

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 25, 2024

Contents

List of Figures

List of Tables

Listings

1.1	sample test intial code	16
1.2	sample test for DHT22	17
1.3	unit test for DFR0026 and MCP3008	17
1.4	unit test for AS312	18
1.5	camera unit test	18
1.6	si powerd SD snippnet	19
A.1	DHT22code	25
A.2	code for AS312	27
A.3	Code for ADC	28
A.4	Code for DFR00026	35
A.5	Code for Camera	36
A.6	Code for memory mangement	37
B.1	DHT22 unit test	39

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 Detection
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	SSSF	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

Methodology

1.1 Introduction

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

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

1.2 Setup of raspberry pi

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

1. once you unpack the pi be sure to connect keyboard mouse and hdmi cable
2. next on a computer you must download the raspberry pi imager and selet the 64 bit recommned os
3. once u have os set 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

1.3 Additional Research

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

1. ADC
2. Radio module

1.3.1 ADC

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

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

1.3.2 Radio module

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

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

Do we need a radio standard?

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

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

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

1. LoRa
2. Zigbee

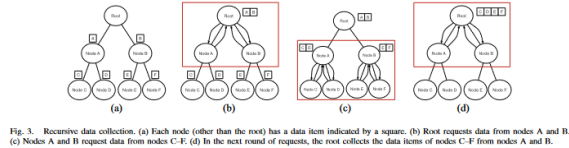


Figure 1.1: protocol Wu used(wu.lie et.al,2023:16705)

3. Sigfox

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

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

for zigbee most of the application that were found where indoor's i.e industial setting

1.4 Software Module Development

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

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

1.4.1 Sensors

This Section will discuss the following:

1. DHT22
2. AS312

To see the light sensor look on page ??

DHT22

For this section we used the following libraries:

```

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

```

This uses the library from this link

1. we define the our class

```

1     class DHT22:
2         ##Set DATA pin to pin 4
3         def __init__(self):
4             """this will setup the data pin for DHT2"""
5             # self.dhtDevice =adafruit_dht.DHT22(board.D4)
6             self.dhtDevice =adafruit_dht.DHT11(board.D4)
7             self.humidity=self.dhtDevice.humidity
8             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.

```

1     def Read_DHT22_data(self)-> tuple[float,float,str]:
2         """This will setup a DHT instance and return the
3         data from the sensor"""
4         try:
5             return self.temperature,self.humidity
6         except RuntimeError as e:
7             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:

```

1         if __name__ == "__main__":
2             DHT22()

```

AS312

For this we import the following libraries:

```

1     #!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/
2     lib/python3.11
3     import RPi.GPIO as GPIO
4     import time

```

1. next we set up our variables in the class

```

1     class AS312:
2         def __init__(self):
3             "connect the AS312 to pin 17"
4             self.pin_number=17
5             self.GPIO=GPIO
6             self.GPIO.setmode(GPIO.BCM)
7             self.GPIO.setup(self.pin_number,GPIO.IN)
8             self.current_state=0

```

This sets current state as 0

2. next we detect movement

```

1         def read_state(self)->bool:
2             time.sleep(0.1)
3             self.current_state =bool(self.GPIO.input(self.
                pin_number))
4             return self.current_state

```

1.4.2 DFR0026

From the repository `DFRobot_ADS1115` we do the following :

import the libraries

```

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

```

Next we define our variables

```

1         class DFR0026():
2             def __init__(self):
3                 self.ADS1115_REG_CONFIG_PGA_6_144V           = 0x00
4                     # 6.144V range = Gain 2/3
5                 self.ADS1115_REG_CONFIG_PGA_4_096V           = 0x02
6                     # 4.096V range = Gain 1
7                 self.ADS1115_REG_CONFIG_PGA_2_048V           = 0x04
8                     # 2.048V range = Gain 2 (default)
9                 self.ADS1115_REG_CONFIG_PGA_1_024V           = 0x06
10                    # 1.024V range = Gain 4
11                 self.ADS1115_REG_CONFIG_PGA_0_512V           = 0x08
12                    # 0.512V range = Gain 8
13                 self.ADS1115_REG_CONFIG_PGA_0_256V           = 0x0A
14                    # 0.256V range = Gain 16
15                 self.ads1115 = ADS1115()
16                 self.ads1115.set_addr_ADS1115(0x48)
17                 self.ads1115.set_gain(self.
                    ADS1115_REG_CONFIG_PGA_6_144V)
18                 self.adc_channel=0

```

This configures all the pins and set the associative gain

Next read the analogue channel

```

1         def read_voltage(self):
2             return self.ads1115.read_voltage(self.adc_channel
                )

```

1.4.3 Camera

Here are the steps for module development of the Camera:

1. install the following libraries:

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

2. we define our class variables

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

3. make the function for taking a picture

```
1      def take_pic(self)-> str:  
2          """this will take a picture from camera"""  
3          self.camera.configure(self.camera_config)  
4          self.camera.start_preview(Preview.QTGL)  
5          self.camera.start()  
6          sleep(self.timeamount)  
7          self.camera.capture_file(self.fname)  
8          return self.fname
```

1.4.4 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:

```
1      #!/home/mistaherd/Documents/Github/  
      meshnetwork_in_forest/env/lib/python3.11  
2      import pandas as pd  
3      from DHT22 import DHT22  
4      from AS312 import AS312  
5      from DFR0026 import DFR0026  
6      import glob  
7      import re  
8      import subprocess
```

2. define our class sensors

```
1      class sensor_data:  
2          def __init__(self):  
3              self.dht22 = DHT22()  
4              self.humidity, self.temperature = self.dht22.  
                  Read_DHT22_data()  
5              self.AS312 = AS312(17)  
6              self.motion_detected = AS312.read_state()  
7              self.DF0026 = DFR0026()  
8              self.light_value = self.DF0026.Read_data()  
9              self.fname = "sensor_data.csv"
```

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

```
1          def write_append_csv(self):  
2              data = { "Timestamp" : self.timestamp,  
3                      "Temperature(oc)" : self.  
                          Temperature,  
4                      "Humidity(%)" : self.humidity,  
5                      "Light(lux)" : self.light_value,  
6                      "Motion_Detected": self.  
                          motion_detected  
7                      }  
8              df = pd.DataFrame(data)  
9              if glob.glob(self.fname):  
10                 df.to_csv(self.fname, mode='a' ,  
                             index=False, header=False)  
11             else:  
12                 df.to_csv(self.fname, mode='w' ,  
                             index=False)
```

4. Next we define our variables for testing memory

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

5. next we check our memory

```
1     def check_memory(self):
2         try:
3             if re.search(self.regex,self.output_bash)
4                 :
5                     value,unit=match.group(0).split()
6                     try:
7                         return float(value)*self.units[
8                             unit]
9                     except KeyError:
10                        raise ValueError(f"unknown unit: {
11                            unit}")
12
13         except subprocess.CalledProcessError as e:
14             raise ValueError(f"Error running script:{
15                 e.output}")
```

6. we then make an error if its using 20 percent memory

```
1     def error_check(self):
2         mem=self.check_memory()
3         max=32*10e9
4         if mem >= 0.2* max:
5             raise MemoryError("memory on pi is about
6                 to be used up")
```

7. to make sure our class run from another python file

```
1     if __name__=="__main__":
2         sensor_data()
3         Memory_tester()
```

1.4.5 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 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°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 stablsh we must make a unit that will do the following:

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

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

Listing 1.1: sample test intial code

```
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 1.2: 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.assertGreaterEqual(hum,0.0)
14        self.assertLessEqual(hum,100.0)

```

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

DFR0026 & MCP3008

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

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

Listing 1.3: unit test for DFR0026 and MCP3008

```

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.assertLessEqual(5.0000000)
8         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 snippet :

Listing 1.4: unit test for AS312

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

Note : Don't forget to import `read_as312` function from `test_file` seen as this is a motion sensor output

Raspberry Pi VR 220 Camera

according to the data sheet ? we the resolution to it uses is 1080p50 which is 1920x1080p so our tests will have to incorporate the following:

1. Test the output shape if open cv is gonna be used
 - (a) test the amount of elements in the 3 dimensional array
2. test the file type is png

this would lead me to the following code snippet.

Listing 1.5: camera unit test

```
1 def test_Raspberry_Pi_VR220_out_shape(self):
2     self.assertEqual(Read_Raspberry_Pi_VR220.shape, (1920, 1080, 3))
```

this function check the pixel count or resolution

Li-polymer Battery HAT

memory modules

in this section will discuss the following:

1. silicon power 32GB
2. Turbo 1GB

for this i will use using 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 first we import the function

Listing 1.6: si powerd SD snippnet

```
1      import unittest
2      from protest import Read_DHT22, Read_MCP3008,
          Read_AS312, Read_Raspberry_PiVR220,
          Read_Memory_module
3
4      def Test_memory_module_turbo_1GB_size(self):
5          #testing turbo 1GB
6          self.assertEqual(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 which output that will be between 1GB and 0

2. silicon power 32GB

MM2 Series 900 MHz

Unit test iterations

the first iterations as seen here have the following problems for the sensors:

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

conclusion

The initial draft code for the test development is the following on page

1.5 Data Analysis Methods

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

1.6 Ethical Considerations

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

1.7 Validity and Reliability

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

1.8 Limitations and Delimitations

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

1.9 Timeline

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

Chapter 2

Results

In 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

2.1 Recorded data from sensors

in this section will have tables from the following components:

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

2.1.1 DHT22

Results during prototypeing

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

Table 2.1: Recorded data from DHT22 on the March 25, 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)  
ImportError: cannot import name 'DHT22' from 'DHT22' (DHT22.py)
```

Figure 2.1: unit test message for DHT22 module

2.1.2 AS312

Results during prototype

date/time of record	motion detected(yes/no)
2024-03-25 ₁₅ – 02 – 57	False
2024-03-25 ₁₅ – 04 – 37	True

Table 2.2: Recorded data from AS312 on the March 25, 2024

2.1.3 DFR0026

Results during prototypes

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

2.1.4 Raspberry Pi VR 220

When testing the Raspberry Pi VR 220

Results during portotyping



Figure 2.2: A photo from 25th of march 2024

2.2 Recorded data from transceiver

2.3 Recorded data from mesh network

Chapter 3

Appendix A

Appendix A

Python Scripts

A.0.1 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 timestamp is not None:  
24             data = [(temperature, humidity, timestamp)]  
25             df = pd.DataFrame(data, columns=['Temperature', '  
                Humidity', 'Timestamp'])  
26             df.to_csv(filename, index=False)  
27         else:  
28             print("Failed_to_retrieve_data_from_sensor._Data_  
                not_written_to_CSV.")
```



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

A.0.2 AS312

Listing A.2: code for AS312

```
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_Dectected": [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_dectected")  
33         pir_sensor.append_data()  
34 except KeyboardInterrupt:  
35     pass  
36 finally:  
37     GPIO.cleanup()
```

A.0.3 ADC

Listing A.3: Code for ADC

```
1      #!/home/mistaherd/Documents/Github/meshnetwork_in_forest/  
      env/lib/python3.11  
2  '''!  
3      @file DFRobot_ADS1115.py  
4      @brief Provides an Raspberry pi library to read ADS1115  
          data over I2C. Use this library to read analog voltage  
          values.  
5      @copyright Copyright (c) 2010 DFRobot Co.Ltd (http://www.  
          dfrobot.com)  
6      @license The MIT License (MIT)  
7      @author [luoyufeng](yufeng.luo@dfrobot.com)  
8      @version V1.0  
9      @date 2019-06-19  
10     @url https://github.com/DFRobot/DFRobot_ADS1115  
11  '''  
12  
13  import smbus  
14  import time  
15  
16  ## Get I2C bus  
17  bus = smbus.SMBus(1)  
18  
19  ## I2C address of the device  
20  ADS1115_IIC_ADDRESS0 = 0x48  
21  ADS1115_IIC_ADDRESS1 = 0x49  
22  
23  ## ADS1115 Register Map  
24  ## Conversion register  
25  ADS1115_REG_POINTER_CONVERT = 0x00  
26  ## Configuration register  
27  ADS1115_REG_POINTER_CONFIG = 0x01  
28  ## Lo_thresh register  
29  ADS1115_REG_POINTER_LOWTHRESH = 0x02  
30  ## Hi_thresh register  
31  ADS1115_REG_POINTER_HITHRESH = 0x03  
32  
33  ## ADS1115 Configuration Register  
34  ## No effect  
35  ADS1115_REG_CONFIG_OS_NOEFFECT = 0x00  
36  ## Begin a single conversion  
37  ADS1115_REG_CONFIG_OS_SINGLE = 0x80  
38  ## Differential P = AIN0, N = AIN1 (default)  
39  ADS1115_REG_CONFIG_MUX_DIFF_0_1 = 0x00  
40  ## Differential P = AIN0, N = AIN3  
41  ADS1115_REG_CONFIG_MUX_DIFF_0_3 = 0x10  
42  ## Differential P = AIN1, N = AIN3  
43  ADS1115_REG_CONFIG_MUX_DIFF_1_3 = 0x20  
44  ## Differential P = AIN2, N = AIN3
```

```

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

```

```

96  ## Latching comparator
97  ADS1115_REG_CONFIG_CLAT_LATCH          = 0x04
98  ## Assert ALERT/RDY after one conversions
99  ADS1115_REG_CONFIG_CQUE_1CONV          = 0x00
100 ## Assert ALERT/RDY after two conversions
101 ADS1115_REG_CONFIG_CQUE_2CONV          = 0x01
102 ## Assert ALERT/RDY after four conversions
103 ADS1115_REG_CONFIG_CQUE_4CONV          = 0x02
104 ## Disable the comparator and put ALERT/RDY in high state (
    default)
105 ADS1115_REG_CONFIG_CQUE_NONE           = 0x03
106
107 mygain=0x02
108 coefficient=0.125
109 addr_G=ADS1115_IIC_ADDRESS0
110 class ADS1115():
111     def set_gain(self,gain):
112         '''!
113             @brief Sets the gain and input voltage
114                 range.
115             @param gain This configures the
116                 programmable gain amplifier
117             @n ADS1115_REG_CONFIG_PGA_6_144V      = 0x00
118                 # 6.144V range = Gain 2/3
119             @n ADS1115_REG_CONFIG_PGA_4_096V      = 0x02
120                 # 4.096V range = Gain 1
121             @n ADS1115_REG_CONFIG_PGA_2_048V      = 0x04
122                 # 2.048V range = Gain 2
123             @n default:
124             @n ADS1115_REG_CONFIG_PGA_1_024V      = 0x06
125                 # 1.024V range = Gain 4
126             @n ADS1115_REG_CONFIG_PGA_0_512V      = 0x08
127                 # 0.512V range = Gain 8
128             @n ADS1115_REG_CONFIG_PGA_0_256V      = 0x0A
129                 # 0.256V range = Gain 16
130         '''
131
132         global mygain
133         global coefficient
134         mygain=gain
135         if mygain == ADS1115_REG_CONFIG_PGA_6_144V:
136             coefficient = 0.1875
137         elif mygain == ADS1115_REG_CONFIG_PGA_4_096V:
138             coefficient = 0.125
139         elif mygain == ADS1115_REG_CONFIG_PGA_2_048V:
140             coefficient = 0.0625
141         elif mygain == ADS1115_REG_CONFIG_PGA_1_024V:
142             coefficient = 0.03125
143         elif mygain == ADS1115_REG_CONFIG_PGA_0_512V:
144             coefficient = 0.015625
145         elif mygain == ADS1115_REG_CONFIG_PGA_0_256V:
146             :

```

```

137         coefficient = 0.0078125
138     else:
139         coefficient = 0.125
140     def set_addr_ADS1115(self, addr):
141         '''!
142         @brief Sets the IIC address.
143         @param addr 7 bits I2C address, the range
144             is 1~127.
145         '''
146         global addr_G
147         addr_G=addr
148     def set_channel(self, channel):
149         '''!
150         @brief Select the Channel user want to use
151             from 0-3.
152         @param channel the Channel: 0-3
153         @n For Single-ended Output:
154         @n 0 : AINP = AIN0 and AINN = GND
155         @n 1 : AINP = AIN1 and AINN = GND
156         @n 2 : AINP = AIN2 and AINN = GND
157         @n 3 : AINP = AIN3 and AINN = GND
158         @n For Differential Output:
159         @n 0 : AINP = AIN0 and AINN = AIN1
160         @n 1 : AINP = AIN0 and AINN = AIN3
161         @n 2 : AINP = AIN1 and AINN = AIN3
162         @n 3 : AINP = AIN2 and AINN = AIN3
163         @return channel
164         '''
165         global mygain
166         self.channel = channel
167         while self.channel > 3 :
168             self.channel = 0
169
170         return self.channel
171
172     def set_single(self):
173         '''!
174         @brief Configuration using a single read.
175         '''
176         global addr_G
177         if self.channel == 0:
178             CONFIG_REG = [
179                 ADS1115_REG_CONFIG_OS_SINGLE |
180                 ADS1115_REG_CONFIG_MUX_SINGLE_0 |
181                 mygain |
182                 ADS1115_REG_CONFIG_MODE_CONTIN,
183                 ADS1115_REG_CONFIG_DR_128SPS |
184                 ADS1115_REG_CONFIG_CQUE_NONE]
185         elif self.channel == 1:
186             CONFIG_REG = [
187                 ADS1115_REG_CONFIG_OS_SINGLE |

```

```

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

```

```

197         ADS1115_REG_CONFIG_MODE_CONTIN,
198         ADS1115_REG_CONFIG_DR_128SPS |
199         ADS1115_REG_CONFIG_CQUE_NONE]
200     elif self.channel == 3:
201         CONFIG_REG = [
202             ADS1115_REG_CONFIG_OS_SINGLE |
203             ADS1115_REG_CONFIG_MUX_DIFF_2_3 |
204             mygain |
205             ADS1115_REG_CONFIG_MODE_CONTIN,
206             ADS1115_REG_CONFIG_DR_128SPS |
207             ADS1115_REG_CONFIG_CQUE_NONE]
208
209     bus.write_i2c_block_data(addr_G,
210                               ADS1115_REG_POINTER_CONFIG, CONFIG_REG)
211
212     def read_value(self):
213         '''!
214         @brief Read ADC value.
215         @return raw adc
216         '''
217         global coefficient
218         global addr_G
219         data = bus.read_i2c_block_data(addr_G,
220                                         ADS1115_REG_POINTER_CONVERT, 2)
221
222         # Convert the data
223         raw_adc = data[0] * 256 + data[1]
224
225         if raw_adc > 32767:
226             raw_adc -= 65535
227         raw_adc = int(float(raw_adc)*coefficient)
228         return {'r' : raw_adc}
229
230     def read_voltage(self, channel):
231         '''!
232         @brief Reads the voltage of the specified
233             channel.
234         @param channel the Channel: 0-3
235         @n For Single-ended Output:
236         @n 0 : AINP = AIN0 and AINN = GND
237         @n 1 : AINP = AIN1 and AINN = GND
238         @n 2 : AINP = AIN2 and AINN = GND
239         @n 3 : AINP = AIN3 and AINN = GND
240         @n For Differential Output:
241         @n 0 : AINP = AIN0 and AINN = AIN1
242         @n 1 : AINP = AIN0 and AINN = AIN3
243         @n 2 : AINP = AIN1 and AINN = AIN3
244         @n 3 : AINP = AIN2 and AINN = AIN3
245         @return Voltage
246         '''
247         self.set_channel(channel)

```



```

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

```

A.0.4 DFR0026

Listing A.4: Code for DFR00026

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

A.0.5 Camera

Listing A.5: Code for Camera

```
1  #!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/  
   lib/python3.11  
2  from picamera import PiCamera  
3  from time import sleep  
4  from datetime import datetime  
5  class Raspberry_Pi_VR_220:  
6      def __init__(self):  
7          """setup an instan for the camera"""  
8          self.timestamp=datetime.now().strftime("%Y-%m-%d_%H:%  
           M:%S")  
9          self.fname = '/home/mistaherd/Documents/Github/  
           meshnetwork_in_forest/{}.png'.format(self.  
           timestamp)  
10         self.camera=PiCamera()  
11         self.timeamount=2  
12     def take_pic(self)-> str:  
13         """this will take a picture from camera"""  
14         self.camera.start_preview()  
15         sleep(self.timeamount)  
16         self.camera.capture(self.fname)  
17         self.stop_preview()  
18         return self.fname  
19 camera=Raspberry_Pi_VR_220()  
20 picture=camera.take_pic()
```

A.0.6 Memory mangement

Listing A.6: Code for memory mangement

```
1  #!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/
   lib/python3.11
2  import pandas as pd
3  # from DHT22 import DHT22
4
5  # from AS312 import AS312
6  # from MCP3008 import DF0026
7  import pandas as pd
8  import glob
9  import re
10 import subprocess
11 class sensor_data:
12     def __init__(self):
13         self.dht22 = DHT22()
14         self.humidity,self.temperature,self.timestamp
           =self.dht22.Read_DHT22_data()
15         self.AS312=AS312(17)
16         self.motion_dected =AS312.read_state()
17         self.DF0026 =DF0026()
18         self.light_value=self.DF0026.Read_data()
19         self.fname="sensor_data.csv"
20     def write_append_csv(self):
21         data = { "Timestamp" : self.timestamp,
22                 "Temperature(oc)" : self.Temperature,
23                 "Humidity(%)" : self.humidity,
24                 "Light(lux)" :self.light_value,
25                 "Motion_Dected": self.motion_dected
26                 }
27         df = pd.DataFrame(data)
28         if glob.glob(self.fname):
29             df.to_csv(self.fname,mode='a' ,index=
               False,header=False)
30         else:
31             df.to_csv(self.fname,mode='w' ,index=
               False)
32 class Memory_tester():
33     def __init__(self):
34         self.units={"K":10e3,"M": 10e6,"G":10e9}
35         self.regex ="\\d{4}\\.[0-9]{1,3}[K,M,G]"
36         self.fname="./bash_scrpits/memorytest.sh"
37         self.output_bash=subprocess.check_output(["
           bash",self.fname],universal_newlines=True)
38     def check_memory(self):
39         try:
40             if re.search(self.regex,self.
               output_bash):
41                 value,unit=match.group(0).
                   split()
```

```

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

```

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.
10                               Read_DHT22_data, 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)
```