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 13, 2024

Contents

List of Figures

List of Tables

Listings

Glossary

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

Chapter 1

Methodology

1.1 Introduction

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

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

1.2 Setup of raspberry pi

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

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

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

   this will downlaod the nessary 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 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

1.4 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

1.4.1 Sensors

DHT22

AS312

1.4.2 ADC

1.4.3 Camera

1.4.4 Memory Mangement

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^oc 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^oc for the lowest temp we have 6 bits to represent - 40 due to 2s complement so lowest will be -40.3^oC so with that that stablish we must make a unit that will do the following:

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

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

Listing 1.1: sample test intial code

```python
import unittest
from protest import Read_DHT22
class test_project_code(unittest.TestCase):
    def test_DHT_22_temp_output_type(self):
        self.assertIsInstance(Read_DHT22, float)
    def test_DHT22_temp_range(self):
        self.assertGreaterEqual(Read_DHT22,-30.3)
        self.assertLessEqual(Read_DHT22,80.3)
```

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

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

this lead to the following code

Listing 1.2: sample test for DHT22

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

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

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

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

DFR0026 & MCP3008

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

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

Listing 1.3: unit test for DFR0026 and MCP3008

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

this code is in the same in theres of limits

AS312

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

1. test for type is boolean

we can now add to the snipppet :

Listing 1.4: unit test for AS312

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

Note : Don't forget to import read$_as12functionfromtestfileseenasthhisisamotionsensorourououtwillbe$

Raspberry Pi VR 220 Camera

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

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

this would lead me to the following code snippet.

Listing 1.5: camera unit test

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

this function check the pixeal count or resoulkation

Li-polymer Battery HAT

memory moduldes

in this setion will dicuss the following:

1. **silicon power 32GB**
2. **Turbo 1GB**

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

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

   Listing 1.6: si powerd SD snippnet

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

           def Test_memory_module_turbo_1GB_size(self):
               #testing  turbo 1GB
               self.assertLessEqual(Read_Memory_module,1e9)
               self.assertGreaterEqual(Read_Memory_module,0)
   ```

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

2. **silicon power 32GB**

MM2 Series 900 MHz

Unit test iterations

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

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

conculsion

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

1.5 Hardware updates

1.6 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.

1.7 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.

1.8 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.

1.9 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.

1.10 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 2

Results

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

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

2.1 Recorded data from sensors

in this section will have tables from the following componets:

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

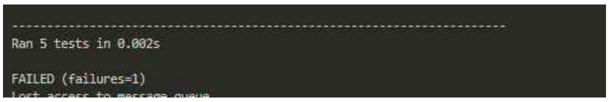
2.1.1 DHT22

Results during protypeing

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

Table 2.1: Recorded data from DHT22 on the March 13, 2024

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



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

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

Figure 2.1: unit test message for DHT22 module

2.1.2 AS312

Results during protype

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

Table 2.2: Recorded data from AS312 on the March 13, 2024

2.1.3 DFR0026

Results during protypes

for our first test we got the following table

Date/time of record	lux vaules

Table 2.3: Recorded data from DFR0026 on the March 13, 2024

2.1.4 Raspberry Pi VR 220

When testing the Raspberry Pi VR 220

Results during portotypeing

Figure 2.2: A photo from March 13, 2024

2.2 Recorded data from transciver

2.3 Recorded data from mesh network

Chapter 3

Appendix A

Appendix A

Python Scripts

A.1 Sensor Scripts

A.1.1 DHT22

Listing A.1: DHT22code

```
#!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/lib/python3.11
import adafruit_dht
import board
import datetime
import pandas as pd
class DHT22:
##Set DATA pin to pin 4
    def __init__(self):
        # self.dhtDevice =adafruit_dht.DHT22(board.D4)
        self.dhtDevice =adafruit_dht.DHT11(board.D4)
    def Read_DHT22_data(self)-> tuple[float,float,str]:
        try:
            Humidity=self.dhtDevice.humidity
            Temperature=self.dhtDevice.temperature
            timestamp =datetime.datetime.now()
            timestamp = timestamp.strftime("%Y-%m-%d␣%H:%M:%S")
            return Temperature,Humidity,timestamp
        except RuntimeError as e:
            print(f"Error␣reading␣sensor:␣{e}")
            return None, None
    def write_to_csv(self,filename:str):
        temperature, humidity, timestamp = self.Read_DHT22_data()
        if temperature is not None and humidity is not None and timestamp is not None:
            data = [(temperature, humidity, timestamp)]
            df = pd.DataFrame(data, columns=['Temperature', 'Humidity', 'Timestamp'])
            df.to_csv(filename, index=False)
        else:
            print("Failed␣to␣retrieve␣data␣from␣sensor.␣Data␣not␣written␣to␣CSV.")
dht_sensor = DHT22()
```

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

A.1.2 AS312

Listing A.2: code for AS312

```
#!/home/mistaherd/Documents/Github/meshnetwork_in_forest/env/lib/python3.11
import RPi.GPIO as GPIO
import time
import datetime
import pandas as pd
#pin 17
class AS312:
        def __init__(self,pin_number:int):
                self.pin_number=pin_number
                self.GPIO=GPIO
                self.GPIO.setmode(GPIO.BCM)
                self.GPIO.setup(self.pin_number,GPIO.IN)
                self.current_state=0
                self.timestamp=datetime.datetime.now().strftime("%Y-%m-%d␣%H:%M:%S")
        def read_state(self)->int:
                self.current_state =self.GPIO.input(self.pin_number)
                return self.current_state
        def append_data(self):
                data={
                        "Motion␣Dectected": [self.current_state],
                        "Timestamp": [self.timestamp]
                }
                df =pd.DataFrame(data)
                df.to_csv(’sensor_data.csv’,mode=’a’ ,index=False,header=False)
pir_sensor = AS312(17)
try:
        time.sleep(0.1)
        current_state =pir_sensor.read_state()
        timestamp=pir_sensor.timestamp
        print("GPIO␣pin␣%s␣is␣%s" % (pir_sensor.pin_number,current_state))
        if current_state == 1:
                print("Motion␣dectected")
        pir_sensor.append_data()
except KeyboardInterrupt:
        pass
finally:
        GPIO.cleanup()
```

Appendix B

TDD Script

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

1. **DHT22**
2. **AS312**
3.

B.0.1 DHT22

Listing B.1: DHT22 unit test

```python
from DHT22 import DHT22
import board
dht22_instance=DHT22()
hum,temp,ts=dht22_instance.Read_DHT22_data()
class test_project_code(unittest.TestCase):
    # DHT22
    def test_DHT22_output_type(self):

        self.assertIsInstance(dht22_instance.Read_DHT22_data, tuple)

    def test_DHT_22_temp_output_type(self):
        self.assertIsInstance(temp, (int,float) )

    def test_DHT22_temp_range(self):
        self.assertGreaterEqual(temp,-30.3)
        self.assertLessEqual(temp,80.3)

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

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