
 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 1 of 41 Date: 28 Oct 2022
--	---	---

Air Quality Test Report

Project: UAVPAYG19 WP Name: Subsystem Testing WP Number: WP-AQS-06	Type of Test: Unit Test	
Test Article: Raspberry Pi 3B+ Pimoroni Enviro+ Enclosure	Part Number: N/A	Serial Number: N/A
System Requirements: [REQ-M-02] [REQ-M-03] [REQ-M-05]	Test Equipment: See “equipment used” section of each test	
Test Operators: Alexander Gray	Test Engineers: Alexander Gray	
Project Manager: Marissa Bowen	Project Supervisor: Dr Felipe Gonzalez	
Test Summary This documents details the processes carried out to test the individual components of the air quality subsystem. All tests listed in this report make use of the Pimoroni Enviro+ sensor cluster and raspberry pi. In addition to these devices, to calibrate the system a room thermometer and access to current altitude (in meters) is required. The purposes of these tests are to ensure that the system requirements related to the air quality subsystem are met. These requirements are described below in Table 1. However, the requirements that are tested in this report are REQ-M-01, REQ-M-02 and REQ-M-13 . All tests completed for this subsystem were successful in collecting accurate data. Although it is noted that a large amount of calibration is required for any data collected by the BME280 sensor due to heat interference. It is suggested that a better cooling solution is provided in the future to alleviate this issue.		

Queensland University of Technology
Gardens Point Campus
Brisbane, Australia, 4001.

This document is Copyright 2022 by the QUT. The content of this document, except that information which is in the public domain, is the proprietary property of the QUT and shall not be disclosed or reproduced in part or in whole other than for the purpose for which it has been prepared without the express permission of the QUT

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 2 of 41 Date: 28 Oct 2022
--	--	---

Revision Record

Document Issue/Revision Status	Description of Change	Date	Approved
1.0	Initial Issue	28/10/2022	A.G


 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 3 of 41 Date: 28 Oct 2022
--	--	---

Table of Contents


Paragraph	Page No.
1 Introduction	6
1.1 Scope	6
1.2 Background	6
2 Reference Documents.....	7
2.1 QUT avionics Documents	7
2.2 Non-QUT Documents	7
3 Test Objectives	8
4 Testing	9
4.1 Software Test.....	9
4.1.1 Software Test 1: Temperature Readings	9
4.1.2 Software Test 2: Pressure Readings	12
4.1.3 Software Test 3: Humidity Readings	15
4.1.4 Software Test 4: Light Readings	18
4.1.5 Software Test 5: Proximity Readings.....	21
4.1.6 Software Test 6: Gas Readings	24
4.2 Hardware Tests.....	27
4.2.1 Hardware Test 1: Enclosure Mounting	27
5 Results	29
6 Analysis	31
6.1 Hardware Analysis	31
6.1.1 Hardware Test 1: Enclosure Mounting	31
6.2 Software Analysis.....	31
6.2.1 Software Test 1: Temperature Readings	31
6.2.2 Software Test 2: Pressure Readings	31
6.2.3 Software Test 3: Humidity Readings	32
6.2.4 Software Test 4: Light Readings	32
6.2.5 Software Test 5: Proximity Readings.....	32
6.2.6 Software Test 6: Gas Readings	32
7 Conclusions and Recommendations.....	33
8 Appendix	35

List of Figures

Figure	Page No.
Figure 1: Temperature Readings No Calibration	11
Figure 2: Temperature Reading with Calibration.....	11
Figure 3: Pressure Data	14
Figure 4: Humidity Data.....	17
Figure 5: Light Data	20
Figure 6: Proximity Data.....	23
Figure 7: Gas Data.....	26
Figure 8: Enviro+ in Enclosure	28


List of Tables

Table	Page No.
Table 1: Air Quality Subsystem Requirements	8
Table 2: Results from tests	29
Table 3: Requirements Met	33

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 5 of 41 Date: 28 Oct 2022
--	--	---

Definitions

ASP	Advance Sensor Payload
UAV	Unmanned Aerial Vehicle
AQS	Air Quality Sensor

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 6 of 41 Date: 28 Oct 2022
--	--	---

1 Introduction

The team members of Group 19 have been appointed to research, design, plan and implement an Advance Sensor Payload (ASP) for Unmanned Aerial Vehicle (UAV) target detection and air quality monitoring in GPS denied environments. The group has committed to the specified budget whilst implementing the project requirements stated by the client. The team has also committed to meeting the deadline date specified by the client with a full functioning ASP that has been tested to ensure the client requirements have been met. This document outlines the hardware and software tests completed to ensure all requirements for the air quality subsystem are met.

1.1 Scope


The scope of the project is to research, plan, design, implement and test the ASP for UAV target detection in GPS denied environments. This document contains the objectives of the test, the equipment used, in depth descriptions of the tests, results, an analysis of these results and a conclusion with recommendations. The purpose of this test document see if the tests satisfies the state System Requirements/HLO's in RD-1

1.2 Background

The Queensland University of Technology's Airborne System Lab (ASL) has commissioned the group UAVPAYG19 to design and develop a payload capable in detecting specific objects, recording air quality data to be displayed on a web interface and to pierce a ground sample. This payload is to be attached to a S500 UAV which will complete an automated flight path. The payload is mounted on the bottom of the UAV using a provided bracket. This payload must contain all components to complete its required tasks. These components are:

- Raspberry Pi 3b+
- Raspberry Pi Camera
- Pimoroni Enviro+ sensor
- DF15RSMG 360 Degree Motor

The payload is required to identify three targets, a valve (In open or closed position), a fire extinguisher and an ArUCO marker. The Pimoroni sensor is to be used to record air temperature, pressure humidity, light and potentially hazardous gas level data. This data along with a live feed of the Raspberry Pi Camera is to be visualized on a Web Interface. Lastly a soil sample must be obtained using a sampling mechanism.


 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 7 of 41 Date: 28 Oct 2022
--	--	---

2 Reference Documents

2.1 QUT avionics Documents

RD/1	UA–System Requirements	UAVPayloadTAQ System Requirements
RD/2	UA –Customer Needs	Advanced Sensor Payload for UAV Target Detection and Air Quality Monitoring in GPS Denied Environments
RD/7	UAVPAYG19-PD-AQS-02	AQS prelim report
RD/20	UAVPAYG-19-ED-TR-01	Enclosure Test Report
RD/25	UAVPAYG-19-AQS-TAIP-WVI-TR-01	Air Quality Sensor, Target Acquisition and Image Processing, Web Vision Interface Integration Report
RD/26	UAVPAYG-19-ED-AQS-TAIP-ST-TR-01	Enclosure, Air Quality Sensor, Target Acquisition and Image Processing, Sampling Tube Test Report
RD/28	UAVPAYG-19-TR-AT-01	Enclosure, Air Quality Sensor, Target Acquisition and Image Processing, Sampling Tube Integration Report
RD/29	UAVPAYG19-AQS-FD-01	Air Quality final report

2.2 Non-QUT Documents


 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 8 of 41 Date: 28 Oct 2022
--	--	---

3 Test Objectives

To ensure that the Air Quality subsystem (AQS) was fully functional before integrating with all other subsystems the tests outlined in this document were performed. These tests have been completed to confirm that the AQS system is able to meet the system requirements listed in table 1. It should be noted that some requirements for this subsystem have not been met but the test that completes it is not mentioned in this report. These tests can be found in one of the following documents: RD/25, RD/26, RD/27, RD/28.

Table 1: Air Quality Subsystem Requirements

Requirement Code	Description
REQ-M-01	The UAVPayloadTAQ shall remain under the maximum weight of 320 g and comply with an IP41 rating. The air quality sensors must be exposed to the environment to allow for accurate reading.
REQ-M-02	The UAVPayloadTAQ must measure hazardous gases, (these include Carbene monoxide, Nitrogen dioxide, Ethanol, Hydrogen, Ammonia, methane, and Iso-butane) humidity, pressure, temperature and light via on-board sensors.
REQ-M-03	The UAVPayloadTAQ shall communicate with a ground station computer to transmit video, target detection and air quality data.
REQ-M-05	The Web Interface is required to display real time air sampling data that is recorded directly from the UAVPayloadTAQ and updated dynamically throughout the duration of the flight.
REQ-M-10	Preliminary designs shall be completed by week 7
REQ-M-11	The payload should display its IP address via the integrated Enviro sensor LCD screen
REQ-M-12	The LCD screen should display live feed of target detection as well as temperature readings from the Pi and the Enviro sensor board.
REQ-M-13	The LCD screen shall be placed on the side of the payload in order for the user to easily see its operation during flight.
REQ-M-14	Developed solution shall conform to the systems engineering approach.
REQ-M-15	The system shall have logged functioning operation for a minimal period of 10 minutes prior to acceptance test
REQ-M-19	Live data from the UAV must be made available through the web server within 10 seconds of capture.

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 9 of 41 Date: 28 Oct 2022
--	--	---

4 Testing

The AQS subsystem testing will be split into two separate testing components. Software tests and a single hardware test. The purpose of each of these tests will be to confirm the system requirements listed in table 1. For an overview of the results refer to section 5.

4.1 Software Test

The software testing stage consists of testing the Enviro+ sensor and ensure all required data can be collected. This testing was also completed to ensure that the data is as accurate as the situation demands.

4.1.1 Software Test 1: Temperature Readings

This test aims to confirm that the temperature sensor on the Enviro+ is functional and can accurately collect temperature data. This will consist of testing the current room temperature and ensure temperature changes can be seen.

Equipment used:

The equipment used in this test consists of the following:

- Raspberry Pi
- Enviro+
- PC
- Room thermometer

Procedure:

Following is the procedure used to conduct the test:

1. Connect power to Pi
2. Read and record the IP displayed on the enviro+ LCD
3. Using the IP address ssh into the Pi
4. Navigate to the AQS folder
 - a. `cd ~/AQS`
5. Start the data collection program
 - a. `python3 Data_collection.py`
6. Using the room thermometer take a reading of the current temperature
 - a. This is required to calibrate the sensor to the room
7. When prompted by the script enter the temperature recording
8. When prompted enter the sensors current altitude (in meters) for lv 1 of O block this is 10m
9. Type "No" (case sensitive) when prompted for gas calibration
10. Wait 10 seconds for the system to calibrate and observe the temperature readings
11. Compare the reading with the expected room temperature they should be similar
12. It is possible to simulate an increase in room temperature by placing a finger on the sensor for a few seconds and observing the console to check rising temperatures

Code:

The full code used for all tests can be found in Appendix A. The relevant sections of code from Data_collection.py for this test can be seen below:

From def calibration_setup():

```
while(timer < 10 ):
    timer = finish_time - start_time
    print(timer)
    cpu_temp = get_cpu_temperature()
    # Smooth out with some averaging to decrease jitter
    cpu_temps = cpu_temps[1:] + [cpu_temp]
    avg_cpu_temp = sum(cpu_temps) / float(len(cpu_temps))

    # print("this is the avg CPU temp " + str(avg_cpu_temp))
    finish_time = time.time()

while(adj_temp < float(calibration_temp)):
    raw_temp = bme280.get_temperature()
    adj_temp = raw_temp - ((avg_cpu_temp - raw_temp) / factor)
    factor += 0.01

    print("this is the factor " + str(factor))
    print("this is the adjusted temperature " + str(adj_temp))

def temp():
    # print(factor)


    cpu_temps = []

    unit = "C"
    cpu_temp = get_cpu_temperature()
    # Smooth out with some averaging to decrease jitter
    cpu_temps = cpu_temps[1:] + [cpu_temp]
    avg_cpu_temp = sum(cpu_temps) / float(len(cpu_temps))

    # print("this is the avg CPU temp " + str(avg_cpu_temp))

    raw_temp = bme280.get_temperature()
    # print("this is the adjusted temperature " + str(raw_temp))
    corrected_temp = raw_temp - ((avg_cpu_temp - raw_temp) / factor)

    print("this is the adjusted temperature " + str(corrected_temp))
    # display_text(variables[mode], data, unit)
    return corrected_temp
```

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 11 of 41 Date: 28 Oct 2022
--	--	--

Results and Evidence:

When this test was initially performed it was noticed that the temperature readings was heavily influenced by the heat of the raspberry Pi. To alleviate this a calibration stage was implemented to provide a reference point. This is discussed further in RD/29. Once the calibration stage was implemented it was seen that the temperature readings were accurate to the assumed temperature of the room. Alongside this, changes in temperature were observed when a finger was placed on the sensor. Figure 1 shows the results of the temperature sensor with no calibration. Figure 2 shows the results once calibration was implemented (assumed temperature of 24 degrees) with and without a figure influencing the reading. This test was considered as a success.



Figure 1: Temperature Readings No Calibration

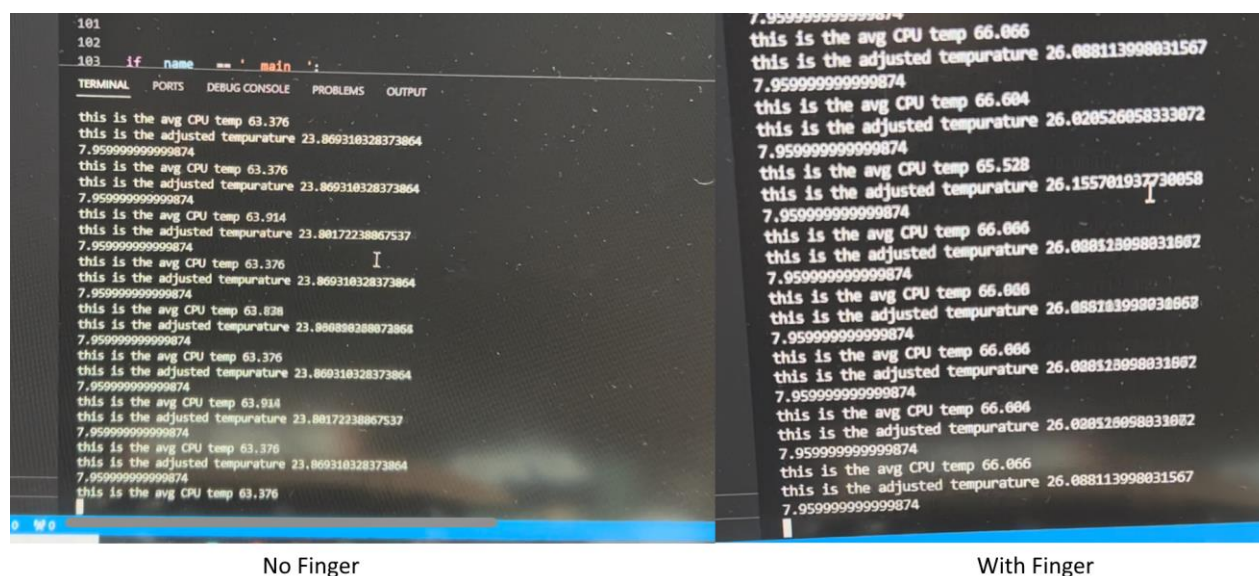



Figure 2: Temperature Reading with Calibration

Video Evidence: <https://youtu.be/kSqZLFVUfRI>

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 12 of 41 Date: 28 Oct 2022
--	--	--

4.1.2 Software Test 2: Pressure Readings

This test aims to confirm that the Pressure sensor on the Enviro+ is functional and can accurately collect atmospheric pressure data. This will consist of finding the current atmospheric pressure and comparing the read data.

Equipment used:


The equipment used in this test consists of the following:

- Raspberry Pi
- Enviro+
- PC
- Room thermometer
- Access to Internet (<http://www.bom.gov.au/qld/observations/brisbane.shtml>)

Procedure:

Following is the procedure used to conduct the test (steps 1-5 can be skipped if these have been completed by a previous test):

1. Connect power to Pi
2. Read and record the IP displayed on the enviro+ LCD
3. Using the IP address ssh into the Pi
4. Navigate to the AQS folder
 - a. `cd ~/AQS`
5. Start the data collection program
 - a. `python3 Data_collection.py`
6. Using the room thermometer take a reading of the current temperature
 - a. This is required to calibrate the sensor to the room
7. When prompted by the script enter the temperature recording
8. When prompted enter the sensors current altitude (in meters) for lv 1 of O block this is 10m
9. Type “No” (case sensitive) when prompted for gas calibration
10. Wait 10 seconds for the system to calibrate and observe the pressure readings
11. Compare the reading with the reading from the website linked above for the area (if operating at QUT this will be Brisbane)
 - a. This sensor has a reported error of ± 1.5 hPa the reading is considered correct if this reading is within that limit

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 13 of 41 Date: 28 Oct 2022
--	---	--

Code:

The full code used for all tests can be found in Appendix A. The relevant sections of code from Data_collection.py for this test can be seen below:

```
def press():
    temp = bme280.get_temperature()
    # print (alt)
    adj = math.pow(1 - (0.0065 * float(alt)/(temp + 0.0065 * float(alt) +
273.15)), -5.257) #calibration equation provided by Roscoe27 in this forum:
    # https://github.com/pimoroni/enviroplus-python/issues/67
    # The user used linear regression on the data set to generate the parameters
they seem to work within the error tollerances of the
    # pressure sensor

    # print (adj)

    pressure = bme280.get_pressure()

    adj_pres = pressure * adj


    # print("this is the sensor pressure " + str(pressure))
    print("this is the adjusted pressure " + str(adj_pres))

    return pressure
```


As stated in the comments of the code the calibration equation used for the pressure sensor was adapted from code found on the pimoroni forum groups. During the creation of these calibrations linear regression analysis was performed to improve the accuracy of the data. As seen in Figure 3 the pressure seen by the sensor before correction is around 10 hPa below what it should be. This is due to the effect that the sensors temperature has on all of its reading and why a correction is needed. Figure 3 also shows the results of the test compared against atmospheric measurements taken on the day. This test was considered as a success.

Figure 3: Pressure Data

Video Evidence: <https://youtu.be/x0obVyatBOo>

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 15 of 41 Date: 28 Oct 2022
--	---	--

4.1.3 Software Test 3: Humidity Readings

This test aims to confirm that the Humidity sensor on the Enviro+ is functional and can accurately collect humidity data. This will consist of finding the current room humidity and comparing this data against BOM recorded data.

Equipment used:


The equipment used in this test consists of the following:

- Raspberry Pi
- Enviro+
- PC
- Room thermometer
- Access to Internet (<http://www.bom.gov.au/qld/observations/brisbane.shtml>)

Procedure:

Following is the procedure used to conduct the test (steps 1-5 can be skipped if these have been completed by a previous test):

1. Connect power to Pi
2. Read and record the IP displayed on the enviro+ LCD
3. Using the IP address ssh into the Pi
4. Navigate to the AQS folder
 - a. `cd ~/AQS`
5. Start the data collection program
 - a. `python3 Data_collection.py`
6. Using the room thermometer take a reading of the current temperature
 - a. This is required to calibrate the sensor to the room
7. When prompted by the script enter the temperature recording
8. When prompted enter the sensors current altitude (in meters) for lv 1 of O block this is 10m
9. Type “No” (case sensitive) when prompted for gas calibration
10. Wait 10 seconds for the system to calibrate and observe the pressure readings
11. Compare the reading with the reading from the website linked above for the area (if operating at QUT this will be Brisbane)

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 16 of 41 Date: 28 Oct 2022
--	---	--

Code:

The full code used for all tests can be found in Appendix A. The relevant sections of code from Data_collection.py for this test can be seen below:

```
def humid():
    # calibration of humidity code created by user Roscoe27 in this forum
    # https://forums.pimoroni.com/t/enviro-readings-unreliable/12754/58

    humi_intercept = 13.686 # was originally at 15.686 this seems to work better
    comp_hum_slope = 0.966

    humi = bme280.get_humidity()
    raw_temp = bme280.get_temperature()
    dew_point = (243.04 * (math.log(humi / 100) + ((17.625 * raw_temp) / (243.04 + raw_temp)))) / (17.625 - math.log(humi / 100) - (17.625 * raw_temp / (243.04 + raw_temp)))
    temp_adjusted_hum = 100 * (math.exp((17.625 * dew_point) / (243.04 + dew_point)) / math.exp((17.625 * temp()) / (243.04 + temp())))
    comp_hum = comp_hum_slope * temp_adjusted_hum + humi_intercept

    # print("This is the dew point " + str(dew_point) + "C")
    # print("This is the sensor humidity " + str(humi) + "%")
    print("This is the adjusted Humidity " + str(comp_hum) + "%")

    return comp_hum
```


Results and Evidence:

As stated in the comments of the code the calibration equation used for the humidity sensor was adapted from code found on the pimoroni forum groups. During the creation of these calibrations linear regression analysis was performed to improve the accuracy of the data. This needs to be completed for all sensors stored in the BME280 chip as these readings are heavily affected by the temperature. As can be seen in Figure 4 the adjusted humidity records the same value as reported by BOM at that time which makes this test successful. The full effect of the board temperature on the sensor readings can be seen as the recorded humidity was 29.9%, well below the supposed value.

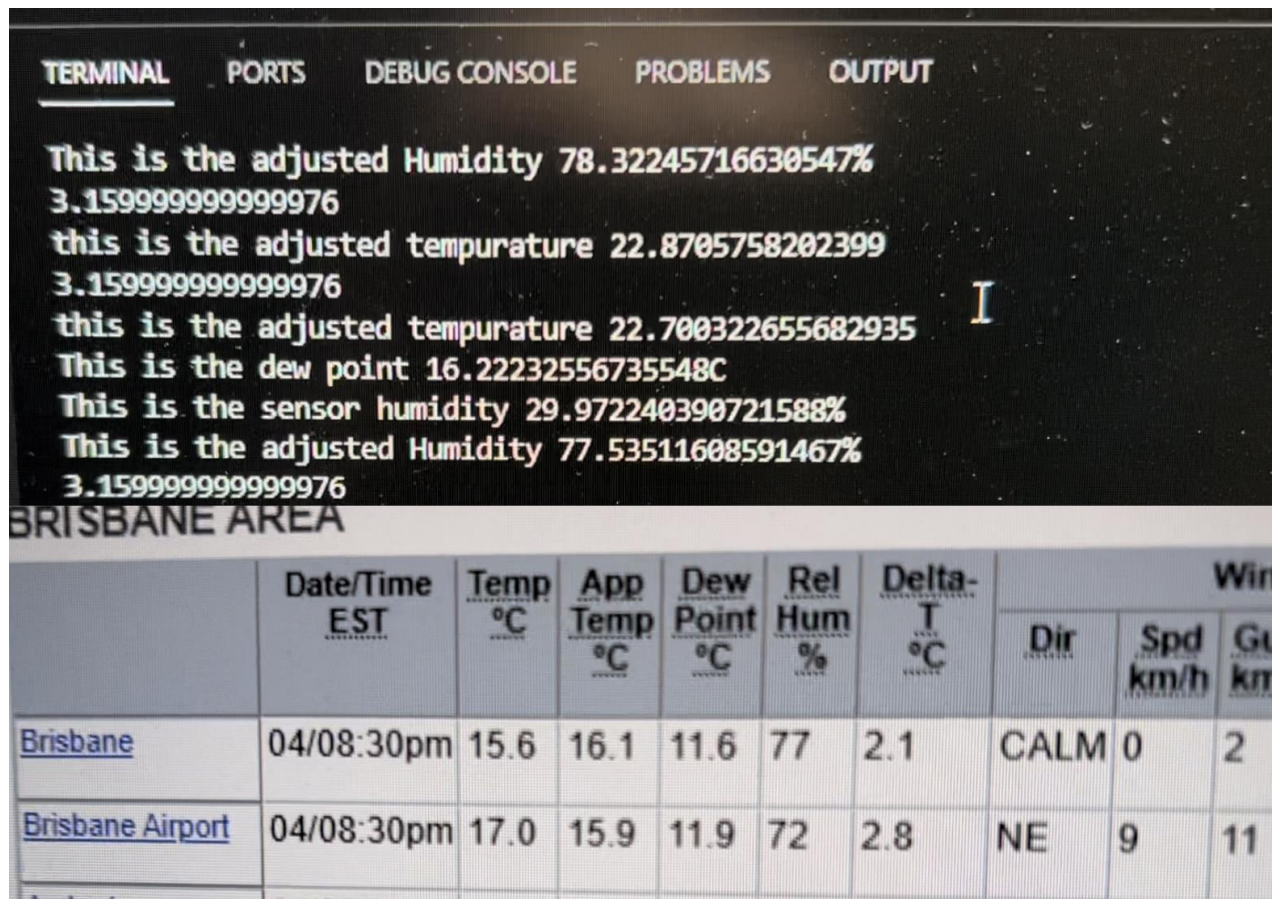



Figure 4: Humidity Data

Video Evidence: <https://youtu.be/pM3MqxbvaEg>

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 18 of 41 Date: 28 Oct 2022
--	---	--

4.1.4 Software Test 4: Light Readings

This test aims to confirm that the Light sensor on the Enviro+ is functional and can accurately collect light data. This will consist of running the light sensor to collect a baseline data then blocking the light or shining a torch to see if the value changes.

Equipment used:


The equipment used in this test consists of the following:

- Raspberry Pi
- Enviro+
- PC
- Room thermometer
- Torch

Procedure:

Following is the procedure used to conduct the test (steps 1-5 can be skipped if these have been completed by a previous test):

1. Connect power to Pi
2. Read and record the IP displayed on the enviro+ LCD
3. Using the IP address ssh into the Pi
4. Navigate to the AQS folder
 - a. `cd ~/AQS`
5. Start the data collection program
 - a. `python3 Data_collection.py`
6. Using the room thermometer take a reading of the current temperature
 - a. This is required to calibrate the sensor to the room
7. When prompted by the script enter the temperature recording
8. When prompted enter the sensors current altitude (in meters) for lv 1 of O block this is 10m
9. Type "No" (case sensitive) when prompted for gas calibration
10. Wait 10 seconds for the system to calibrate and observe the light readings
11. Wait till the light sensor is recording a steady value before testing for light changes
12. This test can either be completed by placing a finger over the sensor and observing a decrease in light. Or shining a torch on the sensor and seeing an increase in light


 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 19 of 41 Date: 28 Oct 2022
--	---	--

Code:

The full code used for all tests can be found in Appendix A. The relevant sections of code from Data_collection.py for this test can be seen below:

```
def light():
    light = ltr559.get_lux()
    print('this is the lux ' + str(light) + ' Lux')

    return light
```

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 20 of 41 Date: 28 Oct 2022
--	--	--

Results and Evidence:

Overall, this test was a success. As the light sensor is separate from the BME280 sensor and not effected by temperature it was not necessary to calibrate the sensor and was assumed that the values were correct. The results of placing a finger on the sensor can be seen in Figure 5. The image on the left was taken before the finger was placed with the image on the right being taken during the finger placement. As can be seen when the finger is placed on the sensor the expected drop in light occurs.

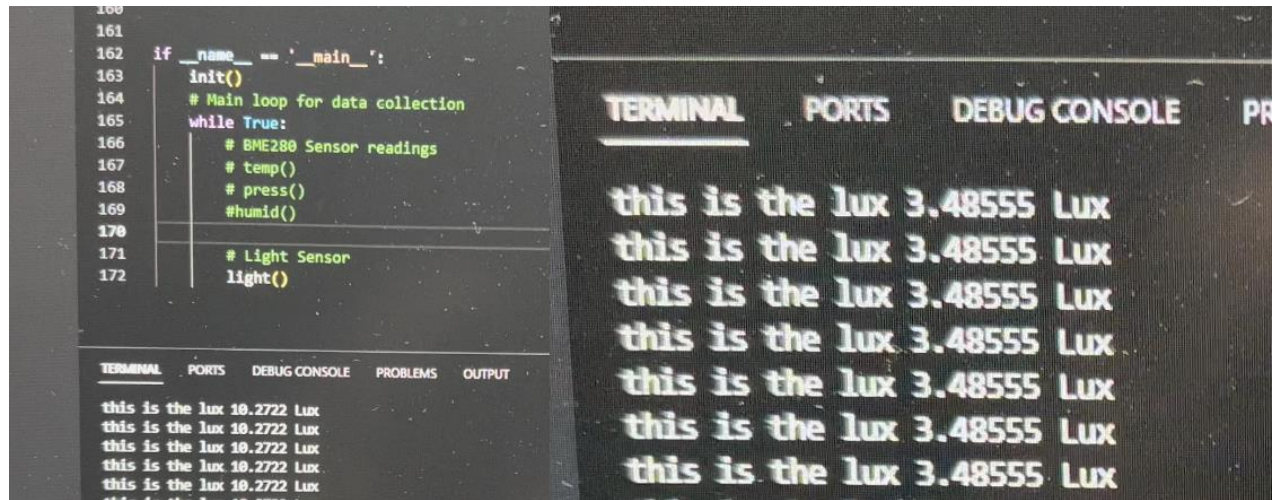



Figure 5: Light Data

Video Evidence: https://youtu.be/Uh4-jTV_Zqk

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 21 of 41 Date: 28 Oct 2022
--	--	--

4.1.5 Software Test 5: Proximity Readings

This test aims to confirm that the Proximity sensor on the Enviro+ is functional and can accurately collect Proximity data. This will consist of running the light sensor to collect a baseline data then blocking the light or shining a torch to see if the value changes.

Equipment used:


The equipment used in this test consists of the following:

- Raspberry Pi
- Enviro+
- PC
- Room thermometer

Procedure:

Following is the procedure used to conduct the test (steps 1-5 can be skipped if these have been completed by a previous test):


1. Connect power to Pi
2. Read and record the IP displayed on the enviro+ LCD
3. Using the IP address ssh into the Pi
4. Navigate to the AQS folder
 - a. `cd ~/AQS`
5. Start the data collection program
 - a. `python3 Data_collection.py`
6. Using the room thermometer take a reading of the current temperature
 - a. This is required to calibrate the sensor to the room
7. When prompted by the script enter the temperature recording
8. When prompted enter the sensors current altitude (in meters) for lv 1 of O block this is 10m
9. Type "No" (case sensitive) when prompted for gas calibration
10. Wait 10 seconds for the system to calibrate and observe the proximity readings
11. Place a finger on the proximity sensor and observe the changes in the console

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 22 of 41 Date: 28 Oct 2022
---	--------------------------------------	--

Code:

The full code used for all tests can be found in Appendix A. The relevant sections of code from Data_collection.py for this test can be seen below:

```
def proximity():  
    prox = ltr559.get_proximity()  
    print('this is the proximity ' + str(prox))  
  
    return prox
```


 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 23 of 41 Date: 28 Oct 2022
--	---	--

Results and Evidence:

Overall, this test was a success. As the proximity sensor is combined with the light sensor and as such is separate from the BME280 sensor. As it is not part of this sensor cluster readings are not effected by temperature. Thus, it was not necessary to calibrate the sensor and was assumed that the values were correct. The results of placing a finger on the sensor can be seen in Figure 6. The image on the left was taken before the finger was placed with the image on the right being taken during the finger placement. As can be seen when the finger is placed on the sensor the reading increase as proximity increases up to a maximum value.

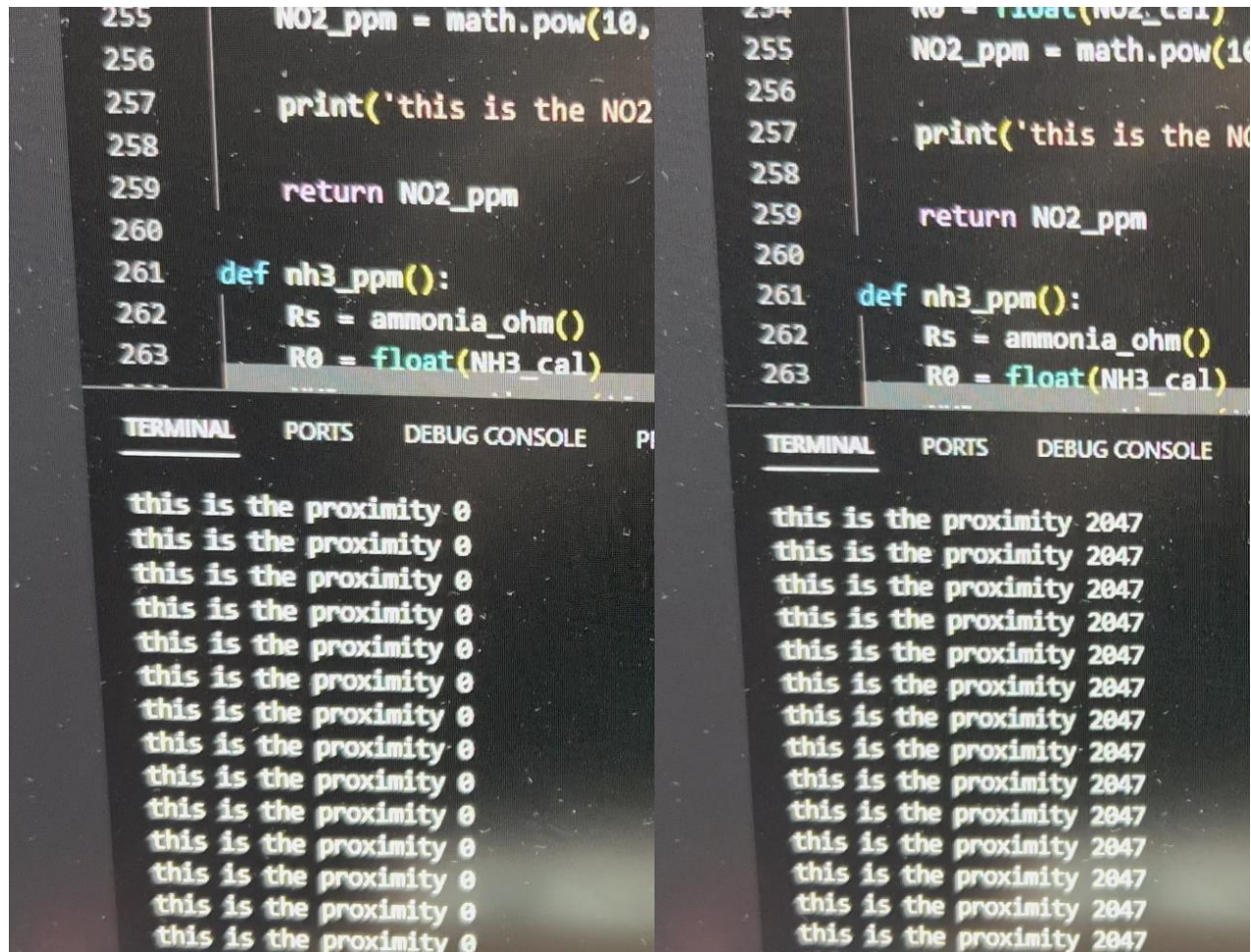



Figure 6: Proximity Data

Video Evidence: <https://youtu.be/IKByNjCxmMQ>

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 24 of 41 Date: 28 Oct 2022
--	---	--

4.1.6 Software Test 6: Gas Readings

This test aims to confirm that the Gas sensor on the Enviro+ is functional and can accurately collect all three types of gas data. This will consist of calibrating the gas sensor to the room environment before checking the readings in ppm. Note that due to no gas calibrators being present the reading will be checked based on feasibility.

Equipment used:


The equipment used in this test consists of the following:

- Raspberry Pi
- Enviro+
- PC
- Room thermometer

Procedure:

Following is the procedure used to conduct the test (steps 1-5 can be skipped if these have been completed by a previous test):

1. Connect power to Pi
2. Read and record the IP displayed on the enviro+ LCD
3. Using the IP address ssh into the Pi
4. Navigate to the AQS folder
 - a. `cd ~/AQS`
5. Start the data collection program
 - a. `python3 Data_collection.py`
6. Using the room thermometer take a reading of the current temperature
 - a. This is required to calibrate the sensor to the room
7. When prompted by the script enter the temperature recording
8. When prompted enter the sensors current altitude (in meters) for lv 1 of O block this is 10m
9. If the gas sensor has not been previously calibrated to the room (within the past day) type “Yes” (case sensitive) when prompted for gas calibration. Else if it has been calibrated recently type “No” (case sensitive)
 - a. Note that wait times for a full calibration is 5 minutes the system must not be disturbed in this time
10. Wait for the system to calibrate and observe the gas readings
 - a. A screenshot can be taken to get a single frame of the readings
11. Check the gas sensor readings by either observing the results to see if they make sense of by using hand sanitiser and holding recently sanitised hands up to the sensor looking for changes in readings

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 25 of 41 Date: 28 Oct 2022
--	---	--

Code:

The full code used for all tests can be found in Appendix A. The relevant sections of code from Data_collection.py for this test can be seen below:

```
def co_ppm():
    Rs = reduced_ohm()
    R0 = float(CO_cal)
    CO_ppm = math.pow(10, -1.25 * math.log10(Rs/R0) + 0.64) # equation provided
by Roscoe27 sourced from https://forums.pimoroni.com/t/enviro-ohms-to-ppm/12207/5

    print('this is the CO ppm ' + str(CO_ppm))

    return CO_ppm

def no2_ppm():
    Rs = oxidised_ohm()
    R0 = float(NO2_cal)
    NO2_ppm = math.pow(10, math.log10(Rs/R0) - 0.8129) # equation provided by
Roscoe27 sourced from https://forums.pimoroni.com/t/enviro-ohms-to-ppm/12207/5


    print('this is the NO2 ppm ' + str(NO2_ppm))

    return NO2_ppm

def nh3_ppm():
    Rs = ammonia_ohm()
    R0 = float(NH3_cal)
    NH3_ppm = math.pow(10, -1.8 * math.log10(Rs/R0) - 0.163) # equation provided
by Roscoe27 sourced from https://forums.pimoroni.com/t/enviro-ohms-to-ppm/12207/5

    print('this is the NH3 ppm ' + str(NH3_ppm))

    return NH3_ppm
```

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 26 of 41 Date: 28 Oct 2022
--	--	--

Results and Evidence:

Overall, this test was a success. As the gas sensor returns readings in Ohms in order to receive the reading as a ppm value a conversion needs to take place. The data sheet for this sensor does not include the equation to make this conversion but instead provides graphs that explain the relationship between baseline reading (taken as an average from the five minutes of calibration for each class of gas). Using these graphs, it is possible to create a formula that finds the ppm reading. As this is a common request for this sensor there are many existing examples of this calculation due to the accuracy provided by other calibration methods an equation made by the same user as before was tweaked for this situation and used again here. Overall, the gas readings seem to be accurate with what a room should be reading these can be seen in Figure 7 below. During the acceptance testing the other method was used to confirm that gas readings were being taken. This is expanded upon further in RD/28.

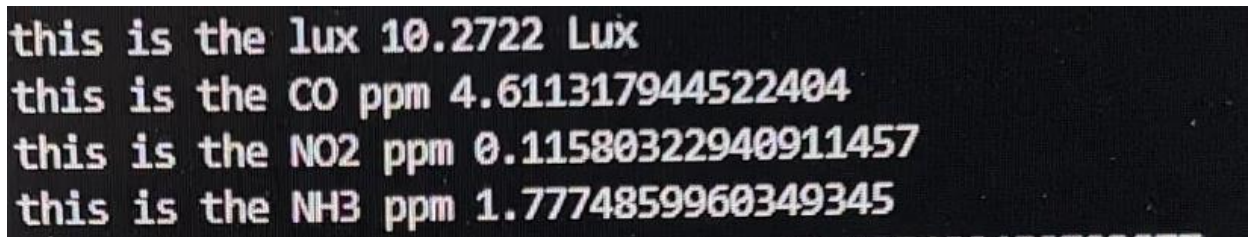



Figure 7: Gas Data

Video Evidence: <https://youtu.be/WhgC50L6Ces>

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 27 of 41 Date: 28 Oct 2022
--	---	--

4.2 Hardware Tests

There is only one hardware test that pertains to this subsystem. This being mounting the Enviro+ sensor to the enclosure subsystem.

4.2.1 Hardware Test 1: Enclosure Mounting

This hardware test consists of mounting the enviro+ to the enclosure using the integrated retainers. Once mounted the enviro+ can be checked to see if all sensor are open to the air and the LCD screen is visible.

Equipment Used:


The equipment used in this test consists of the following:

List all equipment used when conducting the test

- Raspberry Pi
- Enviro+
- PC
- Enclosure

Procedure:

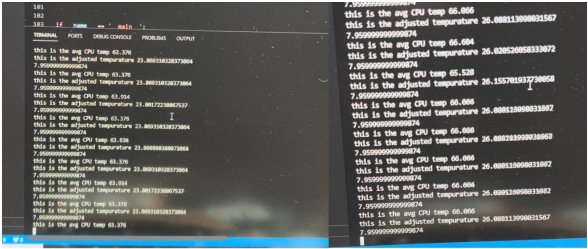
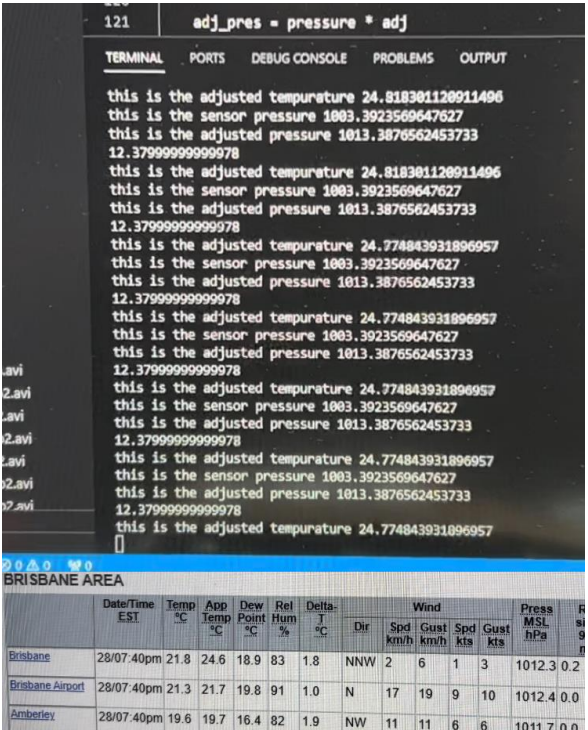
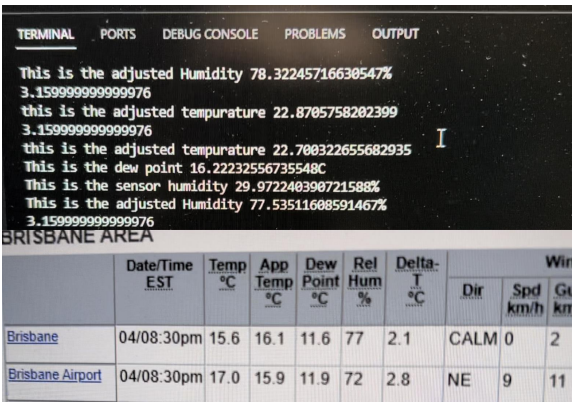
1. Ensure the Enviro+ is wired to the Raspberry Pi as per the ICD documentation
2. Mount the Raspberry Pi inside the enclosure
3. Slot the Enviro+ into its retainers
4. Place and secure the lid onto the enclosure
5. Confirm that all sensors are accessible to the air and the LCD screen is still visible


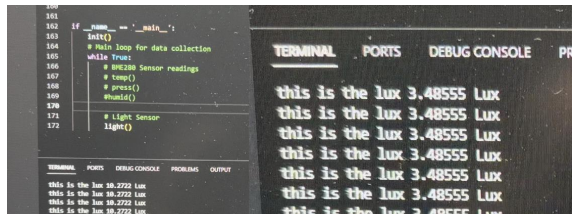
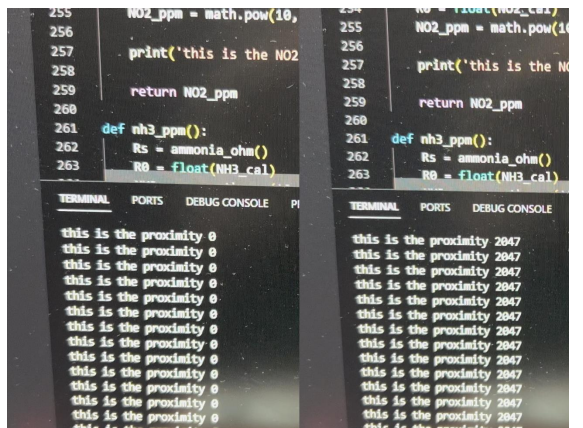
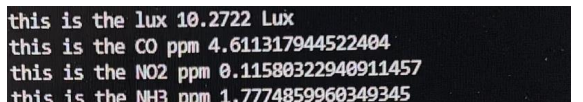

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 29 of 41 Date: 28 Oct 2022
--	--	--


5 Results

Table 2 provides an overview of all tests completed in this subsystem and what requirement the test is aimed to complete. As can be seen from the table all tests passed successfully indicating that no major design changes are required.

Table 2: Results from tests

Test Title	Result	IMG	Requirement Met
Software Test 1	Passed		REQ-M-02
Software Test 2	Passed		REQ-M-02
Software Test 3	Passed		REQ-M-02

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 30 of 41 Date: 28 Oct 2022	
Software Test 4	Passed		REQ-M-02
Software Test 5	Passed		REQ-M-02
Software Test 6	Passed		REQ-M-02
Hardware Test 1	Passed		REQ-M-01, REQ-M-13

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 31 of 41 Date: 28 Oct 2022
--	---	--

6 Analysis

All tests discussed in this report have been completed with the intension to verify that the subsystem is functional before integration occurs with the other systems. This ensures that integration goes smoothly without many issues.

6.1 Hardware Analysis

6.1.1 Hardware Test 1: Enclosure Mounting

As this subsystem is primarily software focused there is only one hardware test prevalent to this system. This being the mounting of the Enviro+ to the enclosure. This test when run was successful on the first attempt. This indicates that integration with the other hardware components can continue without issue with only minor redesigns required. The other hardware tests relating to the enclosure can be found in RD/20.

6.2 Software Analysis


As this subsystem is primarily focused on the air quality data collection there was a greater focus on the software tests and the accuracy of the data. This was achieved through 6 separate tests each confirming that the different data measurements required under **REQ-M-02** are collectable. The largest issue faced when implementing this system was collecting accurate readings using the BME280 sensor (temperature, pressure and humidity). As all readings on this sensor are effected by the PCB temperature if this increases above the expected levels (such as by a hot Raspberry Pi CPU in proximity) all reading become inaccurate beyond the error tolerances specified.

6.2.1 Software Test 1: Temperature Readings

The purpose of this test was to confirm if the temperature sensor was able to take an accurate temperature of the sensors room. As this sensor is part of the BME280 chip it is heavily influenced by the temperature of the CPU and if this is not compensated for the inaccuracy of the reading will remain. To alleviate this issue a scaling factor was used which is found by scanning all factors until the predicted temperature has been reached hence why the room temperature needs to be entered at the start of the test. Overall, this test was a success with the Enviro+ being successfully used to record temperature readings.

6.2.2 Software Test 2: Pressure Readings

This test aimed to show that the pressure sensor component of the BME280 chip was capable of accurately recording atmospheric pressure. Similar to the temperature sensor this reading is affected by the PCB temperature. In testing without any compensation for this it was found that the sensor would typically record the pressure at 10hPa below the recorded value as seen in B.O.M. To alleviate this issue a calibration taking in the sensors current altitude and sensor temperature was used. As there was no baseline pressure that could be used to calculate a scaling factor off of for this sensor it was necessary to implement code that achieves this. This code was modified from a similar project which was facing similar problems. Once the code was modified to better suit this project it was implemented. As seen from the results of this test it was successful.

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-AQS-TR-01 Issue: 1.0 Page: 32 of 41 Date: 28 Oct 2022
--	---	--

6.2.3 Software Test 3: Humidity Readings

The purpose of this test was to confirm the functionality of the humidity sensor. This is the final sensor that is part of the BME280 chip. As such this sensor also required calibration due to the temperature offset. In this case however this can be done by calculating the dew point temperature based on the sensors reading and does not require user input to calibrate. Similar to the pressure readings, code found in a community forum was implemented with changes made to optimize the system for this application and climate. As can be seen from the results there is a large discrepancy between the read humidity and the actual humidity. However, once the adjustments were made the humidity returned to an expected value.

6.2.4 Software Test 4: Light Readings

This is the first of the sensors tested that was not impacted by the temperature of the PCB. As such this sensor did not require any calibration to gain accurate readings implementation was much simpler. The purpose of this test was to confirm that the light sensor is able to detect changes in ambient light which it was able to leading to a successful test.

6.2.5 Software Test 5: Proximity Readings

As this sensor is integrated within the light sensor again there was not need for calibrations to be performed. The purpose of this test was to confirm the usability of the proximity sensor for future integrations. Overall, this test was completed successfully without any issues.

6.2.6 Software Test 6: Gas Readings

The aim of this test was to use the gas sensor on the Enviro+ to find the ppm values of three gases. These gases being CO, NO₂ and NH₃. It should be noted that whilst these gasses are the gasses targeted by the sensor and thus this application it is able to detect a variety of gasses within range. As such only provides an indication of gas levels. Whilst this sensor was not affected greatly by the PCB temperature calibration is still required for a baseline room reading to be taken. In this instance the calibration time is 5 minutes. However, it is suggested that calibration should take between 5 – 90 minutes to get an accurate reading. If the gas resistance values are known in advance or if calibration in the test room has been completed recently it is possible to ignore the calibration time and enter the values manually or by using a file. In addition to this the reading of the sensor does not natively provide a reading in ppm. Instead, a conversion from the recorded resistance to ppm needs to be completed using the calibrated baseline. Once again, the formulas to do this were adapted from a community forum post. Upon doing this and running the test the values returned seemed to be reasonable for the room environment the test was being run in therefore the test was deemed a success.

7 Conclusions and Recommendations

Overall, The AQS subsystem has passed all testing completed. This testing has shown that the Enviro+ sensor cluster is able to accurately collect air quality data. Some initial testing done on this system found that some data recordings were not accurate when taking the sensor reading directly leading to improvements being made to the system. Among this was research into possible solutions which yielded usable code to calibrate the sensors with minor adjustments made to improve the accuracy for this application. Fit, form and function tests have been completed for the AQS subsystem which has verified that the code is reliable, and the data can be integrated in other subsystems.

An improvement that could be made to this system would be a redesign of the enclose with a larger payload and cost budget. Using this it would be possible to incorporate a design that uses a fan system to keep the CPU and Enviro+ PCB cool during operation. This in turn would reduce the amount of calibration required potentially removing it entirely for the BME280 sensors.

Table 3 lists the requirements related to this subsystem and how each of these have been met. It should be noted that not all requirements have been met by this report. The relevant reports are referenced where possible.

Table 3: Requirements Met

Requirement Code	Description	Requirement Met
REQ-M-01	The UAVPayloadTAQ shall remain under the maximum weight of 320 g and comply with an IP41 rating. The air quality sensors must be exposed to the environment to allow for accurate reading.	Met: - This requirement has been met as seen by the results of Hardware Test 1. The weight requirement has also been met but is discussed further in RD/20 and RD/28
REQ-M-02	The UAVPayloadTAQ must measure hazardous gases, (these include Carbone monoxide, Nitrogen dioxide, Ethanol, Hydrogen, Ammonia, methane, and Iso-butane) humidity, pressure, temperature and light via on-board sensors.	Met: - This has been met by all software tests completed. With the Enviro+ being used to accurately collect the required data.
REQ-M-03	The UAVPayloadTAQ shall communicate with a ground station computer to transmit video, target detection and air quality data.	Met: - This requirement has been met however, tests relating to it are described in the following test reports: RD/24, RD/25, RD/27 and RD/28
REQ-M-05	The Web Interface is required to display real time air sampling data that is recorded directly from the UAVPayloadTAQ and updated dynamically throughout the duration of the flight.	Met: - This requirement has been met however, tests relating to it are described in the following test reports: RD/24, RD/25, RD/27 and RD/28



REQ-M-10	Preliminary designs shall be completed by week 7	Met: - This has already been demonstrated through the submission of the assessment 1 reports
REQ-M-11	The payload should display its IP address via the integrated Enviro sensor LCD screen	Met: - This requirement has been met however, tests relating to it are described in the following test reports: RD/25 and RD/28
REQ-M-12	The LCD screen should display live feed of target detection as well as temperature readings from the Pi and the Enviro sensor board.	Met: - This requirement has been met however, tests relating to it are described in the following test reports: RD/25 and RD/28
REQ-M-13	The LCD screen shall be placed on the side of the payload in order for the user to easily see its operation during flight.	Met: - This requirement has been met however, tests relating to it are described in the following test reports: RD/25 and RD/28
REQ-M-14	Developed solution shall conform to the systems engineering approach.	Met: - This has been demonstrated throughout the semester with reports showing this to be true
REQ-M-15	The system shall have logged functioning operation for a minimal period of 10 minutes prior to acceptance test	Met: - This requirement has been met however, tests relating to it are described in the following test reports: RD/24, RD/25, RD/27 and RD/28
REQ-M-19	Live data from the UAV must be made available through the web server within 10 seconds of capture.	Met: - This requirement has been met however, tests relating to it are described in the following test reports: RD/24, RD/25, RD/27 and RD/28

8 Appendix

```
#!/usr/bin/env python3

import time
from ltr559 import LTR559 #light and proximity
from enviroplus import gas #gas
import logging #for logging
import smbus2
from bme280 import BME280 #temp pressure and humidity
import math

bme280 = BME280()
ltr559 = LTR559()

delay = 0.5 # Debounce the proximity tap
mode = 0 # The starting mode
last_page = 0
light = 1

logging.basicConfig(
    format='%(asctime)s.%(msecs)03d %(levelname)-8s %(message)s',
    level=logging.INFO,
    datefmt='%Y-%m-%d %H:%M:%S')

logging.info("""this collects readings from Enviro plus' sensors

Press Ctrl+C to exit!
""")

def get_cpu_temperature():
    with open("/sys/class/thermal/thermal_zone0/temp", "r") as f:
        temp = f.read()
        temp = int(temp) / 1000.0
        # print('wah ' + str(temp))
    return temp

def oxidised_ohm():
    # find the NO2
    NO2 = gas.read_all()
    NO2 = NO2.oxidising # this is in ohms

    return NO2

def reduced_ohm():
```

```
# find the CO value
CO = gas.read_all()
CO = CO.reducing # this is in ohms

return CO

def ammonia_ohm():
    # find the ethanol
    NH3 = gas.read_all()
    NH3 = NH3.nh3 # this is in ohms

    return NH3

# this function is run once at the start of
# the program to find the appropriate factor for the room
def calibration_setup():
    global factor
    global alt
    global NO2_cal
    global CO_cal
    global NH3_cal

    alt = 0
    factor = 1

    adj_temp = 0
    cpu_temps = []

    cpu_temp = get_cpu_temperature()
    # Smooth out with some averaging to decrease jitter
    cpu_temps = cpu_temps[1:] + [cpu_temp]
    avg_cpu_temp = sum(cpu_temps) / float(len(cpu_temps))

    calibration_temp = input("Enter the current room temperature in degrees
celsius: ")

    # https://en-us.topographic-map.com/maps/vj4v/Ferny-Grove/

    alt = input("Enter the current altitude (above sea level) of your position
(in meters): ")

    gas_cal = input("Does the gas sensor require calibration? Type 'Yes' or 'No':
")

    start_time = time.time()
    finish_time = 0
```



```
if str(gas_cal) == "No":
    try:
        with open("Gas_calibration.txt") as params:
            content = params.read()
            print(content)
        content = content.split(" ")
        index = 1
        for x in content:
            if x == "no2_cal":
                NO2_cal = str(content[index])

            elif x == "co_cal":
                CO_cal = str(content[index])

            elif x == "nh3_cal":
                NH3_cal = str(content[index])
            index += 1

    except:
        print("Calibration file does not exist enter values manually if known:")
        print()
        NO2_cal = input("input the NO2 calibration value (in ohms): ")
        CO_cal = input("input the CO calibration value (in ohms): ")
        NH3_cal = input("input the NH3 calibration value (in ohms): ")

elif str(gas_cal) == "Yes":
    # print("wah")
    min_count = 0
    timer = 0

    while(timer < 300 ): #set this to 300 for the actual implementation as
this is the minimum callibration time (5 minutes) longer would be better
        timer = finish_time - start_time
        print(timer)
        NO2_cal = oxidised_ohm()
        CO_cal = reduced_ohm()
        NH3_cal = ammonia_ohm()

        finish_time = time.time()

    with open("Gas_calibration.txt", 'w') as calibration:
        calibration.write("no2_cal " + str(NO2_cal) + " co_cal " +
str(CO_cal) + " nh3_cal " + str(NH3_cal))
```

```
print("Calibrating the parameters: ")

start_time = time.time()
finish_time = 0
timer = 0

while(timer < 10 ):
    timer = finish_time - start_time
    print(timer)
    cpu_temp = get_cpu_temperature()
    # Smooth out with some averaging to decrease jitter
    cpu_temps = cpu_temps[1:] + [cpu_temp]
    avg_cpu_temp = sum(cpu_temps) / float(len(cpu_temps))

    # print("this is the avg CPU temp " + str(avg_cpu_temp))
    finish_time = time.time()

while(adj_temp < float(calibration_temp)):
    raw_temp = bme280.get_temperature()
    adj_temp = raw_temp - ((avg_cpu_temp - raw_temp) / factor)
    factor += 0.01

    print("this is the factor " + str(factor))
    print("this is the adjusted temperature " + str(adj_temp))

# return factor

# gets the temperature in degrees celsius
def temp():
    # print(factor)

    cpu_temps = []

    unit = "C"
    cpu_temp = get_cpu_temperature()
    # Smooth out with some averaging to decrease jitter
    cpu_temps = cpu_temps[1:] + [cpu_temp]
    avg_cpu_temp = sum(cpu_temps) / float(len(cpu_temps))

    # print("this is the avg CPU temp " + str(avg_cpu_temp))

    raw_temp = bme280.get_temperature()
    # print("this is the adjusted temperature " + str(raw_temp))
```

```
corrected_temp = raw_temp - ((avg_cpu_temp - raw_temp) / factor)

print("this is the adjusted temperature " + str(corrected_temp))
# display_text(variables[mode], data, unit)
return corrected_temp

# Gets the Pressure in hPa
def press():
    temp = bme280.get_temperature()
    # print (alt)
    adj = math.pow(1 - (0.0065 * float(alt)/(temp + 0.0065 * float(alt) +
273.15)), -5.257) #calibration equation provided by Roscoe27 in this forum:
    # https://github.com/pimoroni/enviroplus-python/issues/67
    # The user used linear regression on the data set to generate the parameters
they seem to work within the error tollerances of the
    # pressure sensor

    # print (adj)

    pressure = bme280.get_pressure()

    adj_pres = pressure * adj

    # print("this is the sensor pressure " + str(pressure))
    print("this is the adjusted pressure " + str(adj_pres))

    return pressure

def humid():
    # calibration of humidity code created by user Roscoe27 in this forum
https://forums.pimoroni.com/t/enviro-readings-unreliable/12754/58

    humi_intercept = 13.686 # was orginally at 15.686 this seems to work better
    comp_hum_slope = 0.966

    humi = bme280.get_humidity()
    raw_temp = bme280.get_temperature()
    dew_point = (243.04 * (math.log(humi / 100) + ((17.625 * raw_temp) / (243.04
+ raw_temp)))) / (17.625 - math.log(humi / 100) - (17.625 * raw_temp / (243.04 +
raw_temp)))
    temp_adjusted_hum = 100 * (math.exp((17.625 * dew_point)/(243.04 +
dew_point)) / math.exp((17.625 * temp()) / (243.04 + temp())))
    comp_hum = comp_hum_slope * temp_adjusted_hum + humi_intercept

    # print("This is the dew point " + str(dew_point) + "C")
    # print("This is the sensor humidity " + str(humi) + "%")
    print("This is the adjusted Humidity " + str(comp_hum) + "%")
```



```
    return comp_hum

def light():
    light = ltr559.get_lux()
    print('this is the lux ' + str(light) + ' Lux')

    return light

def proximity():
    prox = ltr559.get_proximity()
    print('this is the proximity ' + str(prox))

    return prox

def init():
    calibration_setup()

def co_ppm():
    Rs = reduced_ohm()
    R0 = float(CO_cal)
    CO_ppm = math.pow(10, -1.25 * math.log10(Rs/R0) + 0.64) # equation provided
by Roscoe27 sourced from https://forums.pimoroni.com/t/enviro-ohms-to-ppm/12207/5

    print('this is the CO ppm ' + str(CO_ppm))

    return CO_ppm

def no2_ppm():
    Rs = oxidised_ohm()
    R0 = float(NO2_cal)
    NO2_ppm = math.pow(10, math.log10(Rs/R0) - 0.8129) # equation provided by
Roscoe27 sourced from https://forums.pimoroni.com/t/enviro-ohms-to-ppm/12207/5

    print('this is the NO2 ppm ' + str(NO2_ppm))

    return NO2_ppm

def nh3_ppm():
    Rs = ammonia_ohm()
    R0 = float(NH3_cal)
    NH3_ppm = math.pow(10, -1.8 * math.log10(Rs/R0) - 0.163) # equation provided
by Roscoe27 sourced from https://forums.pimoroni.com/t/enviro-ohms-to-ppm/12207/5

    print('this is the NH3 ppm ' + str(NH3_ppm))
```




```
    return NH3_ppm

# converts the AQS data into a json object
def to_json():
    AQS_data = {'Temperature': float(temp()), 'Pressure': float(press()),
    'Humidity': float(humid()), 'Light': float(light()), 'Proximity':
    float(proximity()), 'CO_ppm': float(co_ppm()), 'NO2_ppm': float(no2_ppm()),
    'NH3_ppm': float(nh3_ppm())}

    return AQS_data

if __name__ == '__main__':
    init()
    # Main loop for data collection
    while True:
        # BME280 Sensor readings
        temp() # corrected_temp
        press() # pressure
        humid() # comp_hum

        # Light and Proximity Sensor
        light() # light
        proximity() # prox

        #Gas Readings
        co_ppm() # CO_ppm
        no2_ppm() # NO2_ppm
        nh3_ppm() # NH3_ppm
        print(to_json())
```