MSD: Software Status and How To

Status

For this project the software focused mainly on assisting in testing the ADACS and sail deployment, and consisted of software written on an MSP430FR5739 and a Raspberry Pi Zero W. The MSP430 was going to be used for controlling and managing sail deployment, but was put on hold due to changes in the team, and current events. The Raspberry Pi would have been a simulated flight computer, handling all of the things the actual flight computer would. This Pi has a GUI on it, written in python and using Tkinter, that would have sent out signals over I2C and SPI to control the ADACS and MSP430, and also display data on matplotlib graphs. Any functions related to I2C, gyroscope and MSP430 communication, was not completed. There is another python file that has the SPI commands working on it. From observation these SPI signals should be correct, but work on this interface was put on hold due to current events. The buttons on the Pi are all setup and have functions they run when pressed, but they only write to the command line and do not actually send anything out, since design was interrupted. The Pi is accessible over RIT Wi-Fi as "p20101.student.rit.edu". This Pi is currently registered under Charles Nystrom's RIT account, but can be removed and reassigned to a SPEX related account. The Pi is also accessible over USB. Just connect a USB cable to the non-power input, and connect to it using either remote desktop or Tera Term and using the computer name "raspberrypi.local". The password is "MSDP20101". The computer that is accessing the Pi will not be able to use the internet when connected to the Pi in this way. If the Wi-Fi connection does not work, this method will.

Resources Used

Notable Python Modules – Tkinter, spi-dev, matplotlib, simple_pid

-Tkinter is used to power the GUI. It uses buttons to control the flight computer, and uses matplotlib plots that are embedded into the GUI to display data. spi-dev is a relatively simple module to control the SPI bus on the Pi. simple_pid was planned to be used to implement a basic PID controller, but if it works correctly or not is unknown, since it was not able to be tested.

Software Notes

The GUI initializes itself with a "self.pack()" command, which in this case seems to not pose a problem, but when it was trying to be recreated in a different file, the ".pack()" command and the later usage of .grid() command object placement conflicted. The Stop Deployment button exits the GUI, which should not go into any final design. Adding another button to exit the GUI would probably be a better implementation. The placement of blocks can be a little confusing, but the method used uses the ".grid()" method, which organizes the objects based on rows and columns. Frames are placed on the highest-level grid, and buttons are placed in grids inside those frames.

Links

https://pypi.org/project/spidev/ - SPI Module

https://pypi.org/project/simple-pid/ - simple_pid module

https://docs.python.org/3/library/tkinter.html - tkinter

https://pythonprogramming.net/how-to-embed-matplotlib-graph-tkinter-gui/ - matplotlib tkinter