

Developing a mesh network in a wooded area



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.
March 7, 2024

Contents

List of Figures	3
List of Tables	4
1 Literature Review	7
1.1 Introduction	7
1.1.1 Overview	7
1.1.2 Mesh network	9
1.2 Hardware Consideration	9
1.2.1 Sensor considerations	10
1.2.2 Radio Module	17
1.2.3 ADC Considerations	17
1.2.4 Camera	18
1.2.5 Memory module	19
1.2.6 Battery	19
1.2.7 Arduino vs PI Consideration	20
1.2.8 Conclusion	22
1.3 Software considerations	22
1.3.1 Raspberry pi OS	22
1.3.2 Sensor code	23
1.3.3 MCP3008	25
1.3.4 Raspberry Pi VR 220 Camera	26
1.3.5 MM2 Series 900 MHz	26
1.3.6 code structure	26
1.3.7 File structure	26
1.3.8 Test Driven devolmponet	28
1.4 Attenuation	28
1.4.1 Absorption of water	30
1.5 mesh network considerations	31
1.6 Review key of research Papers	31
1.7 Summary	32
2 Methodology	33
2.1 Introduction	33
2.2 Setup of raspberry pi	33
2.3 Data Collection Methods	34
2.4 Software Module Development	34
2.4.1 Sensors	34
2.4.2 ADC	34
2.4.3 Camera	34
2.4.4 Memory Mangement	34

2.4.5	TDD	34
2.5	Hardware updates	38
2.6	Data Analysis Methods	38
2.7	Ethical Considerations	38
2.8	Validity and Reliability	38
2.9	Limitations and Delimitations	38
2.10	Timeline	38
3	Results	39
3.1	Recorded data from sensors	39
3.1.1	DHT22	39
3.1.2	AS312	40
3.1.3	DFR0026	40
3.1.4	Raspberry Pi VR 220	40
3.2	Recorded data from transceiver	40
3.3	Recorded data from mesh network	40
	Bibliography	40
4	Appendix A	41
A	Python Scripts	42
A.1	Sensor Scripts	42
A.1.1	DHT22	42
A.1.2	AS312	44
B	TDD Script	45
B.0.1	DHT22	45

List of Figures

1.1	Basic block diagram of a mesh network	9
1.2	Rough circuit diagram for project	10
1.4	Interface for DFR0026	15
1.5	Interface for AS312	16
1.6	Schematic for MCP3008	18
1.7	Silver Maple in-leaf excess attenuation for the line of trees geometry (receiver antenna height: 3.5 m, SAVAGE ET AL.pg.7	29
1.8	Specific attenuation due to woodland (Recommendation ITU-R P.833-7 (02/2012) Attenuation in vegetation pg.5	29
1.9	Predicted attenuation due to rain for the region, which is measured by using the ITU standards,(Source: Hindawi(2014))	30
1.10	absorption of water	30
3.1	unit test message for DHT22 module	39
3.2	A photo from March 7, 2024	40

List of Tables

1.1	Highest shader air Met Eireann(13 th June 2023)	11
1.2	Lowest shader air Met Eireann(13 th June 2023)	11
1.3	Realtive Humidity(%) according to met eirrean	12
1.4	Comparing of temperature sensors	12
1.5	Comparing DHT22 and DHT11	12
1.6	Illuminates values	14
1.7	table of light sensors	14
1.8	Motion sensor components	15
1.9	Radio modules found in research	17
1.10	Camera module	18
1.11	mirco SDs in consirdation	19
1.12	Memory usb to consider	19
1.13	battery considerations	20
1.14	Advantages /Disadvantages of Arduino vs pi	20
1.15	Table of Raspberry Pi's	21
1.16	Advantages and Disadvantages of LoRa and ZigBee	31
3.1	Recorded data from DHT22 on the March 7, 2024	39
3.2	Recorded data from AS312 on the March 7, 2024	40
3.3	Recorded data from DFR0026 on the March 7, 2024	40

Listings

1.1	Example code for DHT2	24
1.2	Example code for AS312	25
1.3	ADC code	25
1.4	example code for camera	26
1.5	sample code for turning sensor data into a data	27
1.6	example code for storing directory	27
2.1	sample test intial code	35
2.2	sample test for DHT22	35
2.3	unit test for DFR0026 and MCP3008	36
2.4	unit test for AS312	36
2.5	camera unit test	37
2.6	si powerd SD snippnet	37
A.1	DHT22code	42
A.2	code for AS312	44
B.1	DHT22 unit test	45

Glossary

APD	Avalanche PhotoDiode	MC	Multiple-Carrier
API	Application Programming Interface	MIMO	Multiple Input Multiple Output
ASK	Amplitude Shift Keying	MLSE	Maximum Likelihood Sequence Estimation
AWG	Agile Waveform Generator	MMF	Multi Mode Fiber
B2B	Back-2-Back	MSK	Minimum Shift Keying
BBP	Baseband Processor	MSO	Mixed Signal Oscilloscope
BER	Bit Error Ratio	MZI	Mach-Zehnder Interferometer
BL	Bandwidth-Length	MZM	Mach-Zehnder Modulator
BLAST	Bell Labs <u>L</u> ayered <u>S</u> pace <u>T</u> ime	NGPON	Next Generation Passive Optical Network
BT	Time Bandwidth Product	NLSE	Non-Linear Schrödinger Equation
CD	Chromatic Dispersion	NRZ	Non-Return to Zero
CDMA	Code Division Multiple Access	ODN	Optical Distribution Network
CPM	Continuous Phase Modulation	OS	operating system (OS)
CSI	Channel State Information	OFDM	Orthogonal Frequency Division Multiplexing
D	Dispersion Coefficient	OOK	On Off Keying
DD	Direct Detection	OSA	Optical Spectrum Analyzer
DECT	Digital Enhanced Cordless Telecommunications	OSNR	Optical Signal to Noise Ratio
DPO	Digital Phosphorous Oscilloscope	PAPR	Peak to Average Power Ratio
DPM	Digital Phase Modulation	PD	Photo Diode
DSP	Digital Signal Processing	P-i-N	P-doped Intrinsic N-doped Photodiode
EDFA	Eridium Doped Fiber Amplifier	PON	Passive Optical Network
FBMC	Filter Bank Multi-Carrier	PRS	Partial Response Signalling
FDM	Frequency Division Multiplex	QMDD	Quadrature Modulation Direct Dectection
FDMA	Frequency Division Multiple Access	RF	Radio Frequency
FEA	Finite Element Analysis	RIN	Relative Intensity Noise
FEC	Forward Error Correction	SCPI	Standard Commands for Programmable Instruments
FFT	Fast Fourier Transform	SISO	Single Input Single Output
FIR	Finite Impulse Response	SMF	Single Mode Fiber
FRS	Full Response Signalling	SNR	Signal to Noise Ratio
FTTx	Fiber To The x	SOA	Semiconductor Optical Amplifier
GASK	Gaussian Amplitude Shift Keying	SPM	Self Phase Modulation
GFDM	Generalised Frequency Division Multiplexing	SS	Spread Spectrum
GIPO	General Purpose Input/Output	SSFM	Split-Step Fourier Method
GLPF	Gaussian Low-Pass Filter	SSSFM	Symmetricised Split Step Fourier Method
GMSK	Gaussian Minimum Shift Keying	TCM	Trellis Coded Modulation
GSM	Global System for Mobile Communications	TDM	Time Division Multiplex
GVD	Group Velocity Dispersion	TDMA	Time Division Multiple Access
IFFT	Inverse Fast Fourier Transform	TFM	Tamed Frequency Modulation
IIR	Infinite Impulse Response	TIA	TransImpedance Amplifier
IMDD	Intensity Modulation Direct Detection	TDD	Test Driven Development
ISI	InterSymbol Interference	UFMC	Universal Filtered Multiple Carrier
IVI	Interchangeable Virtual Instruments	USB	Universal Serial Bus
LAN	Local Area Network	VISA	Virtual Instrument Software Architecture
LD	Dispersion Length	WDM	Wave Division Multiplex
LD	Laser Diode		
LUT	Look-Up Table		

Chapter 1

Literature Review

[

1.1 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

1.1.1 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

1.1.2 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

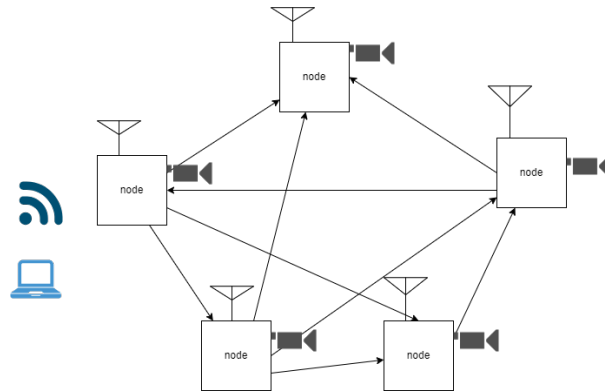


Figure 1.1: Basic block diagram of a mesh network

having a transceiver

1.2 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:

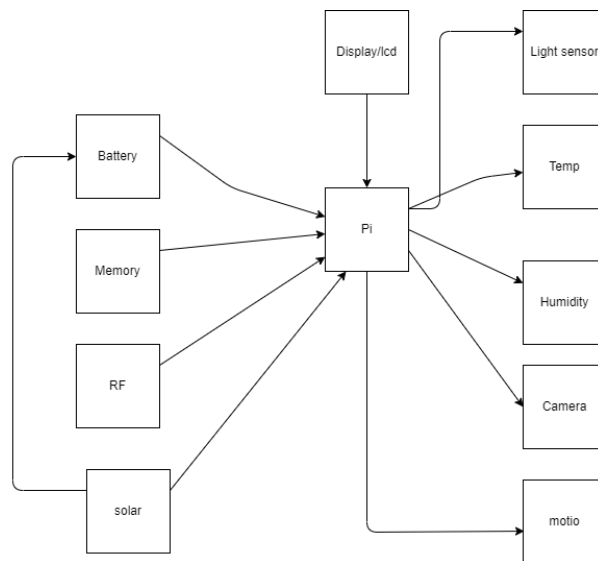


Figure 1.2: Rough circuit diagram for project

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 arduino vs PI

1.2.1 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 1.1: Highest shader air Met Eireann(13th 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 1.2: Lowest shader air Met Eireann(13th 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 humidity this refers to the amount of water vapor in the air. from met eirrean ? got this table: The ranges are 68.3% to 88 % So with these considerations here are

	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 1.3: Relative Humidity(%) according to met eirrean

the different components:

Components	Voltage Range	temperature range	Accuracy	Analogue /Digital outputs	Current in	additional information
DHT22	3-6 volts	-40 to 80 °C	+/-0.5°C	Digital	1.5mA	sample period 2 seconds
LM35D2	4 -30 Volts	-55 to 150	+/-0.5°C (at 25°C)	Analogue	10mA	None
TMP36	2.7 to 5.5 Volts	-40 to 125	+/-1°C (at 25°C)	Analogue	250 µA	NONE
LM75	3.0 to 5.5V	-55 to 125°C	+/-2.0°C (at -55 to 125°C range)	Analogue	100 µA	NONE
DHT11	3-5.5V	0-50 °C	±2°C	Digital	1mA	sample period 1 second

Table 1.4: Comparing of temperature sensors

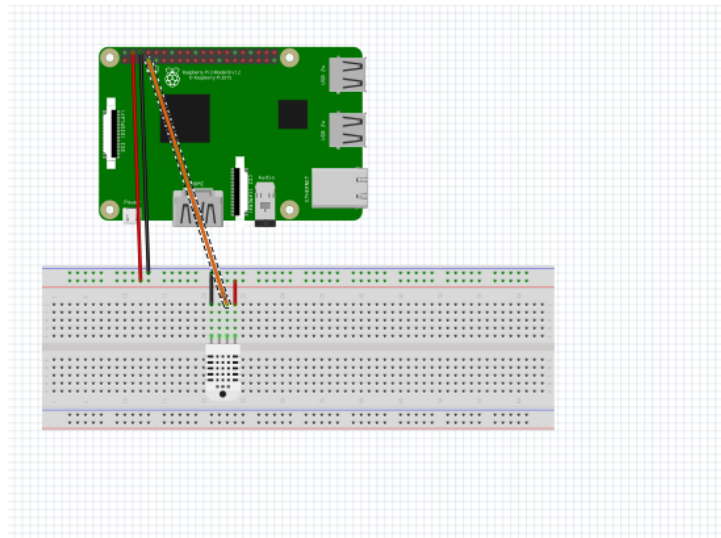
After this, I limited this down to two sensors DHT22 and DHT11. The following are the advantages and disadvantages of the DHT22 and DHT11: So in conclusion I choose DHT22

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

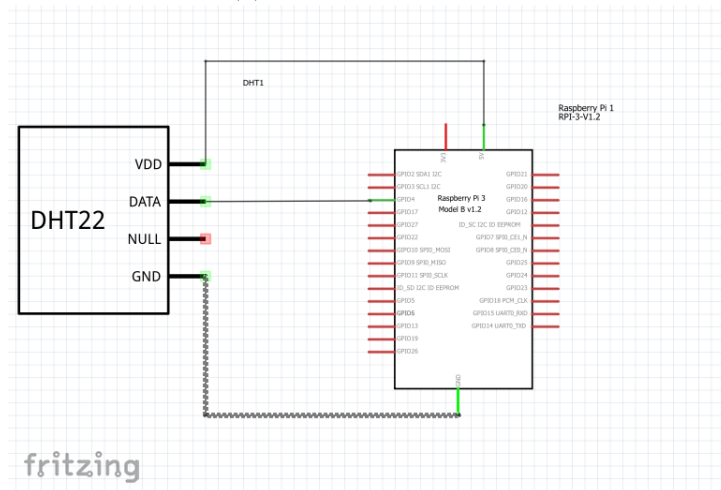
Table 1.5: Comparing DHT22 and DHT11

which is a Digital output. See a wiring diagram below

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 component 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

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 1.6: Illuminates values

our sensor to be 0.002 to 25000 lux ideally, with that in mind here are the components I found through research:

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 1.7: 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:

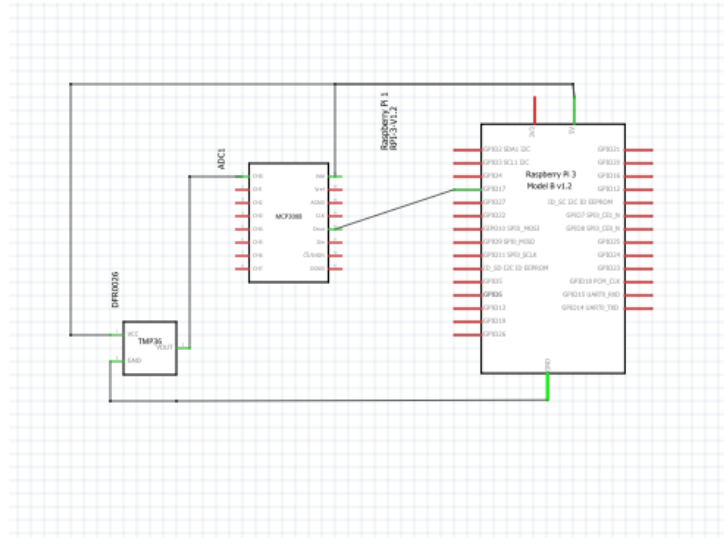


Figure 1.4: Interface for DFR0026

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 page17

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 1.8: 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

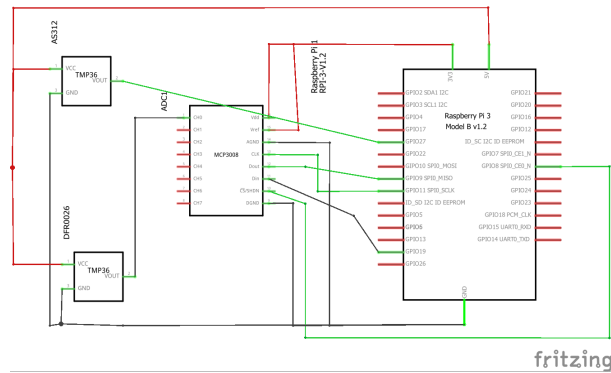
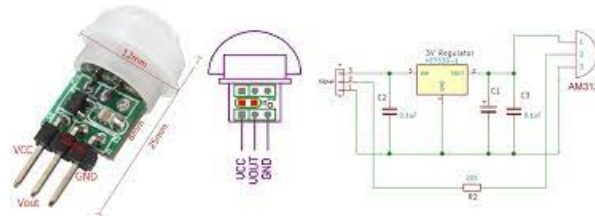


Figure 1.5: Interface for AS312

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° degree
3. A Delay of $15 \mu\text{Seconds}$

Sensor connections

1.2.2 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 -95dbm	9.6kbps (max)	none	-108 dbi	no
MM2 Series 900 MHz ODM 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 dbm @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 - 6.9km	24dbm	10kps ~200kps	NONE	-101dbm	FHSS (Software Selectable Channels)

Table 1.9: 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

1.2.3 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
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

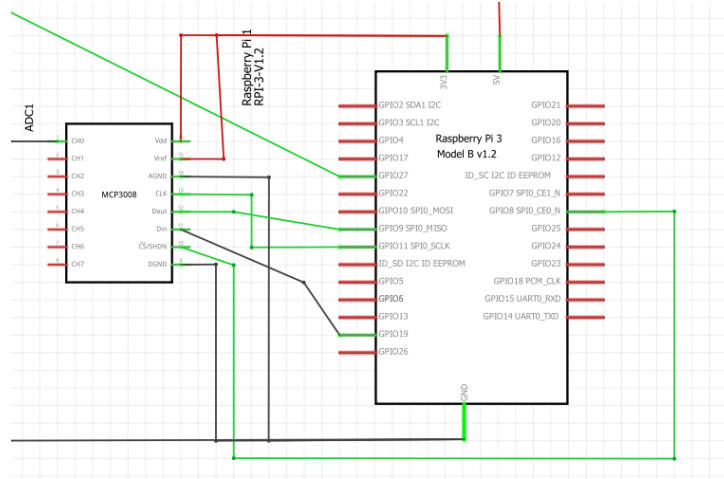


Figure 1.6: Schematic for MCP3008

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 pin is connected to the gnd pin

this component has the following:

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

1.2.4 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

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 1.10: Camera module

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

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 1.11: mirco SDs in consirdation

1.2.5 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 22

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

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 1.12: Memory usb to consider

this, I'll pick the Turbo 1GB USB 2 Flash Drive

1.2.6 Battery

In this section, we want to consider the following:

1. Have enough power for all sensors and radio module

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

Table 1.13: battery considerations

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 :

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

1.2.7 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:

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 1.14: Advantages /Disadvantages of Arduino vs pi

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

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:

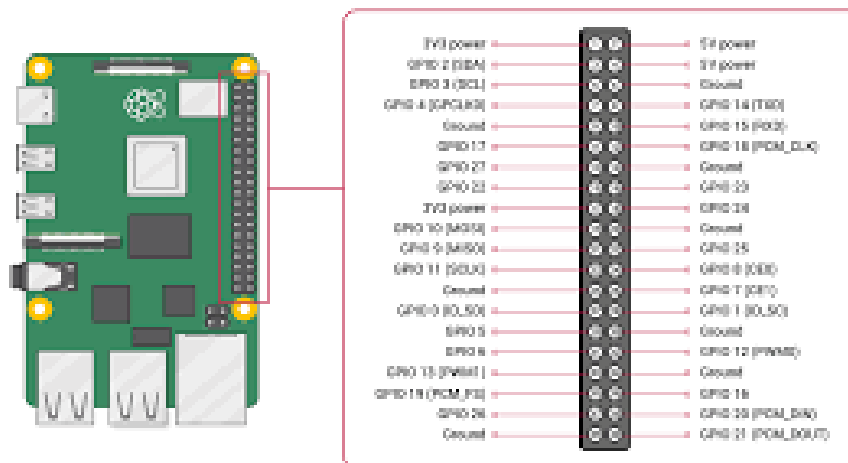
1. The amount of GPIO PORTS we need
2. Nature of the output of the sensor
3. Speed of the clock

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 12, Light on page 14, motion on page 15) 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 1.15: 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 wiring our Pi here is the GPIO pin pinout:



Pinout for the pi

Picking an PI OS

Seen as

1.2.8 Conclusion

In this project the hardware needed is the following components:

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

1.3 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 server and nodes
3. How will data be sent
4. Is this gonna be an OOP or functional approach
5. How to program each device?

1.3.1 Raspberry pi OS

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

1. When we format the OS is the data on the SD lost does it corrupt 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 research the suggestion of backing up the data is a good suggestion. for now I will use the recommended OS and stripdown from there which will be discussed in the methodology section of this report.

1.3.2 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 13 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 1.1: Example code for DHT2

```
1      #Libraries
2      import Adafruit_DHT as dht
3      from time import sleep
4      def setup_DHT22(Gpioport:int):
5          humidityy,temp=dht.read_retry(dht.DHT22, Gpioport
6              )
7              sleep(5)
8              return humundity,temp
9      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 adafruit libaray
2. in the functipon which takes the gpioport as an integer this will read the data on the pin and print it out

AS312

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

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

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

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

which will intall a gpio python module

Then type this into an IDE of your choosing

Listing 1.2: Example code for AS312

```
1      import RPi.GPIO as GPIO
2      import time
3
4      pir_sensor = 27
5      GPIO.setmode(GPIO.BOARD)
6
7      GPIO.setup(pir_sensor, GPIO.IN)
8      current_state = 0
9
10     time.sleep(0.1)
11     current_state = GPIO.input(pir_sensor)
12     if current_state == 1:
13         print("GPIO pin %s is %s" % (pir_sensor,
14                                     current_state))
15         # trigger camera
16         # must look up this
17     GPIO.cleanup()
```

this code does the following:

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

DFR0026

from the last example, nothing has changed from the last component an example code for this can be found on page 25

1.3.3 MCP3008

for this section, we want to consider the following:

1. The MCP3008 data out is GPIO 9

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

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

Listing 1.3: ADC code

```
1      from gpiozero import MCP3008
2      from time import sleep
3      DFR0026 = MCP3008(channel=0, device=0, port=9)
4
5      print('raw: {}'.format(DFR0026.value))
6      sleep(0.1)
```

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

1.3.4 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 1.4: example code for camera

```
1      from picamera import PiCamera
2      from time import sleep
3
4      camera = PiCamera()
5
6      camera.start_preview()
7      sleep(5)
8      camera.stop_preview()
```

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

1.3.5 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

1.3.6 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 humidity 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

1.3.7 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 1.5: sample code for turning sensor data into a data

```

1      import pandas as pd
2      import numpy as np
3      from datetime import datetime
4      cols_name=["Timestamp","Tempeature","Hummidty","
               Light_level","Motion_dected"]
5
6      #assume that being recorded now
7      data=[]
8      timestep=datetime.now()
9      timestep=timestep.strftime("%d/%m/%Y_%H:%M:%S")
10     Current_state=1
11     Heat=0.40
12     Hummidty=1.0
13     Light_level=0.23
14     data=np.array([[timestep],[Heat],[Hummidty],[
               Light_level],[Current_state]])
15     data=data.T
16     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 1.6: example code for storing directory

```

1      import tkinter as tk
2      from tkinter import filedialog
3      import json
4      import os
5
6      root = tk.Tk()
7      root.withdraw()
8      selected_dir = filedialog.askdirectory()
9
10     if not os.path.exists('selected_dir.json'):
11         # Write the selected directory to a JSON
12         # file
13         with open('selected_dir.json', 'w') as f:
14             json.dump(selected_dir, f)
15             print("Successfully saved
16                   selected directory to JSON
17                   file.")
18
19     else:
20         print("File 'selected_dir.json' already
21               exists. Not saving the directory.")
22
23     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 focuss on the size string messeage and then print out the size. this is just a sample script

1.3.8 Test Driven devolmponet

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

1. Advantages
 - (a)
2. Dissadvantages
 - (a)

1.4 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 ?

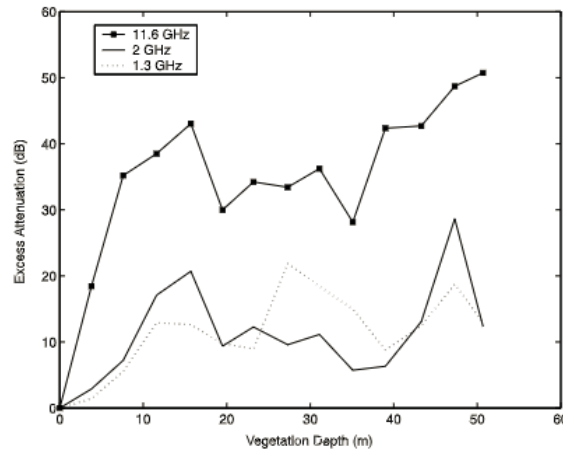


Figure 1.7: Silver Maple in-leaf excess attenuation for the line of trees geometry (receiver antenna height: 3.5 m, SAVAGE ET AL.pg.7

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

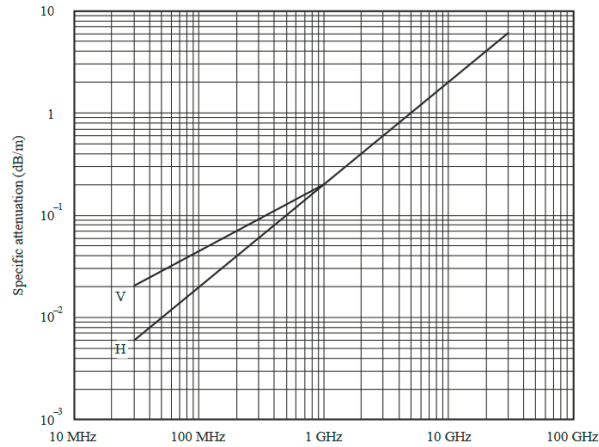


Figure 1.8: Specific attenuation due to woodland (Recommendation ITU-R P.833-7 (02/2012) Attenuation in vegetation pg.5

From this graph we can assume the following:

- (a) From a frequency $\geq 15\text{GHz}$ 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 10^6hz

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 1.9: 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

1.4.1 Absorption of water

for this, I found this graph from Lunken Heimer ?

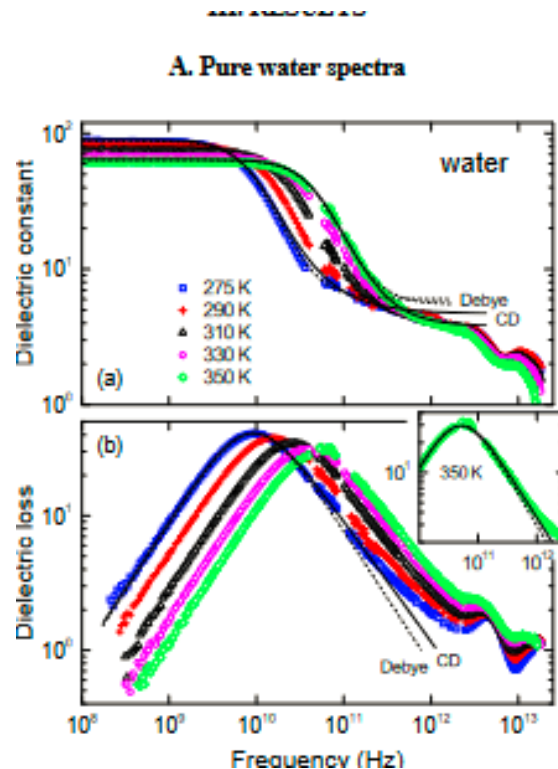


Figure 1.10: absorption of water

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.

1.5 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	Disadvantages
Long transmission distance	Low transmission rate	Low power consumption	Low data rate
Low power consumption	Slow data transfer rate	Long range	Limited range
Multi-channel information procession	Small payload	Scalability	Signal interference
Strong anti-interference ability	Low bandwidth	—	High-sensitivity
High-sensitivity levels	Spectrum interference		

Table 1.16: Advantages and Disadvantages of LoRa and ZigBee

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

1.6 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

1.7 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 2

Methodology

2.1 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

2.2 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 simply 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

2.3 Data Collection Methods

In this section i dicuss the following:

1. the intall unit test this is for what we expect our sensor to output this will be updated
2. How the data from sensor will be stored
3. the code assoicated with the above points

2.4 Software Module Development

this section is here to dicuss the method we took for developing software for the following:

1. Sensors
2. ADC
3. Camera
4. Radio module
5. Memory mangemnet
6. TDD

2.4.1 Sensors

DHT22

AS312

2.4.2 ADC

2.4.3 Camera

2.4.4 Memory Mangement

2.4.5 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°C for temperature 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°C for the lowest temp we have 6 bits to represent - 40 due to 2s complement so lowest will be -40.3°C 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 2.1: sample test initial code

```
1 import unittest
2 from protest import Read_DHT22
3 class test_project_code(unittest.TestCase):
4     def test_DHT_22_temp_output_type(self):
5         self.assertIsInstance(Read_DHT22, float)
6     def test_DHT22_temp_range(self):
7         self.assertGreaterEqual(Read_DHT22, -30.3)
8         self.assertLessEqual(Read_DHT22, 80.3)
```

This code import unittest . the from protest is a python files we can install functions from other python files this can be usefull for testing purposes then we initialized a test class call unittest.testcase our firstion fuction 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 simply 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 2.2: sample test for DHT22

```
1 import unittest
2 from protest import Read_DHT22
3 class test_project_code(unittest.TestCase):
4     hum,temp=Read_DHT22(2)
5     def test_DHT22_output_type(self):
6         self.assertIsInstance(Read_DHT22, tuple)
7         #....
8
9     def test_DHT22_hum_output_type(self):
10        self.assertIsInstance(hum, float)
```

```

11
12     def test_DHT22_hum_range(self):
13         self.assertEqual(hum, 0.0)
14         self.assertEqual(hum, 100.0)

```

seen as we expect our sensor to print out a humidity and temp values we set the output to a tuple to test for this we use `isInstance` 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 2.3: unit test for DFR0026 and MCP3008

```

1     import unittest
2     from protest import Read_DHT22, Read_MCP3008
3     class test_project_code(unittest.TestCase):
4     def test_DFR0026_MCP3008_out_type(self):
5         self.assertIsInstance(Read_MCP3008, float)
6     def test_DFR0026_MCP3008_out_range(self):
7         self.assertEqual(5.0000000)
8         self.assertEqual(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 snippet :

Listing 2.4: unit test for AS312

```

1     def test_AS312_out_type(self):
2         self.assertIsInstance(Read_AS312, bool)

```

Note : Don't forget to import `read_as12` function from test file seen as this is a motion sensor our output will be

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 2.5: camera unit test

```
1 def test_Raspberry_Pi_VR220_out_shape(self):
2     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 using a bash script(see this on page 27) 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 2.6: 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 10^9 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 developemnt si the following on page

2.5 Hardware updates

2.6 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.

2.7 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.

2.8 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.

2.9 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.

2.10 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 3

Results

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

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

3.1 Recorded data from sensors

In this section we will have tables from the following components:

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

3.1.1 DHT22

Results during prototyping

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

Table 3.1: Recorded data from DHT22 on the March 7, 2024

last we tested if our code satisfies our python code after testing the unit test code we updated see the following message

```
-----  
Ran 5 tests in 0.002s  
  
FAILED (failures=1)  
Lost access to message queue
```

Figure 3.1: unit test message for DHT22 module

3.1.2 AS312

Results during prototype

date/time of record	motion detected(yes/no)
---------------------	-------------------------

Table 3.2: Recorded data from AS312 on the March 7, 2024

3.1.3 DFR0026

Results during prototypes

for our first test we got the following table

Date/time of record	lux vaules
---------------------	------------

Table 3.3: Recorded data from DFR0026 on the March 7, 2024

3.1.4 Raspberry Pi VR 220

When testing the Raspberry Pi VR 220

Results during portototypeing

Figure 3.2: A photo from March 7, 2024

3.2 Recorded data from transceiver

3.3 Recorded data from mesh network

Chapter 4

Appendix A

Appendix A

Python Scripts

A.1 Sensor Scripts

A.1.1 DHT22

Listing A.1: DHT22code

```
1  #!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/lib/  
    python3.11  
2  import adafruit_dht  
3  import board  
4  import datetime  
5  import pandas as pd  
6  class DHT22:  
7  ##Set DATA pin to pin 4  
8      def __init__(self):  
9          # self.dhtDevice =adafruit_dht.DHT22(board.D4)  
10         self.dhtDevice =adafruit_dht.DHT11(board.D4)  
11     def Read_DHT22_data(self)-> tuple[float,float,str]:  
12         try:  
13             Humidity=self.dhtDevice.humidity  
14             Temperature=self.dhtDevice.temperature  
15             timestamp =datetime.datetime.now()  
16             timestamp = timestamp.strftime("%Y-%m-%d_%H:%M:%S")  
17             return Temperature, Humidity, timestamp  
18         except RuntimeError as e:  
19             print(f"Error reading sensor: {e}")  
20             return None, None  
21     def write_to_csv(self, filename: str):  
22         temperature, humidity, timestamp = self.Read_DHT22_data()  
23         if temperature is not None and humidity is not None and  
24             timestamp is not None:  
25             data = [(temperature, humidity, timestamp)]  
26             df = pd.DataFrame(data, columns=['Temperature', 'Humidity', 'Timestamp'])  
27             df.to_csv(filename, index=False)  
28         else:  
29             print("Failed to retrieve data from sensor. Data not  
                written to CSV.")  
30     dht_sensor = DHT22()
```

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

A.1.2 AS312

Listing A.2: code for AS312

```
1  #!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/lib/
   python3.11
2  import RPi.GPIO as GPIO
3  import time
4  import datetime
5  import pandas as pd
6  #pin 17
7  class AS312:
8      def __init__(self, pin_number:int):
9          self.pin_number=pin_number
10         self.GPIO=GPIO
11         self.GPIO.setmode(GPIO.BCM)
12         self.GPIO.setup(self.pin_number, GPIO.IN)
13         self.current_state=0
14         self.timestamp=datetime.datetime.now().strftime("
           %Y-%m-%d_%H:%M:%S")
15     def read_state(self)->int:
16         self.current_state =self.GPIO.input(self.
           pin_number)
17         return self.current_state
18     def append_data(self):
19         data={
20             "Motion_Dected": [self.current_state],
21             "Timestamp": [self.timestamp]
22         }
23         df =pd.DataFrame(data)
24         df.to_csv('sensor_data.csv',mode='a' ,index=False
           ,header=False)
25 pir_sensor = AS312(17)
26 try:
27     time.sleep(0.1)
28     current_state =pir_sensor.read_state()
29     timestamp=pir_sensor.timestamp
30     print("GPIO_pin%s_is%s" % (pir_sensor.pin_number,
           current_state))
31     if current_state == 1:
32         print("Motion_dected")
33         pir_sensor.append_data()
34 except KeyboardInterrupt:
35     pass
36 finally:
37     GPIO.cleanup()
```

Appendix B

TDD Script

This section is for All the TDD section of this report in this section will be sharing the TDD of the following:

1. DHT22
2. AS312
- 3.

B.0.1 DHT22

Listing B.1: DHT22 unit test

```
1  from DHT22 import DHT22
2  import board
3  dht22_instance=DHT22()
4  hum,temp,ts=dht22_instance.Read_DHT22_data()
5  class test_project_code(unittest.TestCase):
6      # DHT22
7      def test_DHT22_output_type(self):
8
9          self.assertIsInstance(dht22_instance.Read_DHT22_data,
10                                tuple)
11
12      def test_DHT_22_temp_output_type(self):
13          self.assertIsInstance(temp, (int,float) )
14
15      def test_DHT22_temp_range(self):
16          self.assertGreaterEqual(temp,-30.3)
17          self.assertLessEqual(temp,80.3)
18
19      def test_DHT22_hum_output_type(self):
20          self.assertIsInstance(hum,(int,float))
21
22      def test_DHT22_hum_range(self):
23          self.assertGreaterEqual(hum,0.0)
24          self.assertLessEqual(hum,100.0)
```