 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-WVI-TR-02 Issue: 2.0 Page: 1 of 27 Date: 28 Oct 2022
--	---	---

Web Interface Test Report


Project: UAVPAYG19 WP Name: WP-WVI-06 WP Number: Subsystem Testing		Type of Test: Unit Test	
Test Article: Web Visualization Interface		Part Number: N/A	Serial Number: N/A
System Requirements: REQ-M-03 REQ-M-04 REQ-M-05 REQ-M-06		Test Equipment: See “equipment used” section of each test	
REQ-M-07 REQ-M-10 REQ-M-15 REQ-M-19			
Test Operators: Ryan Brooker		Test Engineers: Ryan Brooker	
Project Manager: Marissa Bowen		Project Supervisor: Dr Felipe Gonzalez	

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

Test Summary

This test report contains the testing of the visualisation in version 1 on the Web Interface. This document contains images and information about the visualization of the Web Interface. A conclusion regarding subsystem requirements has been discussed. The subsystem system requirements have been met. A discussion to improve the design further has been made in this document.

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-WVI-TR-02 Issue: 2.0 Page: 3 of 27 Date: 28 Oct 2022
--	--	---

Revision Record

Document Issue/Revision Status	Description of Change	Date	Approved
1.0	Initial Issue	14/10/2022	R.B
2.0	Test results and conclusion	28/10/2022	R.B

Table of Contents

Paragraph	Page No.
1 Introduction	7
1.1 Scope	7
1.2 Background	7
2 Reference Documents	8
2.1 QUT avionics Documents	8
2.2 Non-QUT Documents	8
3 Test Objectives	9
4 Testing	10
4.1 Software Test	10
4.1.1 Software Test 1: Web Interface connection over a LAN.	11
4.1.2 Software Test 2: Web Interface delay	12
4.1.3 Software Test 3: Server Data Storage and Requirements	13
4.1.4 Software Test 4: Live Image Stream, AQS and TAIP Web Interface data update	14
4.1.5 Software Test 5: Web Socket test	15
4.1.6 Software Test 6: Data Accessibility	16
4.1.7 Software Test 7: API endpoints test	17
4.1.8 Software Test 8: Target Identification Alert test	19
4.2 Hardware Tests	20
5 Results	21
6 Analysis	23
6.1 Hardware Analysis	23
6.2 Software Analysis	23
6.2.1 Software Test 1: Web Interface connection over a LAN.	23
6.2.2 Software Test 2: Web Interface delay	23
6.2.3 Software Test 3: Server Data Storage and Requirements	23
6.2.4 Software Test 4: Live Image Stream, AQS and TAIP Web Interface data update	23
6.2.5 Software Test 5: Web Socket test	23
6.2.6 Software Test 6: Data Accessibility	23
6.2.7 Software Test 7: API endpoints test	24
6.2.8 Software Test 8: Target Identification Alert test	24
7 Conclusions and Recommendations	25
8 Appendix	27
No appendix	27

List of Figures

Figure	Page No.
Figure 1: Web Interface accessible from two devices	11
Figure 2: Request sent from Postman	12
Figure 3: Request received on the Web Interface	13
Figure 4 MySQL code to query latest database table entry	13
Figure 5: Postman request sending data to the server	14
Figure 6: Data from Postman request is saved in the database	14
Figure 7: Web Interface established Web Socket connection to the Web Server	16
Figure 8: 10 minutes of logged AQS data	17
Figure 9: 10 minutes of logged TAIP data	17
Figure 10: Postman test of correctly formatted API requests.....	18
Figure 11: Postman test of incorrectly formatted API requests	18
Figure 12: Postman test results of correctly formatted API requests	19
Figure 13: Postman test results of incorrectly formatted API requests	19

List of Tables

Table	Page No.
Table 1: Web Visualisation and Interfaces Subsystem Requirements	9
Table 2: Results from tests	21
Table 3: Requirements Met	25



Definitions

Acronym	Definition
WVI	Web Visualisation and Interfaces
TAIP	Target Analysis and Image Processing
ST	Sampling Tube
AQS	Air Quality Sensor
UAV	Unmanned Aerial Vehicle
QUT	Queensland University of Technology
API	Application Programming Interface
PC	Personal Computer
Wi-Fi	Wireless Fidelity
GPS	Global Positioning System
LAN	Local Area Network
GUI	Graphical User Interface

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 test report details the testing within the Web Visualization and Interfaces subsystem. The tests in this report have been created to demonstrate that the Web Interface has met the necessary requirements set by the client. Further test has been constructed to also demonstrate the reliability of the subsystem.

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 demonstrates if the test 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-WVI-TR-02 Issue: 2.0 Page: 8 of 27 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/3	UAVPAYG19-PM-PMP-03	PMP document
RD/34	UAVPAYG19-WVI-FD-02	WVI Final Design

2.2 Non-QUT Documents

3 Test Objectives

The primary objective of this test report is to test if the Web Visualization subsystem completes the requirement listed in Table 1. The test that has been completed will provide details to see if the subsystem requirements have been met, and where they have not been met then modification of the subsystem will be required, and further retesting completed. If all the test a proven successful, then there are no further modifications needed for the WVI subsystem.

Table 1: Web Visualisation and Interfaces Subsystem Requirements


Requirements	Description
REQ-M-03	The UAVPayloadTAQ shall communicate with a ground station computer to transmit video, target detection and air quality data.
REQ-M-04	The target identification system shall be capable of alerting the GCS of a target's type.
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-06	The Web Interface is required to display the images of the targets that are taken directly from the UAVPayloadTAQ and updated every time a new picture is taken.
REQ-M-07	The Web Interface shall be designed and run as a web server, which is to be accessible by any computers on the local network. This shall store logged sensor data and target detections with corresponding timestamps.
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.

4 Testing

Testing has been done to ensure that the subsystem solution that is deployed adheres to the requirements and adheres to those requirements reliably. For this test there are no hardware testing as the WVI subsystem only consist of software. Modification will be made to the subsystem if the subsystem does not pass the testing.

4.1 Software Test

Software Test	Description of the test
Web Interface connection over a LAN.	The Web Interface is accessible on any PC that is connected to the same local network.
Web Interface delay.	Data is shown on the Web Interface within 10 seconds of capture from the payload.
Server data storage and requirements.	Data is sent from the Payload to the server and the server records the data in a database. The Data should have corresponding time stamps.
Live Image Stream, AQS and TAIP data update	The Web Interface should update an image and air quality data dynamically when it is sent one from the payload. The Web Interface should show a live image stream.
Web Socket test.	The Web Interface should connect the user to the server through web sockets to update the GUI
Data Accessibility.	The Web Interface should be able to show 10 minutes of Data on request.
API endpoints test.	The Web Interface API endpoints will be tested to see if they will reject a bad/incomplete request and store the data sent in a correctly formatted request.
Target Identification Alert test	The Web Interface will alert if the user is the payload has successfully detected a target.

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-WVI-TR-02 Issue: 2.0 Page: 11 of 27 Date: 28 Oct 2022
--	--	--

4.1.1 Software Test 1: Web Interface connection over a LAN.

The Web Interface connection over a LAN test is to ensure that the server is accessible to multiple users that are on the same local network.

Equipment used:

The equipment used in this test consists of the following:

- 2 x Laptop
- Wireless Router
- Safari or Google Chrome browser
- MySQL Database

Procedure:

1. Connect both laptops to the wireless router using Wi-Fi.
2. Start the Database service on the first laptop
3. Start the Web Interface Server on the first laptop by using the command `./node server` in the server files directory.
4. Connect to the Web Interface using a Safari or Google Chrome Browser by going to the web address `Localhost:3000` on the laptop running the Server
5. Using the server address displayed on the Web Interface, connect to the Web Interface using a Safari or Google Chrome Browser by going to that server address on port 3000 on the second laptop. (Eg address. `172.0.0.4:3000`)

Results and Evidence:

The result from this test was that the Web Interface was able to be accessed successfully from multiple users on the same network across multiple devices. The Web Interface showed the same data across both users and the interface was consistent across both users (Figure 1).

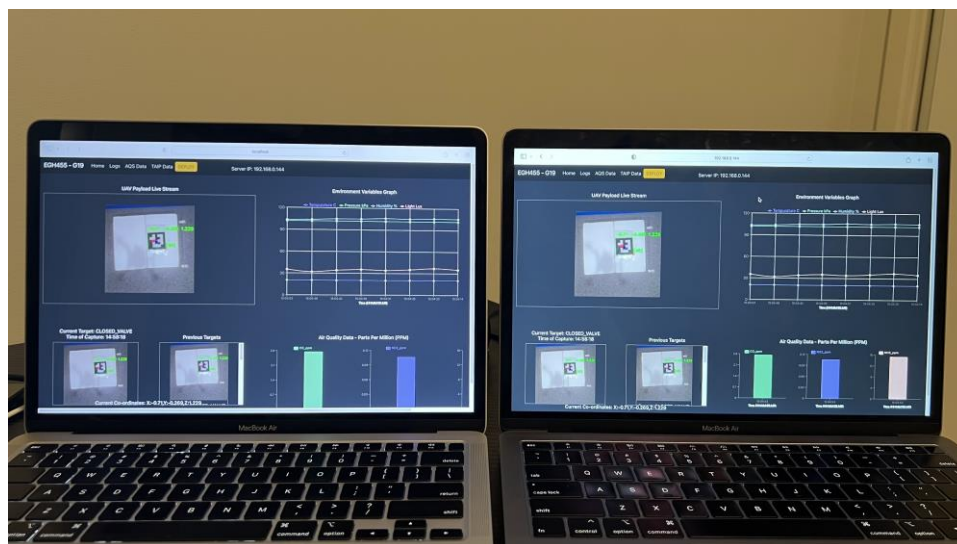


Figure 1: Web Interface accessible from two devices

4.1.2 Software Test 2: Web Interface delay

The Web Interface delay test is to ensure that the web interface is updated within 10 seconds of receiving the data from the payload.

Equipment used:

The equipment used in this test consists of the following:

- Laptop
- Postman (Free software for creating HTTP requests)
- Safari or Google Chrome browser
- MySQL Database

Procedure:

1. Start the Database service on the laptop
2. Start the Web Interface Server on the laptop by using the command `./node server` in the server files directory.
3. Open the web interface on <http://localhost:3000>
4. Open the browsers developer tools and record network activity
5. Using Postman, send a request to the server with mock data to mimic the payloads requests.
6. Using the timestamps recorded in Postman, compare the timestamp to the time the Web Interface gets the response to render the data on the Web Interface.

Results and Evidence:

The Results from this test showed that the Web Interface was able to update the interface within 10 seconds of receiving the data. Following the procedure showed that a request was sent from Postman at 12:19:58pm GMT (Figure 2) and the Web Interface has completed the request from the server to collect that data at 12:19:59 GMT (Figure 3). After comparing the time sent from Postman to the time the Web Interface received that data, the calculations showed that the Web Interface on average received the data within 1-2 Seconds. As the request for this data is handles the re-render of the component, for the response on the Web Interface to be completed the new data must be mounted, thus meaning the data is accessible within 1-2 seconds from the Payload sending the data meaning that the Web Interface delay test was successful.

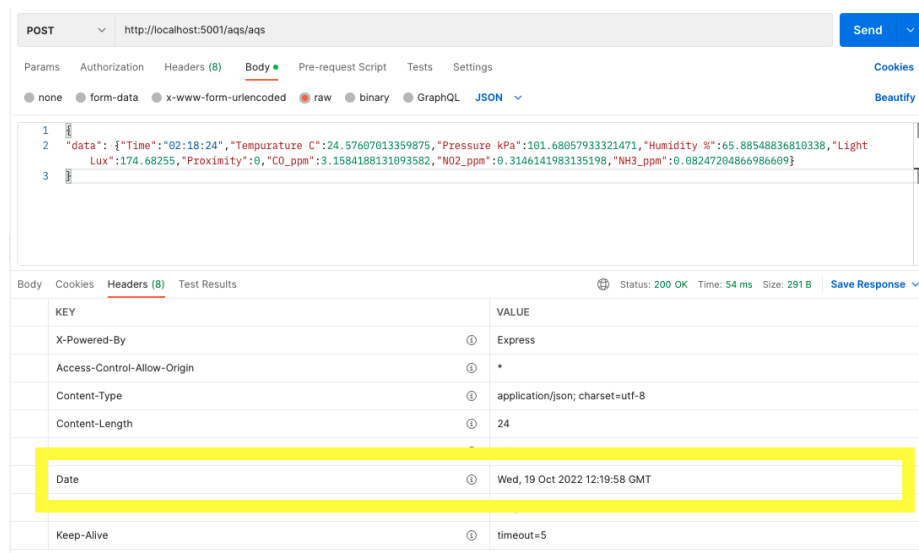



Figure 2: Request sent from Postman

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-WVI-TR-02 Issue: 2.0 Page: 13 of 27 Date: 28 Oct 2022
--	--	--

Summary

```

URL: http://localhost:3000/aqs/getAqsData
Status: 200 OK
Source: Network
Address: 127.0.0.1:3000
Initiator: xhr.js:220

```

Request

```

GET /aqs/getAqsData HTTP/1.1
Accept: application/json, text/plain, */*
Accept-Encoding: gzip, deflate
Host: localhost:3000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/15.6.1 Safari/605.1.15
Accept-Language: en-AU,en;q=0.9
Referer: http://localhost:3000/
Connection: keep-alive

```

Response

```

HTTP/1.1 200 OK
Access-Control-Allow-Methods: *
Content-Type: application/json; charset=utf-8
Connection: close
Vary: Accept-Encoding
Transfer-Encoding: Identity
Date: Wed, 19 Oct 2022 12:19:59 GMT
Content-Encoding: gzip
ETag: W/"a80-qlDEBxtmjJTdBYg9rZd+G+6kYFc"
Access-Control-Allow-Origin: *
x-powered-by: Express

```

Figure 3: Request received on the Web Interface

4.1.3 Software Test 3: Server Data Storage and Requirements

The Server Data Storage and Requirements test is to ensure that data sent from the Payload is saved in a Database with corresponding timestamps.

Equipment used:

The equipment used in this test consists of the following:

- Laptop
- Postman
- MySQL Database

Procedure:

1. Start the MySQL Database service on the laptop
2. Start the Web Interface Server on the laptop by using the command `./node server` in the server files directory.
3. Using Postman, send a request to the server with mock data to mimic the payloads requests.
4. Query the Database to ensure that the data is saved with corresponding timestamps

Code

The code used to Query the MySQL Database (Figure 4)

```

1 #Select Most Recent Entry From Air Quality Table in Database
2 • SELECT * FROM EGH455.aqs LIMIT 1;

```

Figure 4 MySQL code to query latest database table entry

Results and Evidence:

The results from this test show that after sending data to the server using a Postman request (Figure 5). The server receives the data, it stores the data in the database and saves the corresponding timestamps (Figure 6). This shows that the data was successfully saved, thus this test was successful.

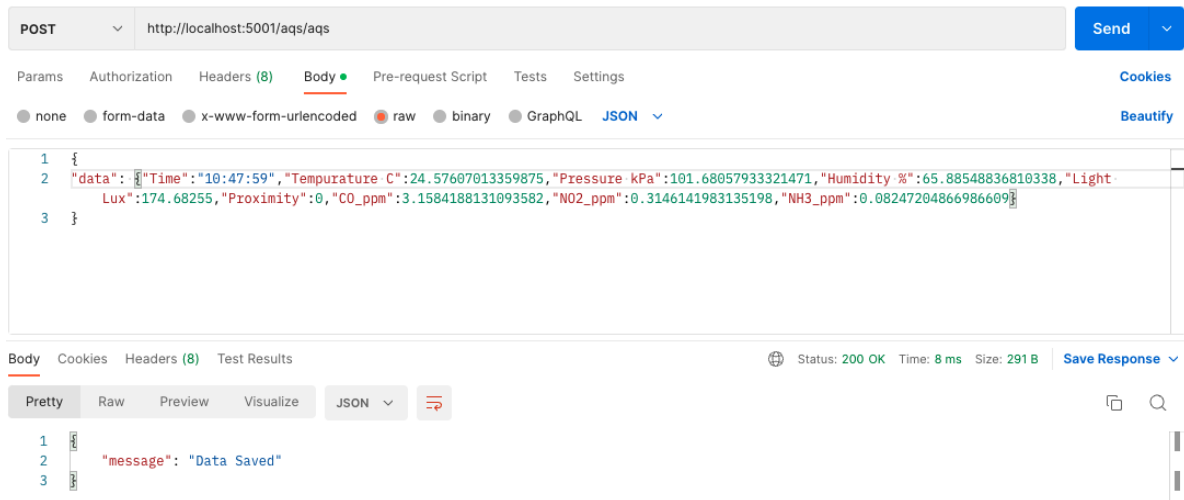


Figure 5: Postman request sending data to the server

id	airQualityData	airQualityDate
5605	{"Time":"10:47:59","Temperature C":24.57607013359875,"Pressure kPa":101.680579333214...	2022-10-19 22:48:01

Figure 6: Data from Postman request is saved in the database

4.1.4 Software Test 4: Live Image Stream, AQS and TAIP Web Interface data update

The Live Image Stream, AQS and TAIP Web Interface data update test is to ensure that data sent from the Payload is updated dynamically on the Web Interface.


Equipment used:

The equipment used in this test consists of the following:

- Laptop
- Wireless Router
- Safari or Google Chrome browser
- MySQL Database
- Raspberry Pi
- Raspberry Pi camera

Procedure:

1. Start the MySQL Database service on the laptop
2. Start the Web Interface Server on the laptop by using the command `./node server` in the

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-WVI-TR-02 Issue: 2.0 Page: 15 of 27 Date: 28 Oct 2022
--	--	--

server files directory.

3. Run the main script on the Raspberry Pi
4. Open the Web Interface by navigating to <http://localhost:3000> to visually confirm that the data is updating on the Web Interface dynamically

Results and Evidence:

The results from this test showed that as the Payload recorded and sent data to the web server, the user interface updated dynamically. This is shown in the video recording of the Web Interface shown, thus making this test successful.

Video Evidence: <https://youtu.be/LGQwPKWBF4g>

4.1.5 Software Test 5: Web Socket test

The Web Socket test is to ensure that the Server and the Interface has consistent communication without needing to manually refresh the page.

Equipment used:

The equipment used in this test consists of the following:

- Laptop
- Safari or Google Chrome browser
- MySQL Database

Procedure:

1. Start the MySQL Database service on the laptop
2. Start the Web Interface Server on the laptop by using the command `./node server` in the server files directory.
3. Open the internet browser
4. Open the developer tools in the browser and record network traffic and filter to Web Sockets
5. Open the web interface on <http://localhost:3000>
6. Analyse the Web Socket data to ensure that there is a connection open between the server and the interface

Results and Evidence:

The results from this test were that the Web interface was able to establish a Web Socket connection to the server and the Web Socket connection can receive messages from the Server (Figure 7). The results show that this test was successful.

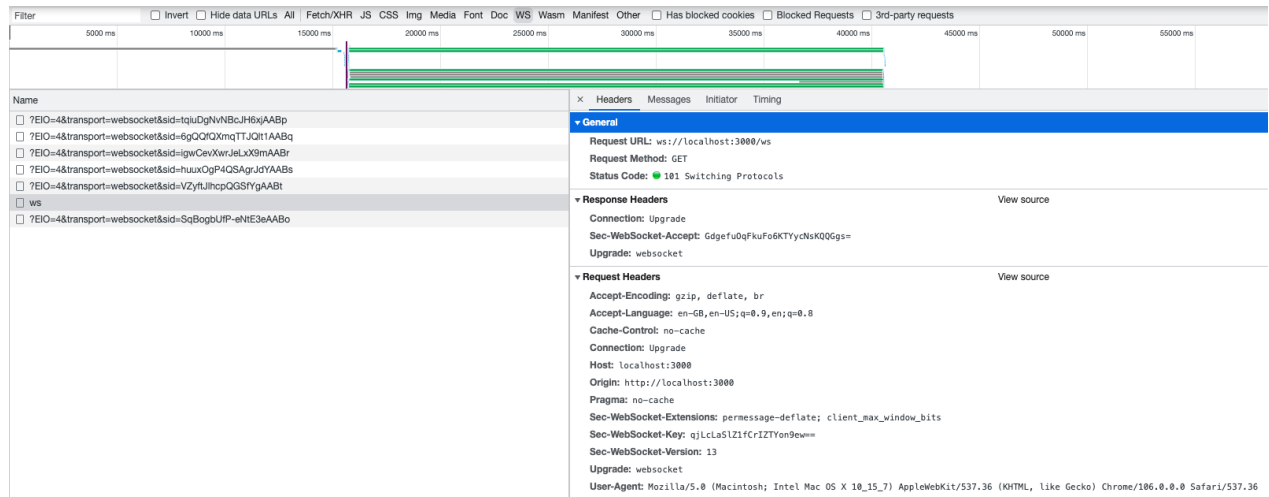


Figure 7: Web Interface established Web Socket connection to the Web Server

4.1.6 Software Test 6: Data Accessibility

The Data Accessibility test is to ensure that the Web Interface can access 10 minutes of Data sent from the payload.

Equipment used:

The equipment used in this test consists of the following:

- Laptop
- Safari or Google Chrome browser
- MySQL Database

Procedure:

1. Start the MySQL Database service on the laptop
2. Start the Web Interface Server on the laptop by using the command `./node server` in the server files directory.
3. Run the main script on the Raspberry Pi for a minimum of 10 minutes
4. Open the internet browser and navigate to <http://localhost:3000>
5. Navigate to the TAIP Tab and the AQS tab to verify that there is over 10 Minutes of data accessible from the Web Interface

Results and Evidence:

The results from this test were that the Web Interface can show 10 minutes of recorded data sent from the Payload. This 10 minutes of logged AQS and TAIP data is shown in their respective tabs that are accessible on the Web Interface (Figure 8 & Figure 9). The results shown verify that this test was successful.

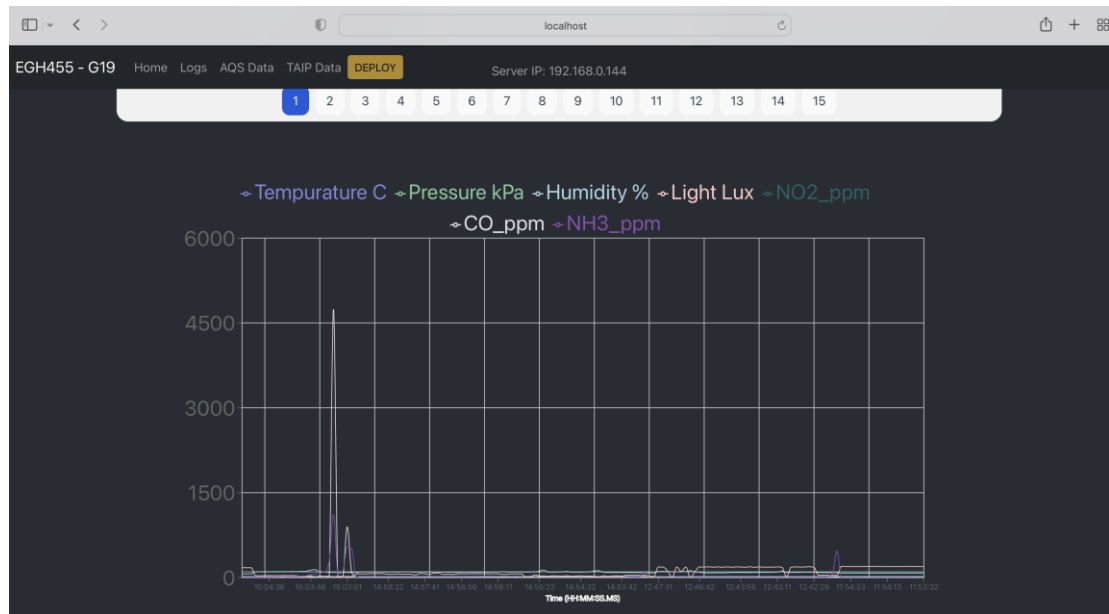


Figure 8: 10 minutes of logged AQS data

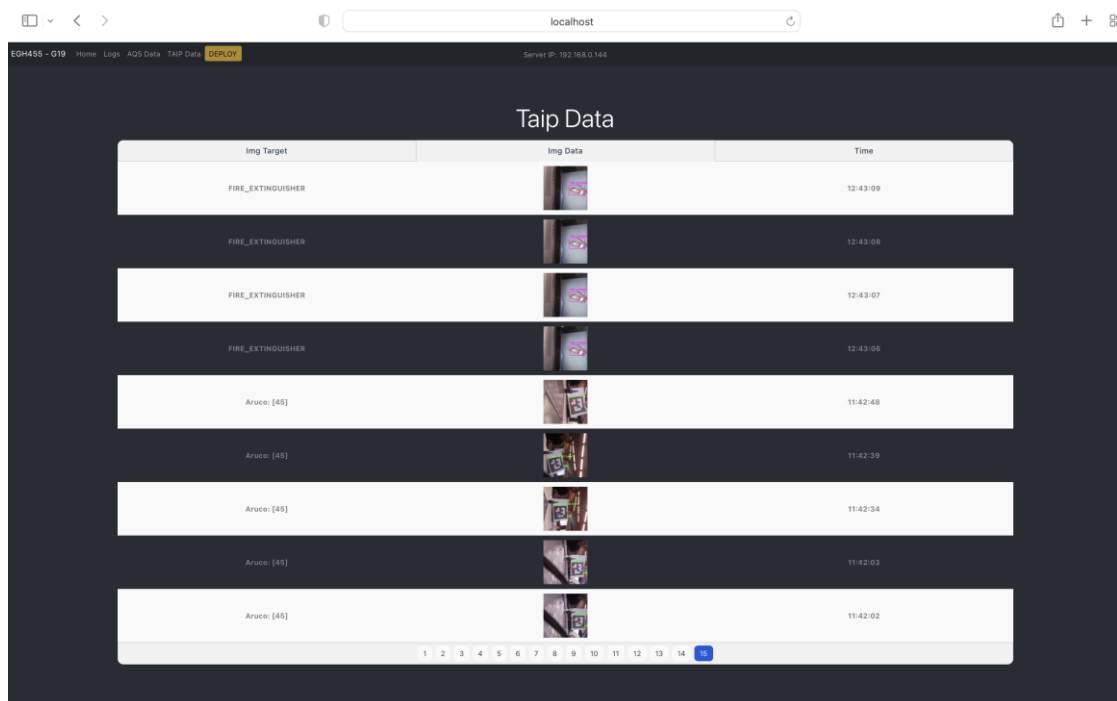


Figure 9: 10 minutes of logged TAIP data


4.1.7 Software Test 7: API endpoints test

The API endpoints test is to ensure that the Web Interface server can receive requests for all the needed API endpoints to receive data from the Payload. The API endpoints should also return an error for an incorrectly formatted API address request.

Equipment used: API endpoints test.

The equipment used in this test consists of the following:

- Laptop

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-WVI-TR-02 Issue: 2.0 Page: 18 of 27 Date: 28 Oct 2022
--	--	--

- Safari or Google Chrome browser
- MySQL Database
- Postman

Procedure:

1. Start the MySQL Database service on the laptop
2. Start the Web Interface Server on the laptop by using the command `./node server` in the server files directory.
3. Run the Postman Test file and record the results

Postman:

The postman file used to run the correctly formatted tests that will be accepted by the server is shown below (Figure 10). The postman file used to run the incorrectly formatted tests that will be rejected by the server is shown below in (Figure 11).

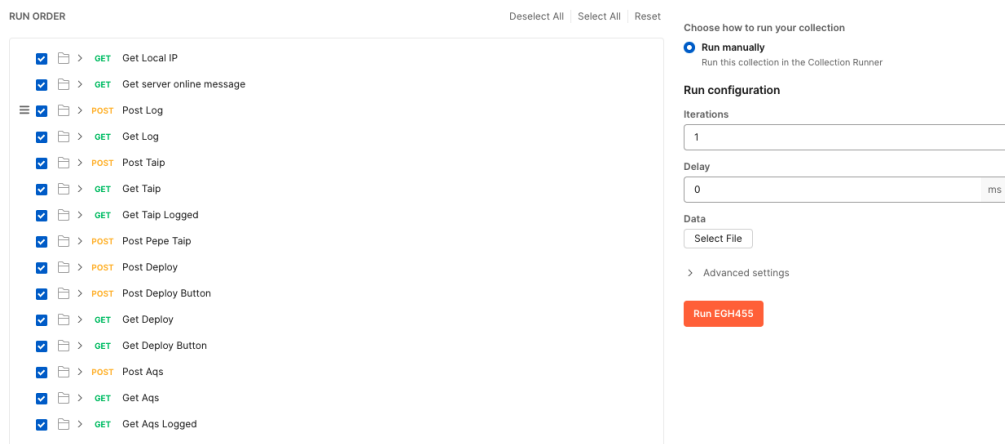


Figure 10: Postman test of correctly formatted API requests

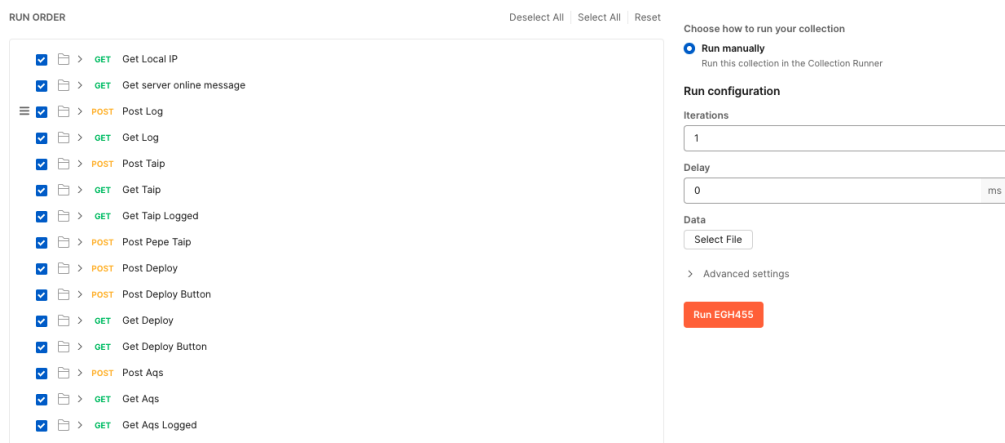


Figure 11: Postman test of incorrectly formatted API requests

Results and Evidence:

The results from this test show that when the API requests are formatted correctly the Postman test returns status codes of 200 for Get requests and 201 for Post Requests (Figure 12). Furthermore, when the API endpoints are formatted incorrectly, the test shows the server rejects the request and returns an error (Figure 13). The results from this show that this test was successful.

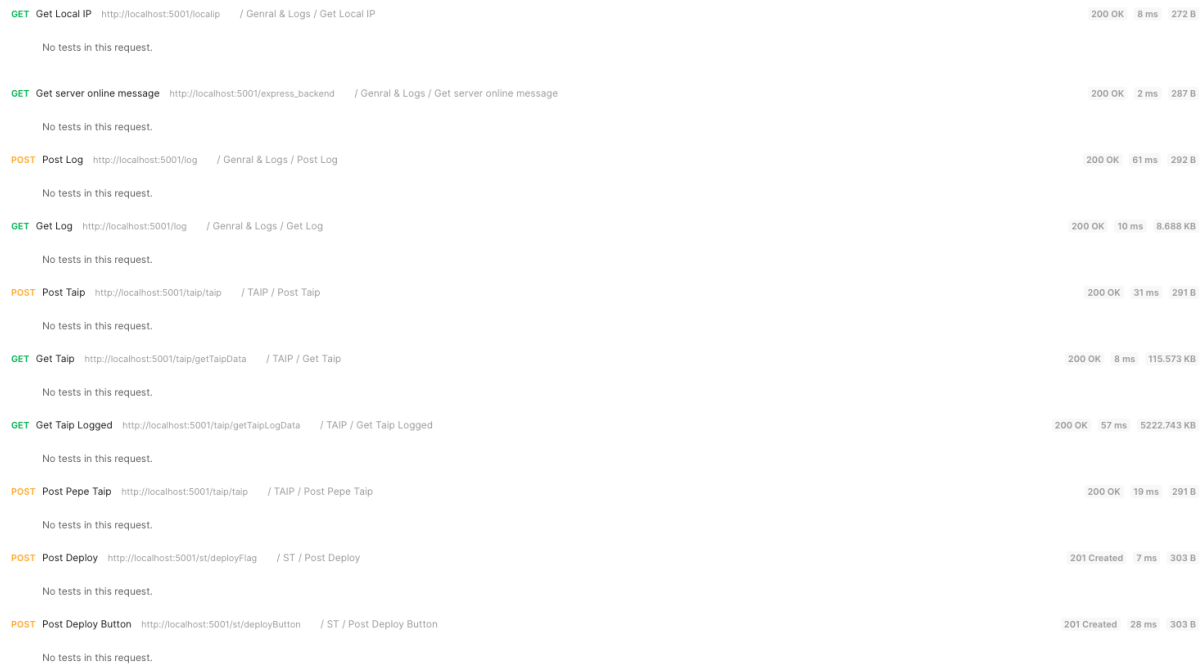


Figure 12: Postman test results of correctly formatted API requests

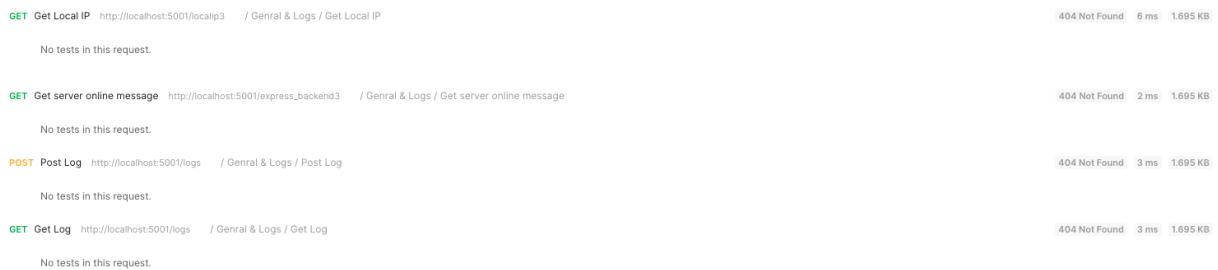


Figure 13: Postman test results of incorrectly formatted API requests

4.1.8 Software Test 8: Target Identification Alert test

The Target Identification Alert test is to ensure the Web Interface vocalises any targets the Payload identifies.


Equipment used:

The equipment used in this test consists of the following:

- Laptop
- Wireless Router
- Safari or Google Chrome browser
- MySQL Database
- Raspberry Pi
- Raspberry Pi camera

Procedure:

1. Start the MySQL Database service on the laptop
2. Start the Web Interface Server on the laptop by using the command `./node server` in the server files directory.

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-WVI-TR-02 Issue: 2.0 Page: 20 of 27 Date: 28 Oct 2022
--	---	--

3. Run the main script on the Raspberry Pi
4. Open the Web Interface by navigating to <http://localhost:3000>
5. Place the Raspberry Pi in front of a target and ensure that the Raspberry Pi Camera can see the target.
6. Once the target data comes through to the Web Interface ensure the volume on the laptop is turned up and verify that the Web Interface is vocalising the targets.


Results and Evidence:

The results from this test were that the Web Interface was able to vocalise targets that the payload has sent to the Web Interface. This is shown in the time stamped video linked below. The results from the video shows that this test was successful.

<https://youtu.be/LGQwPKWBF4g?t=34>

4.2 Hardware Tests

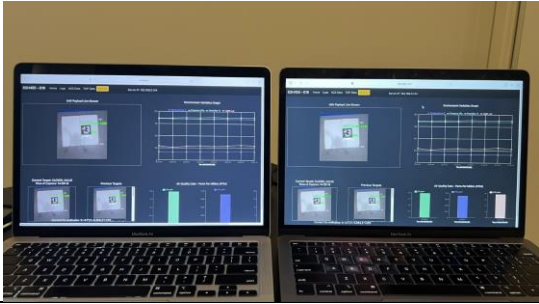

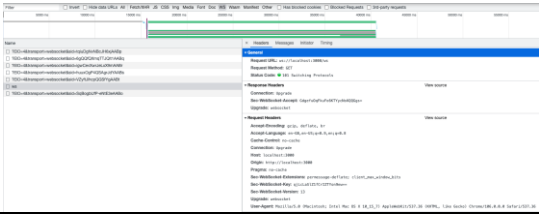

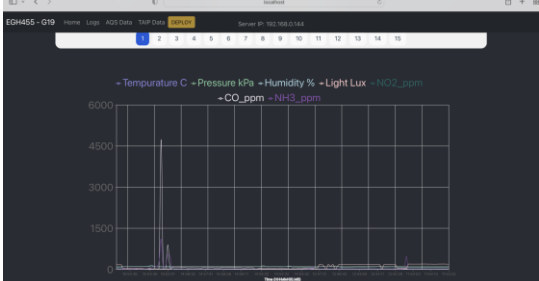
There are no hardware test completed for the WVI subsystem.


	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-WVI-TR-02 Issue: 2.0 Page: 21 of 27 Date: 28 Oct 2022
---	--	--

5 Results


Table 2 show the results of all tests that were completed. All tests passed successfully.

Table 2: Results from tests

Test Title	Result	IMG	Requirement Met
Web Interface connection over a LAN.	Success		REQ-M-03
Web Interface delay.	Success	N/A	REQ-M-19
Server data storage and requirements.	Success		REQ-M-07
Live Image Stream, AQS and TAIP data update	Success	N/A	REQ-M-05 REQ-M-06 REQ-M-03
Web Socket test.	Success		REQ-M-06
API endpoints test.	Success		REQ-M-07
Data Accessibility.	Success		REQ-M-15

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-WVI-TR-02 Issue: 2.0 Page: 22 of 27 Date: 28 Oct 2022
--	--	--

Target Identification Alert test	Success	N/A	REQ-M-04
--	---------	-----	----------

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-WVI-TR-02 Issue: 2.0 Page: 23 of 27 Date: 28 Oct 2022
--	--	--

6 Analysis

The analysis section will discuss if the aims of the test have been achieved and were there any issues involved with this.

6.1 Hardware Analysis

There were no hardware tests performed on the WVI subsystem.

6.2 Software Analysis

6.2.1 Software Test 1: Web Interface connection over a LAN.

The Web Interface connection over a LAN test is successful in that the Web Interface is accessible across multiple devices on the same LAN. If the Payload or other clients are connected on the same network the Web Interface will be accessible via any browser that supports JavaScript and HTML 5.

6.2.2 Software Test 2: Web Interface delay

The Web Interface delay test was successful in that the Web Interface will receive data from the Server within 10 seconds of the data being received from the Payload. Regardless of the information sent from the Payload, the test showed that the server receives that data in full without mutation of the data, such as sending the data to the client over multiple requests

6.2.3 Software Test 3: Server Data Storage and Requirements

The Server Data Storage and Requirements test was successful in that the data sent from the Payload to the server is saved with corresponding timestamps in a local database. The data that has been sent from the Payload to the server is accessible via the Web Interface or by using SQL queries to get the data from the database manually at any time.

6.2.4 Software Test 4: Live Image Stream, AQS and TAIP Web Interface data update


The Live Image Stream, AQS and TAIP Web Interface data update test was successful in that the data sent from both the AQS and TAIP subsystems via the Payload was dynamically updated on the Web Interface. From the TAIP subsystem an image stream was successfully established alongside dynamically updating target identification images, co-ordinate data whilst detecting an Aruco Marker, corresponding timestamps and target information. Data sent from the AQS subsystem such as air quality information and environment sensor information captured, light information and corresponding timestamps was also dynamically updated with the graphs on the user interface also updating dynamically.

6.2.5 Software Test 5: Web Socket test

The Web Socket test was successful in that the server and the Web Interface successfully established a Web Socket connection that was able to send messages that contained data. This test also showed that Web Sockets would be used to meet the requirements of receiving data dynamically within 10 seconds to the user interface as it allows the server to notify the user interface to fetch the new data.

6.2.6 Software Test 6: Data Accessibility

The Data Accessibility test was successful in that the Web Interface was able to show 10 minutes

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-WVI-TR-02 Issue: 2.0 Page: 24 of 27 Date: 28 Oct 2022
--	---	--

of data that was sent from the Payload to the server. The data is accessible from the corresponding subsystem tabs and is displayed in graphs. The logged AQS data is also displayed in a separate graph on the AQS tab to show trends in sensor data over a period of 10 minutes.

6.2.7 Software Test 7: API endpoints test

The API endpoint test was successful in that the Server was able to receive data on the open API endpoint. These API endpoints are used in the Payload main script to send data to the server. Misconfiguring the endpoints or sending data to an incorrect endpoint also returned the user an error code depending on the error.

6.2.8 Software Test 8: Target Identification Alert test


The Target Identification Alert test was successful in that when the Web Interface receives target information and data from the TAIP system running on the payload the Web Interface vocalizes what the identification of the target (E.g., Fire Extinguisher).

7 Conclusions and Recommendations


The conclusion of the test report is a successfully designed Web Interface design capable of handling all the data sent from the Payload with correct methods of storage, display and identification of the data. The Web Interface included a Server component that directly connected to the Web Interface that handled all the data sent from the Payload and notified the Web Interface to get new data. The Web Interface correctly displayed all the data within the set time frame and vocalised identified targets. The outcome of these test proved that all requirements for the Web Interface subsystem has been achieved. This can be seen in table 3.

Table 3: Requirements Met

Requirement Code	Description	Requirement Met
REQ-M-03	The UAVPayloadTAQ shall communicate with a ground station computer to transmit video, target detection and air quality data.	Successful
REQ-M-04	The target identification system shall be capable of alerting the GCS of a target's type.	Successful
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.	Successful
REQ-M-06	The Web Interface is required to display the images of the targets that are taken directly from the UAVPayloadTAQ and updated every time a new picture is taken.	Successful
REQ-M-07	The Web Interface shall be designed and run as a web server, which is to be accessible by any computers on the local network. This shall store logged sensor data and target detections with corresponding timestamps.	Successful
REQ-M-15	The system shall have logged functioning operation for a minimal period of 10 minutes prior to acceptance test.	Successful
REQ-M-19	Live data from the UAV must be made available through the web server within 10 seconds of capture.	Successful

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-WVI-TR-02 Issue: 2.0 Page: 26 of 27 Date: 28 Oct 2022
--	--	--

Recommendations for future design plans are to have data that could be filtered and toggles for both the AQS and TAIP data allowing the user to have more control over the data being sent. Further recommendations would be for new data to flow from the right-hand side of the AQS graphs as this would make the new data coming into the Web Interface easier to read. The current system is currently capable of implementing these recommendations, however, due to time constraints these features were not implemented, moreover, the lack of implementation did not negatively impact the requirements met.

 Queensland University of Technology	QUT Systems Engineering UAVPAYG19	Doc No: UAVPAYG19-WVI-TR-02 Issue: 2.0 Page: 27 of 27 Date: 28 Oct 2022
--	--	--

Appendix

No appendix