Test Report

|  |  |
| --- | --- |
| **Project**: UAVPayloadTAQ-TAIP-TEST  **WP Name**: Component Testing  **WP Code**: WP-TAIP | **Type of Test**: Unit Test |
| **System Requirements**:  **[REQ-M-03]**  **[REQ-M-04]**  **[REQ-M-08]**  **[REQ-M-10]**  **[REQ-M-12]**  **[REQ-M-13]**  **[REQ-M-14]** | **Test Equipment**:  Documented in Section 4 |
| **Test Article**: 1 Laptop with Raspberry Pi OS(32-bit)Lite running on 1 Raspberry Pi 3 Model B, Raspberry Pi camera module |  |
| **Test Operator**: Arik Intenam Mir | **Test Engineer**: Arik Intenam Mir |
| **WP Group Manager**: Celine Blumer | **WP Supervisor**: Dr. Felipe Gonzalez |

**Revision Record**

|  |  |  |  |
| --- | --- | --- | --- |
| Document Issue/Revision Status | **Description of Change** | **Date** | **Approved** |
| 1.0 | Initial Issue | 17 October 2020 | … |

Queensland University of Technology

Gardens Point Campus

Brisbane, Australia, 4001.

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

Contents

[Contents 4](#_Toc41563574)

[**List of Figures** 6](#_Toc41563575)

[**Definitions** 7](#_Toc41563576)

[1 Introduction 8](#_Toc41563577)

[1.1 Scope 8](#_Toc41563578)

[1.2 Background 8](#_Toc41563579)

[2 Reference Documents 9](#_Toc41563580)

[2.1 QUT Avionics Documents 9](#_Toc41563581)

[2.2 Non-QUT Documents 9](#_Toc41563582)

[3 Test Objectives 10](#_Toc41563583)

[3.1 Future Test Objectives 11](#_Toc41563584)

[4 Test Setup and Equipment 12](#_Toc41563585)

[4.1 Test Setup 12](#_Toc41563586)

[4.2 Test Equipment 13](#_Toc41563587)

[4.2.1 Raspberry Pi 3 Model B 13](#_Toc41563588)

[4.2.2 Raspberry Pi Camera Module V2.1 14](#_Toc41563589)

[5 IMG-TAQ Testing Procedure 16](#_Toc41563590)

[5.1 Initial Hardware Inspection 16](#_Toc41563591)

[5.2 Initial Software Initialisation 16](#_Toc41563592)

[5.3 Raspicam\_node Test 16](#_Toc41563593)

[5.4 RQT image-view Test 17](#_Toc41563594)

[6. Results 18](#_Toc41563595)

[6.1 Initial Hardware Inspection 18](#_Toc41563596)

[6.2 Initial Software Initialization 19](#_Toc41563597)

[6.3 Raspicam\_node and Rqt image-view Tests 19](#_Toc41563598)

[7. Analysis 21](#_Toc41563599)

[8. Conclusions and Recommendations 22](#_Toc41563600)

[8.1 Conclusions 22](#_Toc41563601)

[8.2 Recommendations 22](#_Toc41563602)

**List of Figures**

[Figure 1: Test Setup 10](#_Toc54098549)

[Figure 2:Raspberry Pi 3 Model B 11](#_Toc54098550)

[Figure 3:Raspberry Pi Camera Module 12](#_Toc54098551)

[Figure 4:USB to serial cable 13](#_Toc54098552)

[Figure 5: Serial pins connected to Pi 13](#_Toc54098553)

[Figure 6: ARUCO marker detected 18](#_Toc54098554)

**Definitions**

|  |  |
| --- | --- |
| UAVPayloadTAQ | Unmanned Aerial Vehicle: PayloadTAQ |
| IP  UAV | Image Processing  Unmanned Aerial Vehicle |
| QUT  G16  REQ  HLO  SSH  OpenCV | Queensland University of Technology  Group 16  Requirement  High Level Objective  Secure Shell  Open Source Computer Vision Library |
|  |  |

# Introduction

Proper unit testing of any subsystems in frequent manners throughout the production phase is the key in producing a well-structured system. The unit tests for each subsystem, will allow to get rid of the potential problems and concerns prior to integration with other subsystems. This allows the implementation of high-quality subsystems and also assures the team that any sorts of errors, are due to integration, not because of a specific subsystem. This test report document will be aiming towards conducting a system engineering approach to achieve the customer needs as per the system requirements REQ-M-08 and REQ-M-10.

## Scope

This test report document will ai to present the testing procedure for the image processing subsystem, which includes testing the Raspberry Pi microcomputer, Raspberry Pi camera module and the relevant software components. The test tasks and the results listed in this document are to be used by the UAVPayloadTAQG16, the customer and future peers to see how the subsystem was tested and functionality verified.

## Background

Group 16 from EGH 455 was appointed by the QUT Airborne Sensing systems to design and construct a UAV (unmanned aerial vehicle) payload that will be installed on a S500 UAV that can navigate under a GPS denied environment. The payload constructed must have the capability to measure the air quality in a simulated underground mine environment and can detect multiple markers placed around the mine. The payload will be incorporating a board camera and other sensors to autonomously measure air quality, identify, and locate three targets. Furthermore, the telemetry, imagery and air quality data will be displayed through a web interface in real time.

# Reference Documents

## QUT Avionics Documents

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | RD/1 | -SUP-Customer Needs | 2020: Customer needs | | RD/2 | SR-UAVPayloadTAQ-20 | System Requirements Document | | RD/3 | -PM-PMP-01 | Project Management Plan Document | | RD/4 | -TAIP-FD-01 | Target Acquisition &Image Processing Final Design Document | | RD/5 | -TAIP-PD-01 | Target Acquisition & Image Processing Preliminary Design Document | |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Non-QUT Documents

|  |  |  |
| --- | --- | --- |
| RDn/1 | Raspberry Pi Camera | <https://www.raspberrypi.org/documentation/hardware/camera/> |
| RDn/2 | Raspberry Pi 3 Model | <https://www.raspberrypi.org/documentation/usage/gpio/> |
| RDn/3 | Cascade Classifer, OpenCV | <https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html> |
| RDn/4 | Aruco Tag  Detection, OpenCV | <https://docs.opencv.org/trunk/d5/dae/tutorial_aruco_detection.html> |

# Test Objectives

The unit tests entailed in this document along with the outcome that are listed, are aiming to allow the reader to witness whether the chosen components will work as intended. There are several system requirements that these tests are aiming to meet such as:

**[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 targets type.

**[REQ-M-12]:** The UAVPayloadTAQ shall process all imagery data on-board via the on-board computer.

**[REQ-M-13]:** The processing must be able to analyse all data from the flight path while the UAV moves at a maximum speed of 1m/s.

**[REQ-M-14]:** The processing must be able to analyse all data from the flight path while the UAV operates at an altitude of 2m.

Tests will be done for this subsystem according to the system requirements mentioned above. The purpose of conducting these tests will be to verify the feasibility of the subsystem on its own and when its integrated with the other subsystems to achieve a complete working system as per the customer needs.

# Test Setup and Equipment

## Test Setup

The Image processing and the target acquisition subsystem will tested on the Raspberry Pi Model 3 over Wi-fi connection in order to simulate the situations similar to demo conditions. Components required for the tests are listed below. The scripts for the target detection algorithms can be found in the Appendices section.

A computer sitting on top of a table

Description automatically generated

Figure 1: Test Setup

## Test Equipment and Software Components

The main components used for the tests, specifically, the raspberry Pi running on Raspberry OS Lite, laptop with putty connected to the Pi over WiFi through SSH and the Raspberry Pi camera module.

### Raspberry Pi 3 Model B

The model being utilized for this project is the Raspberry Pi 3 Model B. The Raspberry OS(32-bit) Lite was installed in the Pi as its base operating system. SSH and camera modules were enabled for the Pi, in order to perform the desired functionalities required for this project.

A circuit board

Description automatically generated

Figure 2:Raspberry Pi 3 Model B

### Raspberry Pi Camera Module

The Raspberry Pi camera module is attached to the Raspberry Pi 3 computer via an integrated cable.

A circuit board

Description automatically generated

Figure 3:Raspberry Pi Camera Module

### USB to Serial Cable

In order to flash the Pi OS and connecting to the Laptop, a USB to serial cable is required. Before enabling the SSH and connecting over local Wi-fi network, the USB to Serial cable is used. The serial portion of the cable are connected to pins 6,8 and 10 respectively for establishing a connection between the Pi and the laptop over the cable.

### Python Scripts & OpenCV

The target detection mechanisms/algorithms were implemented using the Python programming language and sent to the Raspberry Pi via the SSH connection. These algorithms can be modified and triggered when required. OpenCV was used mainly for obtaining and implementing these target detecting mechanisms. The relevant scripts are attached in the Appendices section.

An open computer sitting on top of a wooden table

Description automatically generated

Figure 4:USB to serial cable

A circuit board

Description automatically generated

Figure 5: Serial pins connected to Pi

# IP Testing Procedure

## Initial Hardware Inspection

For the purpose of testing and assessing the status of this subsystem, the hardware will be inspected visually first. These equipments are the Raspberry Pi, the Raspberry Pi camera module, USB to serial cable and the operating laptop.

**Machine Condition**

1. Machine Condition
   1. Visually inspecting the device to confirm it matches expected images
   2. Look for any damage to the exterior of the model/case
   3. Inspect ports to check cleanliness and confirm no bent pins
2. Cable Condition
   1. Visually inspect each cable to confirm it is the required cable
   2. Look for any damage to the exterior of the cable
   3. Inspect cable ends to check cleanliness and confirm no bent pins

## Software Components Initialisation

The stages required for testing the full functionality of all software packages in the system:

1. After the Raspberry OS Lite has been flashed onto the system, it will be tested by connecting the Pi to the laptop over a serial to USB connection cable.
2. The Raspberry Pi should be the initialized using Putty over a serial connection with a baud rate of 115200 and successful connection will be tested.
3. Test will be done as the SSH module is connected, whether Wi-fi connection is possible and the Pi should be able to be connected over a local Wi-fi network and Pi is operable over SSH.
4. Tests on the verification of whether OpenCV and other required modules are installed.
5. Testing if the XMing Server is installed so the camera is operable, and the captured images are visible via a window as the Lite OS does not have a GUI.
6. Tests regarding the system’s capability to detect both Type A and Type B targets.

## Target Detection Algorithm Tests

The tests on the detection algorithms’ functionality will involve capturing images and then processing the captured images by running the python scripts. The outcome of these tests will be verified visually by looking at the screen capture of the outcome.

# Results

## 6.1 Initial Hardware Inspection

|  |  |  |
| --- | --- | --- |
| **Component** | **Hardware Condition** | **Cable Connection Condition** |
| **Raspberry Pi 3 Model B** | Good | Good |
| **Raspberry Pi Camera Module V2.1** | Good  A circuit board  Description automatically generated | Good |
| **USB to Serial Cable** | Good | Good |

## 6.2 Initial Software Initialization

|  |  |
| --- | --- |
| **Software Components Installed/initialized** | **Functionality** |
| **Raspberry Pi OS(32-bit) Lite** | Yes |
| **SSH & Camera Module** | Yes |
| **Successful Wi-fi Connection** | Yes |
| **OpenCV and other modules** | Yes |

## 6.3 Target Detection Algorithm Tests

As for the target detection mechanisms, algorithm was only implemented for the type B target- the ARUCO marker using the OpenCV library’s ARUCO extension. Defined by the customer, the ARUCO marker detection algorithm was setup to detect targets from the ARUCO dictionary 5X5\_100. After running several tests, it was observed, that the algorithm was successful to detect the ARUCO marker every single time. Tests were conducted for the ARUCO marker using 3 markers with 3 different IDs- 11,25 and 17 respectively. Unfortunately, the cascade classifier implemented for detecting the type A targets were not successful in detections and thus the output of those results are not discussed in this test report.

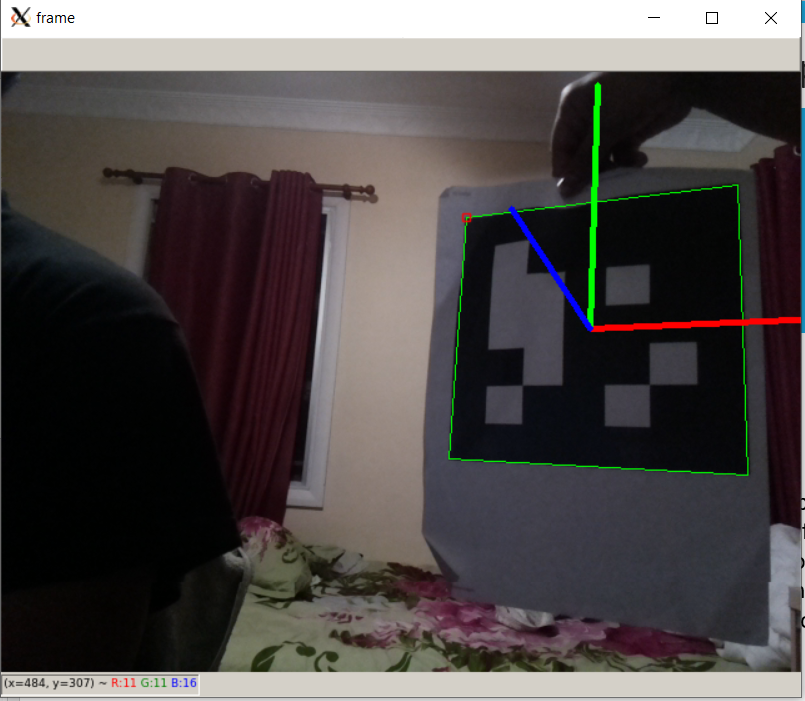


Figure 6: ARUCO marker detected

# Analysis

After conducting the tests, it was observed that the desired results were achieved as the Raspberry pi computer and the pi camera module were functioning as per requirements along with the connection with the local Wi-Fi network was established successfully. As for the detection of targets, only the type B target- ARUCO marker was successful, while the detection of the type A targets were not successful as there were errors while implementing the algorithm for cascade classifiers.

# Conclusions and Recommendations

From the tests conducted and discussed above, the Raspberry Pi 3 and the Raspberry Pi camera module was observed to be adequate for the UAVPayloadTAQ project. As the SSH was successfully enabled and tested by connecting over a local Wi-fi connection, it can be stated that the Pi will be able to transmit the processed imagery data to the Web Interface subsystem located in the GCS. Requirements [REQ-M-12], [REQ-M-13] and [REQ-M-14] were partially fulfilled, as the detection mechanisms for the type-A targets were not successful.

# Appendices

## Appendix I

2. import numpy as np
3. import cv2
4. import cv2.aruco as aruco
5. import sys, time, math
6. marker\_size  = 10 #- [cm]
7. Rt = np.transpose(R)
8. shouldBeIdentity = np.dot(Rt, R)
9. I = np.identity(3, dtype=R.dtype)
10. n = np.linalg.norm(I - shouldBeIdentity)
11. return n < 1e-6
12. def rotationMatrixToEulerAngles(R):
13. assert (isRotationMatrix(R))
14. sy = math.sqrt(R[0, 0] \* R[0, 0] + R[1, 0] \* R[1, 0])
15. singular = sy < 1e-6
16. if not singular:
17. x = math.atan2(R[2, 1], R[2, 2])
18. y = math.atan2(-R[2, 0], sy)
19. z = math.atan2(R[1, 0], R[0, 0])
20. else:
21. x = math.atan2(-R[1, 2], R[1, 1])
22. y = math.atan2(-R[2, 0], sy)
23. z = 0
24. return np.array([x, y, z])
25. #--- Get the camera calibration path
26. calib\_path  = ""
27. camera\_matrix   = np.loadtxt(calib\_path+'cameraMatrix\_raspi.txt', delimiter=',')
28. camera\_distortion   = np.loadtxt(calib\_path+'cameraDistortion\_raspi.txt', delimiter=',')
29. #--- 180 deg rotation matrix around the x axis
30. R\_flip  = np.zeros((3,3), dtype=np.float32)
31. R\_flip[0,0] = 1.0
32. R\_flip[1,1] =-1.0
33. R\_flip[2,2] =-1.0
34. #--- Define the aruco dictionary
35. aruco\_dict  = aruco.getPredefinedDictionary(aruco.DICT\_5X5\_100)
36. parameters  = aruco.DetectorParameters\_create()
37. #--- Capture the videocamera (this may also be a video or a picture)
38. cap = cv2.VideoCapture(0)
39. #-- Set the camera size as the one it was calibrated witxh
40. cap.set(cv2.CAP\_PROP\_FRAME\_WIDTH, 640)
41. cap.set(cv2.CAP\_PROP\_FRAME\_HEIGHT, 480)
42. #-- Font for the text in the image
43. font = cv2.FONT\_HERSHEY\_PLAIN
44. while True:
45. #-- Read the camera frame
46. ret, frame = cap.read()
47. #-- Convert in gray scale
48. gray    = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY) #-- remember, OpenCV stores color images in Blue, Green, Red
49. #-- Find all the aruco markers in the image
50. corners, ids, rejected = cv2.aruco.detectMarkers(image=gray, dictionary=aruco\_dict, parameters=parameters,
51. cameraMatrix=camera\_matrix, distCoeff=camera\_distortion)
53. # if ids is not None and ids[0] == id\_to\_find:
54. if np.all(ids is not None):
56. ret = aruco.estimatePoseSingleMarkers(corners, marker\_size, camera\_matrix, camera\_distortion)

        #-- Unpack the output, get only the first

1. rvec, tvec = ret[0][0,0,:], ret[1][0,0,:]
2. #-- Draw the detected marker and put a reference frame over it
3. print(aruco.drawDetectedMarkers(frame, corners))
4. #--- Display the frame
5. cv2.imshow('frame', frame)
6. #--- use 'q' to quit
7. key = cv2.waitKey(1) & 0xFF
8. if key == ord('q'):
9. cap.release()
10. cv2.destroyAllWindows()
11. break