
 Queensland University of Technology	QUT Systems Engineering UAVPayload^{TAQ}-G2	Doc No: SD-UAVPayload ^{TAQ} -G2-WEB-01 Issue: 1 Page: 1 of 14 Date: 19 October 2020
--	--	---

UAVPayload^{TAQ}

Subsystem Design – Web Visualisation

Prepared by	BS _____ Brian Sivertsen, WEB lead	Date	_____
Checked by	AI _____ Alexander Inftene, Project Manager	Date	_____
Approved by	BS _____ Brian Sivertsen, WEB lead	Date	_____
Authorised for use by	_____	Date	_____
	Dr. Felipe Gonzalez, Project Coordinator		

This document is Copyright 2020 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 UAVPayload^{TAQ}-G2	Doc No: SD-UAVPayload ^{TAQ} -G2-WEB-01 Issue: 1 Page: 2 of 14 Date: 19 October 2020
--	--	---

Revision Record

Document Issue/Revision Status	Description of Change	Date	Approved
1.0		19 October 2020	Brian Sivertsen


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 Subsystem Introduction.....	8
4 Subsystem Architecture.....	9
4.1 Interfaces	9
5 Subsystem Design Description.....	10
6 Conclusion.....	11
8Appendices	12
6.1 WEB Subsystem Architecture.....	12
6.2 WEB UI.....	12
6.3 Sensor Data JSON object	13
6.4 FLASK Server.....	13
6.5 Front End API Call.....	14

 Queensland University of Technology	QUT Systems Engineering UAVPayload^{TAQ}-G2	Doc No: SD-UAVPayload ^{TAQ} -G2-WEB-01 Issue: 1 Page: 4 of 14 Date: 19 October 2020
--	--	---


List of Figures

Figure	Page No.
Figure 1: WEB Subsystem Architecture	12

 Queensland University of Technology	QUT Systems Engineering UAVPayload^{TAQ}-G2	Doc No: SD-UAVPayload ^{TAQ} -G2-WEB-01 Issue: 1 Page: 5 of 14 Date: 19 October 2020
--	--	---

Definitions

UAV	Unmanned Aerial Vehicle
UAVPayload	Unmanned Aerial Vehicle Payload
GCS	Ground Control Station
WEB	Web Visualization Subsystem
FTP	File Transfer Protocol
HLO	High Level Objective
SR	System Requirements
POC	Proof of Concept
LAN	Local Area Network
TBD	To Be Determined

 Queensland University of Technology	QUT Systems Engineering UAVPayload^{TAQ}-G2	Doc No: SD-UAVPayload ^{TAQ} -G2-WEB-01 Issue: 1 Page: 6 of 14 Date: 19 October 2020
--	--	---

1 Introduction

This subsystem design outlines the WEB subsystem for the UAVPayload project. By following the systems engineering design system this subsystem forms the web design and visualisation subsystem. This subsystem design document outlines the integration and operation of the WEB subsystem.

1.1 Scope

This subsystem design outlines the WEB subsystem and how it interfaces with the other subsystems within the UAVPayload project. High level architecture of the WEB subsystem as well as diagrams and pseudocode and small code snippets are used to outline the subsystem design.

The WEB system is required to visualise the temperature, pressure, humidity, light, noise level, gas sensors, and imagery data. The interface should be designed and run as a web server but can be hosted on a local machine. The web interface must be accessible from another computer on the Local Area Network (LAN). Taken from RD/1.

1.2 Background

The Queensland University of Technology (QUT) Airborne Sensing Lab have appointed Group 2 of the EGH455 (Advanced Systems Design) class to design a UAV Payload for indoor air quality to be installed on a S500 UAV designed for navigating in GPS denied environments. The UAVPayloadTAQ is required to conduct constant air quality sampling in a simulated underground mine. During monitoring, it must find and identify multiple markers placed by miners around the mine. Additionally, QUT Airborne Sensing Systems requires that the UAVPayloadTAQ is designed and developed using Systems Engineering to ensure QUT Airborne Sensing Systems requirements are met. Taken from RD/3.


2 Reference Documents

2.1 QUT Avionics Documents

RD/1	CN-UAVPayloadTAQ-01	UAV Payload Customer Needs
RD/2	SR-UAVPayloadTAQ-01	UAV Payload System Requirements
RD/3	PMP-UAVPayloadTAQ-01	UAV Payload Project Management Plan
RD/4	SD-UAVPayloadTAQ-G2-WEB-01	Preliminary Web Visualisation Subsystem Design
RD/5	SD-UAVPayloadTAQ-G2-DAT-01	Preliminary Sensor Data Subsystem Design
RD/6	SD-UAVPayloadTAQ-G2-IMG-01	Preliminary Image Processing Subsystem Design
RD/7	SD-UAVPayloadTAQ-G2-ENC-01	Preliminary Payload Enclosure Subsystem Design
RD/8	TR-UAVPayloadTAQ-G2-ENC-01	Subsystem Unit Testing – Web Visualisation

2.2 Non-QUT Documents

RD/8	https://flask.palletsprojects.com/en/1.1.x/	Flask User Guide
RD/9	https://expressjs.com/	ExpressJS Documentation
RD/10	https://reactjs.org/	React Documentation
RD/11	https://chakra-ui.com/getting-started	Chakra UI Documentation
RD/12	https://github.com/jerairrest/react-chartjs-2	Chartjs React Integration


 Queensland University of Technology	QUT Systems Engineering UAVPayload^{TAQ}-G2	Doc No: SD-UAVPayload ^{TAQ} -G2-WEB-01 Issue: 1 Page: 8 of 14 Date: 19 October 2020
--	--	---

3 Subsystem Introduction

The WEB subsystem is required to display the data and real-time in-flight information. Information such as the temperature, pressure, light, humidity noise level, gas sensor data and imagery data need to be displayed. Data will be transmitted via the Raspberry Pi 3B+ to a web server on a local machine which is to make the interface available on the Local Area Network to be accessed by another machine.

The requirements above are taken from RD/1 in relation to the HLO-M-3, this high-level objective can be broken up into various requirements. The relevant requirements for the WEB subsystem taken from RD/2 are:

- i. REQ-M-3: The UAVPayload shall communicate with a ground station computer to transmit video, target detection and air quality data.
- ii. REQ-M-4: The target identification system shall be capable of alerting the GCS of a targets type.
- iii. REQ-M-5: The Web Interface is required to display real time air sampling data that is recorded directly from the UAVPayload and updated dynamically throughout the duration of the flight.
- iv. REQ-M-6: The Web Interface is required to display the images of the targets that are taken directly from the UAVPayload and updated every time a new picture is taken.
- v. REQ-M-7: The Web Interface shall be designed and run as a web server, which is to be accessible by any computers on the local network.
- vi. REQ-M-12: The UAVPayload shall process all imagery on-board via the on-board computer.
- vii. REQ-M-15: 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 UAVPayload^{TAQ}-G2	Doc No: SD-UAVPayload ^{TAQ} -G2-WEB-01 Issue: 1 Page: 9 of 14 Date: 19 October 2020
--	--	---

4 Subsystem Architecture


The WEB subsystem architecture is outlined in Appendix 1. The Subsystem itself consists of a Flask Server, Router, GCS/Web Server and an End User Computer. These four components within the subsection communicate with each other over the LAN in order to satisfy the requirements listed in the subsection introduction.

A further breakdown of the Flask Server and GCS/Web Server is given in section 5, Subsystem Design Breakdown.

4.1 Interfaces

As can be seen in Appendix 1, Web Subsystem Architecture, the WEB Subsystem Interfaces with the Sensor Data and Imagery Data through the Raspberry Pi 3B+. The Sensor Data will supply the Flask server with the relevant sensor data to achieve REQ-M-3 of RD/2, traced from HLO-M-03 of RD/1.

The WEB Subsystem also interfaces with the Image Processing subsystem, this subsystem will supply the imaging data required to satisfy REQ-M-3 of RD/2.

 Queensland University of Technology	QUT Systems Engineering UAVPayload^{TAQ}-G2	Doc No: SD-UAVPayload ^{TAQ} -G2-WEB-01 Issue: 1 Page: 10 of 14 Date: 19 October 2020
--	--	--

5 Subsystem Design Description


The third HLO, HLO-M-3 from RD/1 outlines that a web interface with the purpose of visualising real time flight data has been developed. This data needs to be able to be viewed by any device connected to the LAN. As the Raspberry Pi 3B+ which is to be used for the UAVPayload is very limited in resources as much as is allowed of the WEB subsystem will be computed by a separate Web Server Machine.

As stated in REQ-M-12 of RD/2 all of the image processing must be completed on board the UAVPayload which will take up much of the compute power of the Raspberry Pi and will need as much compute time as possible to gain fast results and reliable data. In order to reduce the compute power utilised by the WEB subsystem raw sensor data will be transmitted via a Flask server, RD/9, on the Raspberry Pi to the Web Server. The sensor data will be transmitted via a JSON object, an example of which can be seen in appendix 6. Imagery data will be sent via the same Flask server with a separate endpoint, any detections and the time of the request will be sent via the JSON object in Appendix 3, the “id” is added to the json object by reading it from an input file. The static images after processing will be sent via an FTP request.

The Web Server acts as an interface between any End User Computers (Appendix 1) and the UAVPayload. The web server is running an Express server, RD/10, and serves a React, RD/11, application to the end user. The React application sends requests to the Flask server (appendix 5) operating on the Raspberry Pi and stored the data it receives from the image processing and sensors within an array locally, the data from the local array will be used to render out a visualisation to the end user. Once the local array reaches a length of 50 datapoints, old data will be popped and new data will be pushed, allowing the application to stay relatively lightweight and only display current relevant data.

RD/1 supplies an example web interface layout, but this layout contains data that is not required and not necessary, such as the UAV position and UAV state. Because of this a new web interface layout has been designed which contains only relevant data and a much clearer interface (Appendix 2). Chakra UI, RD/12, will be used to simplify the design and implementation of various components within the web interface as it integrates well with the React application and supplies components such as the Image wrapper.

A ChartJS React integration RD/13 is used to graph and display the data, an example of this is shown in appendix 2.

 Queensland University of Technology	QUT Systems Engineering UAVPayload^{TAQ}-G2	Doc No: SD-UAVPayload ^{TAQ} -G2-WEB-01 Issue: 1 Page: 11 of 14 Date: 19 October 2020
--	--	--

6 Conclusion

This subsystem design document outlines all the requirements pertaining to the WEB subsystem outlined in RD/2 have defined the overall design of the subsystem. This subsystem design serves as an overview of the overall design for the subsystem.

The WEB Subsystem meets all the requirements addressed in section 4 by using various frameworks and languages to provide the optimal result. Evidence of succession within all these criteria can be found in RD/8.

8 Appendices

6.1 WEB Subsystem Architecture

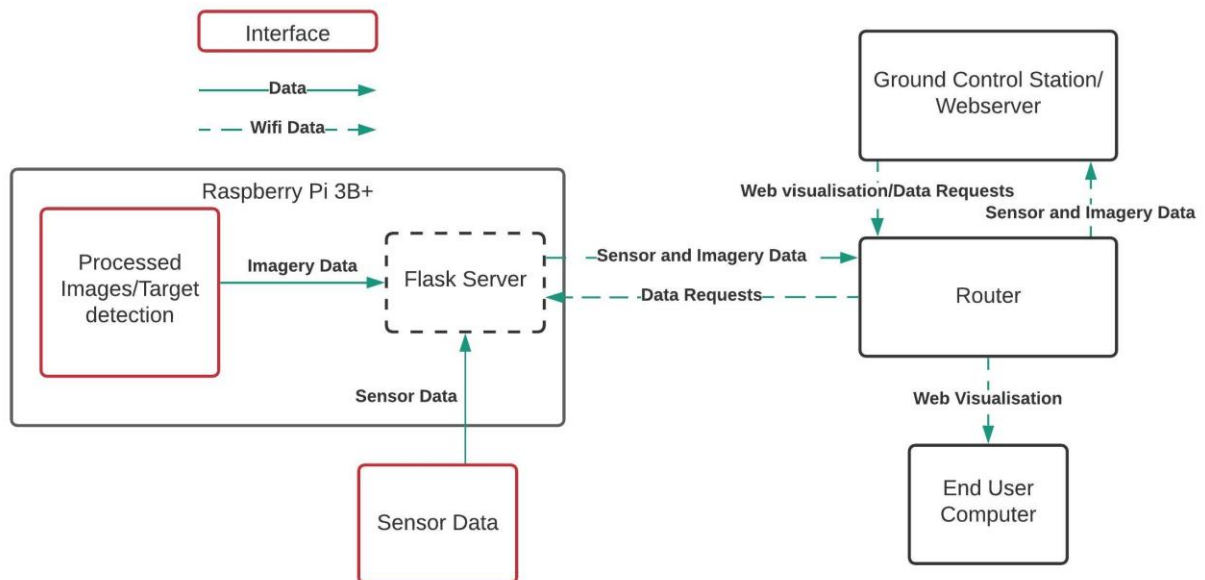


Figure 1: WEB Subsystem Architecture

6.2 WEB UI





6.3 Sensor Data JSON object

```
data = {  
    humidity: 36.312575027605625,  
    id: "0",  
    light: 45.2082,  
    nh3: 0.023396810545597963,  
    noise: 15.996364231394265,  
    oxidised: 0.24942645135306174,  
    pressure: 1013.8826293833631,  
    reduced: 1.0039478116889995,  
    temperature: 6.132050825639343,  
    time: 1603106876.655459  
}
```

6.4 FLASK Server

```
import time  
from flask import Flask, send_file  
from random import seed  
from random import randint  
from flask import jsonify  
import sys  
# insert at 1, 0 is the script path (or '' in REPL)  
sys.path.insert(1, '/home/pi/enviropius/examples')  
sys.path.insert(2, '/home/pi/pythonscripts')  
  
import dataSense  
  
app = Flask(__name__)  
  
@app.route('/data')  
def get_current_data():  
    id = open("/home/pi/pythonscripts/detectionFile.txt", "r")  
    d = dataSense.returnSensors()  
    d["random"] = randint(0, 100)  
    d["id"] = id.read()  
    return jsonify(d)  
  
@app.route('/get_image')  
def get_image():  
    return send_file('/home/pi/photos/output/outputimage.jpg')
```



6.5 Front End API Call

```
useEffect(() => {  
  fetch('/data').then(res => res.json()).then(data => {  
    data.Time=Date.now();  
    setData(data);  
    let newGraphs = graphs;  
  
    if (graphs.length < 50)  
      newGraphs.push(data);  
    else {  
      newGraphs.shift()  
      newGraphs.push(data);  
    }  
  
    if (data.id == "1"){  
      let toxic = new Audio("./toxic.ogg")  
      toxic.play();  
    } else if ( data.id == "2"){  
      let dangerous = new Audio("./dangerous.ogg")  
      dangerous.play();  
    } else if ( data.id == "3"){  
      let aruco = new Audio("./aruco.ogg")  
      aruco.play();  
    }  
  
    setGraphs(newGraphs);  
  });  
});
```