Atlantis-I Payload Documentation

Nathan Meulenbroek

June 2017

Contents

1 Introduction

This document describes the procedures, pitfalls, and the architecture of the code for the Atlantis I rocket as well as the ground control code. Please make sure that all procedures are followed, especially those regards to the pre-flight checks. You will need an internet connection to grab code from the repositories. There is a designated laptop for competition that has everything besides processing configured.

Atlantis I Microcontroller Code: `https://github.com/StudentOrganisationForAreospaceResearch/Atlantis-I-Payload`

SOAR Ground Control System: `https://github.com/StudentOrganisationForAreospaceResearch/GroundControl`

2 Dependencies

2.1 Ground Control System

This program was developed on Windows 10 using Python 3.6. To future proof your computer for any updates to this program, please install the latest Anaconda distribution, found at `https://www.continuum.io/downloads`. If you would like to stay more lightweight for a computer that does not have as much storage, please install all of the below libraries through command prompt (or terminal) using the command "pip install *library*".

Libraries:

1. matplotlib
2. pyqt5
3. numpy
4. pyserial

2.2 Microcontroller:

This program is built for a Teensy 3.5, BMP180/BMP085, Razor 9DOF IMU AHRS, Zigbee, and a GPS, stepper motors and fan with model numbers that I currently don't have on hand. The GitHub repository also includes modified firmware for the Razor IMU. It does not have any non-default library requirements. To interface with the Teensy 3.5, you must have installed Arduino IDE with a version number larger than 1.6.X and smaller than 1.8.2, as well as the Teensyduino drivers.

To calibrate the IMU, a recent version of processing is needed.

3 Pitfalls and Known Issues

3.1 Ground Control System

The UI for the Ground Control System is very finicky. There are multiple errors that have to do with the graphics library that have not been able to be pinpointed or reproduced reliably. We have, however, confirmed that if the program runs for more than about a minute the likely hood that it will crash is extremely small. If it doesn't work with one computer, it will sometimes work with another. The graphics library that we use is known to not work on Ubuntu based Linux flavours.

The main points of failure for this program will most likely be in the recording of data points (the arrays are hard-coded for each data point and it's possible that something is wrong with passing the data points through appropriately) or the UI, of which only the former is truly debuggable.

If there is a problem with data points, please look first at the data file, though it may be the connection between any of the methods that pass these back and forth.

3.2 Microcontroller Code

The microcontroller has unfortunately not been exhaustively tested. You will find that data collection and all of the core sensors work, and that all of the algorithms have had logical tests completed, but no operational tests. The BMP180 may also need to be glued into place still before the flight. It is assumed that all algorithms will work as expected, though this has not been tested. Main points of failure will include the downlinkn and anything associated with the aerosol sampler. The altitude ranges for the aerosol sampler also have not been verified, though there may be changes to this before launch day.

There is a chance that the microcontroller could have a memory leak because of the string construction as well. We don't know how long this would let the Teensy run since we didn't have time to test. I've tried to minimise the impact of this as much as possible, but there's still a small chance that it won't work. Possible fix involving creating a deconstructor in the library and is described here: `http://forum.arduino.cc/index.php?topic=48480.0`

3.3 Downlink Connection

There has also been no time to test the communications with the payload as, at the time of writing of this document, it is not possible to complete a connection from the payload to a computer. The Zigbee seems to work, though the relay box does not. This means that there has been no opportunity to test the combination of the two programs.

If there is a problem with the serial reading, please look at the dataStream file.

4 Procedures

4.1 IMU Calibration

Supplementary instructions can be found at `https://github.com/Razor-AHRS/razor-9dof-ahrs/wiki/Tutorial#setting-up-the-software`.

1. Load the sketch found at Atlantis-I-Payload ¿ Razor_AHRS ¿ CONFIGURATION_ONLY ¿ Arduino ¿ Razor_AHRS onto the IMU.
2. Start the file found at Atlantis-I-Payload ¿ Razor_AHRS ¿ CONFIGURATION_ONLY ¿ Processing ¿ Magnetometer_calibration ¿ Magnetometer_calibration.pde
3. Once the GUI has loaded, start spinning the Magnetometer around to cover the entire circle (front and back) with dots
4. Once you are satisfied or the program can no longer accept any more points, press space bar to close the window and print out the calibration values to the console.
5. Return to Atlantis-I-Payload ¿ Razor_AHRS, load the sketch, and input the calibration values into the Razor_AHRS file.
6. Load the now calibrated sketch onto the IMU
7. Re-mount the IMU on the payload

4.2 Testing

1. Open the sketch found at Atlantis-I-Payload ¿ Teensy_rocket_loop
2. In the user setup area, there are several debugging options. Setting SERIAL_DEBUGGING to true will print out to the USB serial (you will get messages through serial monitor). Setting LED_DEBUGGING to true will slow down the loop so that each iteration takes one second, and flash the LED every time an iteration is completed.

Setting SKIP_SAFETY_CHECKS to true would more or less immediately deploy the parachutes upon start up and then go into the sequence for the aerosol sampler. This is extremely dangerous and should never be left enabled.

3. Use these tools to help diagnose issues if they should arise.

4.3 Startup

1. To start the Teensy with code already loaded on, plug in the battery to the board and to the fan and pull the pull tab. Everything will then start up automatically.
2. To start the Teensy with new code, complete the above, upload the code using the Arduino IDE and Teensyduino drivers. You may need to press the reset button on the board if a software reboot does not get a response.
3. If for some reason the Teensy does not respond to an upload at all even with a manual reboot, plug in the pulltab, compile the code, unplug the pulltab, wait until you hear the computer register the connection (have volume turned up for this) and then quickly hit upload.

4.4 Pre-Flight Checklist

1. Check that the correct firmware is installed on the IMU. This can be done by opening serial monitor with the IMU connected by FTDI. If you see one long line of characters, you have the right firmware installed.
2. Check that all debugging options are disabled and reload the code onto the Teensy. This step is critical as otherwise the rocket could be made unsafe when the payload is activated.
3. If the down link is working, make sure that the radio relay is turned on and you are receiving data on the computer.
4. Before leaving the rocket, make sure that the LEDs are turned on for the IMU, downlink, and other sensors.

5 Emergency Contact

If you are in desparate need of help with the rocket, please contact Nathan Meulenbroek on slack, email (`mailto:nathan.meulenbroek@ucalgary.ca`) or by facebook messenger. I will do my best to get back to you as soon as possible.