Mini-project A

MSMJOS003, MNCXOL005

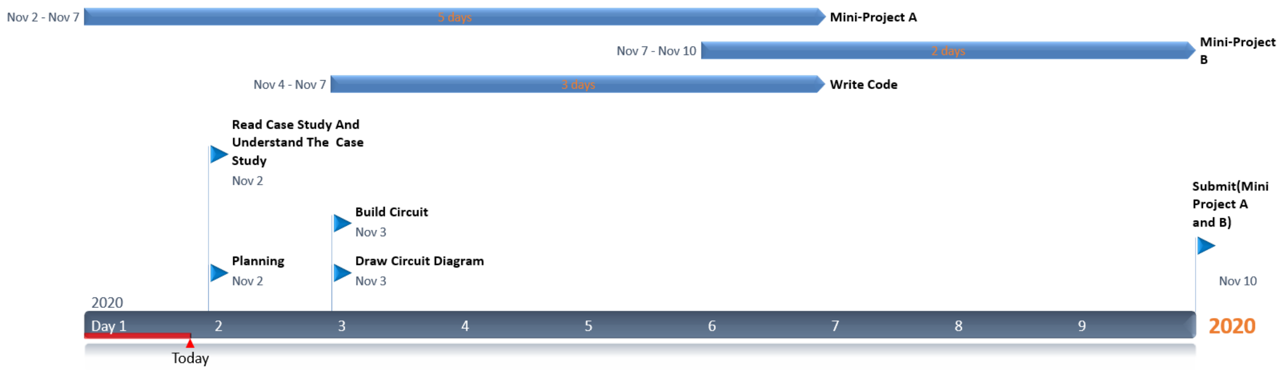
EEE3096S 2020

1 Introduction

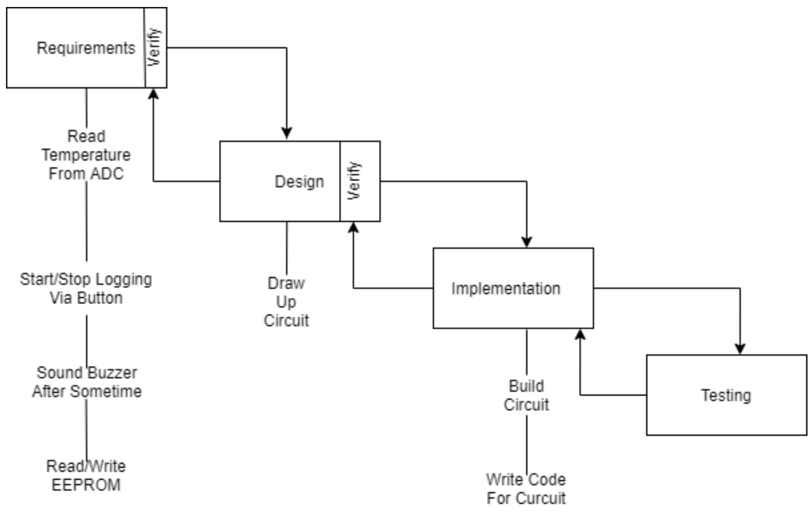
In this project, we designed and built an environment logger system for my uncle to assist him with taking care of his pet bearded dragon. The system monitors the temperature in the terrarium. To plan and design the system I used various models such as a gantt chat(see diagram under planning), UML use case diagram, waterfall diagram and flow chat which details how the system operates. These models helped me in allocating our time, to plan our design before actually building it.

2 Planning

The project was suppose to take about 10 days as shown in the diagram but due to the challenges we faced doing the project it ended up taking longer than expected. (See below)



The waterfall diagram below shows the process we followed from looking at our requirements to completing the project.



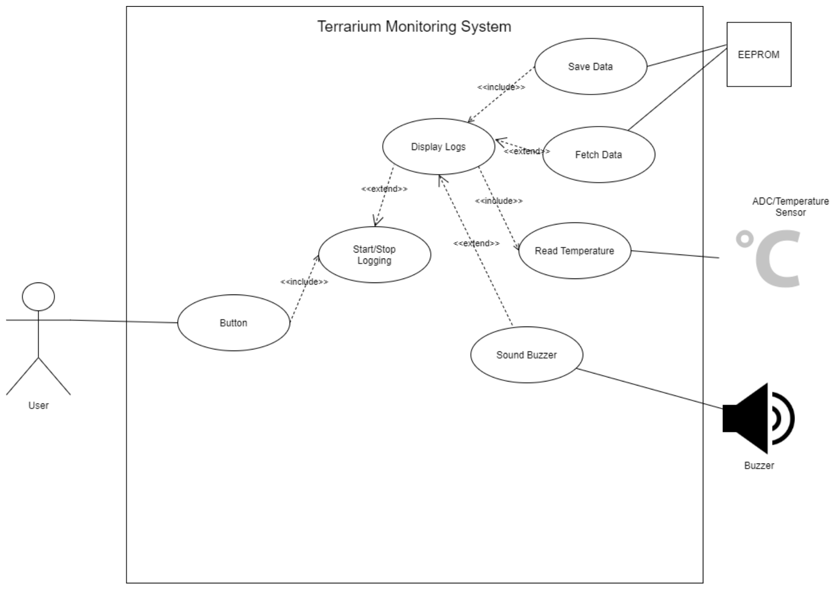
3 Methology

Requirements:

- Breadboard
- MCP9700A Temperature sensor
- MCP3008 ADC
- Raspberry Pi Zero
- Button
- Resistor
- Transistor
- Buzzer
- EEPROM chip

UML Use Case:

The UML diagram below show how the user interact with our system and also how various components in our system interact with each other



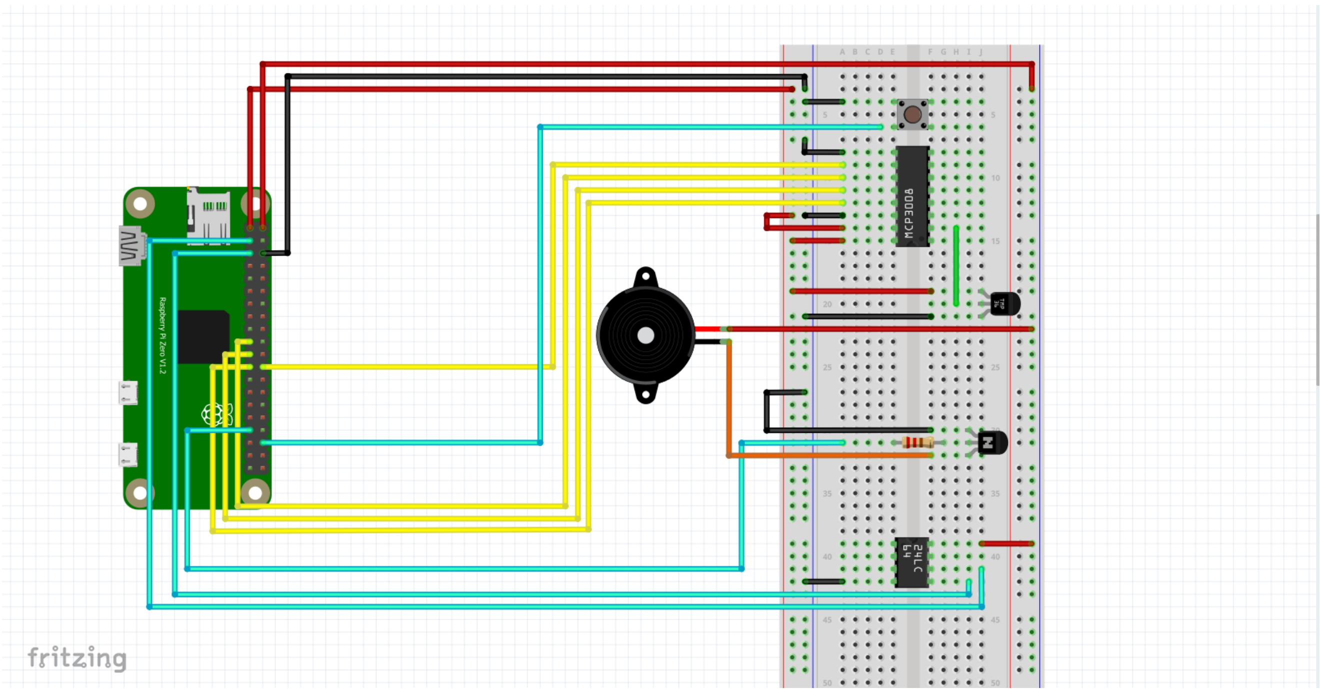
Connecting the circuit:

- Please see circuit diagram under Specification and Design

4 Specification and Design

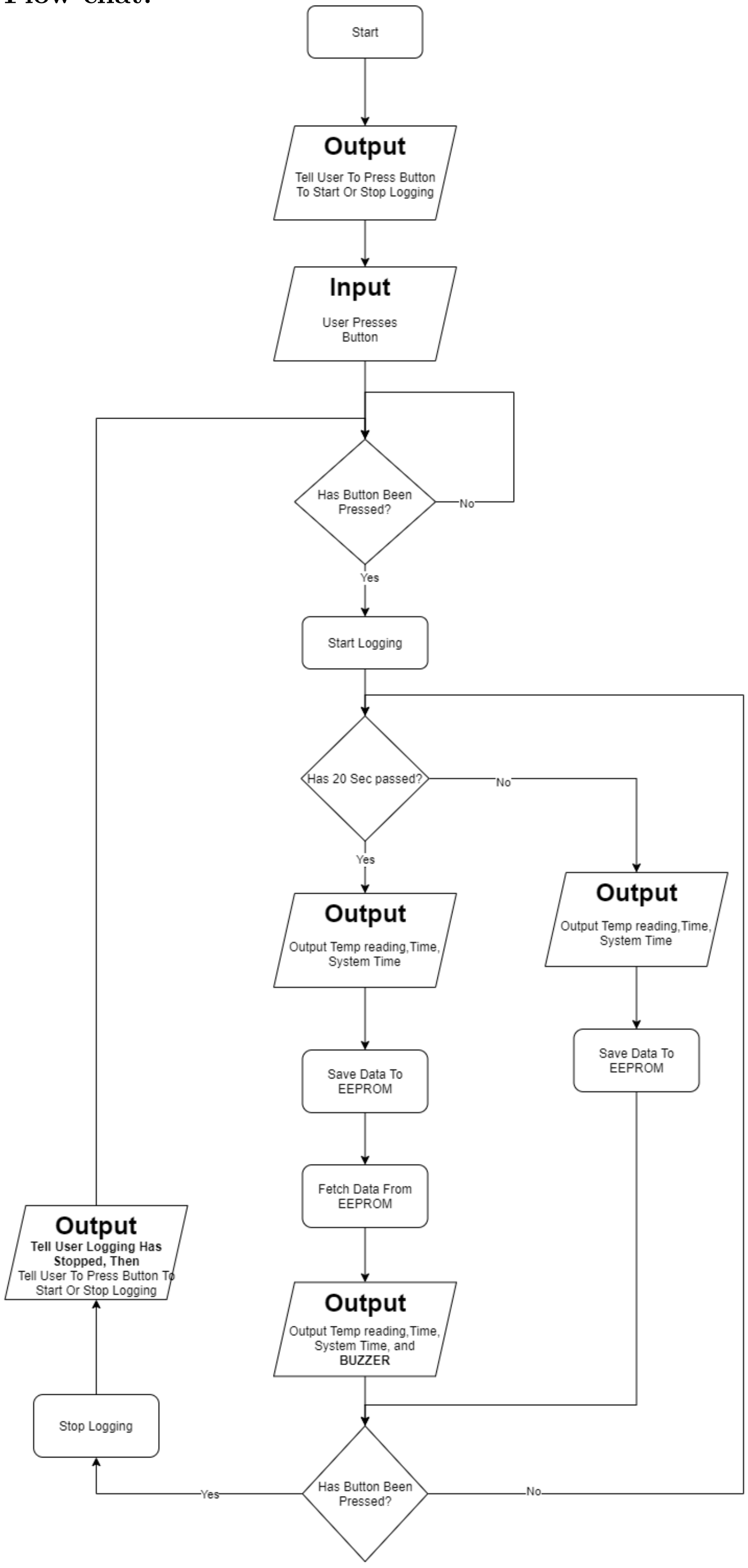
The system reads the temperature in the terrarium using the MCP9700A temperature sensor. Then use the MCP3008 ADC which is connected to the raspberry pi which converts the analog signal received from the temperature sensor into a digital signal. The system uses threads to read the temperature from MCP3008 ADC and the main thread to display(log) the information on the screen. The button is used to start and stop logging(monitoring) the terrarium. When the button is pressed to start logging, the system starts to read the temperature of the terrarium after 5 seconds and saving this data(includes the current time, system time and temperature) in EEPROM. This information is also being displayed on the screen while the system is running. For every 20 seconds that have passed while the system is running, we fetch and display the last saved data then we sound the buzzer.(see the system design/circuit below)

Circuit diagram for the system:


fritzing
Raspberry Pi Zero V1.2
MCP3008
24LC64

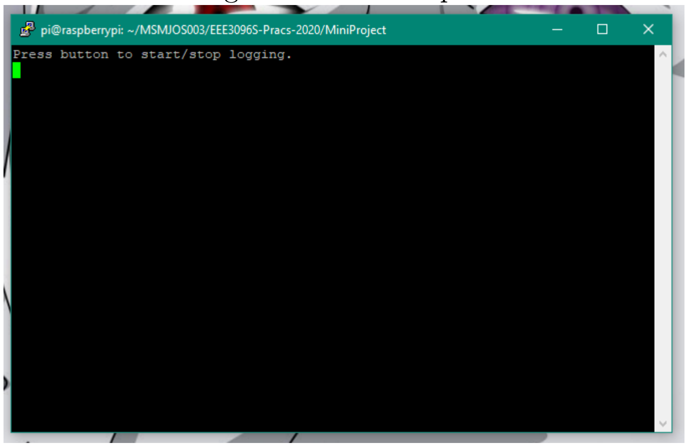
The flow chat below shows the details on how our the system operates.

Flow chat:


Start
Output
Tell User To Press Button To Start Or Stop Logging
Input
User Presses Button
Has Button Been Pressed?
No
Yes
Start Logging
Has 20 Sec passed?
No
Yes
Output
Output Temp reading,Time, System Time
Save Data To EEPROM
Output
Output Temp reading,Time, System Time
Save Data To EEPROM
Fetch Data From EEPROM
Output
Output Temp reading,Time, System Time, and BUZZER
Output
Tell User Logging Has Stopped, Then
Tell User To Press Button To Start Or Stop Logging
Stop Logging
Has Button Been Pressed?
Yes
No

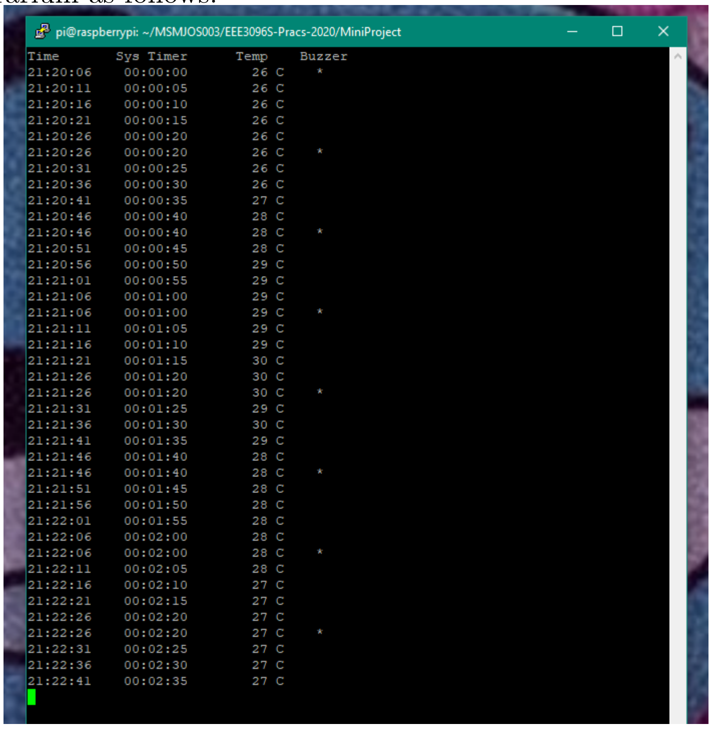
5 Implementation

When the program is run for the first time it will send out a message on the screen instructing the user to press the button to start and stop logging of data.



```
Press button to start/stop logging.
```

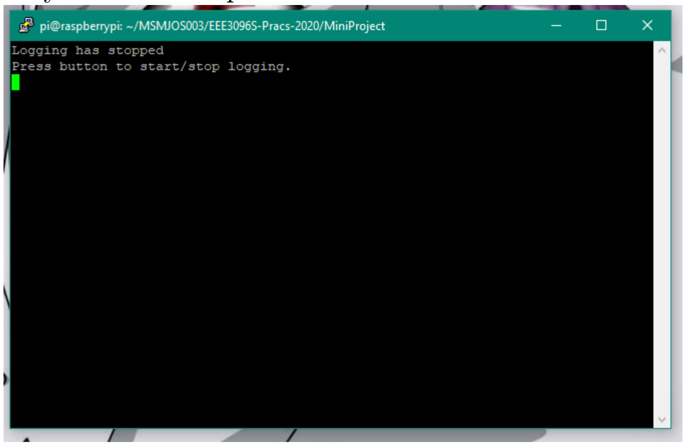
Once the user presses the button the system will start to log the temperature of the terrarium as follows.



```
Time        Sys Timer     Temp   Buzzer
21:20:06     00:00:00       26 C    *
21:20:11     00:00:05       26 C
21:20:16     00:00:10       26 C
21:20:21     00:00:15       26 C
21:20:26     00:00:20       26 C
21:20:26     00:00:20       26 C    *
21:20:31     00:00:25       26 C
21:20:36     00:00:30       26 C
21:20:41     00:00:35       27 C
21:20:46     00:00:40       28 C
21:20:46     00:00:40       28 C    *
21:20:51     00:00:45       28 C
21:20:56     00:00:50       29 C
21:21:01     00:00:55       29 C
21:21:06     00:01:00       29 C
21:21:06     00:01:00       29 C    *
21:21:11     00:01:05       29 C
21:21:16     00:01:10       29 C
21:21:21     00:01:15       30 C
21:21:26     00:01:20       30 C
21:21:26     00:01:20       30 C    *
21:21:31     00:01:25       29 C
21:21:36     00:01:30       30 C
21:21:41     00:01:35       29 C
21:21:46     00:01:40       28 C
21:21:46     00:01:40       28 C    *
21:21:51     00:01:45       28 C
21:21:56     00:01:50       28 C
21:22:01     00:01:55       28 C
21:22:06     00:02:00       28 C
21:22:06     00:02:00       28 C    *
21:22:11     00:02:05       28 C
21:22:16     00:02:10       27 C
21:22:21     00:02:15       27 C
21:22:26     00:02:20       27 C
21:22:26     00:02:20       27 C    *
21:22:31     00:02:25       27 C
21:22:36     00:02:30       27 C
21:22:41     00:02:35       27 C
```

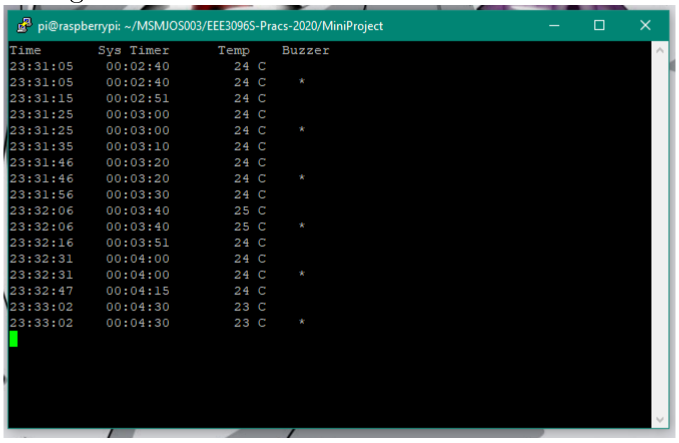
In the figure above you see the temperature rising from 26 degrees C to 30 degrees C then drops again. This is because we increased the room temperature by turning on the fan heater and hence the increase of the temperature.

When the user decides to stop logging the data by pressing the button again, the following screen will be displayed letting them know that logging has stopped and to start logging again they will need to press the button.



```
pi@raspberrypi: ~/MSMJOS003/EEE3096S-Pracs-2020/MiniProject
Logging has stopped
Press button to start/stop logging.
```

Then when the user presses the button the system continues to logging the data as shown in the figure below.



```
pi@raspberrypi: ~/MSMJOS003/EEE3096S-Pracs-2020/MiniProject
Time        Sys Timer      Temp    Buzzer
23:31:05     00:02:40        24 C
23:31:05     00:02:40        24 C     *
23:31:15     00:02:51        24 C
23:31:25     00:03:00        24 C
23:31:25     00:03:00        24 C     *
23:31:35     00:03:10        24 C
23:31:46     00:03:20        24 C
23:31:46     00:03:20        24 C     *
23:31:56     00:03:30        24 C
23:32:06     00:03:40        25 C
23:32:06     00:03:40        25 C     *
23:32:16     00:03:51        24 C
23:32:31     00:04:00        24 C
23:32:31     00:04:00        24 C     *
23:32:47     00:04:15        24 C
23:33:02     00:04:30        23 C
23:33:02     00:04:30        23 C     *
```

When the user presses the button again the system will stop logging.

6 Validation and Performance

To validate our system we did it step by step following the one of the models stated above under planning. The waterfall model. This model allows us to code and implementing our system at the same time. When reading from the adc our temperature is suppose to be between -50 and 280 degrees due to the input voltage of 3.3V. To validate and test our system allowed the temperature sensor to read the room temperature by allowing it read values for over 10 min which was around 24 degrees C. Then increased the room temperature by turning the fan heater on which increase the room temperature. After a while we decreased the room temperature.

7 Conclusion

The system successfully met our uncle's expectations because it allowed him to be able to monitor the terrarium remotely. Even though it meets our uncles expectation there is still room for improvement. The product has potential to become a useful product with more features.

8 The Python Code

The initialisations and imports are as follows:

```python
import time
import board
import busio
import digitalio
import RPi.GPIO as GPIO
import adafruit_mcp3xxx.mcp3008 as MCP
from adafruit_mcp3xxx.analog_in import AnalogIn
import threading
from datetime import datetime
import ES2EEPROMUtils
import os

#spi setup
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
cs = digitalio.DigitalInOut(board.D5)

mcp = MCP.MCP3008(spi, cs)
chan = AnalogIn(mcp, MCP.P1)
#Button Buzzer pin numbers
button1 = 16
button2 = 23
buzzer1 = 13

#declare variables
hour = 0
min = 0
sec = 0
value = 5
firstTime = 0
log = False
getCurrentTime = None
Sec20 = None
thread = None
now = None
systClockString = None
tempString = None
```

```
pi_pwm_buzzer = None

#EEPROM object
eeprom = ES2EEPROMUtils.ES2EEPROM()
eeprom.write_block(0, [0])
```

The `fetch_temp_read` method is as follows:

```
#fetch EEPROM
def fetch_temp_read():
    temp_count = eeprom.read_byte(0)
    temp_reads = eeprom.read_block(1,temp_count*4)

    data = []
    i = 0

    while i < temp_count*4:
        data2 = []
        data2.append(chr(temp_reads[i])+""+chr(temp_reads[i+1])+""+
        chr(temp_reads[i+2])+""+chr(temp_reads[i+3])+""+
        chr(temp_reads[i+4])+""+chr(temp_reads[i+5])+""+
        chr(temp_reads[i+6])+""+chr(temp_reads[i+7]))

        data2.append(chr(temp_reads[i+8])+""+chr(temp_reads[i+9])+""+
        chr(temp_reads[i+10])+""+chr(temp_reads[i+11])+""+
        chr(temp_reads[i+12])+""+chr(temp_reads[i+13])+""+
        chr(temp_reads[i+14])+""+chr(temp_reads[i+15]))

        data2.append(chr(temp_reads[16])+""+chr(temp_reads[i+17]))
        data2.append(chr(temp_reads[i+19]))
        data.append(data2)
        i = i + 20
        pass
    return temp_count,data
```

The `save_temp_read` method is as follows:

```
#Writes data to EEPROM
def save_temp_read():
    global now,systClockString,tempString
    time.sleep(0.2)
    temp_count = eeprom.read_byte(0)
    temp_reads = eeprom.read_block(1,temp_count*4)
    temp_count = temp_count + 5

    eeprom.write_block(0, [temp_count])

    add = "{}".format(now)+""+"{}".format(systClockString)+""+
    "{}".format(tempString)+" C"
    for letter in add:
        temp_reads.append(ord(letter))
    eeprom.write_block(1, temp_reads)
    time.sleep(1)
    eeprom.write_block(0, [temp_count])
```

The `setup` method is as follows:

```
# Setup Pins
def setup():
    global pi_pwm_buzzer
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(button1,GPIO.IN,pull_up_down=GPIO.PUD_UP)
    GPIO.setup(button2,GPIO.IN,pull_up_down=GPIO.PUD_UP)
    GPIO.add_event_detect(button1,GPIO.FALLING,callback =
    callback, bouncetime = 500)

    GPIO.add_event_detect(button2,GPIO.FALLING,callback =
    ChangeInterval, bouncetime = 500)

    GPIO.setup(buzzer1,GPIO.OUT)
    pi_pwm_buzzer = GPIO.PWM(buzzer1,1000)
```

The `callback` method is as follows:

```

#Callback is called when button pressed
def callback(channel):
    global log
    if (not log == False):
        os.system('clear')
        print('Logging has stopped')
        print("Press button to start/stop logging.")
        log = not log
    else:
        os.system('clear')
        print ("{:<10}{:>10}{:>10}{:>10}".format("Time",
        "Sys Timer","Temp","Buzzer"))

        log = not log
```

The `ChangeInterval` method is as follows:

```
#Change interval
def ChangeInterval(channel):
    global value

    if(value == 5):
        value = 10
    elif(value == 2):
        value = 5
    elif(value == 10):
        value = 2
    time.sleep(0.1)
```

The `trigger_buzzer` method is as follows:

```
#Buzzer
def trigger_buzzer():
    global pi_pwm_buzzer
    pi_pwm_buzzer.start(0)
    pi_pwm_buzzer.ChangeDutyCycle(50)
    start = time.time()
    while True:
        time.sleep(0.1)
        try:
            #pi_pwm_buzzer.ChangeFrequency(1)
            time.sleep(0.1)
            stop = time.time()
            if(stop-start>0.1):
                pi_pwm_buzzer.stop()
                break
        except Exception as e:
            print("some error")
```

The `readADC` method is as follows:

```
#Read Temperature From ADC
def readADC():
    global temp,thread,value
    thread = threading.Timer(value, readADC) #after 5 sec print temp
    thread.daemon=True
    thread.start()
    temp = round(((chan.voltage-0.5)/0.01),3)
    pass
```

The `SysClock` method is as follows:

```
#System clock
def SysClock():
    global hour,min,sec,getCurrentTime,firstTime,Sec20
    if(firstTime==0):
        getCurrentTime = time.time()
        Sec20 = time.time()
        firstTime = firstTime+1
    sec = int(time.time() - getCurrentTime)
    if sec > 59:
        getCurrentTime = time.time()
        sec = 0
        min = min+1
    if min > 59:
        min = 0
        hour= hour +1
    if hour > 23:
        hour = 0
        min = 0
    return("{:0>2}".format(hour)+":"+"{:0>2}".format(min)+":"+
    "{:0>2}".format(sec))
```

The `main` method is as follows:

```
def main():
    global getCurrentTime,thread,log,value,Sec20,now,systClockString,tempString

    getCurrentTime = time.time()
    setup()
    readADC()
    os.system('clear')
    print("Press button to start/stop logging.")
    try:
        while True:
            if log == True:
                thread.join()
                systClockString = SysClock()

                tempString = ""+str(round(temp))
                now = datetime.now().strftime("%X")

                count,arr = fetch_temp_read()
                time.sleep(0.3)
                save_temp_read()
                if count == 0:
                    Asterisk = "*"
                    display = "{:<10}".format(now)+""+"{:>10}".format(
                    systClockString)+""+"{:>10}".format(tempString)+""+
                    "{:>2}".format("C")+""+"{:>5}".format(Asterisk)

                    print(display)

                else:
                    if(time.time()-Sec20>19):
                        Sec20 = time.time()
                        Asterisk = " "
                        display = "{:<10}".format(now)+""+"{:>10}".format(
                        systClockString)+""+"{:>10}".format(tempString)+""+
                        "{:>2}".format("C")+""+"{:>5}".format(Asterisk)

                        print(display)
                        count,data = fetch_temp_read()
                        Asterisk = "*"
                        #display = "{:<10}".format(data[len(data)-1][0])+""+"{:>10}".format(
                        #data[len(data)-1][1])+""+"{:>10}".format(data[len(data)-1][2])+""+
                        #"{:>2}".format(data[len(data)-1][3])+""+"{:>5}".format(Asterisk)


                        display = "{:<10}".format(now)+""+"{:>10}".format(
                        systClockString)+""+"{:>10}".format(tempString)+""+
                        "{:>2}".format("C")+""+"{:>5}".format(Asterisk)

                        print(display)
                        trigger_buzzer()

                    else:
                        Asterisk = " "
                        display = "{:<10}".format(now)+""+"{:>10}".format(
                        systClockString)+""+"{:>10}".format(tempString)+""+
                        "{:>2}".format("C")+""+"{:>5}".format(Asterisk)

                        print(display)
            time.sleep(0.1)
    except KeyboardInterrupt:
        print("Keyboard interrupt")
    except Exception as e:
        print(e)
    finally:
        GPIO.output(buzzer1,0)
        GPIO.cleanup()
```

Call main to execute the program

```
if __name__ == "__main__":
   main()
```

Declaration of non-plagiarism

- I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
- I have used the ________________ convention for citation and referencing. Each contribution to, and quotation in, this essay/report/project/submission from the work(s) of other people has been attributed, and has been cited and referenced.
- This essay/report/project/submission is my own work.
- I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.

Name Joseph Msimango Student Number MSMJOS003

Signature J.M Date 15/11/20

Declaration of non-plagiarism

- I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
- I have used the ________________ convention for citation and referencing. Each contribution to, and quotation in, this essay/report/project/submission from the work(s) of other people has been attributed, and has been cited and referenced.
- This essay/report/project/submission is my own work.
- I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.

Name Xola Manciya Student Number MNCXOL005

Signature X.M Date 15/11/20