220()
picture=camera.take_pic()
```

Camera alt

Listing A.5: Code for alternaive code for Camera

```
#!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/
   lib/python3.11
import subprocess
class camera:
    def __init__(self):
        self.run=subprocess.run(["bash","/home/mistaherd/
           Documents/Github/meshnetwork_in_forest/
           bash_scrpits/camerea.sh"])
if __name__=="__main__":
    camera()
```

A.1.5 Memory management

Listing A.6: Code for memory mangement

```
#!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/lib/python3.11
import pandas as pd
from DHT22 import DHT22
from AS312 import AS312
from MCP3008 import DF0026
import glob
import re
import subprocess
class sensor_data:
    def __init__(self):
        self.dht22 = DHT22()
        self.humidity,self.temperature,self.timestamp=self.dht22.Read_DHT22_data()
        self.AS312=AS312(17)
        self.motion_dected =AS312.read_state()
        self.DF0026 =DF0026()
        self.light_value=self.DF0026.Read_data()
        self.fname="sensor_data.csv"
    def write_append_csv(self):
        data = { "Timestamp" : self.timestamp,
            "Temperature(oc)" : self.Temperature,
            "Humidity(%)" : self.humidity,
            "Light(lux)" :self.light_value,
            "Motion Dected": self.motion_dected
            }
        df = pd.DataFrame(data)
        if glob.glob(self.fname):
            df.to_csv(self.fname,mode='a' ,index=False,header=False)
        else:
            df.to_csv(self.fname,mode='w' ,index=False)
class Memory_tester():
    def __init__(self):
        self.units={"K":10e3,"M": 10e6,"G":10e9}
        self.regex ="\d{4}\.\[0-9]{1,3}[K,M,G]"
        self.fname="../bash_scrpits/memorytest.sh"
        self.output_bash=subprocess.check_output(["bash",self.fname],universal_newlines=True)
    def check_memory(self):
        try:
            if re.search(self.regex,self.output_bash):
                value,unit=match.group(0).split()
                try:
                    return float(value)*self.units[unit]
                except KeyError:
                    raise ValueError(f"unknown unit: {unit}")

```

```
        except subprocess.CalledProcessError as e:
            raise ValueError(f"Error running script:{e.output}")
    def error_check(self):
        mem=self.check_memory()
        max=32*10e9
        if mem >= 0.2* max:
            raise MemoryError("memory on pi is about to  used up"
                )
```

A.1.6 Radio module

Listing A.7: Code for Radio module

```
#!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/
    lib/python3.11
import time
import serial
import pandas as pd
import numpy as np
import threading
import subprocess
import base64
from memory_mangment import sensor_data
class Transciever:
    def __init__(self):
        self.transceive_ser=serial.Serial(port='/dev/ttyS0',
            baudrate=9600,parity=serial.PARITY_NONE,stopbits=
            serial.STOPBITS_ONE,bytesize=serial.EIGHTBITS,
            timeout=1)
        self.message="Hello world!"
        self.chunk_size=240
        self.txt_fname="/home/mistaherd/Documents/Github/
            meshnetwork_in_forest/Tests/transmited_text.txt"
        self.csv_fname=sensor_data().fname
        self.timelimit=time.time()+6
        self.recived=self.transceive_ser.in_waiting
        self.event=threading.Event()
    def serial_interrupt(self):
        if self.recived:
             self.event.set()
    def cal_bytes(self)-> int:
        return len([bytes(self.data[i],'utf-8').hex() for i
            in range(len(self.data))])

    # hello world
    def transceive_test_message(self,transceive:bool):
        """send /recive a hello world"""
        if transceive:
             # self.message
             #transmite
```

```
                self.transceive_ser.write(bytes(self.message,'utf
                    -8'))
                time.sleep(0.2)
            if not transceive:
                while time.time()< self.reive_timelimit:
                    self.transceive_ser.attachInterrupt(self.
                        serial_interrupt)
                    if self.event.is_set():
                        data_read=self.transceive_ser.readline()
                        data=data_read.decode("utf-8")
                        print("message␣received:",data)
                        self.event.clear()
        # Text file
        def transceive_test_txt_file(self,transceive:bool):
            """send␣/revive␣a␣txt␣file"""
            if transceive:
                with open(self.txt_fname,'r') as f:
                    data=f.read()

                self.transceive_ser.write(bytes(data,'utf-8'))
                time.sleep(0.2)
            if not transceive:
                while time.time()< self.timelimit:
                    self.transceive_ser.attachInterrupt(self.
                        serial_interrupt)
                    if self.event.is_set():
                        data_read=self.transceive_ser.readline()
                        data=data_read.decode("utf-8")
                        print(data)
        #test csv file
        def transceive_test_csv(self,transceive:bool):
            if transceive:
                with open('/home/mistaherd/Documents/Github/
                    meshnetwork_in_forest/main/sensor_data.csv','r
                    ') as f:
                    data=f.readlines()
                data=''.join(data)
                lora.write(bytes(data,'utf-8'))
                time.sleep(0.2)
            if not transceive:
                while time.time() <self.timelimit:
                    self.transceive_ser.attachInterrupt(self.
                        serial_interrupt)
                    if self.event.is_set():
                        data=self.transceive_ser.readlines()
                        output=[data[i].decode()[:-1].split(",")
                            for i in range(len(data))]
                        df=pd.DataFrame(output)
                        print(df)

        #Test png,jpg
```

```python
        def Transcevie_png_file(self):
            """Transmit a PNG file"""
            if transceive:
                with open(self.png_fname, 'rb') as f:
                    data = f.read()
                chunks=[data[i:i+self.chunk_size] for i in range(0,len(data),self.chunk_size)]
                for chunk in range(len(chunks)):
                    encoded_chunk=base64.b64encode(chunk)
                    self.transceive_ser.write(encoded_chunk)
            if not transceive:
                output=[]
                self.transceive_ser.attachInterrupt(self.serial_interrupt)
                if self.event.is_set():
                    while(self.transceive_ser.read() != b''):
                        data_read = self.transceive_ser.read()
                        print("bytes reviced %a"%data_read)
                        output.append(base64.b64decode(data_read))
                    output=b"".join(output)
                    with open("recived_img.png", 'wb') as f:
                        f.write(output)
        def transive_choice(self,arugement):
            """ run this for demo"""
            if not self.event.is_set():
                #transmit something
                self.transmit=True
                choice ={
                    1:lambda :self.transceive_test_message(self.transmit),
                    2:lambda :self.transceive_test_txt_file(self.transmite),
                    3:lambda :self.transceive_test_csv(self.transmit),
                    4:lambda :self.Transcevie_png_file(self.transmit)}
                choice[arugement]()
            #revived somthing
            self.transmit=False
            choice[self.user_message]()
    if __name__=='__main__':
        Transciever()
```

Appendix B

TDD Script

B.1 TDD scripts

This section is for All the TDD sections of this report this section will share the TDD of the following:

1. DHT22
2. AS312
3. DFR0026
4. Raspberry Pi VR 220 Camera
5. Memory management
6. SB Components LoRa HAT for Raspberry Pi

B.1.1 DHT22

Listing B.1: DHT22 unit test

```
#!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/
    lib/python3.11
from DHT22 import DHT22
import unittest
dht22_instance=DHT22()
hum, temp,ts=dht22_instance.Read_DHT22_data()
class test_project_code(unittest.TestCase):
    # DHT22
    def test_DHT22_output_type(self):

        self.assertIsInstance(dht22_instance.Read_DHT22_data,
            tuple)

    def test_DHT_22_temp_output_type(self):
        self.assertIsInstance(temp, (int,float) )

    def test_DHT22_temp_range(self):
        self.assertGreaterEqual(temp,-30.3)
        self.assertLessEqual(temp,80.3)

```

```
        def test_DHT22_hum_output_type(self):
            self.assertIsInstance(hum,(int,float))

        def test_DHT22_hum_range(self):
            self.assertGreaterEqual(hum,0.0)
            self.assertLessEqual(hum,100.0)
    if __name__ == '__main__':
        unittest.main()
```

B.1.2 AS312

Listing B.2: Code for unit test of AS312

```python
#!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/
    lib/python3.11
import unittest
from AS312 import AS312
class test_project_code(unittest.TestCase):
    def test_AS312_out_type(self):
        self.assertIsInstance(AS312_instance.read_state,bool)
if __name__ == '__main__':
    unittest.main()
```

B.1.3 DFR0026

Listing B.3: Code for unit test of DFR0026

```
#!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/
   lib/python3.11
import unittest
from DFR0026 import DFR0026
class test_project_code(unittest.TestCase):
    def test_DFR0026_out_type(self):
        self.assertIsInstance(DFR0026().read_voltage(),float)
    def test_DFR0026_out_range(self):
        self.assertLessEqual(DFR0026().read_voltage(),5)
        self.assertGreaterEqual(DFR0026().read_voltage(),0)
if __name__ == '__main__':
    unittest.main()
```

B.1.4 Memory Management

Listing B.4: Code for unit test of memory module

```
#!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/
   lib/python3.11
import unittest
from memory_mangment import sensor_data,Memory_tester
memorytest_obj=Memory_tester()
class test_project_code(unittest.TestCase):
    def Test_memory_silicon_power_32GB(self):
        self.assertLessEqual(memorytest_obj.check_memory,32e9
           )
        self.assertGreaterEqual(memorytest_obj.check_memory
           ,0)

if __name__ == '__main__':
    unittest.main()
```

B.1.5 Radio Module

Listing B.5: unit test code for Radio module

```python
#!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/lib/python3.11
import unittest
from Radiomodule import Transciever
Transciever_instance=Transciever()
class test_project_code(unittest.TestCase):
    def test_serial_connection(self):
        self.assertIsInstance(Transciever_instance.transceive_ser,serial.Serial)
    def test_serial_interrupt(self):
        self.assertEqual(Transciever_instance.event.is_set(),(False,True))
    def test_transceiver_test_message(self):
        message=Transciever_instance.message
        Transciever_instance.transceive_test_message(True)
        received_message=Transciever_instance.transceive_test_message(False)
        self.assertEqual(message,received_message)
    def test_transceiver_test_txt_file(self):
        txt_fname=Transciever_instance.txt_fname
        with open(txt_file,'r') as f:
            expected_txt=f.read()
        Transciever_instance.transceive_test_txt_file(True)
        received_txt_file=Transciever_instance.transceive_test_txt_file(False)
        self.assertEqual(expected_txt,received_txt_file)
    def test_transciver_test_csv(self):
        csv_fname=Transciever_instance.csv_fname
        expected_df=pd.read_csv(csv_fname)
        Transciever_instance.transceive_test_csv(True)
        reviced_df=Transciever_instance.transceive_test_csv(False)
        self.assertEqual(expected_df,reviced_df)
    def test_trancsive_img_file(self):
        img_fname=Transciever_instance.png_fname
        with open(img_fname,'rb')as f:
            expted_out=f.read()
        Transciever_instance.Transcevie_png_file(True)
        received_bin=Transciever_instance.Transcevie_png_file(False)
        self.assertEqual(expted_out,received_bin)
if __name__ == '__main__':
    unittest.main()
```

Appendix C

Bash scripts

C.1 Bashscripts

in this section we will have the following bash files:

1. Camerea
2. main
3. memorytest
4. radiomodule

C.1.1 Camerea

Listing C.1: Code for triggering the camerea

```
#!/bin/bash
timestamp=$(date +"%Y-%m-%d_%H_%M_%S")
fname="camera_output_$timestamp.png"
output_dir="Images_camera"
if [ ! -d "$output_dir" ]; then
# Create the directory if it doesn't exist
mkdir -p "$output_dir"
fi
rpicam-still --raw -o "$output_dir/$fname"
```

C.1.2 Main

Listing C.2: Code for runing the main function

```
#!/bin/bash
is_root() {
if [[ $EUID -ne 0 ]]; then
    echo "This script requires root privileges. Please run with sudo."
    exit 1
fi
}
if [[ $1 -eq 0 ]]; then
    echo "Error no arugments provided"
    echo -e "enter what is transmited:\n\r1:hello world \n\r2:text file \n\r3:csv file\n\r4:PNG\n\r"
    exit 1
fi
# Call the is_root function to verify permissions
is_root
sudo chmod g+rw /dev/ttyS0
#get current time
current_time=$(date +%H:%M)
current_hour=$(echo $current_time | cut -d: -f 1)
previous_hour=$((current_hour-1))
while [ $current_time != "12:00" ]&&[ $current_time != "9:00" ]; do
if [ $current_time == "$current_hour:00" ]; then
    # python Documents/Github/meshnetwork_in_forest/main/main.py $1
    python main/main.py $1
    echo "file ran successfully"
fi
break #because everyone needs a break sometime
done
```

C.1.3 Radio Module

Listing C.3: Code for the testing serial of the radio module

```
#!/bin/bash
#only use this  for  transcive module
# Function to check if the script is run with root privileges
is_root() {
if [[ $EUID -ne 0 ]]; then
    echo "This script requires root privileges. Please run with sudo."
    exit 1
fi
}
# Call the is_root function to verify permissions
is_root
# Set appropriate permissions for /dev/ttyS0 (consider group or user access)

sudo chmod g+rw /dev/ttyS0

if [[ "$1" == "1" ]]; then
python test_tranmiter.py
elif [[ "$1" == "0" ]]; then
python test_reciver.py
else
echo "Invalid argument. Please provide 1 (transmitter) or 0 (receiver)."
exit 1
fi
```