Software Formalization

Year: 2022 Semester: Fall Team: 8 Project: Hermes

Creation Date: ­9/26/2022 Last Modified: October 1, 2022

Author: Jonathan Wosiak Email: jwosiak@purdue.edu

Assignment Evaluation:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item** | **Score (0-5)** | **Weight** | **Points** | **Notes** |
| **Assignment-Specific Items** | | | | |
| **Third Party Software** |  | x2 |  |  |
| **Description of Components** |  | X3 |  |  |
| **Testing Plan** |  | x3 |  |  |
| **Software Component Diagram** |  | x4 |  |  |
| **Writing-Specific Items** | | | | |
| **Spelling and Grammar** |  | x2 |  |  |
| **Formatting and Citations** |  | x1 |  |  |
| **Figures and Graphs** |  | x2 |  |  |
| **Technical Writing Style** |  | x3 |  |  |
| **Total Score** |  | | |  |

5: Excellent 4: Good 3: Acceptable 2: Poor 1: Very Poor 0: Not attempted

General Comments:

1. Utilization of Third Party Software

Currently the only third-party software we are using are CMSIS headers for the convenience of not having to use the respective hex values. We have found that our time-of-flight sensor requires some interfacing with their library [2], but we are currently testing ways to get around this by creating our own write from DMA function, thus not impacting our engineering effort.

2.0 Description of Software Components

DSHOT:

We are using the DSHOT protocol for communication between our FC and ESC. Some benefits of using DSHOT are no oscillator drift or calibration of our ESC, more accurate ESC signal by noise reduction, high resolution, and automatic rejection of corrupted data. At the moment, we are using a rudimentary PWM protocol for testing our interfacing, an analog precursor to DSHOT. This solution has its limitations, beyond our current implementation used purely for testing, which is why we are not going to use it in our final software.

Radio Software:

Our ESP8266 radio receiver onboard our drone and our BetaFPV transmitter use a protocol named ExpressLRS, or ELRS. The receiver uses a wireless access point to login and configure the receiver over Wi-Fi, however the transmitter requires a USB connection with a precise power up sequence to connect through a configurator. Once the two are bound together packets can be received and reported through UART.

IMU Software:

We are using the LSM6DSOWTR as our IMU. The protocol we are using to communicate with our microcontroller is SPI. The data read from the IMU will be then sent to a PID loop to keep the drone steady and hopefully smooth in flight. We will need to constantly tune the PID controller as it loops to keep flight as smooth as possible.

Time-of-Flight Software:

We are using the VL53L1x Time-of-Flight sensor to sense how far objects are away from the drone. To do this, we are using I2C, reading/writing from DMA. As mentioned previously we are required to interface minimally through the product’s library. However, we will be writing our own functions to read/write with DMA for our engineering effort.

3.0 Testing Plan

DSHOT:

To test DSHOT, we will be reading the traces between our FC and ESC. If we are getting the correct output on our motors, we know DSHOT is implemented correctly.

Radio Software:

To test our radio software, we read the signals received on an oscilloscope.

IMU Software:

To test the IMU software we will read

Time-of-Flight Software:

4.0 Sources Cited:

[1] “CMSIS device headers for STM32” [www.github.com](http://www.github.com)

<https://github.com/modm-io/cmsis-header-stm32>

[2] “VL53L1X library for Arduino” [www.github.com](http://www.github.com).

<https://github.com/pololu/vl53l1x-arduino>

Appendix 1: Software Component Diagram