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 3, 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

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Chapter 1

Introduction

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 a lot of issues associated with this communication such as signal loss due to:

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

In this project I want to explore mesh networks and transmit data across them, a mesh network is a type of network where no node(a node is just a device which has a transceiver) in the network acts as a master. As we look at the environment in which this project aims to be, we 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" when transmitting any radio wave it takes energy another factor to consider is whether wind which will cause a delay in the signal. this report aims to show my findings and try to count for Environmental conditions

2.1.1 Overview

This section provides a brief overview of my 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 Non-line of sight

2. What sensors /senor modules do we use

   - What sensors will give us a good range in an Irish forest
   - What are the limitations on the board we use
   - Do we need to have any additional hardware to accommodate a specific board

3. What microprocessor/hardware do we use?

   - What 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 we looking to use 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 each node will be attached to a tree each having a transceiver

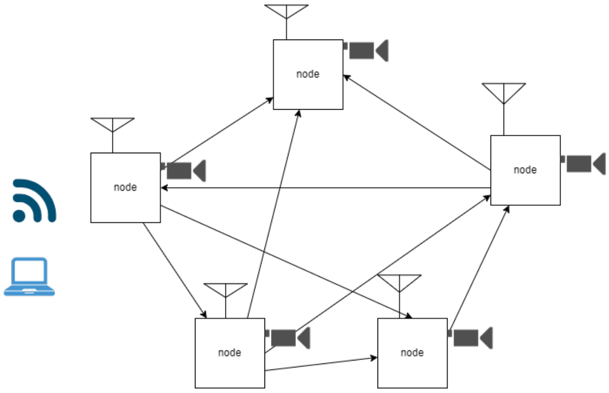


Figure 2.1: Basic block diagram of a mesh network

2.2 Hardware Consideration

In this project we need to have data to transmit firstly let's describe what we want our network to have:

1. we want our mesh network to transmit data for example temperature, humidity and light and camera
2. I want there to be data read every hour and stored as a CSV file the image file will depend on the module I pick
3. I want to have a motion to detect any animal that passes the node

the following is a rough circuit diagram for the project: firstly let's establish the following:

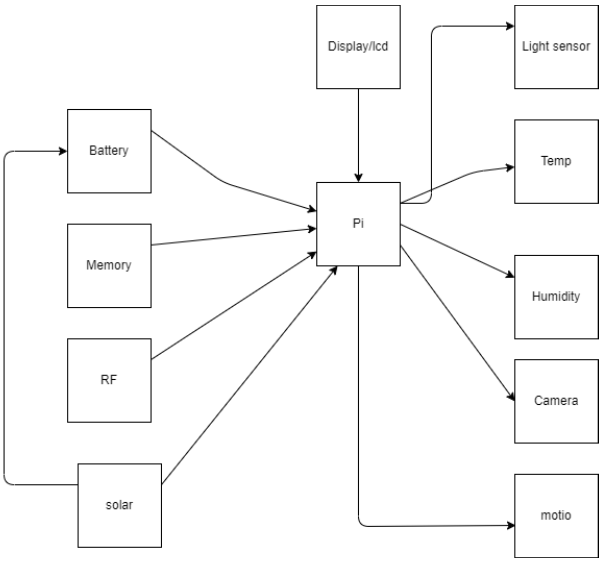


Figure 2.2: Rough circuit diagram for project

1. I can't use PCB due to the ordering process taking too long to come due to the time given to me
2. using any type of board like wire wrap would take too long and is outside of the goals of this project
3. This leaves with a choice of either the Arduino or pi

This section will Dicuss the following:

1. The sensors we will use in the project
2. the ADC we will have to will have to consider
3. the camera i picked considation for in this project
4. the memory module condestrations
5. the battery i picked
6. Considering the ardunio vs PI

2.2.1 Sensor considerations

In this section, we will discuss the process of considering each commponet of the sensors these sensor will be the following:

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

Temperature & Humidity sensor

In our consideration for this sensor we can establish that we want our sensor to work in the following conditions:

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

From that knowledge, I researched the temperature range in Ireland,

According to Met eireann**?**, we get the following table which the highest temperature in a Shaded

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

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

According to the table, the highest temperature is 33.3 now to look at the other extreme for the Lowest temperature:

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

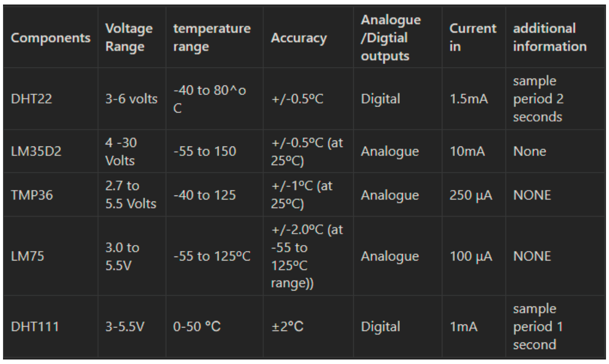
Table 2.2: Lowest shader air Met Eireann(13^{th} June 2023)

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

I also looked at humdity this referes to the amount of water vaper in the air. from met eirrean **?** got this table: The ranges are 68.3% to 88 % So with these conserdations here are the diffrent components:

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

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



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

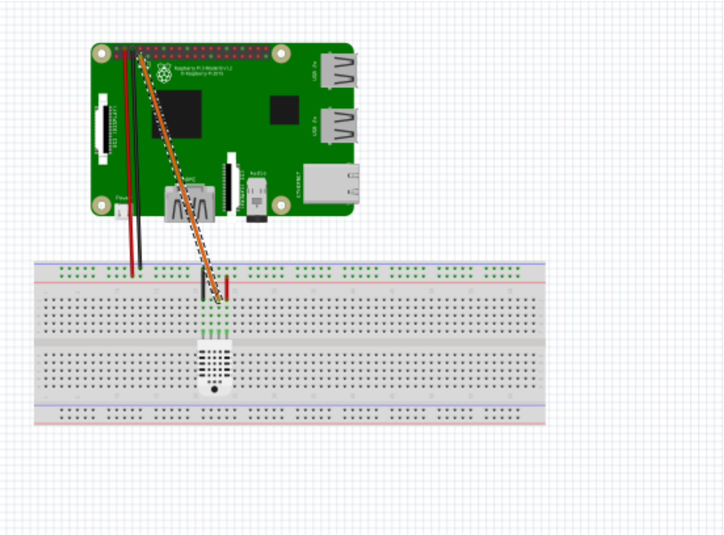
Table 2.4: Comparing of temperature sensors

After this, I limited this down to two sensors DHT22 and DHT11. The following are the advantages and disadvantages of the DHT22 and DHT11: So in conclusion I choose DHT22 which is a Digital output. See a wiring diagram below

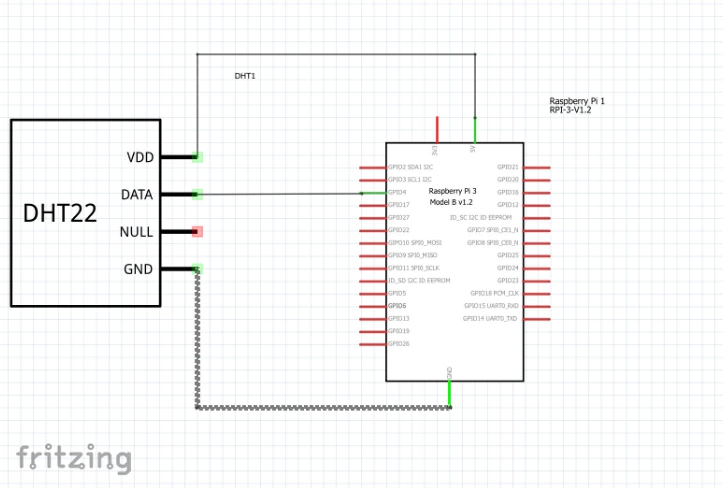
Device	Advantages	Disadvantages
DHT22	good accuracy has temp and humidity, falls in our temp range	sample period 2 seconds
DHT11	OK voltage,better sample period	draws a lot of current , and our of range

Table 2.5: Comparing DHT22 and DHT11

This will have an Interface of the following:



(a) Interface for DHT22



(b) Schematic for DHT22

From above we see our schematic, DHT22 connections are the following:

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

? The following is the link to the datasheet of this module when reading from this componet there is a delay of 2 second due to the sampling period.

Light sensor

In this section, we want to consider the following:

1. What region are we in
2. What light levels do we expect in this country
3. What sensor will accommodate this range

For this sensor we also must consider the outside aspect of the project i found this table on ? This table is the assoicated lux level incate when the vaules are . From above we want our sensor to be 0.002 to 25000 lux ideally, with that in mind here are the components I found through research:

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

Table 2.6: Illuminates values

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

Table 2.7: table of light sensors

After doing research DFR0026 **?** is the option I propose to use as it is the best for our application which will have an analogue output to see the interface see below:

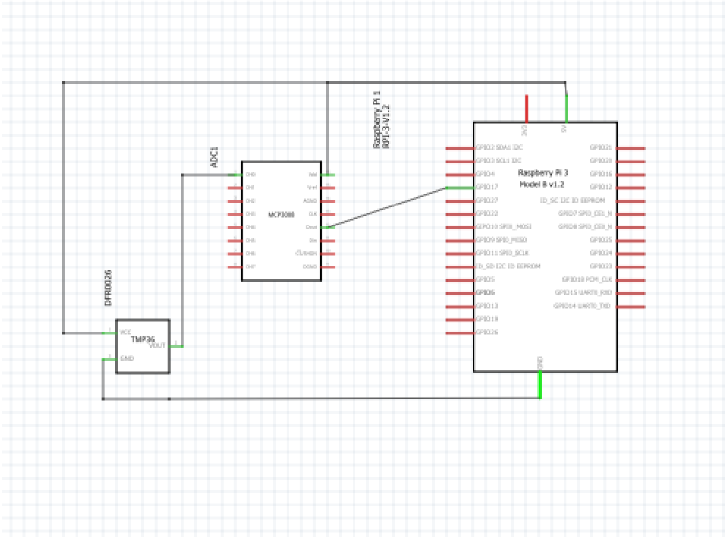


Figure 2.4: Interface for DFR0026

The following are the connections:

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

The commpoent relies on the ADC which is on page**??**

Motion sensor

For this section, we have to consider the following:

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

The following are the components I considered:

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

Table 2.8: Motion sensor components

The sensor I'm choosing is AS312**?**(which has a delay time of 2 seconds) which is a Digital interface to see the wiring see below:

the following is the interface for our device

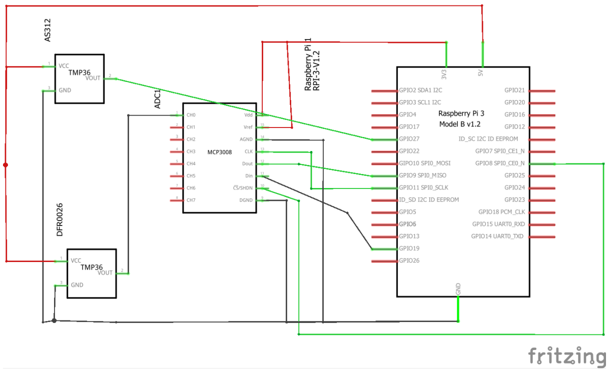
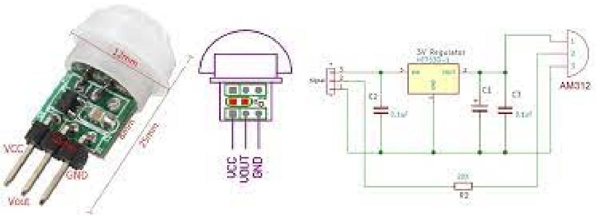


Figure 2.5: Interface for AS312

with The connections are the following:



Pinout of AS312

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

This component has the following:

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

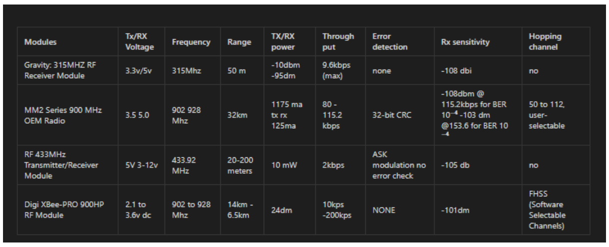
Sensor connections

2.2.2 Radio Module

For this section, we have the following considerations:

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

Through research, I found the following table:



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

Table 2.9: Radio modules found in research

Out of these, I picked the MM2 Series 900 MHz**?**.Note that the seller of this radio module has limited the documentation of this module makes it hard to draw an interface for this module which will be done next semester

2.2.3 ADC Considerations

for the ADC the following considerations:

1. low power
2. high bit resolution
3. low amount of channels
4. high sample rate

the two things we want for this is the high bit Resolution and a high sample rate

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

Above are the components I had to choose from for this project, I picked MCP3008 due to its resolution and sample rate the following is the schematic for the MCP3008**?** the following are connections:

1. VDD is connected to 3v3 pin of the Pi

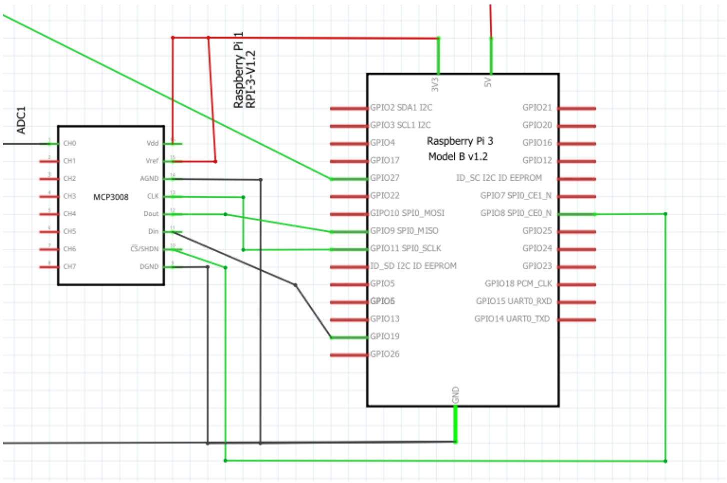


Figure 2.6: Schematic for MCP3008

2. VRef is also connected to 3v3 pin of the Pi

3. AGND is connected to the gnd pin

4. CLK pin is connected to GPIO port 11

5. Dout pin is connected to GPIO port 9

6. Din pin is connected to GPIO port 19

7. CS pin is connected to GPIO port 8

8. DGND ping is connected to the gnd pin

this commponet has the following:

1. A 10 bit resultion

2. seen as the reference will be 3.3v

3. 200ksps meaning the delay to read is 5μ seconds

2.2.4 Camera

For the camera, we have to keep in mind the following:

1. Focal length

2. Resolution

3. Power

4. what lux values it works at

The camera I pick is a Raspberry Pi VR 220**?** Camera to see how to connect look at the following link

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

Table 2.10: Camera module

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

Table 2.11: mirco SDs in consirdation

2.2.5 Memory module

For this section, we consider the following:

1. The file formatting of the sensor data
2. The file formatting of the camera data
3. What are the possible sizes of data
4. what is the size of the raspberry pi OS

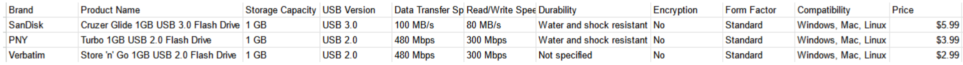
in my project, I plan on using the following:

1. For the sensor I plan on use storing the data in a CSV file with the following heading: (timestamp, heat, humidity, light level, anything detected) which can be around 25KB
2. For the camera in the plan using 10 MB is the largest file size
3. For the raspbery pi i downloaded the raspberry imager this has loads of options such as the following on **??**

after has been we must consider a mircoSD ,here is the following considerations: the best here is the silicon power 32GB due to its tempearure rnage and read and write times now for a testing the data and tranfering code i want an flash drive so i can extract python file and mix and manage sensor data and a back up for the data incase. for this we have the following to consider:

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

here is what I found through research: The Raspberry Pi 4 supports USB 2 and USB 3. For this, I'll pick the Turbo 1GB USB 2 Flash Drive



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

Table 2.12: Memory usb to consider

2.2.6 Battery

In this section, we want to consider the following:

1. Have enough power for all sensors and radio module
2. Have storage of the battery
3. Discharge rate of the battery (how many operating hours can I get out of the battery)

Here are the following Devices I found :

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

Table 2.13: battery considerations

The battery I'm going for is the li-polymer which has a micro USB how to charge:

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

Aside: this commpoent has the following:

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

2.2.7 Arduino vs PI Consideration

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

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

for these will consider the Arduino and the Raspberry Pi the Advantages and disadvantages of these are the following:

Although the Arduino would be more efficient than the Raspberry Pi due to Raspberry Pi has an Operating System I am picking the Pi as I'm more familiar with Python and Linux. Linux can be used to handle the networking side of the project I am willing to lose some efficiency in power for an easier time making the code for this project

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

Table 2.14: Advantages /Disadvantages of Arduino vs pi

Picking a Raspberry Pi

Now that we have picked out a device to use we need to define what we need in terms of the following:

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

GPIO(General purpose input/output) is used to select the input/output the pi can only take in digital signals only Seen as we have our components chosen that require A GPIO port (temperature/ humidity, on page **??**, Light on page **??**, motion on page **??**) we need at least 3 GPIO ports to be available to us as the light sensor and the motion will need an adc as looking through the documentation .firstly let's look at the different models

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

Table 2.15: Table of Raspberry Pi's

The above table displays the modules seen as our radio modules is 900Mhz we want 1.5GHZ which is the Raspberry Pi 4 which needs USB c charger and an HDMI. for wireing our Pi here is the GPIO pin pinout:

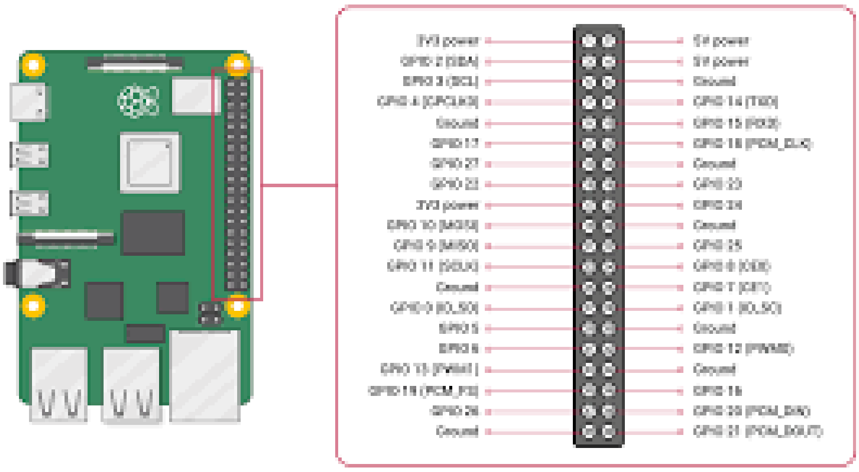
Picking an PI OS

Seen as

2.2.8 Conclusion

In this project the hardware needed is the following commponets:

1. 1 x Raspberry Pi 4 Model B
2. 1 x HDMI cable



Pinout for the pi

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

2.3 Software considerations

now that we have established the essential Hardware needed for our project we must 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. Is this gonna be an OOP or functional approach
5. How to program each device?

2.3.1 Raspberry pi OS

In this Section we want to keep in mind that each os is heavy wegiht we need to consider the following:

1. When we format the os is the data on the sd lost does it croupt the card?
2. We want an os that is low in capacity ?
3. Is 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 reaserch the suggestion of backing up the data is a good sugestion. for now i will use the recommended os and stripdown from there which will be dicussed in the methodolgy section of this report.

2.3.2 Sensor code

in this section, I will discuss the following :

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

this 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 I make 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 I got this from the following link (**Note: I haven't tested this due to the constraints of this year so I can only go off what others have done.**)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 follwoing:

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 libaray
2. in the functipon whch 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

```
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

```
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 dirctory 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 devolmponet

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 and dissadvantges of TDD:

1. Advantages

   (a)

2. Dissadvantages

(a)

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 **?**

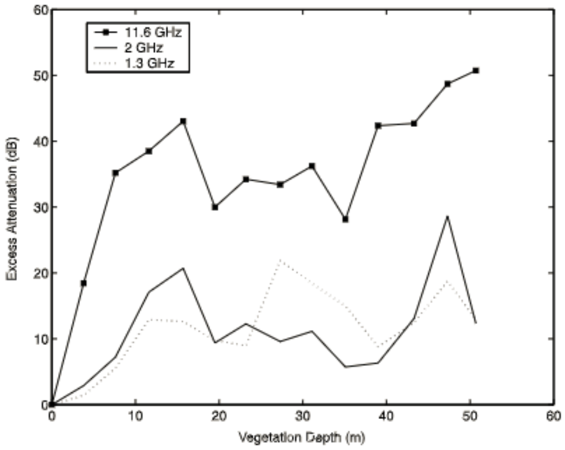


Figure 2.7: Silver Maple in-leaf excess attenuation for the line of trees geometry (receiver antenna height: 3.5 m, SAVAGE ET AL.pg.7

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

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

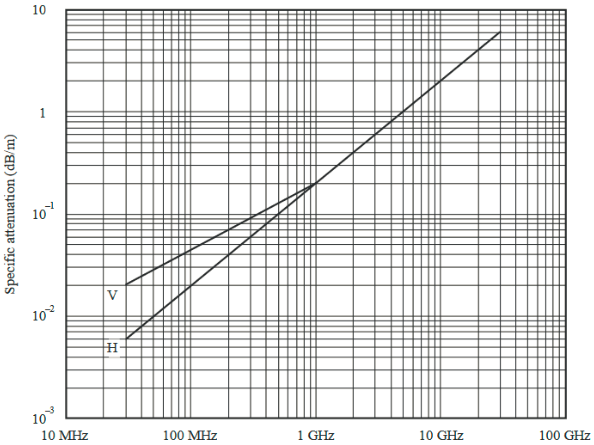


Figure 2.8: Specific attenuation due to woodland (Recommendation ITU-R P.833-7 (02/2012) Attenuation in vegetation pg.5

From this graph we can assume the following:

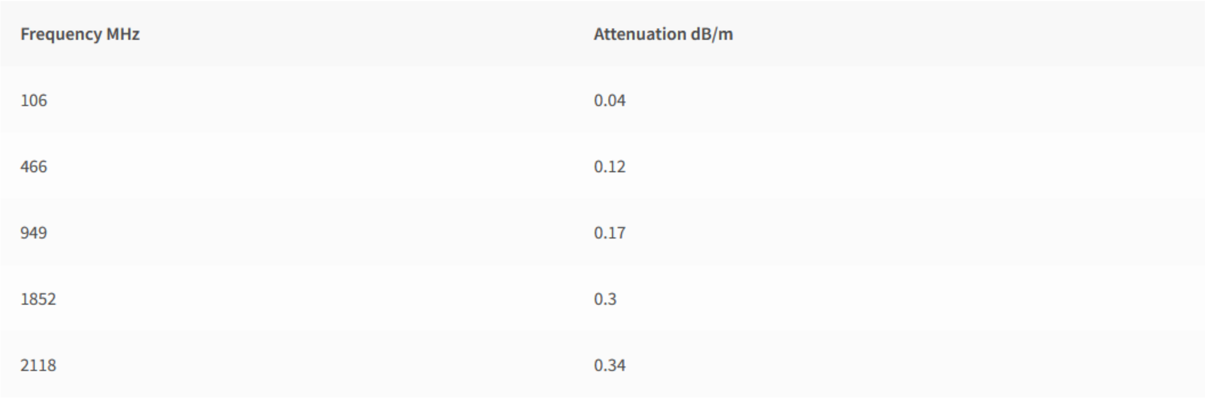
(a) From a frequency ≥15GHz we can assume Attenuation is more components

(b) Around the 1 GHz range we get low values of Attenuation

(c) in the MHz range we get the best response

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

so now that we established our range let us consider what happens when it rains **?**



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

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

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

2.4.1 Absorption of water

for this, I found this graph from Lunken Heimer **?**

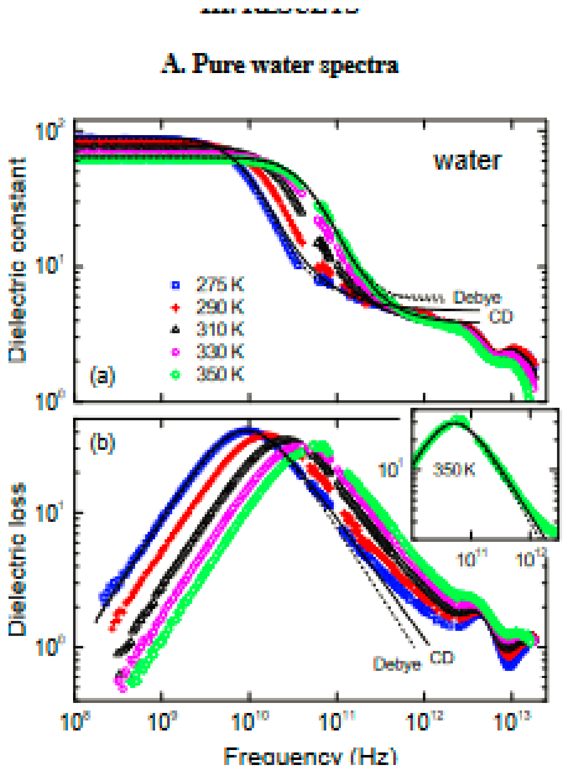


Figure 2.10: absorption of water

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

2.5 mesh network considerations

For this section, we have to consider the following:

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

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

1. LORA
2. Zigbee

According to Chen (2023)**?** "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 rang
Multi-channel information procession	Small payload	Scalability	Signal interfere
Strong anti-interface ability	Low bandwidth	–	High-sensitivity
High-sensitivity levels	Spectrum interference		

Table 2.16: Advantages and Disadvantages of LoRa and ZigBee

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

2.6 Review key of research Papers

The following are the research papers I used

1. zhao

   In my research, I found multiple projects that are similar to mine In Zhao(2023)(**?**, 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 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

3.2 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 eniroment 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

3.3 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

3.3.1 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

3.3.2 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

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.

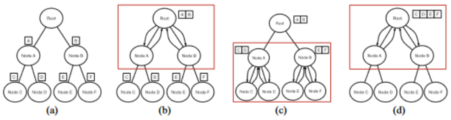


Fig. 3. Recursive data collection. (a) Each node (other than the root) has a data item indicated by a square. (b) Root requests data from nodes A and B. (c) Nodes A and B request data from nodes C–F. (d) In the next round of requests, the root collects the data items of nodes C–F from nodes A and B.

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 .

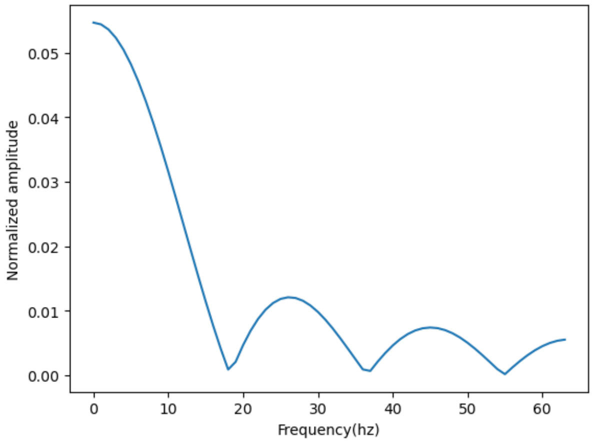


Figure 3.2: sample graph of a FIR response

3.3.3 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."

3.3.4 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

SB Components LoRa HAT

Which has a E22-900T22S on the board which has a throughput rate of 0.3kbps62.5kbps so the maximum time it will take to get to a node will be around 16 seconds depending on distance ,

3.4 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

3.4.1 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:

```
#!/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

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$_{A}DS1115wedothefollowing$:

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)
```

3.4.2 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
```

3.4.3 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()
```

3.4.4 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 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 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 3.2: 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 3.3: 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 3.4: unit test for AS312

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

Note : Don't forget to import read$_{a}s12functionfromtestfileseenasthhisisamotionsensorourouout$

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 3.5: 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 3.6: si powerd SD snippnet

   ```
           import unittest
           from protest import Read_DHT22,Read_MCP3008,
              Read_AS312,Read_Raspberry_PiVR220,
              Read_Memory_module

           def Test_memory_module_turbo_1GB_size(self):
               #testing  turbo 1GB
               self.assertLessEqual(Read_Memory_module,1e9)
               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

3.5 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.

3.6 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.

3.7 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.

3.8 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.

3.9 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 4

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

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**

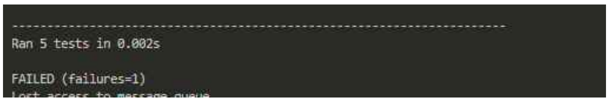
4.1.1 DHT22

Results during protypeing

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

Table 4.1: Recorded data from DHT22 on the May 3, 2024

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



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

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

Figure 4.1: unit test message for DHT22 module

4.1.2 AS312

Results during protype

date/time of record	motion detected(yes/no)
2024-03-25$_15 - 02 - 57$	False
2024-03-25$_15 - 04 - 37$	True

Table 4.2: Recorded data from AS312 on the May 3, 2024

4.1.3 DFR0026

Results during protypes

for our first test we got the following table

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

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

4.1.4 Raspberry Pi VR 220

When testing the Raspberry Pi VR 220

Results during portotypeing

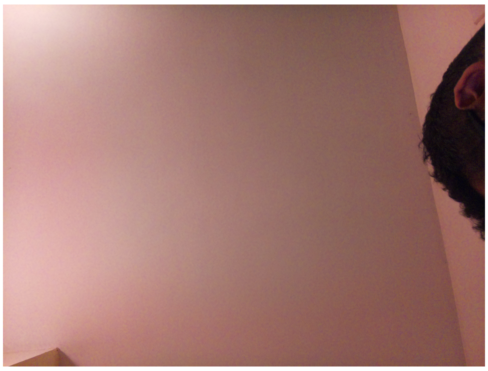


Figure 4.2: A photo from 25th of march 2024

4.2 Recorded data from transceiver

4.3 Recorded data from mesh network

Chapter 5

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.")
dht_sensor = DHT22()
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
ADS1115_REG_CONFIG_MUX_DIFF_2_3         = 0x30
## Single-ended P = AIN0, N = GND
ADS1115_REG_CONFIG_MUX_SINGLE_0         = 0x40
## Single-ended P = AIN1, N = GND
ADS1115_REG_CONFIG_MUX_SINGLE_1         = 0x50
## Single-ended P = AIN2, N = GND
ADS1115_REG_CONFIG_MUX_SINGLE_2         = 0x60
## Single-ended P = AIN3, N = GND
ADS1115_REG_CONFIG_MUX_SINGLE_3         = 0x70
## +/-6.144V range = Gain 2/3
ADS1115_REG_CONFIG_PGA_6_144V           = 0x00
## +/-4.096V range = Gain 1
ADS1115_REG_CONFIG_PGA_4_096V           = 0x02
## +/-2.048V range = Gain 2 (default)
ADS1115_REG_CONFIG_PGA_2_048V           = 0x04
## +/-1.024V range = Gain 4
ADS1115_REG_CONFIG_PGA_1_024V           = 0x06
## +/-0.512V range = Gain 8
ADS1115_REG_CONFIG_PGA_0_512V           = 0x08
## +/-0.256V range = Gain 16
ADS1115_REG_CONFIG_PGA_0_256V           = 0x0A
## Continuous conversion mode
ADS1115_REG_CONFIG_MODE_CONTIN          = 0x00
## Power-down single-shot mode (default)
ADS1115_REG_CONFIG_MODE_SINGLE          = 0x01
## 8 samples per second
ADS1115_REG_CONFIG_DR_8SPS                      = 0x00
## 16 samples per second
ADS1115_REG_CONFIG_DR_16SPS                     = 0x20
## 32 samples per second
ADS1115_REG_CONFIG_DR_32SPS                     = 0x40
## 64 samples per second
ADS1115_REG_CONFIG_DR_64SPS                     = 0x60
```

```
## 128 samples per second (default)
ADS1115_REG_CONFIG_DR_128SPS            = 0x80
## 250 samples per second
ADS1115_REG_CONFIG_DR_250SPS            = 0xA0
## 475 samples per second
ADS1115_REG_CONFIG_DR_475SPS            = 0xC0
## 860 samples per second
ADS1115_REG_CONFIG_DR_860SPS            = 0xE0
## Traditional comparator with hysteresis (default)
ADS1115_REG_CONFIG_CMODE_TRAD           = 0x00
## Window comparator
ADS1115_REG_CONFIG_CMODE_WINDOW         = 0x10
## ALERT/RDY pin is low when active (default)
ADS1115_REG_CONFIG_CPOL_ACTVLOW         = 0x00
## ALERT/RDY pin is high when active
ADS1115_REG_CONFIG_CPOL_ACTVHI          = 0x08
## Non-latching comparator (default)
ADS1115_REG_CONFIG_CLAT_NONLAT          = 0x00
## Latching comparator
ADS1115_REG_CONFIG_CLAT_LATCH           = 0x04
## Assert ALERT/RDY after one conversions
ADS1115_REG_CONFIG_CQUE_1CONV           = 0x00
## Assert ALERT/RDY after two conversions
ADS1115_REG_CONFIG_CQUE_2CONV           = 0x01
## Assert ALERT/RDY after four conversions
ADS1115_REG_CONFIG_CQUE_4CONV           = 0x02
## Disable the comparator and put ALERT/RDY in high state (default)
ADS1115_REG_CONFIG_CQUE_NONE            = 0x03

mygain=0x02
coefficient=0.125
addr_G=ADS1115_IIC_ADDRESS0
class ADS1115():
        def set_gain(self,gain):
                '''!
                  @brief Sets the gain and input voltage range.
                  @param gain  This configures the programmable gain amplifier
                  @n ADS1115_REG_CONFIG_PGA_6_144V     = 0x00 # 6.144V range = Gain 2/3
```

```
              @n ADS1115_REG_CONFIG_PGA_4_096V      = 0x02   # 4.096V range = Gain 1
              @n ADS1115_REG_CONFIG_PGA_2_048V      = 0x04   # 2.048V range = Gain 2
              @n default:
              @n ADS1115_REG_CONFIG_PGA_1_024V      = 0x06   # 1.024V range = Gain 4
              @n ADS1115_REG_CONFIG_PGA_0_512V      = 0x08   # 0.512V range = Gain 8
              @n ADS1115_REG_CONFIG_PGA_0_256V      = 0x0A   # 0.256V range = Gain 16
            '''
            global mygain
            global coefficient
            mygain=gain
            if mygain == ADS1115_REG_CONFIG_PGA_6_144V:
                    coefficient = 0.1875
            elif mygain == ADS1115_REG_CONFIG_PGA_4_096V:
                    coefficient = 0.125
            elif mygain == ADS1115_REG_CONFIG_PGA_2_048V:
                    coefficient = 0.0625
            elif mygain == ADS1115_REG_CONFIG_PGA_1_024V:
                    coefficient = 0.03125
            elif mygain == ADS1115_REG_CONFIG_PGA_0_512V:
                    coefficient = 0.015625
            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]
                elif self.channel == 2:
                        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]
                elif self.channel == 3:
                        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]

                bus.write_i2c_block_data(addr_G,
                   ADS1115_REG_POINTER_CONFIG, CONFIG_REG)

        def set_differential(self):
                '''!
                  @brief Configure as comparator output.
                '''
                global addr_G
                if self.channel == 0:
                        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]
                elif self.channel == 1:
                        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]
                elif self.channel == 2:
                        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]
                elif self.channel == 3:
                        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]

                bus.write_i2c_block_data(addr_G,
                   ADS1115_REG_POINTER_CONFIG, CONFIG_REG)

        def read_value(self):
                '''!
                  @brief  Read ADC value.
                  @return raw  adc
                '''
                global coefficient
                global addr_G
                data = bus.read_i2c_block_data(addr_G,
                   ADS1115_REG_POINTER_CONVERT, 2)

                # Convert the data
                raw_adc = data[0] * 256 + data[1]

                if raw_adc > 32767:
                        raw_adc -= 65535
                raw_adc = int(float(raw_adc)*coefficient)
                return {'r' : raw_adc}

        def read_voltage(self,channel):
                '''!
                  @brief Reads the voltage of the specified
                     channel.
                  @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 Voltage
                '''
                self.set_channel(channel)
                self.set_single()
                time.sleep(0.1)
                return self.read_value()

        def comparator_voltage(self,channel):
                '''!
                  @brief Sets up the comparator causing the ALERT/RDY pin to assert .
                  @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 Voltage
                '''
                self.set_channel(channel)
                self.set_differential()
                time.sleep(0.1)
                return self.read_value()
```

A.0.4 DFR0026

Listing A.4: 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())
```

A.0.5 Camera

Listing A.5: 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()
```

A.0.6 Memory mangement

Listing A.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 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()
                            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")
```

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):
```

```python
            self.assertIsInstance(hum,(int,float))

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