|  |  |  |
| --- | --- | --- |
| Project:  WP Name: Flight Computer Design Testing  WP Number: WP-AP-02 | Type of Test:  Verification | Test Procedure:  Modify flight computer inputs and observe the effects |
| Test Article:  Flight computer performance | Part Number:  None | Serial Number:  None |
| Test Specification:  Flight computer threads updating at 50Hz and gathering sensor data correctly | Test Equipment:  Quadrotor platform with payload  Laptop (running GCS and MATLAB) | |
| Test Operators:  Liam O’Sullivan | Test Engineer:  Liam O’Sullivan | |
| WP Group Manager:  Michael Hamilton | WP Supervisor:  Dr. Luis Mejias | |

**QUT Avionics**

Queensland University of Technology

CRCSS-EESE, GPO Box 2434

Gardens Point Campus

Brisbane, Australia, 4001.

Telephone (+61 7) 3864 1772

Facsimile (+61 7) 3864 1517

e-mail luis.mejias@qut.edu.au

web <http://code.google.com/p/ahns10/>

This document is Copyright 2010 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**

The flight computer implementation was tested by modifying certain inputs to the quadrotor platform and observing the outputs via the GCS. The 5 threads which are executed on the flight computer are all being executed above the minimum 50 Hz requirement. The sensor data from the IMU has been properly converted from its 16 bit raw output to the appropriate double data type and is being received by the GCS. Finally the Arduino is collecting data from the magnetic compass, battery voltage sensor and the altitude sensor it is connected to. This data is being sent correctly to the GCS however some serial errors have been discovered. It is recommended that the serial Arduino library be recoded to remedy these sporadic data errors.

**Table of Contents**

Paragraph Page No.

[1 Test Objectives 4](#_Toc275609447)

[2 Test Set-up & Equipment 5](#_Toc275609448)

[3 Procedure 6](#_Toc275609449)

[4 Results 7](#_Toc275609450)

[4.1 Thread update rates 7](#_Toc275609451)

[4.2 Reconfigure the IMU 10](#_Toc275609452)

[4.3 IMU 16 bit data collection and conversion 10](#_Toc275609453)

[4.4 Arduino sensor data 11](#_Toc275609454)

[4.5 Raw IMU sensor data output 11](#_Toc275609455)

[4.6 Raw Arduino sensor output 15](#_Toc275609456)

[5 Analysis 17](#_Toc275609457)

[5.1 Thread update rates 17](#_Toc275609458)

[5.2 Reconfigure the IMU 17](#_Toc275609459)

[5.3 IMU 16 bit data collection and conversion 17](#_Toc275609460)

[5.4 Arduino sensor data 17](#_Toc275609461)

[5.5 Raw IMU sensor data output 17](#_Toc275609462)

[5.6 Raw Arduino sensor output 17](#_Toc275609463)

[6 Conclusions 18](#_Toc275609464)

[7 Recommendations 19](#_Toc275609465)

**List of Figures**

Figure Page No.

[Figure 4.1 - State estimation thread update rate 7](#_Toc275609433)

[Figure 4.2 - Arduino thread update rate 8](#_Toc275609434)

[Figure 4.3 - UDP thread update rate 8](#_Toc275609435)

[Figure 4.4 - MCU thread update rate 9](#_Toc275609436)

[Figure 4.5 - Control thread update rate 9](#_Toc275609437)

[Figure 4.6 - rateX sensor data (IMU) 12](#_Toc275609438)

[Figure 4.7 - rateY sensor data (IMU) 12](#_Toc275609439)

[Figure 4.8 - rateZ sensor data (IMU) 13](#_Toc275609440)

[Figure 4.9 - accX sensor data (IMU) 13](#_Toc275609441)

[Figure 4.10 - accY sensor data (IMU) 14](#_Toc275609442)

[Figure 4.11 - accZ sensor data (IMU) 14](#_Toc275609443)

[Figure 4.12 - Compass sensor output (from Arduino) 15](#_Toc275609444)

[Figure 4.13 - Ultrasonic sensor output (from Arduino) 16](#_Toc275609445)

[Figure 4.14 - Battery voltage sensor output (from Arduino) 16](#_Toc275609446)

**List of Tables**

No Tables.

# Test Objectives

The test report has the following test objectives:

* Change the flight computer inputs and observe the update rate of each thread. Each thread needs to be updated at the minimum requirement of 50Hz.
* Test functions to reconfigure the IMU baud rate and collect the 16bit data from the IMU sensors.
* Recalibrate the magnetic compass using hard iron and soft iron calibration routines.
* Verify that the Arduino is collecting data from the magnetic compass, the altitude sensor and the battery voltage sensor and passing it to the flight computer.
* Verify that all raw sensor data is being received by the flight computer and is sent to the GCS.

# Test Set-up & Equipment

The following test setup and equipment was used to conduct the test report:

* PC with a Linux based operating system installed
* Quadrotor platform with payload attached (including the IMU, Arduino and compass sensors)
* Flight computer code being executed with the GCS connected to it (as a client)
* PC with MATLAB installed
* Log files recorded by the flight computer
* MATLAB test script file (FlightComputerTest.m)

# Procedure

The test report utilised the following procedure:

1. Observe the thread update rates whilst the flight computer is running
2. Reconfigure the IMU with the new baud rate of 115200 baud.
3. Verify that IMU data can be converted from the 16 bit values to the proper data types.
4. Recalibrate the compass using the OS-4000 specifications (both hard iron and soft iron calibration).
5. Change magnetic compass data, battery voltage data and altitude data and observe the output from the Arduino.
6. Modify each IMU raw sensor variable including: rateX, rateY, rateZ, accX, accY, accZ and observe the outputs.
7. Modify each Arduino raw sensor variable including: compass, altitude sensor and battery voltage and observe the outputs.
8. The log files for all of the above tests should be saved and imported into MATLAB using the script analysis file FlightComputerTest.m.
9. Plots of all sensor data should be generated including the thread update rates.

# Results

The results will be presented per the testing procedures as specified in section 3.

## Thread update rates

Figures 4.1, 4.2, 4.3, 4.4 and 4.5 show the thread update rates for the state estimation, Arduino, UDP, MCU and control threads.

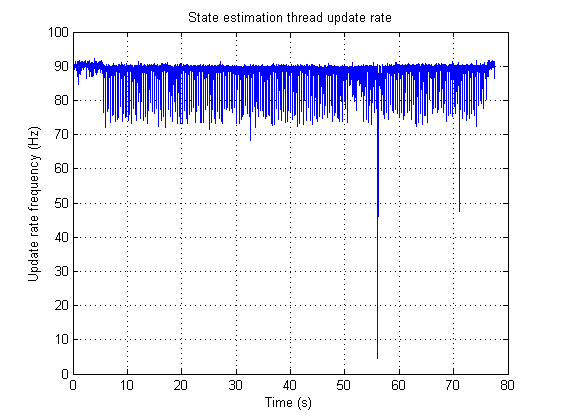


Figure .1 - State estimation thread update rate

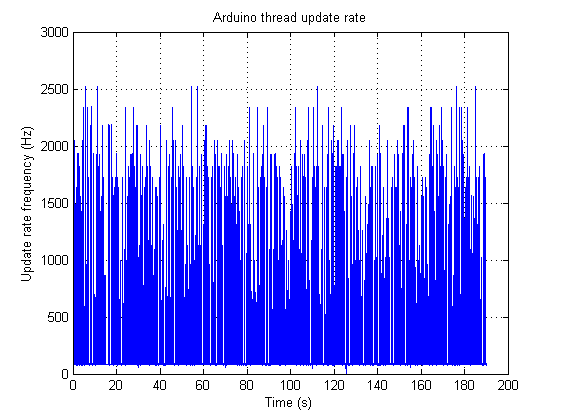


Figure .2 - Arduino thread update rate

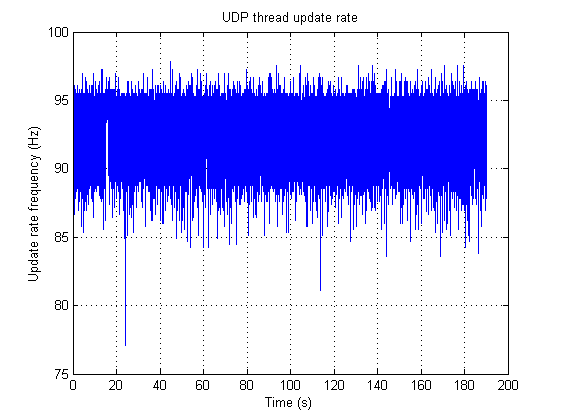


Figure .3 - UDP thread update rate

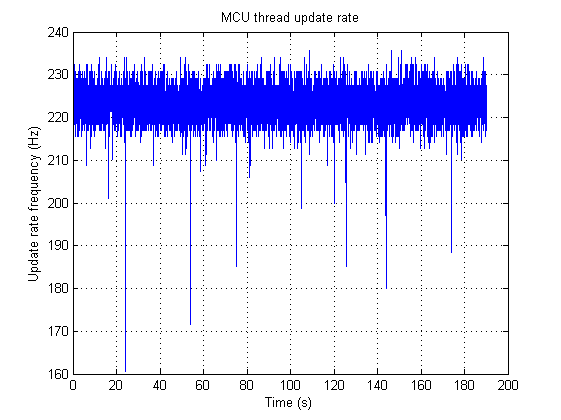


Figure .4 - MCU thread update rate

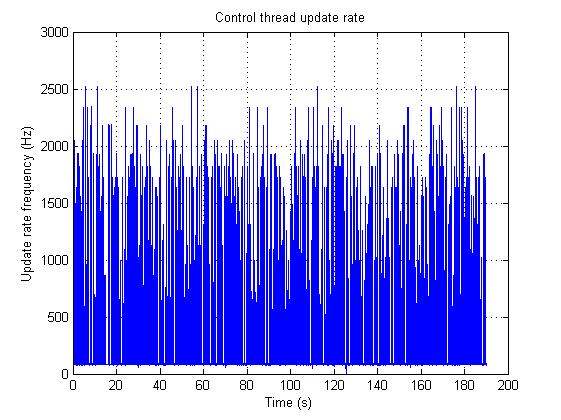


Figure .5 - Control thread update rate

## Reconfigure the IMU

Utilising the setIMUconfig function in the IMU software library allowed the IMU to be reconfigured to allow:

* MR2 rate output
* MR2 accelerometer output
* 115200 baudrate

The output of this function was as follows when executed on the flight computer Overo Fire board:

>> Success - Connected to IMU

>> Set CC0|OK!

>> Read C|C0

>> Set D7F|OK!

>> Read D|7F

>> Change baudrate 3 |OK!

>> IMU reconfigured

## IMU 16 bit data collection and conversion

The following summarised rate and accelerometer data was gathered from the IMU after MR2 output was set:

FFB9,FFB6,FFE4,FFDB,FFC1,0986,0088,0000,0000,2A

FFCF,FFC3,FFB8,FFD5,FFCB,096F,0089,0000,0000,0A

FF85,FFA7,FFE1,FFD4,FFCD,09AD,0089,0000,0000,1E

FFB2,FFB6,FFEE,FFD2,FFC7,09A6,0088,0000,0000,21

FF87,FFB6,000B,FFD9,FFC3,098C,0088,0000,0000,63

FFA9,FFB3,FFEE,FFD4,FFD6,097A,0089,0000,0000,1A

FFB5,FF90,FFCE,FFD6,FFCC,097C,0089,0000,0000,1B

FFD2,FFB5,FFD6,FFD9,FFD0,0992,0088,0000,0000,3B

FFAF,FFB3,FFE6,FFD4,FFCE,0961,0088,0000,0000,20

FF8B,FFA0,FFF1,FFDE,FFBE,097F,0088,0000,0000,0F

The conversion of this 16 bit data using the MR2 rate and accelerometer conversions led to the following output sentences:

Format:

***| MR2 – Rate X | MR2 – Rate Y | MR2 – Rate Z | MR2 – Acc X | MR2 – Acc Y | MR2 – Acc Z |***

-1.0920,-1.1388,-0.4212,-0.1462,-0.2518, 9.9032

-0.7488,-0.9360,-1.1076,-0.1706,-0.2112, 9.8097

-1.9032,-1.3728,-0.4680,-0.1747,-0.2031,10.0616

-1.2012,-1.1388,-0.2652,-0.1828,-0.2275,10.0331

-1.8720,-1.1388, 0.1716,-0.1544,-0.2437, 9.9275

-1.3416,-1.1856,-0.2652,-0.1747,-0.1665, 9.8544

-1.1544,-1.7316,-0.7644,-0.1665,-0.2072, 9.8625

-0.7020,-1.1544,-0.6396,-0.1544,-0.1909, 9.9519

-1.2480,-1.1856,-0.3900,-0.1747,-0.1990, 9.7529

-1.8096,-1.4820,-0.2184,-0.1340,-0.2640, 9.8747

## Arduino sensor data

The Arduino has the following sentence sensor data output:

Format: **Chhh.h,Vvv.vvv,Aa.aaa**

The following summarised sensor output was obtained from the Arduino using the above format:

C226.10,V14.085,A0.190

C226.20,V14.085,A0.312

C226.20,V14.085,A0.298

C226.20,V14.085,A0.312

C226.20,V14.085,A0.312

C226.20,V14.085,A0.312

C226.20,V14.085,A0.285

C226.20,V14.085,A0.312

C226.20,V14.085,A0.312

C226.20,V14.085,A0.298

## Raw IMU sensor data output

Figures 4.6, 4.7, 4.8, 4.9, 4.10 and 4.11 show the raw IMU sensor data outputs for rateX, rateY, rateZ, accX, accY and accZ when the platform attitude is changed.

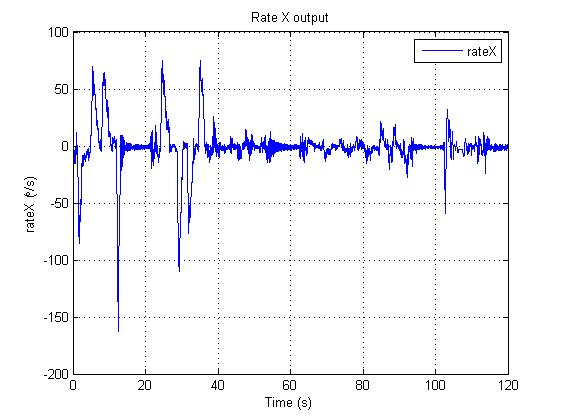


Figure .6 - rateX sensor data (IMU)

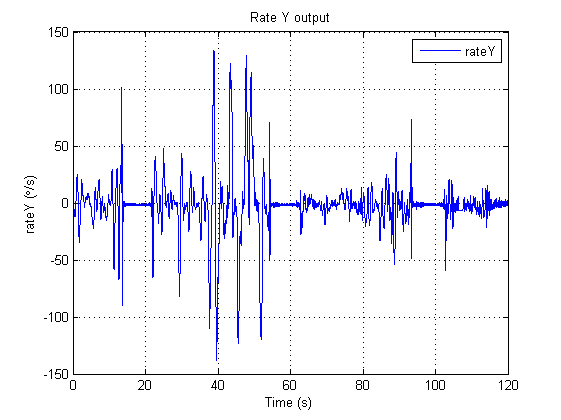


Figure .7 - rateY sensor data (IMU)

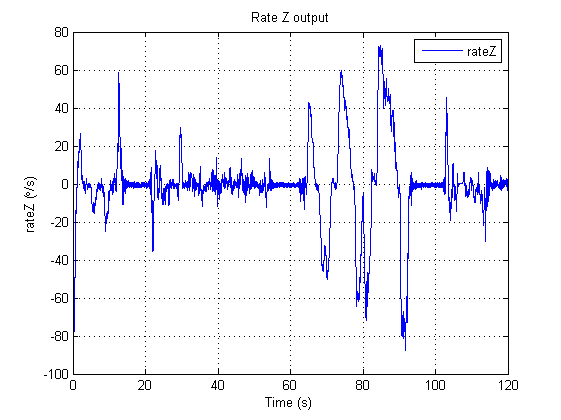


Figure .8 - rateZ sensor data (IMU)

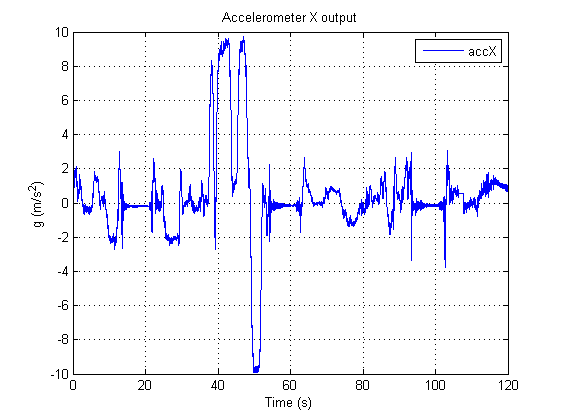


Figure .9 - accX sensor data (IMU)

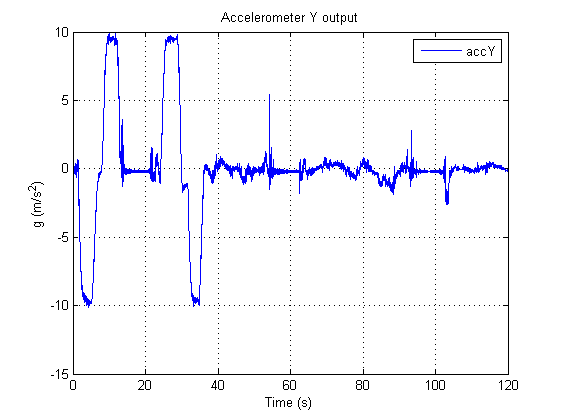


Figure .10 - accY sensor data (IMU)

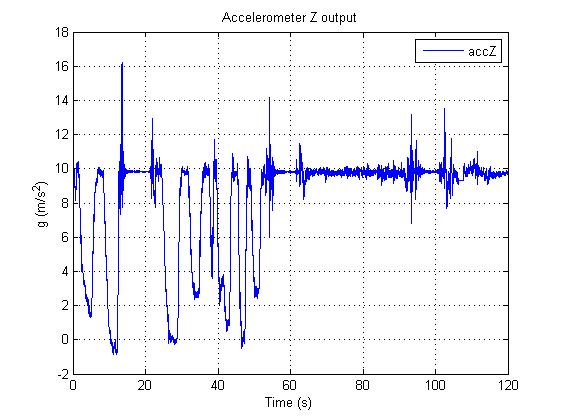


Figure .11 - accZ sensor data (IMU)

## Raw Arduino sensor output

Figures 4.12, 4.13 and 4.14 show the raw Arduino sensor output including the magnetic compass data, the ultrasonic (altitude) sensor data and the battery voltage.

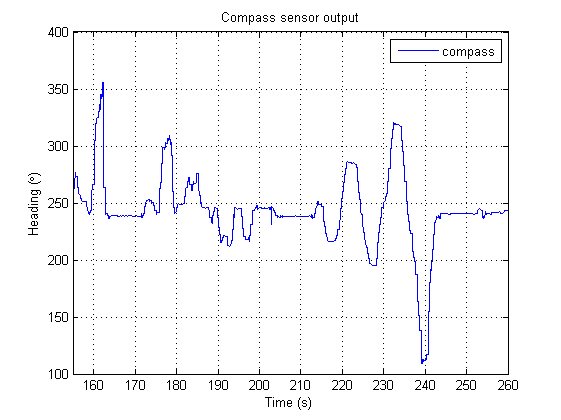


Figure .12 - Compass sensor output (from Arduino)

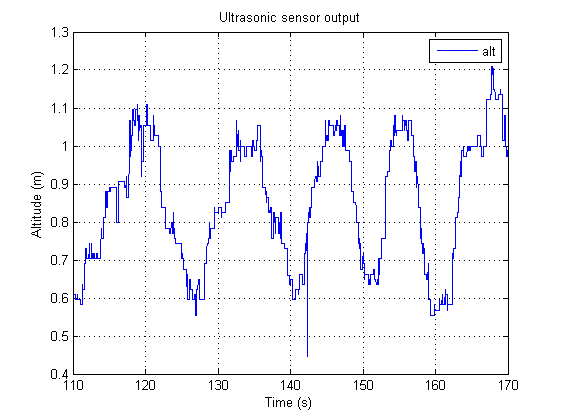


Figure .13 - Ultrasonic sensor output (from Arduino)

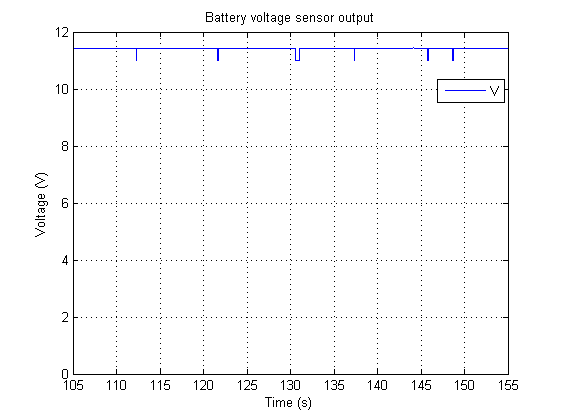


Figure . - Battery voltage sensor output (from Arduino)

# Analysis

The analysis will be presented per the testing procedures as specified in section 3.

## Thread update rates

All threads are being updated above the minimum 50Hz specification on the flight computer.

## Reconfigure the IMU

The IMU has been successfully reconfigured on the flight computer with the appropriate MR2 rate and accelerometer outputs and 115200 baud rate.

## IMU 16 bit data collection and conversion

The IMU raw sensor data has been successfully convert from its 16 bit representation to the required double data type.

## Arduino sensor data

The Arudino has collected the compass data, the battery voltage and the altitude data and transmitted the output sentence to the flight computer.

## Raw IMU sensor data output

Raw IMU sensor data is being sent from the flight computer to the ground control station. All rates and accelerometer values are correctly represented.

## Raw Arduino sensor output

Raw Arduino data is being sent from the flight computer to the ground control station. Some errors in serial readings do occur which is most evident in the ultrasonic sensor output.

# Conclusions

From the plot and data outputs it can be concluded that the flight computer is functioning correctly. Hence requirements SR-D-05 and SR-D-06 have been successfully met. Furthermore other requirements relating to the communications sub system e.g. SR-B-09 have also been achieved. This is because all graphed sensor data in this test report was received via the Wi-Fi link between the flight computer and the GCS.

# Recommendations

It is recommended that the serial communication library which connects the Arduino to the flight computer be redesigned. Either a query/response serial system or a new implementation of the canonical serial connection should be designed. Doing so will enable correct sensor collection between the Arduino and the flight computer on the Overo Fire.