Preliminary Design Web Visualisation

Prepared by ____________________________________ Date ________________
Daniel Huffer, Web Visualization

Checked by ____________________________________ Date ________________

Approved by ____________________________________ Date ________________
Charlie Shaw-Feather, Project Manager

Authorised for use by ____________________________________ Date ________________
Dr. Felipe Gonzalez, Project Coordinator

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

Revision Record

Document Issue/Revision Status	Description of Change	Date	Approved
1.0	Initial Issue	02/05/2017	

Table of Contents

Paragraph **Page No.**

Definitions

UAV	Unmanned Aerial Vehicle
QUT	Queensland University of Technology
GCS	Ground Control Station
WVI	Web Visualisation Interface
LAN	Local Area Network
HLO	High Level Objective
LAMP	Linux, Apache, MySQL, PHP
ROS	Robot Operating System
UAVTAQ	Unmanned Aerial Vehicle: Target Air Quality Monitoring
HTML	Hypertext Mark-up Language
PHP	Hypertext Pre-processor
JS	JavaScript
MySQLi	MySQL improved
API	Application programming interface

1 Introduction

Following the systems engineering approach, the next step following the completion of background research and trade studies is to begin the preliminary design. The preliminary design is important for both the customer and the project team, it provides traceability of design decisions which correlate to customer needs and subsystem requirements satisfaction.

1.1 Scope

In this preliminary design document, the visual web interface is initially introduced with a description of where the subsystem fits within the overall system. The system architecture along with the operation of the system, functions and interfaces are then detailed. Finally, the preliminary designs of the software are presented with evidence and information justifying design decisions.

1.2 Background

Queensland University of Technology Airborne Sensing Systems are a world-leading research team based in Brisbane, Australia. Focusing on the innovation of autonomous UAV flight using on board sensing equipment they strive to turn cutting edge technology along with leading edge concepts from paper into flight tested reality. As a part of the team they have commissioned students EGB349 to design and build a UAVTAQ multi rotor based UAV to take targeted air samples within a simulated mine. The idea is that there are potentially deadly C02 leaks occurring down within a mine which otherwise cannot be physically inspected due to the potential risks involved. This is where the UAVTAQ is required. The UAV will fly in using sensor navigation the known locations and take air samples to confirm the leaks. This information can then be delivered safely to teams outside the mine.

2 Reference Documents

2.1 QUT Avionics Documents

RD/1	UAVTAQ - Customer Requirements	UAVTAQ Customer Requirements 2017
RD/2	PMP of UAVTAQ	The PMP for UAVTAQ Group 1
RD/3	UAVTAQ17-PM-SR	System Requirements
RD/4	TAG16GM2-VWI-TS-02	Web Interface Subsystem Trade Study

2.2 Non-QUT Documents

RD/5 "Apache, The server project", *apache.org*, 2017. [Online]. Available https://httpd.apache.org/. [Accessed: 21- May- 2017].

RD/6 "ROS", *ros.org*, 2014. [Online]. Available hhttp://wiki.ros.org/ROS/Introduction [Accessed: 21- May- 2017].

RD/7 "World's most popular open source Database", *www.oracle.com,* 2017. [Online]. Available https://www.oracle.com/mysql/index.html. [Accessed: 21- May- 2017].

RD/8 "jQuery", jquery.com/, 2017. [Online]. Available https://http. jquery.com/. [Accessed: 21- May- 2017].

3 Subsystem Introduction

The web visual interface subsystem (WVI) derives from a High-Level Objectives (HLO's) specified in the document RD/1, UAVTAQ - Customer Requirements. It is stated that "The system shall have capability to visualise real time flight information, CO2 concentration, temperature, humidity, and imagery data". It is also mentioned that "The web interface must be accessible from another computer on the Local Area Network (LAN)" and "The interface should be designed and run as a web server, but can be hosted on a local machine". These HLO requirements were translated into equivalent system requirements (REQ-M-7-1, REQ-M-7-2, REQ-M-7-3). The core of the WVI is what's commonly referred to as a web development solution stack. A solution stack consists of a bundle of different software with unique functionality designed to interact with one another. The basis of this is that you need a web server to handle client requests, a database to store information and one form or another of scripting language to manipulate data based on client requests.

4 Subsystem Architecture

The subsystems architecture and interface control diagram can be seen below in Figure 1.

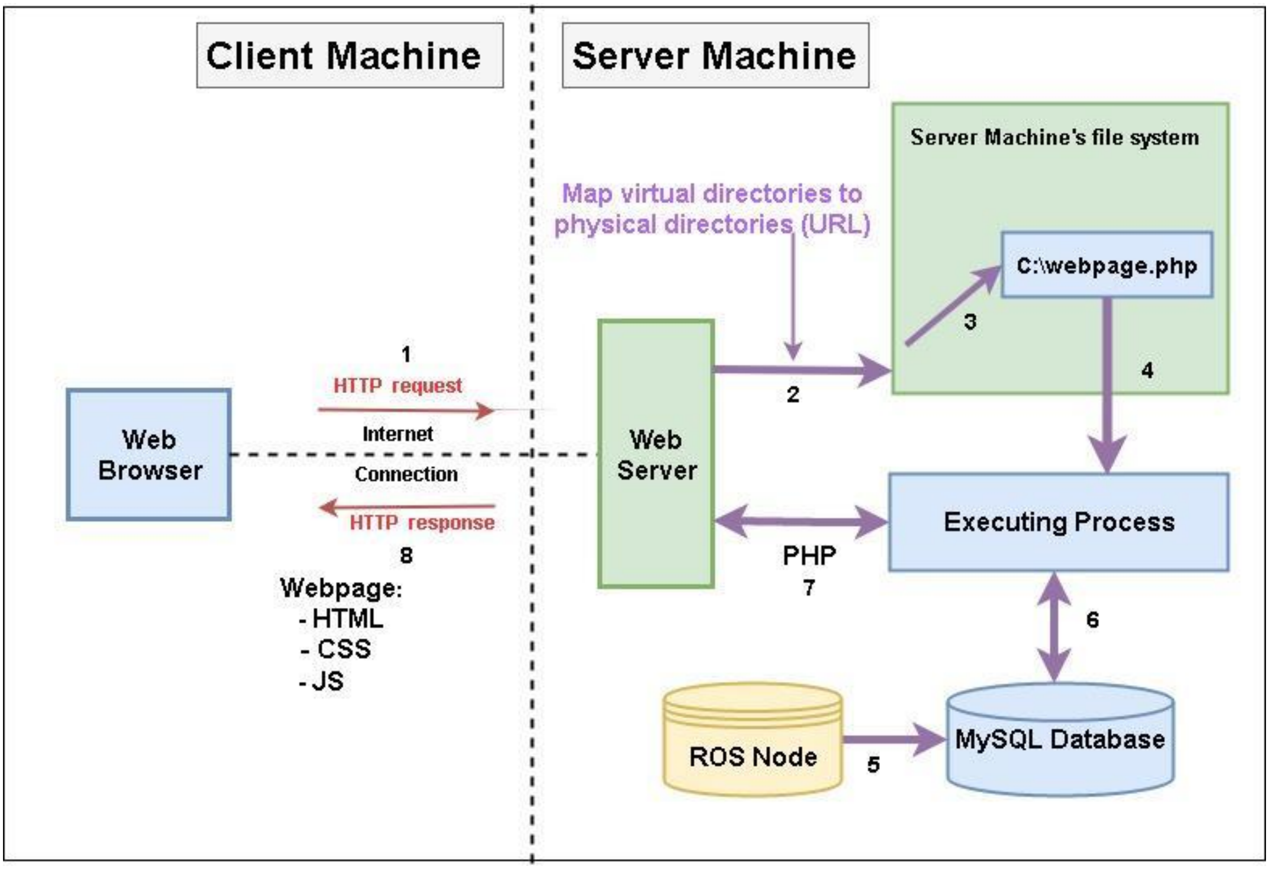


Figure 1 - Web Visualization Subsystem Overview

4.1 Interfaces

Interface 1 Client web browser makes a request, upon request the webserver receives a URL.

Interface 2 Web Server uses URL to map virtual directories to physical directories on the local machine.

Interface 3 Web Server locates PHP file on the machine.

Interface 4 PHP script is executed.

Interface 5 UAV Data from the ROS network is sent to the MySQL database.

Interface 6 PHP can retrieve and modify data stored in the MySQL database.

Interface 7 PHP interfacing with apache webserver, execution output streamed down the HTTP connection.

Interface 8 HTML, CSS and JS and any additional output sent back over the internet / network to the requesting party.

5 Design

Upon finalising and re-evaluating system requirements (RD/3) and VWI trade study (RD/4) It was determined that the LAMP stack is the best solution. This choice primarily boils down to the fact that "ROS currently only runs on Unix-based platforms, where software for ROS is primarily tested on Ubuntu and Mac OS X systems" (RD/6). Further research lead to the understanding that the Ubuntu is the most stable option for working with ROS; Ubuntu is a Debian-based Linux operating system (https://www.ubuntu.com/about/about-ubuntu/ubuntu-and-debian) and LAMP is Apache, MySQL and PHP running on the Linux operating system so it will be a compatible fit.

5.1 Hardware Design

The entirety of the WVI sub system is comprised of various pieces of software. The only hardware interaction been that of the GCS and client computer. Additionally, there are no requirements for these pieces of hardware other than they operate to an acceptable standard. It's for these reasons that no hardware design is specified.

5.2 Software Design

The WVI subsystem is responsible for displaying real time flight and environmental information in a web interface. The interface will be formatted in an easy to understand manner where it can be accessed from any PC on the network. This will be accomplished using the LAMP stack and html coding. In order to run the LAMP stack the ground control station is either going to have Ubuntu installed as the systems primary operating system, or the LAMP stack will run on a virtual machine within a windows operating system. Either one of the two options will be viable, although having Ubuntu installed as the primary operating system will most likely be the best option, as partitioning computer resources to run the oracle virtual machine can cause performance issues and all most the overall UAV software will be running on Linux.

5.2.1 HTML & CSS

Much of the code for the subsystem will be written in HTML and CSS, the HTML code accounts for the structure of the webpage. While the CSS enables the customisation and styling of all structured HTML elements.

5.2.2 LAMP

The LAMP software bundle is completely open source and as previously mentioned runs on the

Linux operating system, a further break down of the components can be seen below.

5.2.2.1 APACHE

Apache provides a secure, efficient and extensible server that provides HTTP services in sync with current HTTP standards (RD/5) As part of the solution stack version 2.4.x from the stable branch will be used.

5.2.2.2 MySQL

MySQL is a relational database management system based on Structured Query Language (SQL). SQL is labelled one of the most popular languages for managing database content due to its flexibility and ease of use (RD/7). The MySQL database will be interfaced with PhpMyAdmin to assist with development of the WVI. All the data except the POV image from the UAV that the customer would like to be able to visualise will be sent to tables constructed in MySQL. The live POV images will be stored on a local directory to avoid additional overheads associated with image storage within MySQL. MySQL queries in PHP will be done using the PDO API over MySQLi. For the subsystem, the latest release MySQL 5.7.x will be used.

5.2.2.3 PHP

PHP which is a recursive acronym for PHP: Hypertext Pre-processor, is a server side scripting language for web development. Writing PHP code into your HTML files essentially allows you to generate HTML code dynamically, manipulate data and interface with databases. The latest release PHP 7.1.x will be used to interface with Apache and MySQL.

5.2.3 jQuery & Plotly

jQuery is a feature rich JavaScript library that makes tasks like HTML document traversal and manipulation, event handling, animation and Ajax much simpler to do (RD/8). The jQuery library will be used in the WVI to aid in generate graphs on the WVI. API from plotly will also be used to aid in the creation of 3D graphing within the JavaScript framework. The 3D graph will be used as a model to map the current and previous UAV locations.

5.2.4 Software Flow Diagram

To help visualise the systematic flow and function of the software used in the WVI subsystem a small flow diagram can be seen below in Figure 2.

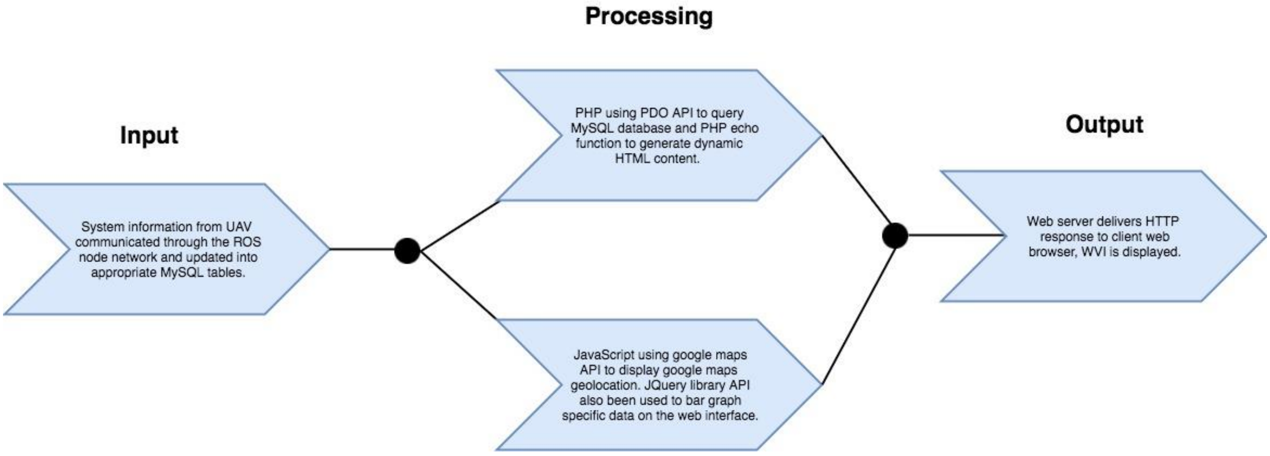
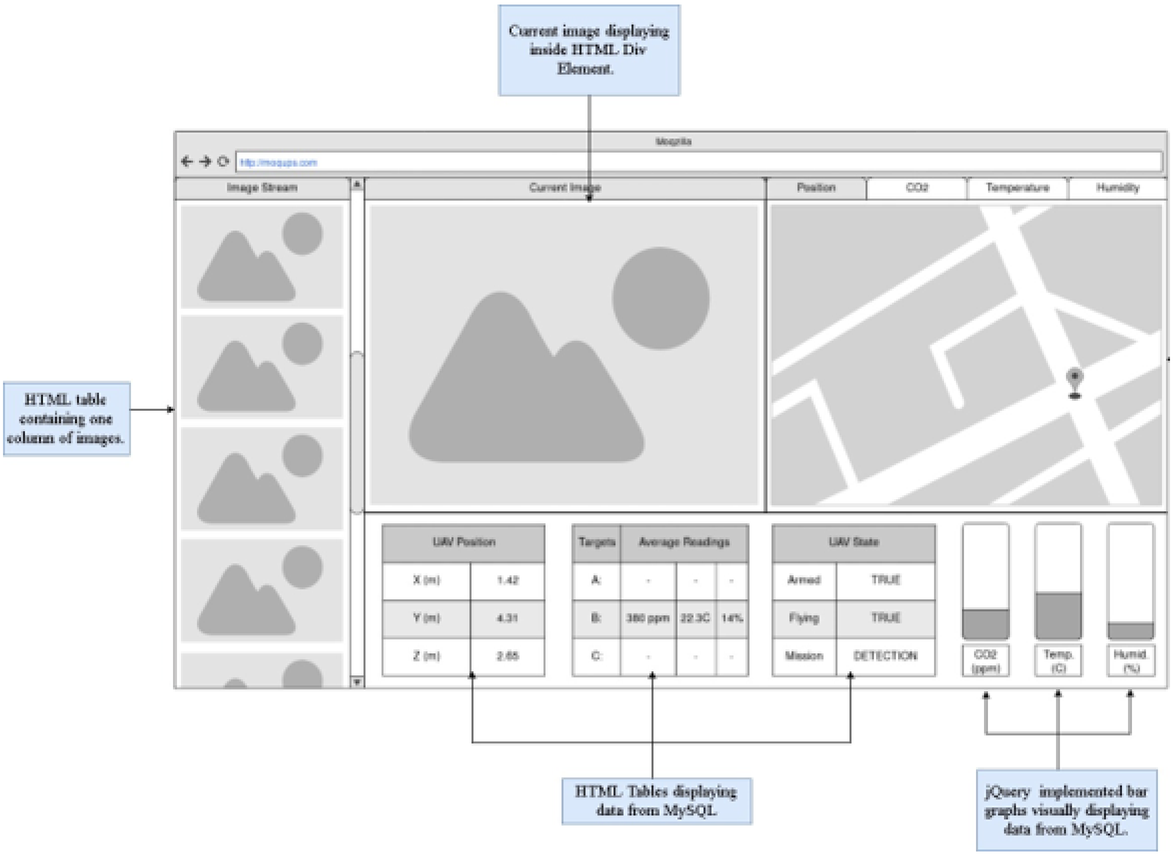

Processing
Input
Output
System information from UAV communicated through the ROS node network and updated into appropriate MySQL tables.
PHP using PDO API to query MySQL database and PHP echo function to generate dynamic HTML content.
JavaScript using google maps API to display google maps geolocation. JQuery library API also been used to bar graph specific data on the web interface.
Web server delivers HTTP response to client web browser, WVI is displayed.

Figure 2- Software Flow

These are the three general processes that are required for the subsystem. First, required data must be obtained through the ROS communication network and uploaded into MySQL database tables (Input). This will most likely be an action that will be configured such that the MySQL table data is refreshed upon a client HTTP request. Secondly web development software PHP and JS will manipulate all the data from the MySQL tables generating a dynamic HTML page for the HTTP request (Processing). Lastly the web server will deliver the response back to the requesting user in the form of JS/HTML/CSS (Output).

5.2.4 Example Mark-up Diagram

An example layout of the visual web interface can be seen above in Figure 3. This example layout was provided as part of the customer needs documentation (RD/1). The core elements of the example have been labelled with a small description briefly relating the mentioned software in this document to its display functionality.



Current image displaying inside HTML Div Element.

HTML table containing one column of images.

HTML Tables displaying data from MySQL

jQuery implemented bar graphs visually displaying data from MySQL.

UAV Position	
X (m)	1.42
Y (m)	4.31
Z (m)	2.65

Targets	Average Readings		
A:	-	-	-
B:	380 ppm	22.3C	14%
C:	-	-	-

UAV State	
Armed	TRUE
Flying	TRUE
Mission	DETECTION

Figure 3 - Example Mark-up Display

6 Conclusion

In conclusion, the web visualisation subsystem has minimal hardware components and is comprised almost entirely of different software components. The subsystem trade studies and preliminary design research have determined the all the best components for the subsystem to satisfy all subsystem requirements (RD/3). The Apache webserver within the LAMP stack makes it possible to host the interface making it available to users over the local area network satisfying requirements (REQ-M-6-1), (REQ-M-6-3). JavaScript and its feature rich library JQuery provide all the functionality required to display visual aids for the data been handled by PHP and MySQL, satisfying the final requirement (REQ-M-6-2). Lastly and most importantly, compatibility of components within the subsystem and compatibility of the subsystem with the overall system has been confirmed.

7 Appendix A – Using MySQL

Code examples for working with the database using PDO API in PHP.

1. Connecting to the database

```
$dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass);
```

In the following connection to the MySQL database the host is set to localhost which Is what will be used during implementation. The database name will be changed to whatever is setup inside PhpMyAdmin. The PHP variables $user and $pass are the username and password for the MySQL database and are usually set to "root" and an empty password by default.

2. Querying the database with prepare

```
$sth = $dbh->prepare('SELECT * FROM foo');
```

In the query example the PDO prepare statement executes an SQL statement and returns a result that is set as a PDOStatment object. The SQL statement itself 'SELECT * FROM foo' is selecting everything from a table named foo which would be defined in the test database in section 1.

3. Executing the prepare statement

```
$sth->execute();
```

Once the query has been prepared it can be executed using the execute function as can be seen in the above pseudocode.

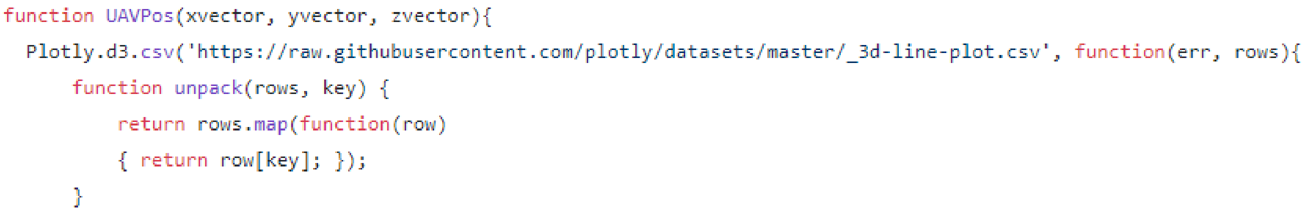
4. Working with the data returned as a PDOStatement.

```
$result = $sth->fetchAll();
```

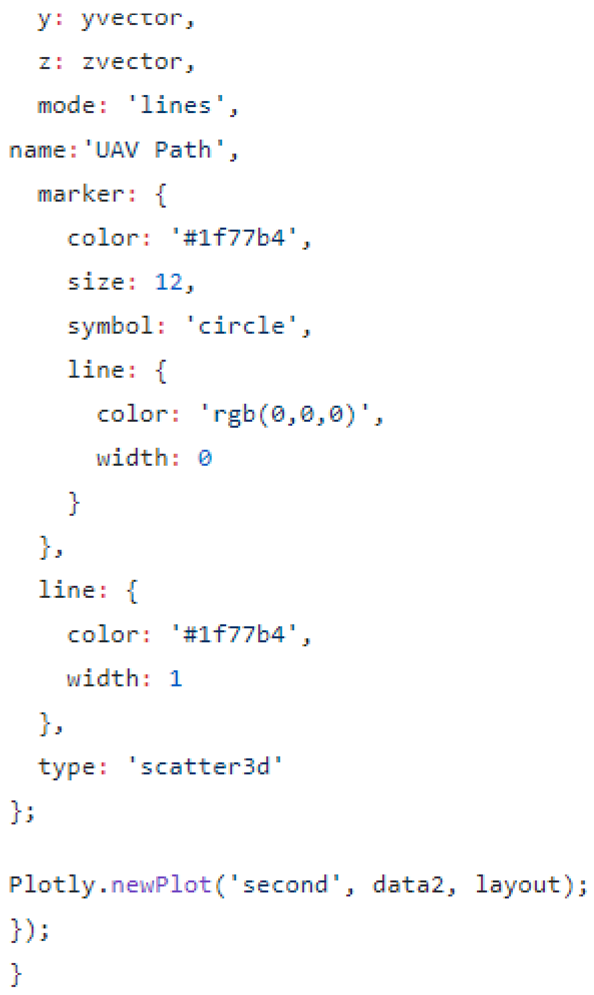
To work with the data that was returned as a PDOStatement object, there is still one more step. The data must be "fetched", there are several different functions and methods you can use to extrapolate the data returned from the query. In the following example above fetchAll() is called which returns a PHP array with containing all the results.

8 Appendix B – Plotly 3D JavaScript Graphing

Below is an example JavaScript mark-up function using plotly API. In the code below, the function UAVPos takes in three input parameters xvecor, yvector, zvector. All three of the input parameters are arrays containing the positional information for each axis of the plot. This data will be plotted to represent the current flight path of the UAV, current goals and identified targets will also be marked within the 3D Map.



```
function UAVPos(xvector, yvector, zvector){
 Plotly.d3.csv('https://raw.githubusercontent.com/plotly/datasets/master/_3d-line-plot.csv', function(err, rows){
    function unpack(rows, key) {
        return rows.map(function(row)
        { return row[key]; });
    }
```



```
  y: yvector,
  z: zvector,
  mode: 'lines',
name:'UAV Path',
  marker: {
    color: '#1f77b4',
    size: 12,
    symbol: 'circle',
    line: {
      color: 'rgb(0,0,0)',
      width: 0
    }
  },
  line: {
    color: '#1f77b4',
    width: 1
  },
  type: 'scatter3d'
};

Plotly.newPlot('second', data2, layout);
});
}
```