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.
April 16, 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

Chapter 1

Methodology

1.1 Introduction

In this Section i will discuss the proposed methodology of this project this will cover the following:

1. The setup of the raspberry pi
2. The Data Collection Methods
3. The Model Development
4. The Data Analysis Methods
5. The Ethical Considerations
6. The validity and reliability
7. The Limitations and Delimitation
8. The timeline

1.2 Setup of raspberry pi

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

1. once you unpack the pi be sure to connect keyboard mouse and hdmi cable
2. next on a computer you must download the raspberry pi imager and selet the 64 bit recommned os
3. once u have os set simpley put the mircosd card into the pi once the pi is setup you can make sub dirrys for this project type the following:

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

   this will downlaod the nessary enirоment for setinng up the pi intiall this will have to built out through the process of the project look at the timeline Section
4. next simply follow the ReadME.md file to understand how to setup the py

1.3 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

1.3.1 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

1.3.2 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

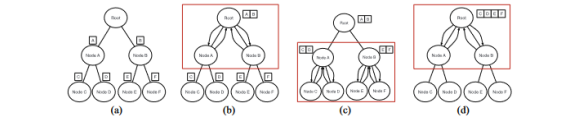

Fig. 3. Recursive data collection. (a) Each node (other than the root) has a data item indicated by a square. (b) Root requests data from nodes A and B. (c) Nodes A and B request data from nodes C–F. (d) In the next round of requests, the root collects the data items of nodes C–F from nodes A and B.

Figure 1.1: protocol Wu used(wu_lie et.al,2023:16705)

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.

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

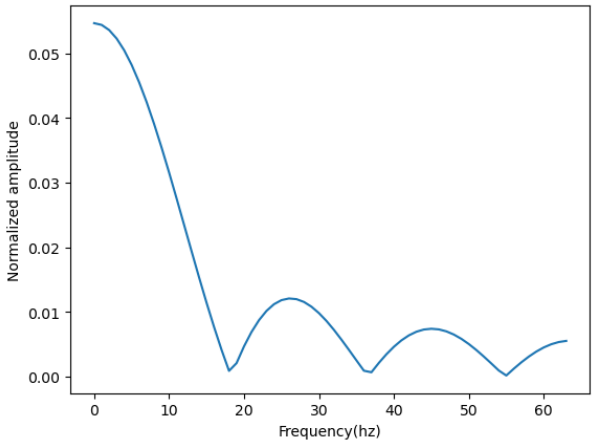

Figure 1.2: sample graph of a FIR response

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 .

1.3.3 What is the difference between a port and a channel

Why the MM2 Series 900 MHz wasnt 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 1.1: Comparing New Radio modules

SB Components LoRa HAT for Raspberry Pi was picked which has the according to its datasheet to install it onto the pi the designer has to

1.4 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

1.4.1 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:

```
1 #!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/lib/
    python3.11
2 import adafruit_dht
3 import board
4 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

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
```

1. 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

2. 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$_{A}DS1115we do the following$:
import the libraries

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

Next we 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 set the associative gain

Next read the analogue channel

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

1.4.2 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
```

1.4.3 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()
```

1.4.4 TDD

Fristly i want to made some unit tests the aim of this is the following:

- 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 MM2 Series 900 MHz
5. 1 x MCP3008
6. 1 x Raspberry Pi VR 220 Camera
7. 1 x Li-polymer Battery HAT
8. 1 x Turbo 1GB

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 1.1: 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 import unitest . the from protest is a python files we can install functions from other python files this can be usefull for testing purposes then we initalized a test class call Unittest.testcase our firstion fucntion of the class we check if the number of the output is a float or not this is for testing tempearture the next function we test for is the range i look at the datasheet online this code is simpley testing the limits of the DHT22 for humidity the Datasheet which ranges from 0 to 100 % we want to test for the following:

1. Test if the output is a float
2. Test if the output ranges 0 to 100

this lead to the following code

Listing 1.2: sample test for DHT22

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

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

    def test_DHT22_hum_range(self):
        self.assertGreaterEqual(hum,0.0)
        self.assertLessEqual(hum,100.0)
```

seen as we expect our sensor to print out a humdity 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 & MCP3008

According to the datasheet **?** we must keep in mind that this componet is connected to an ADC this will give me the following test conditions:

1. Test if the output is a float
2. Test the range of this with the upper limit being 5v
3. test the lover limit being 0

Listing 1.3: unit test for DFR0026 and MCP3008

```
    import unittest
    from protest import Read_DHT22,Read_MCP3008
    class test_project_code(unittest.TestCase):
    def test_DFR0026_MCP3008_out_type(self):
        self.assertIsInstance(Read_MCP3008,float)
    def test_DFR0026_MCP3008_out_range(self):
        self.assertLessEqual(5.0000000)
        self.assertGreaterEqual(0.0000000)
```

this code is in the same in theres of limits

AS312

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

1. test for type is boolean

we can now add to the snipppet :

Listing 1.4: unit test for AS312

```
    def test_AS312_out_type(self):
        self.assertIsInstance(Read_AS312,bool)
```

Note : Don't forget to import read$_a s12 function from test file seen as thhis is a motion sensor our ouout$

Raspberry Pi VR 220 Camera

according to the data sheet **?** we the resoultion to it uses is 1080p50 which is 1920x1080p so our tests will have to in copoarte the followoing:

1. Test the output shape if open cv is gonna be used
   (a) test the amout of elelecelm in the 3 dimesional array
2. test the file type is png

this would lead me to the following code snippet.

Listing 1.5: camera unit test

```
    def test_Raspberry_Pi_VR220_out_shape(self):
    self.assertEqual(Read_Raspberry_PiVR220.shape
       ,(1920,1080,3))
```

this function check the pixeal count or resoulkation

Li-polymer Battery HAT

memory moduldes

in this setion will dicuss the following:

1. silicon power 32GB
2. Turbo 1GB

for this i will use useing a bash script(see this on page **??**) and what we are doing is testing the size in a certain range for the silicon SD card

1. Turbo 1GB as from above we are import the file at which where our functions live in code frist we import the function

Listing 1.6: si powerd SD snippnet

```
1          import unittest
2          from protest import Read_DHT22,Read_MCP3008,
               Read_AS312,Read_Raspberry_PiVR220,
               Read_Memory_module
3
4          def Test_memory_module_turbo_1GB_size(self):
5              #testing  turbo 1GB
6              self.assertLessEqual(Read_Memory_module,1e9)
7              self.assertGreaterEqual(Read_Memory_module,0)
```

then simply we call assert and greater than which sets the bounds of the modes the 1e9 is a way to put 110^9 whcih output that will between 1GB and 0

2. silicon power 32GB

MM2 Series 900 MHz

Unit test iterations

the frist iteatarations as see here has the following problems for the sensors:

1. time stamp for DHT22 wasnt in a string format
2. forget to look for but a float and int in the DHT22.read fucntion

conculsion

The intiall draft code for the test devlopemnt si the following on page

1.5 Data Analysis Methods

Statistical and machine learning techniques are employed to analyze the data collected from both computational models and real-world sources. These techniques are used to identify patterns, trends, and relationships within the data.

1.6 Ethical Considerations

The use of computational methods raises ethical concerns regarding data privacy and security. To address these concerns, data anonymization and encryption techniques are employed to protect sensitive information. Additionally, informed consent is obtained from participants when applicable.

1.7 Validity and Reliability

Validation of computational models is achieved through rigorous testing and evaluation. This involves comparing model predictions with real-world data and examining the sensitivity of the models to different parameters. Reliability is ensured through the use of standardized methods and procedures for data collection, analysis, and interpretation.

1.8 Limitations and Delimitations

The computational nature of the research introduces limitations due to the complexity of the systems being modeled and the potential for errors in modeling and data analysis. Moreover, the generalizability of the findings may be limited to the specific contexts and conditions considered in the research.

1.9 Timeline

The model development phase of the research is scheduled to take place from [start date] to [end date]. The data collection and analysis phases are scheduled to take place from [start date] to [end date]. The final write-up of the research is scheduled to be completed by [deadline date].

Chapter 2

Results

In thes 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

2.1 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**

2.1.1 DHT22

Results during protypeing

date/time of record	Temperature	Humidity
2024-02-21 00:03:56	22	66

Table 2.1: Recorded data from DHT22 on the April 16, 2024

last we tested if our code satisfies our python code after testing the unit test code we upadated see the foolwing message

Figure 2.1: unit test message for DHT22 module

2.1.2 AS312

Results during protype

date/time of record	motion detected(yes/no)
2024-03-25$_1$5 − 02 − 57	False
2024-03-25$_1$5 − 04 − 37	True

Table 2.2: Recorded data from AS312 on the April 16, 2024

2.1.3 DFR0026

Results during protypes

for our first test we got the following table

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

Table 2.3: Recorded data from DFR0026 on the 25th of march 2024

2.1.4 Raspberry Pi VR 220

When testing the Raspberry Pi VR 220

Results during portotypeing

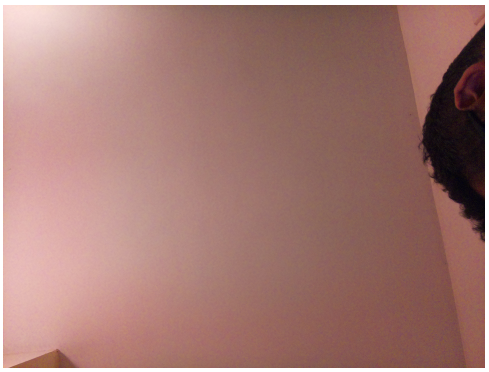

Figure 2.2: A photo from 25th of march 2024

2.2 Recorded data from transceiver

2.3 Recorded data from mesh network

Bibliography

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.

F. . 2003 and A. C. Information, "An introduction to spread-spectrum communications," Jan 2023.

sparkfun.

ada.

Chapter 3

Appendix A

Appendix A

Python Scripts

A.0.1 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.")
```

```
29 dht_sensor = DHT22()
30 dht_sensor.write_to_csv("sensor_data.csv")
```

A.0.2 AS312

Listing A.2: code for AS312

```
#!/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.0.3 ADC

Listing A.3: Code for ADC

```
    #!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/lib/python3.11
'''!
  @file DFRobot_ADS1115.py
  @brief Provides an Raspberry pi library to read ADS1115 data over I2C. Use this library to read analog voltage values.
  @copyright   Copyright (c) 2010 DFRobot Co.Ltd (http://www.dfrobot.com)
  @license     The MIT License (MIT)
  @author [luoyufeng](yufeng.luo@dfrobot.com)
  @version  V1.0
  @date  2019-06-19
  @url https://github.com/DFRobot/DFRobot_ADS1115
'''

import smbus
import time

## Get I2C bus
bus = smbus.SMBus(1)

## I2C address of the device
ADS1115_IIC_ADDRESS0                                = 0x48
ADS1115_IIC_ADDRESS1                                = 0x49

## ADS1115 Register Map
## Conversion register
ADS1115_REG_POINTER_CONVERT                         = 0x00
## Configuration register
ADS1115_REG_POINTER_CONFIG                          = 0x01
## Lo_thresh register
ADS1115_REG_POINTER_LOWTHRESH           = 0x02
## Hi_thresh register
ADS1115_REG_POINTER_HITHRESH            = 0x03

## ADS1115 Configuration Register
## No effect
ADS1115_REG_CONFIG_OS_NOEFFECT          = 0x00
## Begin a single conversion
ADS1115_REG_CONFIG_OS_SINGLE            = 0x80
## Differential P = AIN0, N = AIN1 (default)
ADS1115_REG_CONFIG_MUX_DIFF_0_1         = 0x00
## Differential P = AIN0, N = AIN3
ADS1115_REG_CONFIG_MUX_DIFF_0_3         = 0x10
## Differential P = AIN1, N = AIN3
ADS1115_REG_CONFIG_MUX_DIFF_1_3         = 0x20
## Differential P = AIN2, N = AIN3
```

```
45 ADS1115_REG_CONFIG_MUX_DIFF_2_3         = 0x30
46 ## Single-ended P = AIN0, N = GND
47 ADS1115_REG_CONFIG_MUX_SINGLE_0         = 0x40
48 ## Single-ended P = AIN1, N = GND
49 ADS1115_REG_CONFIG_MUX_SINGLE_1         = 0x50
50 ## Single-ended P = AIN2, N = GND
51 ADS1115_REG_CONFIG_MUX_SINGLE_2         = 0x60
52 ## Single-ended P = AIN3, N = GND
53 ADS1115_REG_CONFIG_MUX_SINGLE_3         = 0x70
54 ## +/-6.144V range = Gain 2/3
55 ADS1115_REG_CONFIG_PGA_6_144V           = 0x00
56 ## +/-4.096V range = Gain 1
57 ADS1115_REG_CONFIG_PGA_4_096V           = 0x02
58 ## +/-2.048V range = Gain 2 (default)
59 ADS1115_REG_CONFIG_PGA_2_048V           = 0x04
60 ## +/-1.024V range = Gain 4
61 ADS1115_REG_CONFIG_PGA_1_024V           = 0x06
62 ## +/-0.512V range = Gain 8
63 ADS1115_REG_CONFIG_PGA_0_512V           = 0x08
64 ## +/-0.256V range = Gain 16
65 ADS1115_REG_CONFIG_PGA_0_256V           = 0x0A
66 ## Continuous conversion mode
67 ADS1115_REG_CONFIG_MODE_CONTIN          = 0x00
68 ## Power-down single-shot mode (default)
69 ADS1115_REG_CONFIG_MODE_SINGLE          = 0x01
70 ## 8 samples per second
71 ADS1115_REG_CONFIG_DR_8SPS                      = 0x00
72 ## 16 samples per second
73 ADS1115_REG_CONFIG_DR_16SPS                     = 0x20
74 ## 32 samples per second
75 ADS1115_REG_CONFIG_DR_32SPS                     = 0x40
76 ## 64 samples per second
77 ADS1115_REG_CONFIG_DR_64SPS                     = 0x60
78 ## 128 samples per second (default)
79 ADS1115_REG_CONFIG_DR_128SPS            = 0x80
80 ## 250 samples per second
81 ADS1115_REG_CONFIG_DR_250SPS            = 0xA0
82 ## 475 samples per second
83 ADS1115_REG_CONFIG_DR_475SPS            = 0xC0
84 ## 860 samples per second
85 ADS1115_REG_CONFIG_DR_860SPS            = 0xE0
86 ## Traditional comparator with hysteresis (default)
87 ADS1115_REG_CONFIG_CMODE_TRAD           = 0x00
88 ## Window comparator
89 ADS1115_REG_CONFIG_CMODE_WINDOW         = 0x10
90 ## ALERT/RDY pin is low when active (default)
91 ADS1115_REG_CONFIG_CPOL_ACTVLOW         = 0x00
92 ## ALERT/RDY pin is high when active
93 ADS1115_REG_CONFIG_CPOL_ACTVHI          = 0x08
94 ## Non-latching comparator (default)
95 ADS1115_REG_CONFIG_CLAT_NONLAT          = 0x00
```

```
96  ## Latching comparator
97  ADS1115_REG_CONFIG_CLAT_LATCH           = 0x04
98  ## Assert ALERT/RDY after one conversions
99  ADS1115_REG_CONFIG_CQUE_1CONV           = 0x00
100 ## Assert ALERT/RDY after two conversions
101 ADS1115_REG_CONFIG_CQUE_2CONV           = 0x01
102 ## Assert ALERT/RDY after four conversions
103 ADS1115_REG_CONFIG_CQUE_4CONV           = 0x02
104 ## Disable the comparator and put ALERT/RDY in high state (
    default)
105 ADS1115_REG_CONFIG_CQUE_NONE           = 0x03
106
107 mygain=0x02
108 coefficient=0.125
109 addr_G=ADS1115_IIC_ADDRESS0
110 class ADS1115():
111         def set_gain(self,gain):
112                 '''!
113                   @brief Sets the gain and input voltage
                          range.
114                   @param gain  This configures the
                          programmable gain amplifier
115                   @n ADS1115_REG_CONFIG_PGA_6_144V     = 0x00
                           # 6.144V range = Gain 2/3
116                   @n ADS1115_REG_CONFIG_PGA_4_096V     = 0x02
                           # 4.096V range = Gain 1
117                   @n ADS1115_REG_CONFIG_PGA_2_048V     = 0x04
                           # 2.048V range = Gain 2
118                   @n default:
119                   @n ADS1115_REG_CONFIG_PGA_1_024V     = 0x06
                           # 1.024V range = Gain 4
120                   @n ADS1115_REG_CONFIG_PGA_0_512V     = 0x08
                           # 0.512V range = Gain 8
121                   @n ADS1115_REG_CONFIG_PGA_0_256V     = 0x0A
                           # 0.256V range = Gain 16
122                 '''
123                 global mygain
124                 global coefficient
125                 mygain=gain
126                 if mygain == ADS1115_REG_CONFIG_PGA_6_144V:
127                         coefficient = 0.1875
128                 elif mygain == ADS1115_REG_CONFIG_PGA_4_096V:
129                         coefficient = 0.125
130                 elif mygain == ADS1115_REG_CONFIG_PGA_2_048V:
131                         coefficient = 0.0625
132                 elif mygain == ADS1115_REG_CONFIG_PGA_1_024V:
133                         coefficient = 0.03125
134                 elif mygain == ADS1115_REG_CONFIG_PGA_0_512V:
135                         coefficient = 0.015625
136                 elif  mygain == ADS1115_REG_CONFIG_PGA_0_256V
                        :
```

```
                        coefficient = 0.0078125
                else:
                        coefficient = 0.125
        def set_addr_ADS1115(self,addr):
                '''!
                  @brief Sets the IIC address.
                  @param addr  7 bits I2C address, the range
                     is 1~127.
                '''
                global addr_G
                addr_G=addr
        def set_channel(self,channel):
                '''!
                  @brief Select the Channel user want to use
                     from 0-3.
                  @param channel  the Channel: 0-3
                  @n For Single-ended Output:
                  @n     0 : AINP = AIN0 and AINN = GND
                  @n     1 : AINP = AIN1 and AINN = GND
                  @n     2 : AINP = AIN2 and AINN = GND
                  @n     3 : AINP = AIN3 and AINN = GND
                  @n For Differential Output:
                  @n     0 : AINP = AIN0 and AINN = AIN1
                  @n     1 : AINP = AIN0 and AINN = AIN3
                  @n     2 : AINP = AIN1 and AINN = AIN3
                  @n     3 : AINP = AIN2 and AINN = AIN3
                  @return channel
                '''
                global mygain
                self.channel = channel
                while self.channel > 3 :
                        self.channel = 0

                return self.channel

        def set_single(self):
                '''!
                  @brief Configuration using a single read.
                '''
                global addr_G
                if self.channel == 0:
                        CONFIG_REG = [
                           ADS1115_REG_CONFIG_OS_SINGLE |
                           ADS1115_REG_CONFIG_MUX_SINGLE_0 |
                           mygain |
                           ADS1115_REG_CONFIG_MODE_CONTIN,
                           ADS1115_REG_CONFIG_DR_128SPS |
                           ADS1115_REG_CONFIG_CQUE_NONE]
                elif self.channel == 1:
                        CONFIG_REG = [
                           ADS1115_REG_CONFIG_OS_SINGLE |
```

```
                                ADS1115_REG_CONFIG_MUX_SINGLE_1 |
                                mygain |
                                ADS1115_REG_CONFIG_MODE_CONTIN,
                                ADS1115_REG_CONFIG_DR_128SPS |
                                ADS1115_REG_CONFIG_CQUE_NONE]
179             elif self.channel == 2:
180                     CONFIG_REG = [
                                ADS1115_REG_CONFIG_OS_SINGLE |
                                ADS1115_REG_CONFIG_MUX_SINGLE_2 |
                                mygain |
                                ADS1115_REG_CONFIG_MODE_CONTIN,
                                ADS1115_REG_CONFIG_DR_128SPS |
                                ADS1115_REG_CONFIG_CQUE_NONE]
181             elif self.channel == 3:
182                     CONFIG_REG = [
                                ADS1115_REG_CONFIG_OS_SINGLE |
                                ADS1115_REG_CONFIG_MUX_SINGLE_3 |
                                mygain |
                                ADS1115_REG_CONFIG_MODE_CONTIN,
                                ADS1115_REG_CONFIG_DR_128SPS |
                                ADS1115_REG_CONFIG_CQUE_NONE]
183
184             bus.write_i2c_block_data(addr_G,
                   ADS1115_REG_POINTER_CONFIG, CONFIG_REG)
185
186     def set_differential(self):
187             '''!
188               @brief Configure as comparator output.
189             '''
190             global addr_G
191             if self.channel == 0:
192                     CONFIG_REG = [
                                ADS1115_REG_CONFIG_OS_SINGLE |
                                ADS1115_REG_CONFIG_MUX_DIFF_0_1 |
                                mygain |
                                ADS1115_REG_CONFIG_MODE_CONTIN,
                                ADS1115_REG_CONFIG_DR_128SPS |
                                ADS1115_REG_CONFIG_CQUE_NONE]
193             elif self.channel == 1:
194                     CONFIG_REG = [
                                ADS1115_REG_CONFIG_OS_SINGLE |
                                ADS1115_REG_CONFIG_MUX_DIFF_0_3 |
                                mygain |
                                ADS1115_REG_CONFIG_MODE_CONTIN,
                                ADS1115_REG_CONFIG_DR_128SPS |
                                ADS1115_REG_CONFIG_CQUE_NONE]
195             elif self.channel == 2:
196                     CONFIG_REG = [
                                ADS1115_REG_CONFIG_OS_SINGLE |
                                ADS1115_REG_CONFIG_MUX_DIFF_1_3 |
                                mygain |
```

```
                                ADS1115_REG_CONFIG_MODE_CONTIN,
                                ADS1115_REG_CONFIG_DR_128SPS |
                                ADS1115_REG_CONFIG_CQUE_NONE]
197             elif self.channel == 3:
198                     CONFIG_REG = [
                                ADS1115_REG_CONFIG_OS_SINGLE |
                                ADS1115_REG_CONFIG_MUX_DIFF_2_3 |
                                mygain |
                                ADS1115_REG_CONFIG_MODE_CONTIN,
                                ADS1115_REG_CONFIG_DR_128SPS |
                                ADS1115_REG_CONFIG_CQUE_NONE]
199
200             bus.write_i2c_block_data(addr_G,
                   ADS1115_REG_POINTER_CONFIG, CONFIG_REG)
201
202         def read_value(self):
203                 '''!
204                   @brief  Read ADC value.
205                   @return raw  adc
206                 '''
207                 global coefficient
208                 global addr_G
209                 data = bus.read_i2c_block_data(addr_G,
                      ADS1115_REG_POINTER_CONVERT, 2)
210
211                 # Convert the data
212                 raw_adc = data[0] * 256 + data[1]
213
214                 if raw_adc > 32767:
215                         raw_adc -= 65535
216                 raw_adc = int(float(raw_adc)*coefficient)
217                 return {'r' : raw_adc}
218
219         def read_voltage(self,channel):
220                 '''!
221                   @brief Reads the voltage of the specified
                         channel.
222                   @param channel  the Channel: 0-3
223                   @n For Single-ended Output:
224                   @n     0 : AINP = AIN0 and AINN = GND
225                   @n     1 : AINP = AIN1 and AINN = GND
226                   @n     2 : AINP = AIN2 and AINN = GND
227                   @n     3 : AINP = AIN3 and AINN = GND
228                   @n For Differential Output:
229                   @n     0 : AINP = AIN0 and AINN = AIN1
230                   @n     1 : AINP = AIN0 and AINN = AIN3
231                   @n     2 : AINP = AIN1 and AINN = AIN3
232                   @n     3 : AINP = AIN2 and AINN = AIN3
233                   @return Voltage
234                 '''
235                 self.set_channel(channel)
```

```
236                 self.set_single()
237                 time.sleep(0.1)
238                 return self.read_value()
239 
240         def comparator_voltage(self,channel):
241                 '''!
242                   @brief Sets up the comparator causing the
                          ALERT/RDY pin to assert .
243                   @param channel  the Channel: 0-3
244                   @n For Single-ended Output:
245                   @n     0 : AINP = AIN0 and AINN = GND
246                   @n     1 : AINP = AIN1 and AINN = GND
247                   @n     2 : AINP = AIN2 and AINN = GND
248                   @n     3 : AINP = AIN3 and AINN = GND
249                   @n For Differential Output:
250                   @n     0 : AINP = AIN0 and AINN = AIN1
251                   @n     1 : AINP = AIN0 and AINN = AIN3
252                   @n     2 : AINP = AIN1 and AINN = AIN3
253                   @n     3 : AINP = AIN2 and AINN = AIN3
254                   @return Voltage
255                 '''
256                 self.set_channel(channel)
257                 self.set_differential()
258                 time.sleep(0.1)
259                 return self.read_value()
```

A.0.4 DFR0026

Listing A.4: 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.0.5 Camera

Listing A.5: 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()
```

A.0.6 Memory mangement

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 pandas as pd
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()
```

```
42                                      try:
43                                              return float(value)*
                                                    self.units[unit]
44                                      except KeyError:
45                                              raise ValueError(f"
                                                    unknown␣unit:␣{
                                                    unit}")
46
47                      except subprocess.CalledProcessError as e:
48                              raise ValueError(f"Error␣running␣
                                    script:{e.output}")
49              def error_check(self):
50                      mem=self.check_memory()
51                      max=32*10e9
52                      if mem >= 0.2* max:
53                              raise MemoryError("memory␣on␣pi␣is␣
                                    about␣to␣␣␣used␣up")
```

Appendix B

TDD Script

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.

B.0.1 DHT22

Listing B.1: DHT22 unit test

```
    from DHT22 import DHT22
    import board
    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)
```