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
Liam Hogan

Supervisor
Philip Creevy

South East Technological University
Department Of Engineering Technology
School of Engineering
Ireland.

May 20, 2024

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 reference to learn about the technology and why it is used

Chapter 1

Introduction

1.1 Motivation

The motive 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 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 of issues associated with this communication such as signal loss due to:

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

In this project aims 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 which 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 **?** "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:

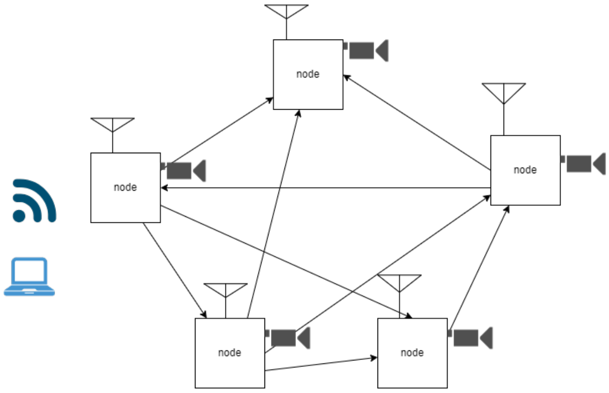


Figure 2.1: Basic block diagram of a mesh network

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

2.2 Hardware Consideration

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 heavy weight 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 suggestions. 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 to be split up into different parts so that all that is needed is the libraries used and code that won't all have to compiled in one file.

DHT22

In this section we have to consider the following:

1. The GPIO port as on page ?? 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 on to device
2. Then change dirertorys 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):
humidityy,temp=dht.read_retry(dht.DHT22, Gpoiport)
    sleep(5)
    return humundity,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 gpioport 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 intall 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. onece it sences a pulse it will tiggerr the camerae

DFR0026

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

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 selcect a channel and device , port and print the vaules 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 fron 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 with an make a 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

```python
import pandas as pd
import numpy as np
from datetime import datetime
cols_name=["Timestamp","Tempeature","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

```python
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 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 home directory this will cal the space if 13K the ”— cut -f1” will only foucs on the size string messeage and then print out the size. this is just a sample script

2.3.8 Test Driven development

In this project ill will be useing Test Driven Development (TDD) is a software development approach where tests are written before the actual code the following is 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 behaviour 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 **?**

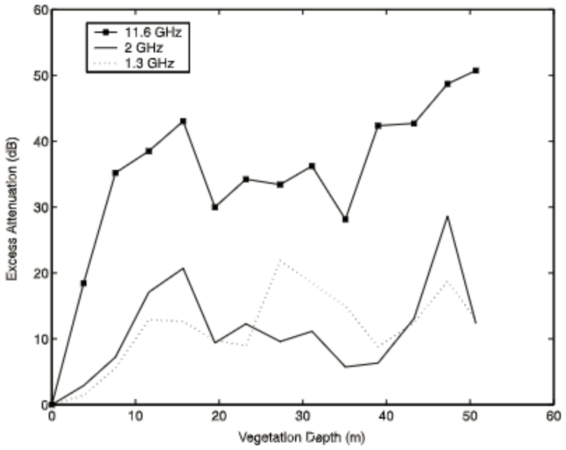


Figure 2.2: Silver Maple in-leaf excess attenuation for the line of trees geometry (receiver antenna height: 3.5 m, SAVAGE ET AL.pg.7

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 **?** recommendations for Attenuation in wooded areas

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

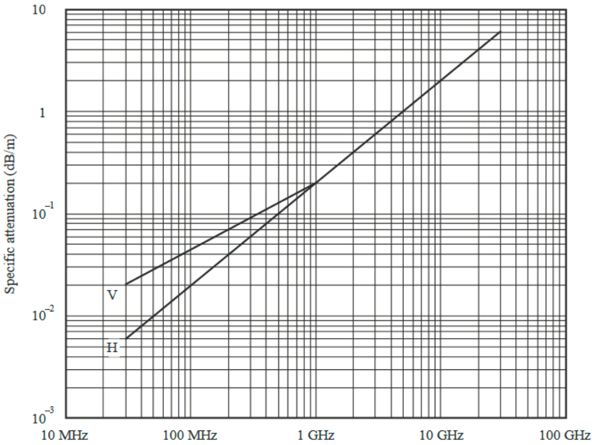


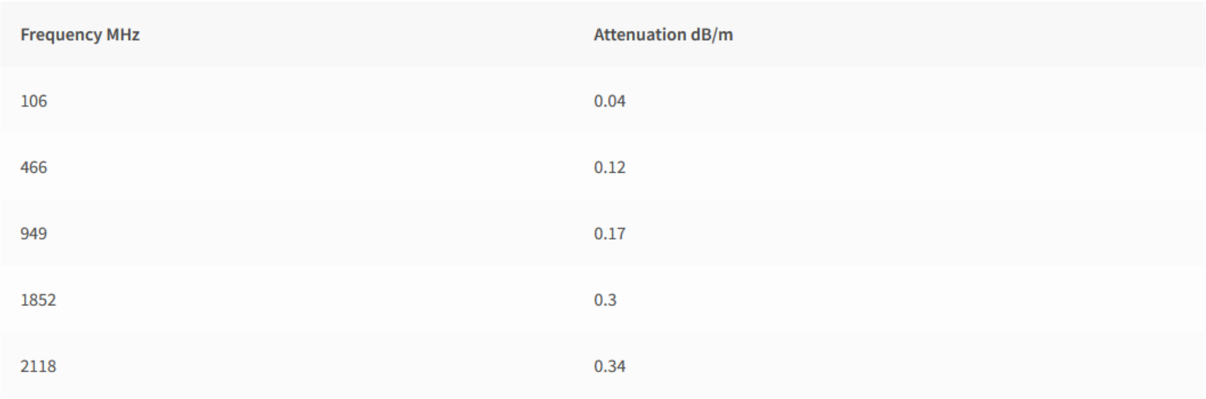
Figure 2.3: Specific attenuation due to woodland (Recommendation ITU-R P.833-7 (02/2012) Attenuation in vegetation pg.5

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?



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

Figure 2.4: 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 ?

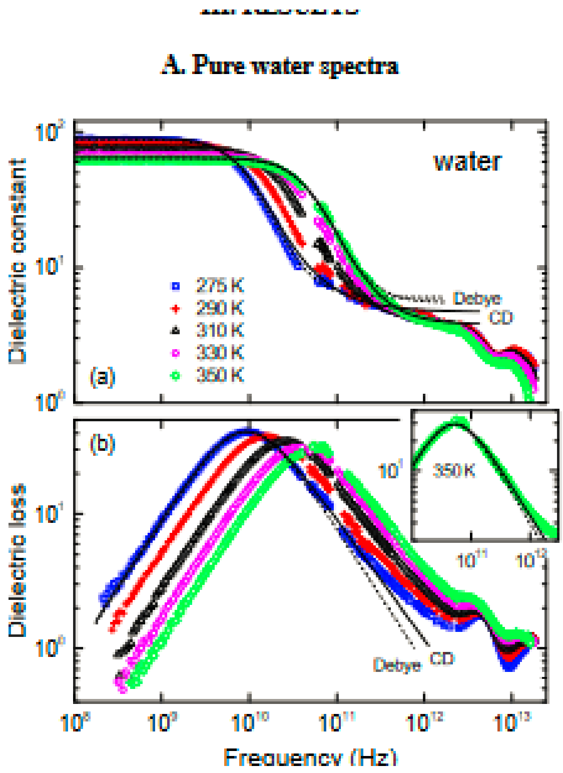


Figure 2.5: absorption of water

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)**?** "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**?**, "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 2.1: 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)(**?**, 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 **?** 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

   ? 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

   ? 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 is the steps of this project:

1. Consider the environment in which we commutate across
2. Determine the desired range for sensor to operate in.
3. Find the 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 setup the software
8. State the software needed to drive these sensor
9. Write unit test to develop the software
10. Dicuss how to track the sensor data
11. Dicuss the limitations
12. Dicuss the timeline at which the project occurred

3.3 Setup of raspberry pi

Firstly once you have your pi heres a quick guide to setup the pi are the following:

1. Unpack the pi be sure to connect keyboard mouse and hdmi cable
2. Download the raspberry pi imager and select the 32 bit recommend os
3. Put the micro_SD card into the pi once the pi is setup you can make sub directories for this project type the following:

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

   This will download the essentially environment for setting up the Pi initial this will have to 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 virtual environment

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

3.4 Additional Research

In 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 choosen 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 chanel's
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 received 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?

Lets 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 of 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 open source project that has sorted out the routeing for you. only issue with this approach is if there is any issues that come from the open source project we will inherit the bugs with this in mind the following standards were found

1. LoRa

LoRa

In **?** lora is used that will organize sensor data from all nodes in the spanning tree toward the root(laptop /PC) this can be show by the following: this proves it possible to make a mesh network using Lora.

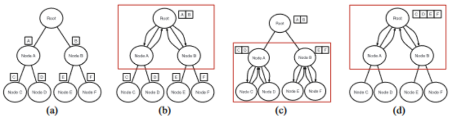


Fig. 3. Recursive data collection. (a) Each node (other than the root) has a data item indicated by a square. (b) Root requests data from nodes A and B. (c) Nodes A and B request data from nodes C–F. (d) In the next round of requests, the root collects the data items of nodes C–F from nodes A and B.

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

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 **?** 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 communciated 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 the how the signal is spread out.

To simulate the system we can use the following FIR response as an example in a given medium of transmit each bandwidth is the length the of the sinc-roll-off which degrade depends on the impulse response in this given bandwidth channels are separated in the same fashion .

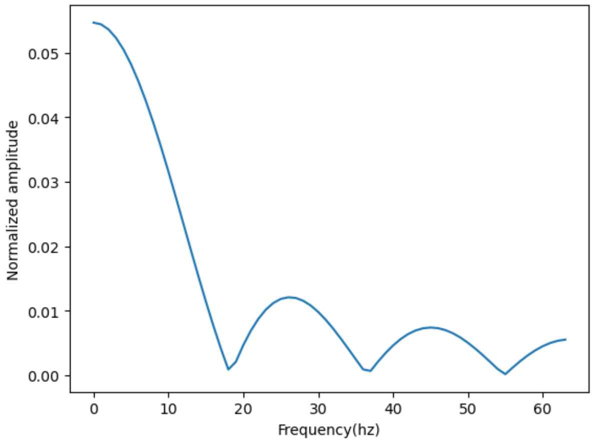


Figure 3.2: sample graph of a FIR response

3.4.3 What is the difference between a port and a channel

In **?** "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 where due to 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

Which has a E22-900T22S on the board which 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 distance ,the module has two was of interacting with the board:

1. Pi
2. USB to windows desktop in this configuration we can test single node from our laptop this is just to test sending message 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 discuss above section when a single is sent over long distance it gets weaker, walls floors and other objects can block or distort this signal , A repeater reive these signal's and simply amplified and retransmits the data,Multi-level have secondary repeaters 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 **??**

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 the our class

   ```
       class DHT22:
       ##Set DATA pin to pin 4
           def __init__(self):
               """this will setup 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 define our DhT device as 11 seen as the DHT22 was broken so we set our gpio pin 4 and setup 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 setup 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 out 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 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$_ADS1115 thefollowingisconsidered$:

Import the libraries:

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

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.

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. install 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 instan  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 takeing a picture

```
        def take_pic(self)-> str:
            """this will take  a picture from 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 it 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 its useing 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 run from another python file

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

3.5.4 Radio module

This section is based off the github repository: `https://github.com/sbcshop/Lora-HAT-for-Raspberry` here are the following approach 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.transcive$_s$*er”setourserialportupwhichinlinuxis′/dev/ttyS*0*′wecancontroltimeouttobe*

3. Setup a 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 calculated 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 /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()
```

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 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. have file as a module

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

The following is the process of devlop:

- To make test that will be there for the codeing 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 **?** seen as the data is 8 bits and the range at which this operates at -40 to 80^{o}c for tempeature meaning we have at least 7 bit 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 mean we have a 2 bits dedicated to decimal place so the high temperature to be 80.3^{o}c for the lowest temp we have 6 bits to represent - 40 due to 2s complement so lowest will be -40.3^oC so with that that stablish 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 steps for folder setup follow 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 unittest . the from DHT22 is a python files we can install functions from other python files this can be useful for testing purposes then we initialized a test class call 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 data sheet online for . This code is simply testing the limits of the DHT22 for humidity the Data sheet which 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 lead 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 a humidity and temp values we set the output to a tuple to test for this we use isInstacne which will test if its a tuple next we test for the limits of the humidity

DFR0026

According to the data sheet **?** 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 intger in it.
2. Test the range of this with the upper limit being 5v the analogue voltage meaning:
$$\text{output} = \frac{2^n \cdot \text{Anal}}{\text{Refe}} = \frac{2^{16} \cdot 5}{5} =$$
3. test the lover limit being 3.3v as the analogue$\frac{2^{16}3.3}{5} = 43253$

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 data sheet **?** The resolution to it uses is 1080p50 which is 1920x1080p so our tests will have to in corporate 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 check 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 **??**) 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 a 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

the first iterations as see here has the following problems for the sensors:

1. Time stamp for DHT22 wasn't in a string format
2. Forget to look for but a float and int in the DHT22.read function
3.

3.6 Data Analysis Methods

In this section we will Discuss the tool we use to analyze the sensor data the following will be dicussed:

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 will the output of this API

3.6.2 Matplotlib

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

3.7 Timeline

The research phase will be conducted from 20/09/2023 to May 20, 2024.The prosed impanation phase is from january till May 20, 2024

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 transceiver
3. Recorded data from testing the mesh network

4.1 Recorded data from sensors

in 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 height

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 see the following message

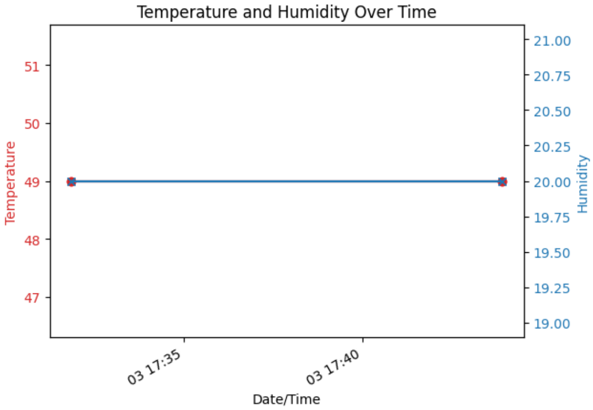
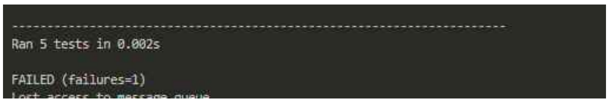


Figure 4.1: Temperature and Humidity plotted overtime



```
----------------------------------------------------------------------
Ran 5 tests in 0.002s

FAILED (failures=1)
Lost access to message queue
```

Figure 4.2: unit test message for DHT22 module

4.1.2 AS312

date/time of record	motion detected(yes/no)
2024-03-25$_{1}5 - 02 - 57$	False
2024-03-25$_{1}5 - 04 - 37$	True
2024-05-03$_{1}8 - 07 - 51$	False
2024-05-03$_{1}8 - 18 - 37$	True height

Table 4.2: Recorded data from AS312 on the May 20, 2024

4.1.3 DFR0026

For 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$_{1}5 - 02 - 57$	940
2024-03-25$_{1}5 - 03 - 13$	945
2024-03-25$_{1}5 - 04 - 37$	4963
2024-05-03$_{1}8 - 54 - 57$	1284 height

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

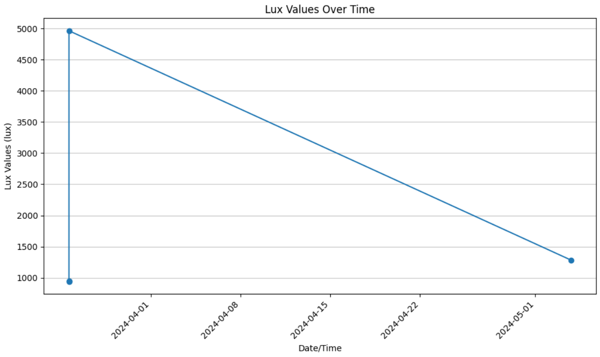


Figure 4.3: lux values overtime

4.1.4 Raspberry Pi VR 220

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

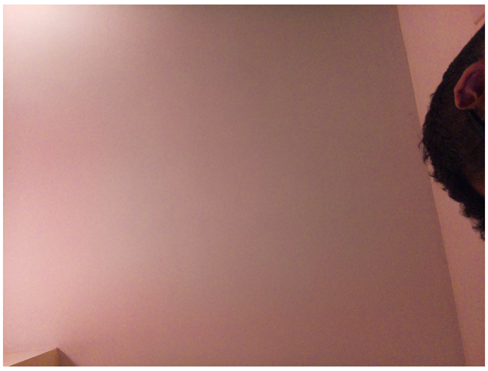


Figure 4.4: A photo from 25th of march 2024

4.2 Recorded data from transceiver

When testing the radio module the follow 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 a image file across the serial

   none results where obtained due to factor on **??**

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 gather from sensor will be discussed Note:(**all test here where conducted indoors**):

1. DHT22 (Temperature & Humidity):

   On page **??** compared to the average room temperature which is 20^{o}c ,The range of humidity in a room is from 30% to 60% in the plot on page **??** note in the file sensor$_{d}ata.csvthereentriesthatare0, thisisduetotestingdifferentcomponentsandmakethesevalues$

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

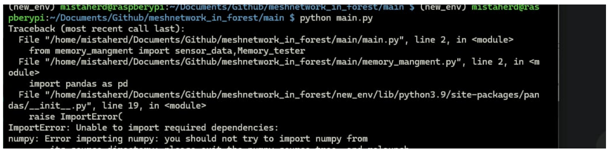
3. DFR0026 (Lux): As seen on page **??** is a table full of lux values according to this link we ideally want a lux value of 800 to 1700 lux which satisfies these conditions.

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 repository for this section the Pi's sd cards where corrupted due to a fault binary file which was later fixed but this error lead to the following error message:



```
(new_env) mistaherd@raspberypi:~/Documents/Github/meshnetwork_in_forest/main $ (new_env) mistaherd@ras
pberypi:~/Documents/Github/meshnetwork_in_forest/main $ python main.py
Traceback (most recent call last):
  File "/home/mistaherd/Documents/Github/meshnetwork_in_forest/main/main.py", line 2, in <module>
    from memory_mangment import sensor_data,Memory_tester
  File "/home/mistaherd/Documents/Github/meshnetwork_in_forest/main/memory_mangment.py", line 2, in <m
odule>
    import pandas as pd
  File "/home/mistaherd/Documents/Github/meshnetwork_in_forest/new_env/lib/python3.9/site-packages/pan
das/__init__.py", line 19, in <module>
    raise ImportError(
ImportError: Unable to import required dependencies:
numpy: Error importing numpy: you should not try to import numpy from
```

Figure 5.1: environment error message
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 [briefly summarize your methods], we have successfully demonstrated the [key results or achievements]. Our findings underscore the potential for mesh networks to revolutionize [specific areas or applications within forest environments].

6.1.1 Key Contributions

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

- **Developinga 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 image 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:** this we can schedule the sending and receiveing of data we can use this to extract the sensor data

6.2 Future Work

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

- **Sockets:** The project could of ventrured 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 was of connecting our sensor

6.2.1 Final Remarks

The deployment of mesh networks in forest environments holds immense promise. The work presented here serves as a solid foundation for future endeavors.into the actual routeing of the network and testing this

Appendix A

Appendix

Appendix B

Python Scripts

B.1 Python Scripts

B.1.1 DHT22

Listing B.1: DHT22code

```python
#!/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")
```

B.1.2 AS312

Listing B.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()
```

B.1.3 DFR0026

Listing B.3: Code for DFR00026

```python
#!/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())
```

B.1.4 Camera

Listing B.4: Code for Camera

```python
#!/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 B.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()
```

B.1.5 Memory management

Listing B.6: Code for memory mangement

```python
#!/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")
```

B.1.6 Radio module

Listing B.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
    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 C

TDD Script

C.1 TDD scripts

This section is for All the TDD section of this report in this section will be shareing 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

C.1.1 DHT22

Listing C.1: DHT22 unit test

```python
#!/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()
```

C.1.2 AS312

Listing C.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()
```

C.1.3 DFR0026

Listing C.3: Code for unit test of DFR0026

```python
#!/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()
```

C.1.4 Memory Management

Listing C.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()
```

C.1.5 Radio Module

Listing C.5: unit test code for Radio module

```
#!/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 D

Bash scripts

D.1 Bashscripts

in this section we will have the following bash files:

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

D.1.1 Camerea

Listing D.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"
```

D.1.2 Main

Listing D.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
```

D.1.3 Radio Module

Listing D.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
```