TFAC Flight Computer

This is a white paper documenting my work related to the design (schematic/board design/layout, manufacturing of board, ordering parts, assembly (soldering), and testing (software/manual) of TFAC (Third Flyable Avionics Computer) which is basically a Flight Computer for TVC (Thrust Vector Control) and HPR (High Power Rocketry).

A short summary of what is covered in this white paper is

1. Schematic Design - Using KiCAD
2. PCB Design/Layout - Using KiCAD
3. Ordering of the Components required to go onto the PCB - sourced from various vendors
4. Physical PCB Manufacturing - Using PCB Fabrication service vendors
5. Soldering - Done by me, using the PCB & Components
6. Verification of the Interfaces, Sensors, Memory devices etc through Software written by me, using C++
7. Verification of the Functionality of the board - Manual movement of the Flight Computer and capturing of the transmitted data.

Overview

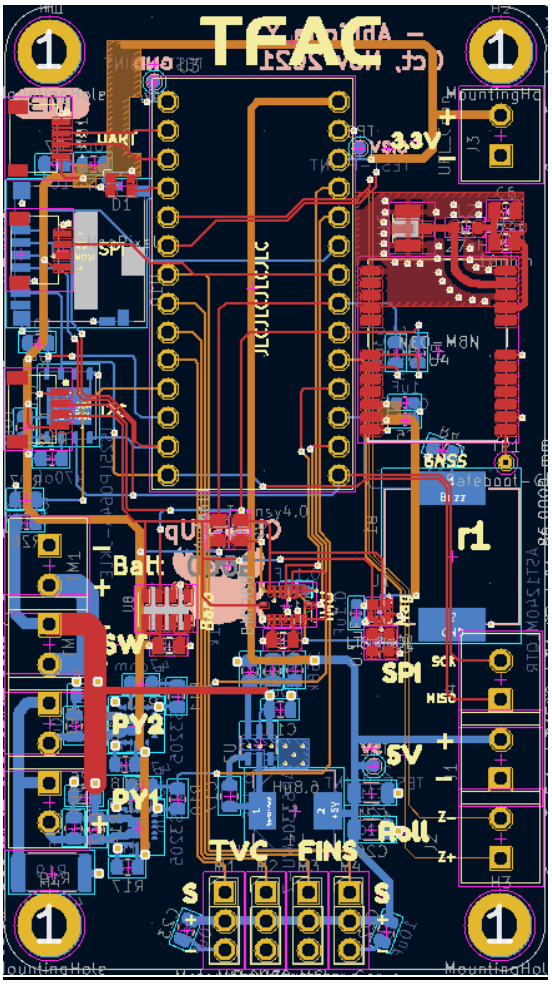
TFAC (Third Flyable Avionics Computer) is the third flight computer (FC) I've designed and built (*figs* 1 and 2) and TFAC is the first one that works nominally (prior attempts provided learning but were suboptimal in design/performance). The FC was designed in the KiCAD EDA, r1 in early October 2021, and r2 in late August 2022. The board has 4 layers - In1 being a solid GND plane and In2 being a mixed signal.

Microcontroller and Framework

At the heart of TFAC is a Cortex M7-based Teensy 4.0 which runs at 600mHz (TFAC is also cross-compatible with the Teensy 3.2). The software for TFAC is written in C++ with the Arduino framework in the Visual Studio Code IDE.

Hardware - Power

TFAC can be powered by any battery with a current output of at least 2A and a voltage above 6V. A 3s (three-cell) 11.1V LiPo battery is ideal to use the Pyrotechnic Channels. A Diodes Incorporated AP63300 buck converter steps the 11.1V from the LiPo down to 5V, which powers the Teensy and servo outputs. The Teensy's onboard regulator steps the buck's 5V to 3.3V (the operating voltage for most of the sensors/peripherals on the board). The total current consumption of the sensors, memory devices, etc is ~180mA.



(*fig* 1 - TFAC Board Layout)

(*fig* 2 - TFAC Board: Fully soldered)

Hardware - I/O

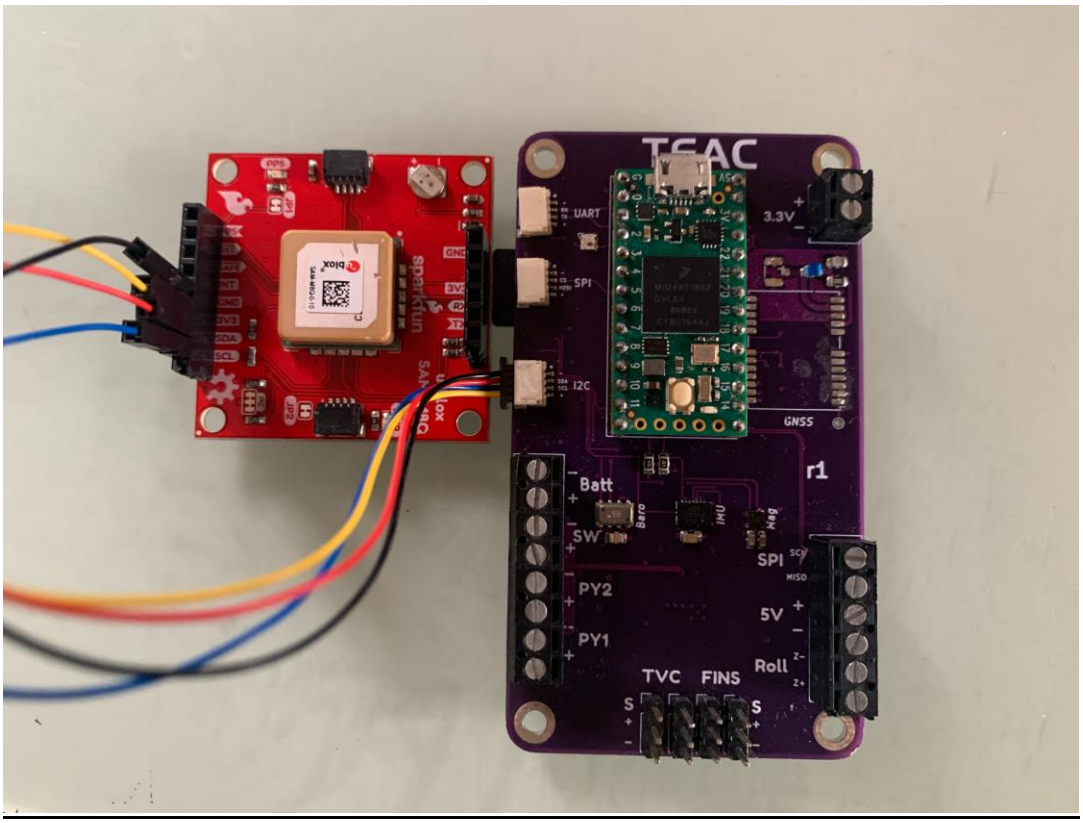
One of my goals with TFAC was expandability, which means extra I/Os. In terms of power, TFAC has 3.3V and 5V outputs. In terms of communication, TFAC has UART, SPI, and I2C outputs on SMD JST-SH connectors (everything else is on terminal blocks). Other I/Os include 2 GPIO PWM pins labeled for roll control (can be done with a 3D printed reaction wheel a DC motor, and a motor driver) and 4 servo pins.

Hardware - Sensors

TFAC is well equipped with a 10DOF MEMS sensor suite, comprising of a Bosch Sensortech BMI088 as the IMU for acceleration and angular velocity data, a TE MS5607 as the barometer for pressure, altitude, and temperature data, and an STMicroelectronics LIS3MDL as the magnetometer. All the onboard sensors communicate with the microcontroller via the I2C communication protocol.

Hardware - GNSS

Onboard TFAC, there is a uBlox NEO-M8N GNSS radio (chosen because it was in stock when I was assembling the board). Any NEO series module after the NEO-6 family can be used as they are all pin compatible. Along with the GNSS module is circuitry for an active antenna with the UFL layout. The UFL antenna and the Bias T are RF-shielded. While assembling TFAC, I realized that I made an error while designing the board itself - pin 2 or D_SEL on the NEO modules is the interface selector pin (the uBlox NEOs can be used with I2C, SPI or UART, I'm using I2C) which can be pulled either high (3.3V) or low (GND) to select the communication protocol with I2C. To use with I2C, it should be pulled high but I pulled it low, alienating the possibility of using the onboard GNSS functionality. For rockets that go higher and faster, you would want to know where exactly they are (for recovery) i.e GNSS coordinates, so I decided to hook up a Sparkfun SAM-M8Q GNSS breakout to the I2C connector on TFAC (*fig* 3).



(*fig* 3 - TFAC hooked up to the SAM-M8Q)

Hardware - Memory

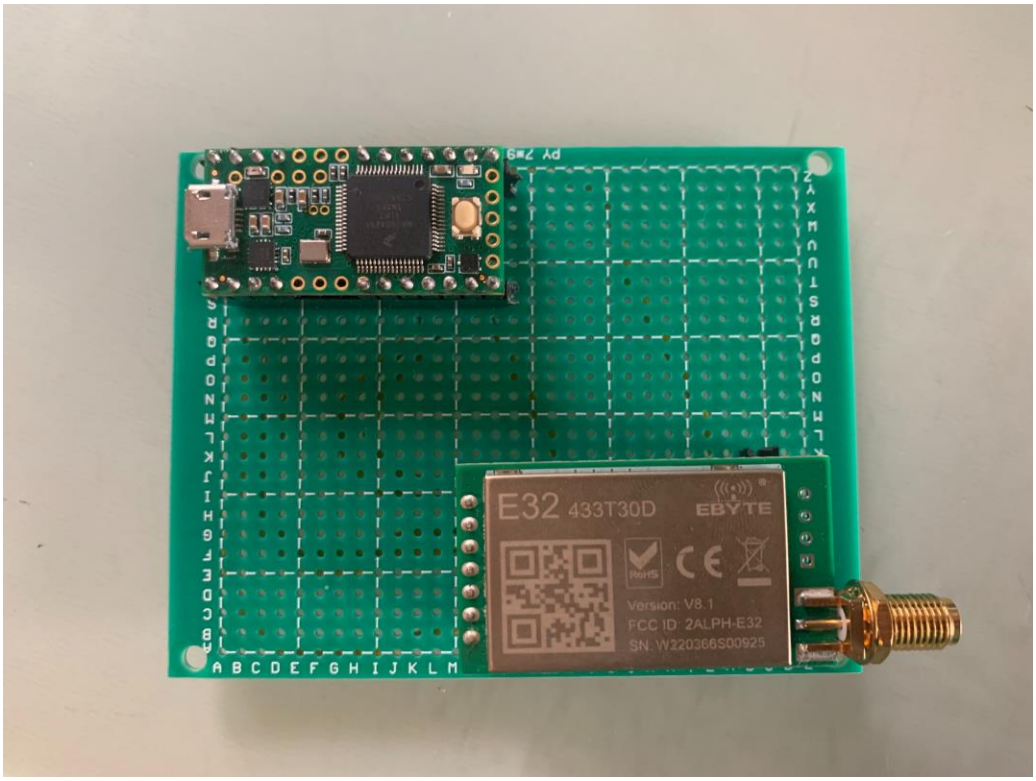
TFAC has two memory devices. The first is an SPI Infineon S25FL064LABNFI011 8MB flash chip. Data is logged to the flash chip at a rate of 0.5Hz on the pad and every 20ms in flight. Data is dumped to an onboard uSD-card when landed.

Hardware - Pyrotechnic Channels

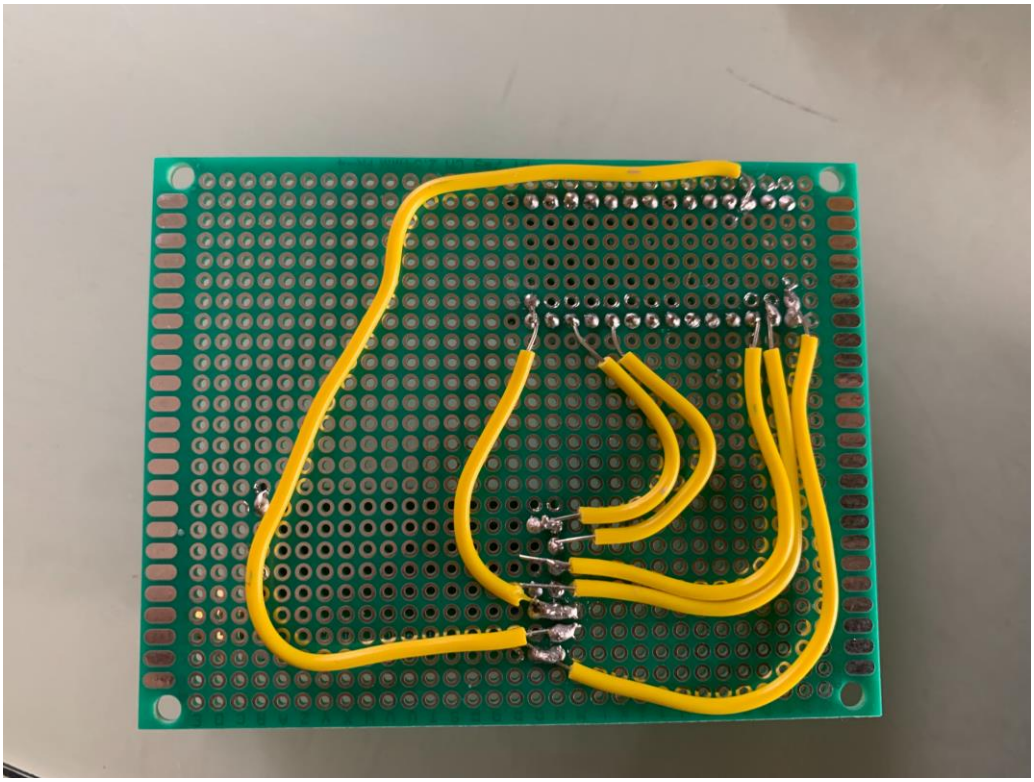
TFAC's pyro channels can be used to ignite motors, eject parachutes via black powder ejection charges, etc. This is done by passing high current/voltage to the other end of the channel for a small period of time. The Pyro Channel utilizes a UCS3205 load switch made by Microchip. The total current draw shouldn't exceed 1.5A or we are at risk of browning out the FC (done with a current limiting resistor).

Hardware - Telemetry Radio

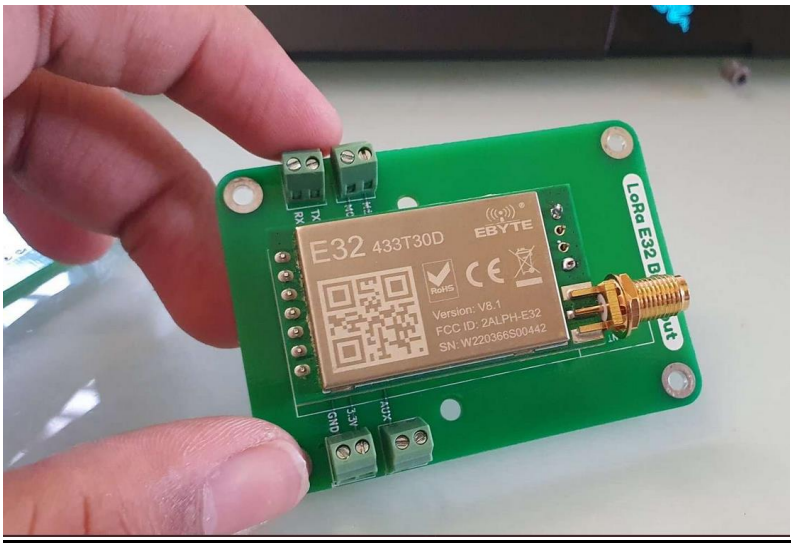
TFAC uses the Ebyte E32 LoRa 433mHz for Telemetry. 50 channels of useful pre-flight and post-flight data are sent to the ground station (*figs* 4 and 5) every 60ms. The LoRa requires at least 700mA @3.3V to operate nominally. (initially, I planned on using a 3.3V Li-Ion battery, but moved on to the following idea when I accidentally burned a LoRa Module). Since the Teensy's onboard VREG can only supply 250mA, we power the LoRa separately with the 5V line and an external 5V to 3.3V 800mA step-down buck converter. Additionally, I designed a breakout board for the LoRa to help with cable management (*fig* 6).



(*fig* 4 - Front of the ground Station)



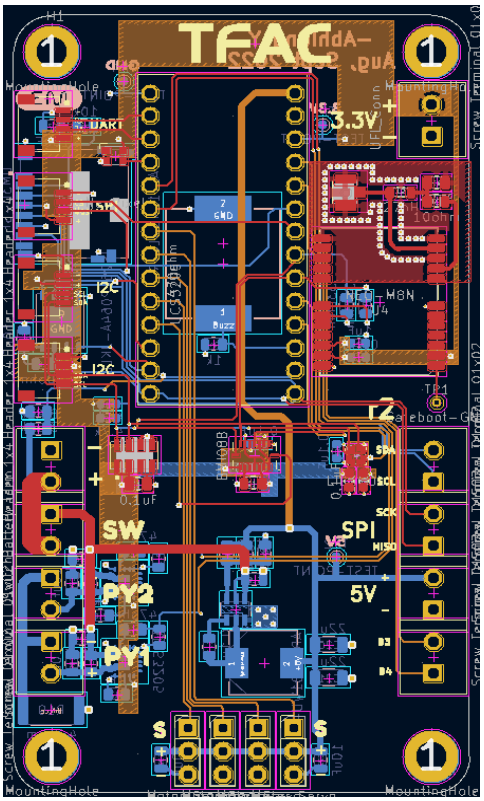
(*fig* 5 - Back of the Ground Station)



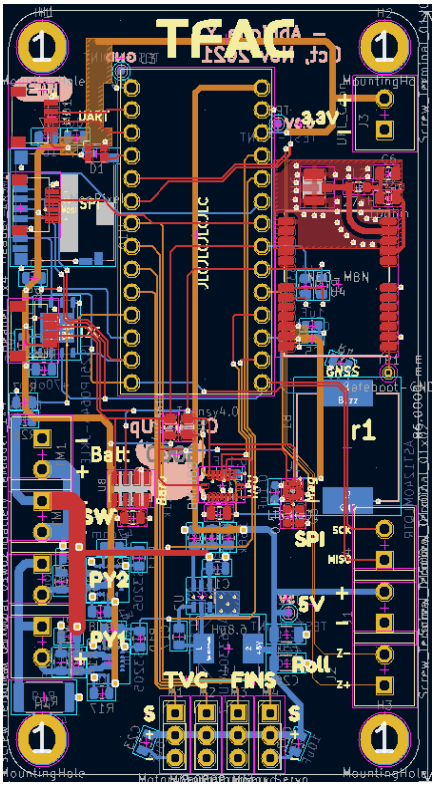
(*fig* 6 - LoRa module breakout)

Hardware - Revisions

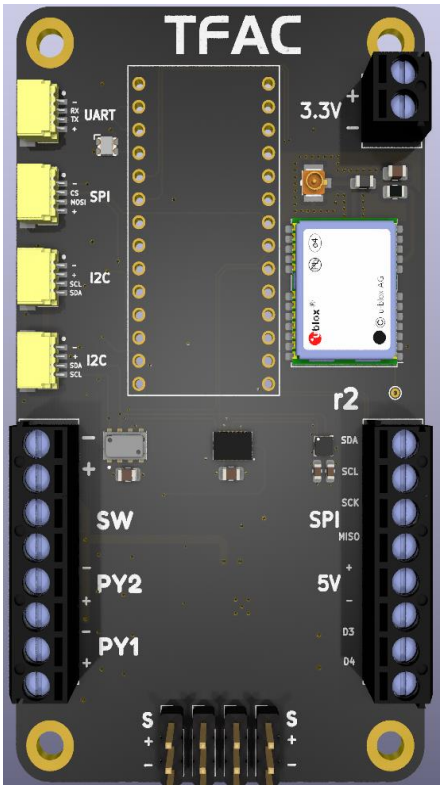
So far two revisions of TFAC have been designed. TFAC r1 is described in this document. The main changes between TFAC r1 and TFAC r2 (*figs* 7, 8, 9, and 10) are extra IO, a working onboard GNSS, and a refined layout. I don't plan on producing this board anytime soon (since the changes aren't really dealbreakers), but I have the design ready if I change my mind!



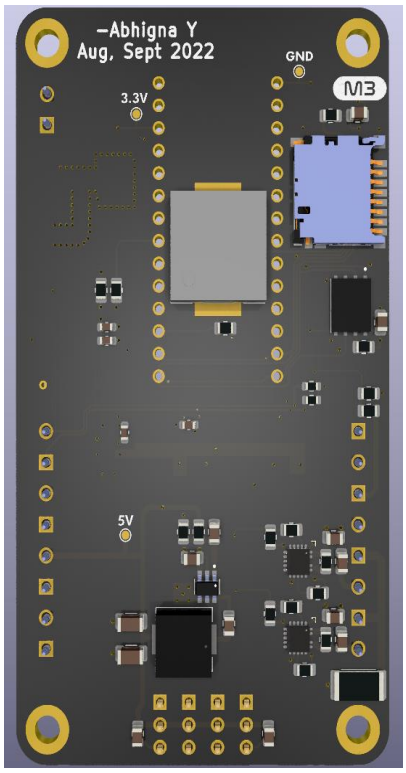
(*fig* 7 - TFAC r2 PCB Layout)



(*fig* 8 - TFAC r1 PCB Layout)



(*fig* 9 - TFAC r2 3D Model: Front)



(*fig* 10 - TFAC r2 3D Model: Back)

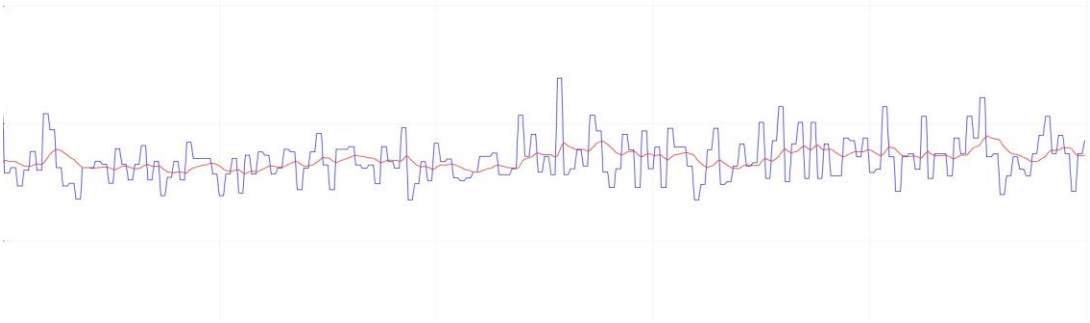
Software - State Machine

In TFAC's software, the state machine is basically a couple of "if", and "else if" statements that run a couple of functions based on what the FC is doing. The states are stored in an enum and there are 6 of them:

1. State 0 is **INIT** - Sensors are offset and FC is initialized.
2. State 1 is **IDLE** - Data is logged at a rate of 0.5Hz, and the FC is waiting for the rocket to lift off (done by checking if the vertical accelerometer readings are above 2G).
3. State 2 is **POWERED FLIGHT** - 70 channels of data are logged to the flash chip at 50Hz. The FC checks whether it's reached apogee, which is done by updating an altitude variable (the previous altitude) every 2 seconds and comparing it with the current altitude, to check if it's greater than a set threshold. If so, the FC shifts states.
4. State 3 is **DESCENT** - Pyro Channel 1 is fired to deploy 'chutes. FC checks if the altitude is less than 5m for 5 seconds, to move to the next state. Data is also logged at 50Hz to the flash.
5. State 4 is **LANDED** - Data is dumped from the flash chip to the micro SD Card.
6. State 5 is **PARTY** - The flight's over! FC cycles through RGB with happy buzzer tones.

Software - Kalman Filter

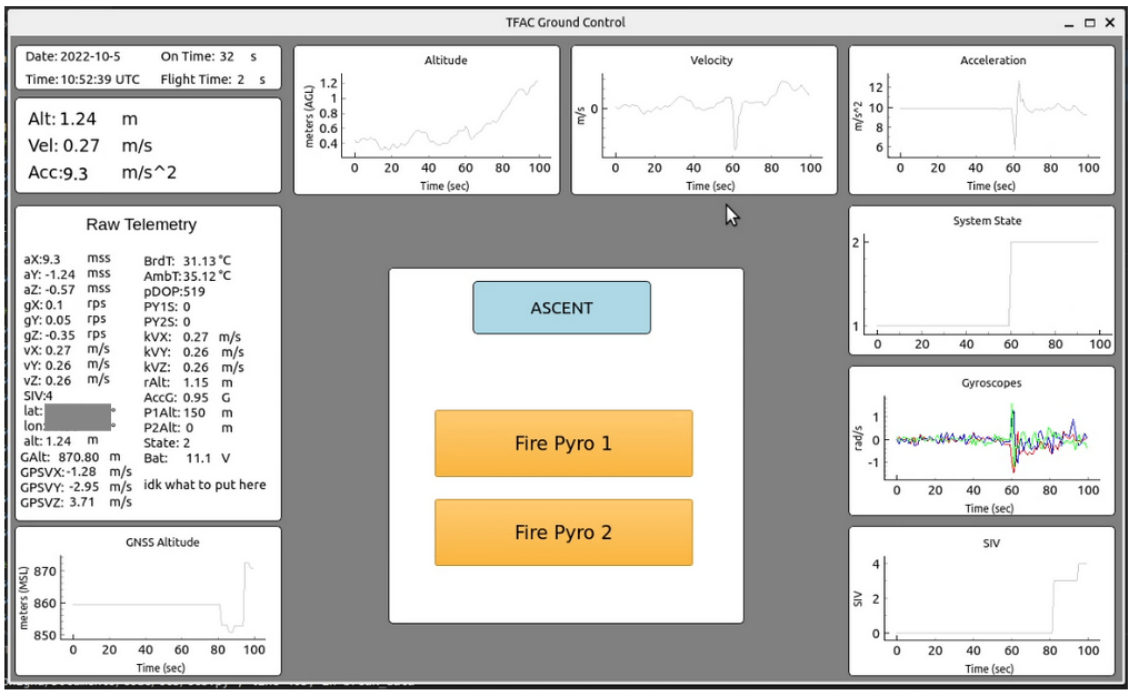
Since the sensors on TFAC are cell-phone grade MEMS sensors, some kind of filter is needed to acquire reliable sensor data. A Kalman filter was chosen to do this (*fig* 11). I decided to use the Kalman mainly because they're ideal for real-time systems and are very fast.



(*fig* 11 - Filtered vs Raw altitude)

Software and Hardware - Ground Station

The Hardware for TFAC's Ground Station is nothing more than a Teensy 3.2 hooked up to an Ebyte E32 in a 3D-printed case. The data sent by TFAC is printed to the Serial Port separated by commas. The Ground Control UI (*fig* 12) is used to display and update telemetry data in real time and can control some of TFAC's in-flight decisions/functionalities - currently can only fire either of the Pyro-Channels for 0.5 seconds. The GUI is written in Python with the PyQt5 and PyQtGraph libraries. In the center of the GUI is a white box that displays the computer's state and contains the "Fire Pyro" buttons, and is surrounded by various graphs - a box with all the telemetry data sent by TFAC, a box for the date and time, and a box that displays the altitude, vertical velocity, and acceleration in a bigger font. All the code for the Ground Control UI is here.



(*fig* 12 - TFAC Ground Control UI)

Avionics Stack - Overview

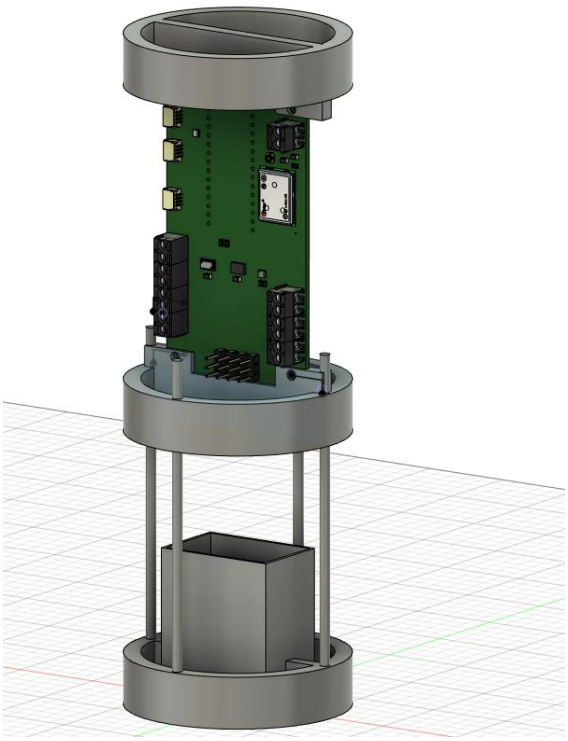
This section is more specific to Air MK1 (the vehicle for TFAC's first flight), however, it still has some amount of relevance to TFAC as a whole and that is why it's included. The avionics stacks are designed in Autodesk Fusion360 and are fully 3D-printed, usually as multiple components. They hold the TFAC, a LiPo battery (a 3s LiPo for versions 1 to 3, and a 2s LiPo for version 4), a switch, a GNSS, and the Lora breakout board (the latter two are found only in the later avionics stack versions).

Avionics Stack - Versions

So far, I have designed four versions of the avionics stack, and new versions have come around when I've made significant changes to Air MK1's design. The newest version (v4) is something I'm not totally happy with and am working on improving (this document will be updated with details on v5 when it's built).

Avionics Stack - v1

Avionics Stack v1 (*fig* 13*)* was inspired by the "mounting bracket" design found in TVC rockets, having two 'brackets', that match the inner diameter of the airframe and holes that match the location of the PCB's holes. Avionics Stack v1 was designed with three components, the first one being an upper mounting bracket. The second one is nearly identical to the upper mounting bracket, with the only change being a couple of M3 holes for the third component. The third component is similar to the previous mounting brackets but has a "box" where the LiPo could be placed vertically and four rods that screw into the M3 holes in the second component. However, I had to scrap this design, as in my view it seemed inherently unreliable.



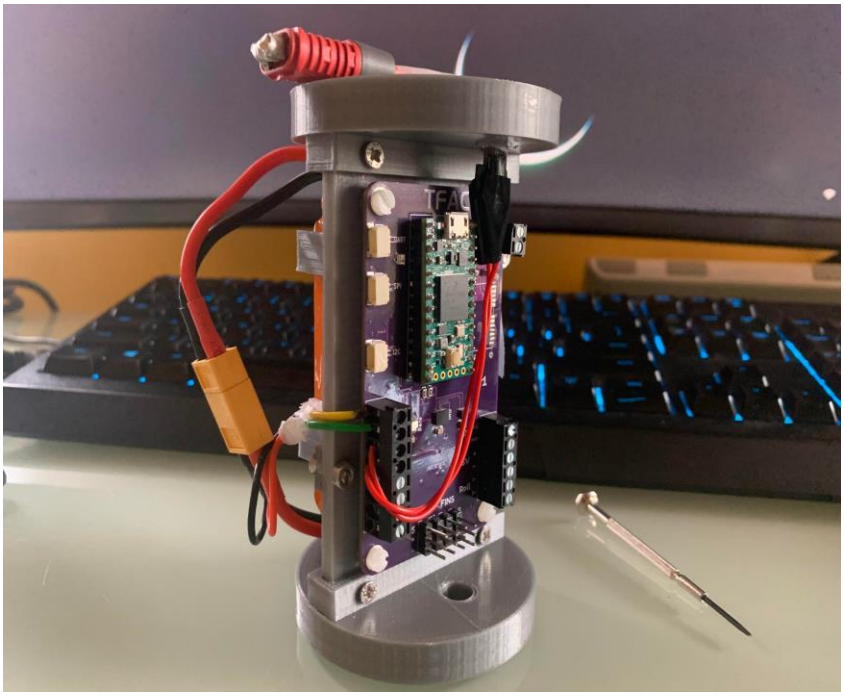
(*fig* 13 - Avionics Stack v1)

Avionics Stack - v2

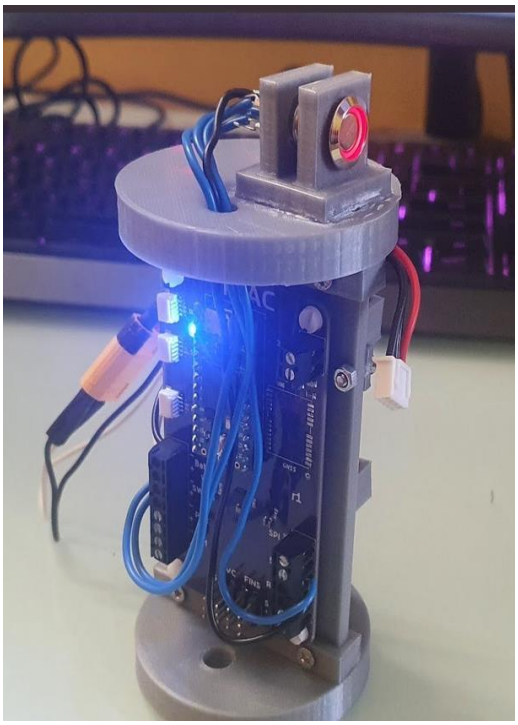
With Avionics Stack v2 (*figs* 14, 15, 16, and 17*)* I adopted the more proven-out design of the main avionics sled with electronics screwed onto it. This version also included a switch to power on/off TFAC. The original idea was to have a “power hatch”. The hatch was basically a cut-out piece of the airframe with a female T-plug glued onto it, a male T-plug would’ve been glued to the top of the avionics stack. This was later replaced with a regular-old push-button switch. The stack is again designed with three components, a sled, and two identical bulkheads. The extrusion for the push button switch was an afterthought and was glued onto the top bulkhead with epoxy. The front of the avionics sled has 4 extruded holes that line up with the mounting holes on TFAC. The back holds the LiPo. There are two plates, where the LiPo rests that are spaced according to the length of the battery. There are also two battery clips to hold the LiPo in place, each screwed into the sled on one side - this was done as I couldn’t figure out how to design the other side in Fusion360. The top plate is shorter in width than the bottom plate to make way for the LiPo’s balance connector and XT-60 plug.

(*fig* 14 - Avionics Stack v2 Front)

(*fig* 15 - Avionics Stack v2 Back)



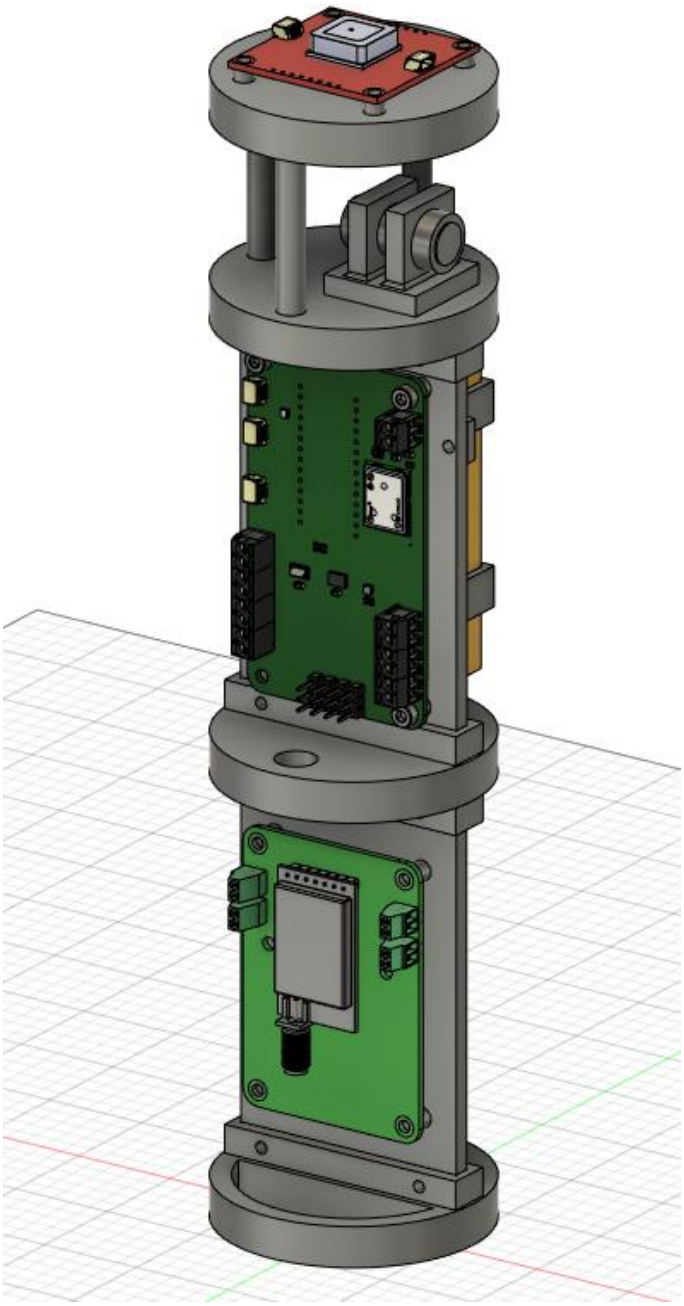
(*fig* 16 - Avionics Stack v2 with the T-plug power on/off method)



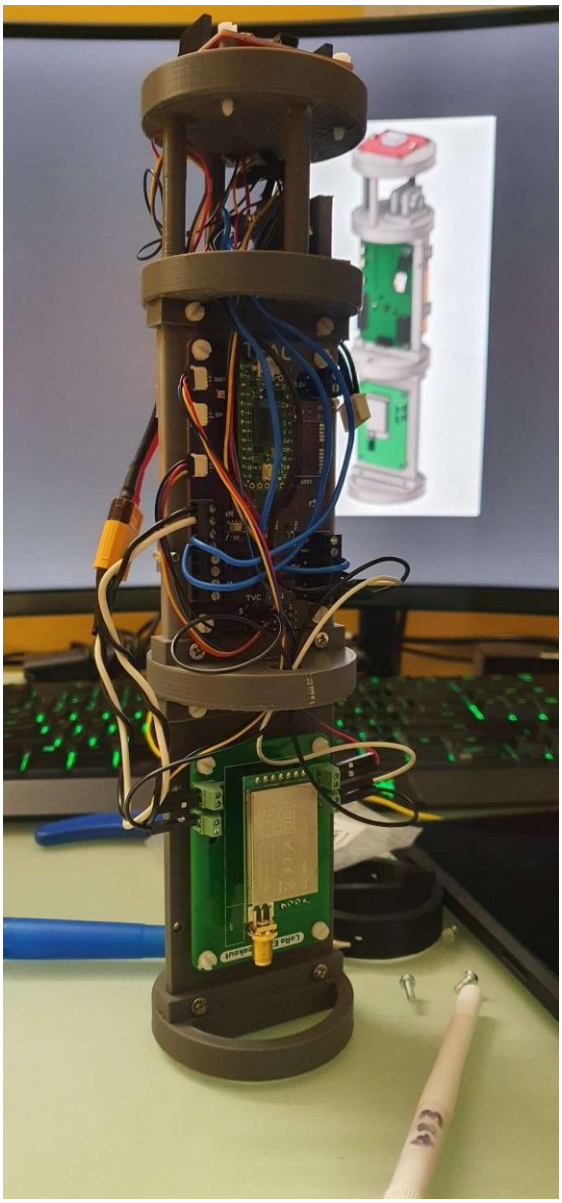
(*fig* 17 - Avionics Stack v2 with a push-button switch)

Avionics Stack - v3

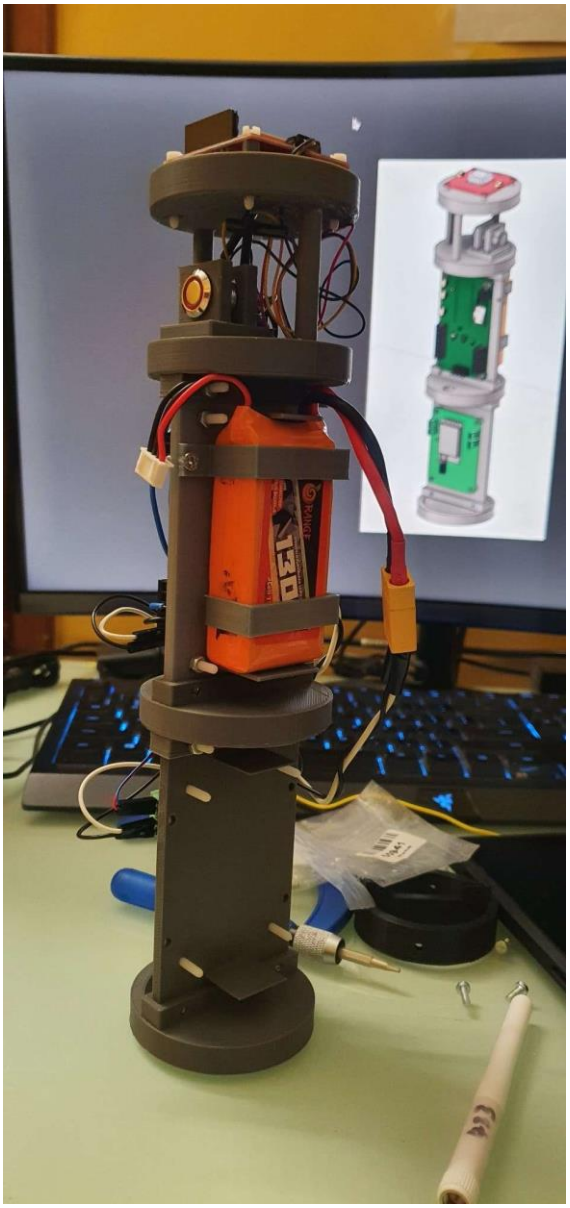
Avionics Stack v3 (*figs* 18, 19, and 20) is basically v2 in the center but with an extra sled and a different top coupler. Avionics Stack v3 was designed to accommodate the GNSS module and telemetry breakout plus the Li-Ion battery for it. The new top coupler is the old top coupler with an additional layer for the GNSS module attached on three stilts. The sled for the LoRa is the same as the one for TFAC but a little shorter and has a different arrangement of the battery holder. This design was scrapped because of the chaotic mess of wires it had turned into which would make it a pain to troubleshoot and the inherent structural flaws in the design, multiple parts broke during integration.



(*fig* 18 - Avionics Stack v3 CAD)



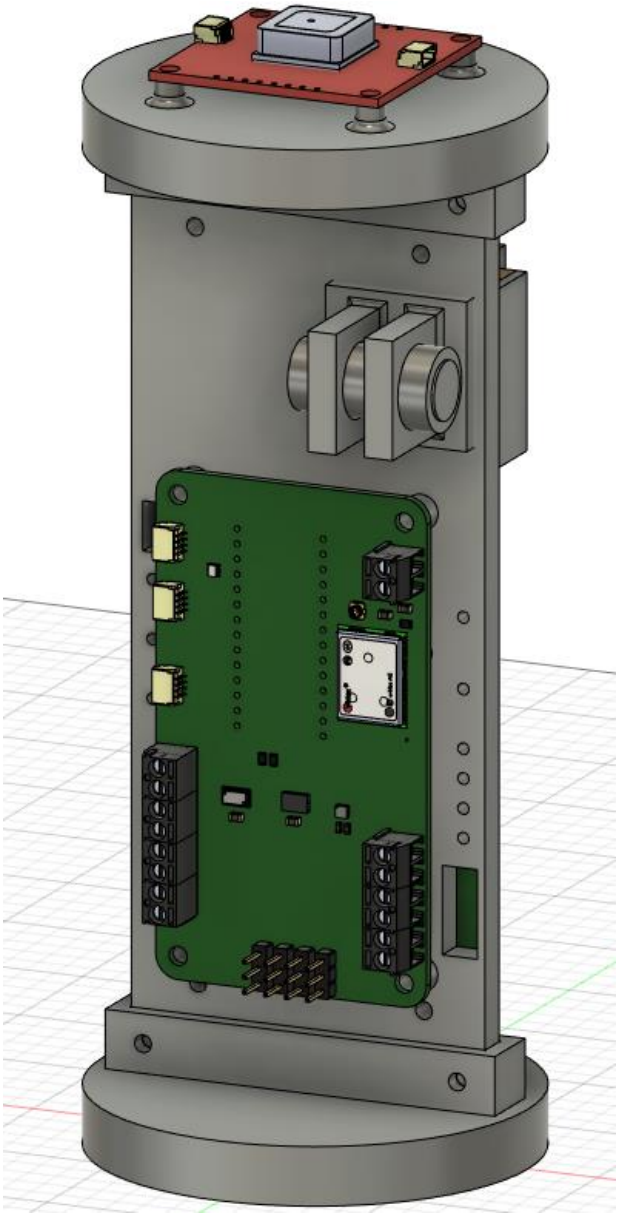
(*fig* 19 - Avionics Stack v3 Front)



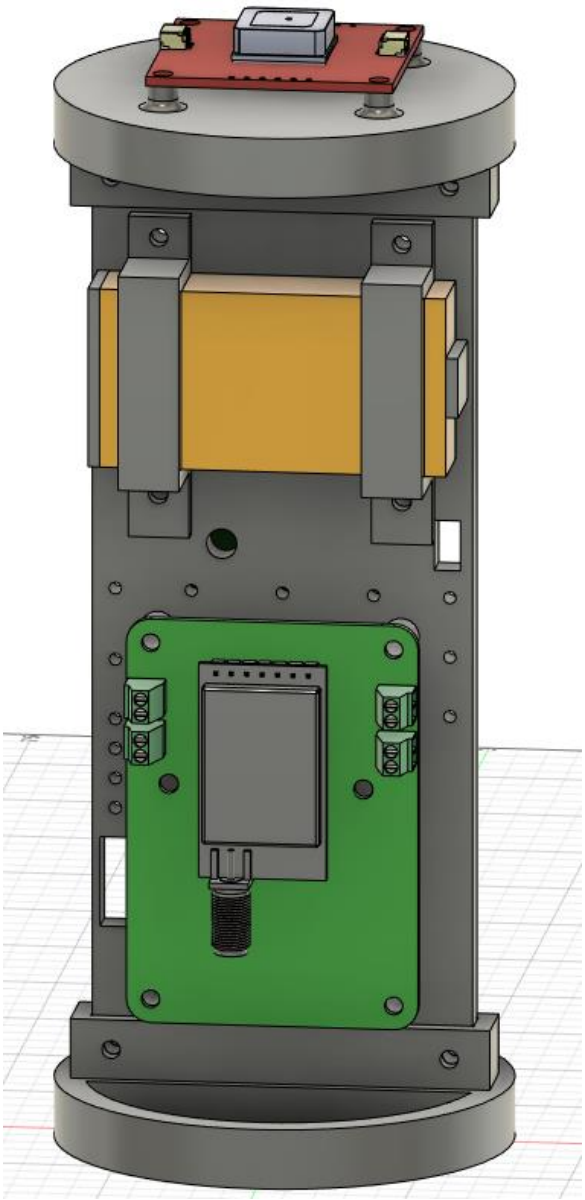
(*fig* 20 - Avionics Stack v3 Back)

Avionics Stack - v4

Avionics Stack v4 (*figs* 21 and 22) is a total redesign of the originals. It is physically larger, in diameter and width, but still consists of the same electronics except for a 2s LiPo instead of a 3s Lipo. There are two bulkheads - the one at the bottom with a cutout for the LoRa's antenna and the one at the top with extrusions for mounting the GPS. There is one big main sled instead of 2 smaller ones. The front of the sled contains the holder for the push button switch and mounting holes for TFAC. The back of the sled has the holder for the battery (same kind of design) and mounting holes for the LoRa breakout. While assembling this stack I found a lot of weak points which could compromise the structural integrity of the Avionics bay, which brought about the thought of another complete redesign.



(*fig* 21 - Avionics Stack v4 CAD Front)



(*fig* 22 - Avionics Stack v4 CAD Back)