**Teensy User Guide:**

In order to program the Teensy download Microsoft Visual Studio, with the Teensy/Arduino add on. You are programming it for the Teensy 3.6 (this is necessary to say so it does memory mapping correctly). Also download the teensy loader from the teensyduino website. Visual Studios should automatically open it up upon compiling, and this is how it flashes the code to the Teensy.

The pushbutton on the PCB (NOT ON THE ACTUAL TEENSY, NEXT TO IT) should be held on startup in order to run a calibration on the gyroscope. If you are running the later code with the patches for no comms board it will also calibrate the pressure for the ground, as well as restart the SD card system and RTC back to 0. No matter which version you are using, on launch day this button should be held while outside, while the board is being held still. This will make sure all of the calibrations are run successfully.

All of the self test results, as well as the calibration results, are printed out to the terminal via Serial Interface (Baud Rate 9600), you should be able to connect to it with Putty. The self test results are also displayed on the red LED. For a "Hard failure" the red LED is turned on and held on. A hard failure is either both of the IMUs failing, or no SD Card present. Any other failure will result in a soft failure and the red LED blinking a specified number of times. The number of times can be seen in the code. If there are multiple faults there are two second delays in between each fault.

All Raw Data is written to the SD card, it is not converted on board. Use the "DecodeData" spreadsheet under the documentation folder of the bad-habips github to convert all of the data. If the data from the SD Card is copied onto the first sheet, the second sheet should automatically calculate all of the real values. MAKE SURE YOU HAVE MACROS ENABLED.

**IMPORTANT:**

If you are not going to fly with the Comms board use the most recent version of the code in Github. This has checks for the altitude and all the other patch functionality. If you do have a fully working system, use the version that was uploaded on 03/27/2018 (SPI with DAQCS Board).

There is some functionality from the later versions that you may want to port over to the 03/27/2018 version. There are certain LEDs that are turned on for failures (ie. if the battery dies it toggles a GPIO to the MSP430 to display a Red LED indicating the failure). If you do a compare of the two versions you can look through and see what was added, and decide if you want to port it or not.

**Remaining Problems:**

RTC loses time when power goes down. We thought the super cap was enough to keep power to the RTC if the rest of the system goes down, but that does not seem to work for long enough. We believe this may be a hardware issue, look into this. Or it might be a settings issue in the Teensy. In the version with the patches in it I store the number of SD files created in flash memory. A new file is created every five minutes, and so this is used to estimate the time of the RTC. On startup it reads this number from flash and then sets the clock to that time. This system is reset if the pushbutton is held down on startup. I recommend porting this over to the working code (03/27/2018 version). This will fix that problem.

SPI with the MSP430 is not working. The logic to parse and then construct messages is working, a test bench was written and it was successfully tested. The setup of SPI is not working, an interrupt is never triggered when a message is received. The teensy is configured as the slave on the bus. This shouldn't be that hard to fix, not much time was spent investigating it.

Feel free to reach out to me with any questions or concerns (probably a lot of concerns, I'm an EE writing crappy code). Best way would be on Facebook (my name is Brandon Codi) or by email brandoncodi01@gmail.com