OSU AIAA

Oregon State University Chapter of the American Institute of Aeronautics and Astronautics

Team 41 Progress Report

30k Spaceport America Cup 2017-18

Oregon State UNIVERSITY

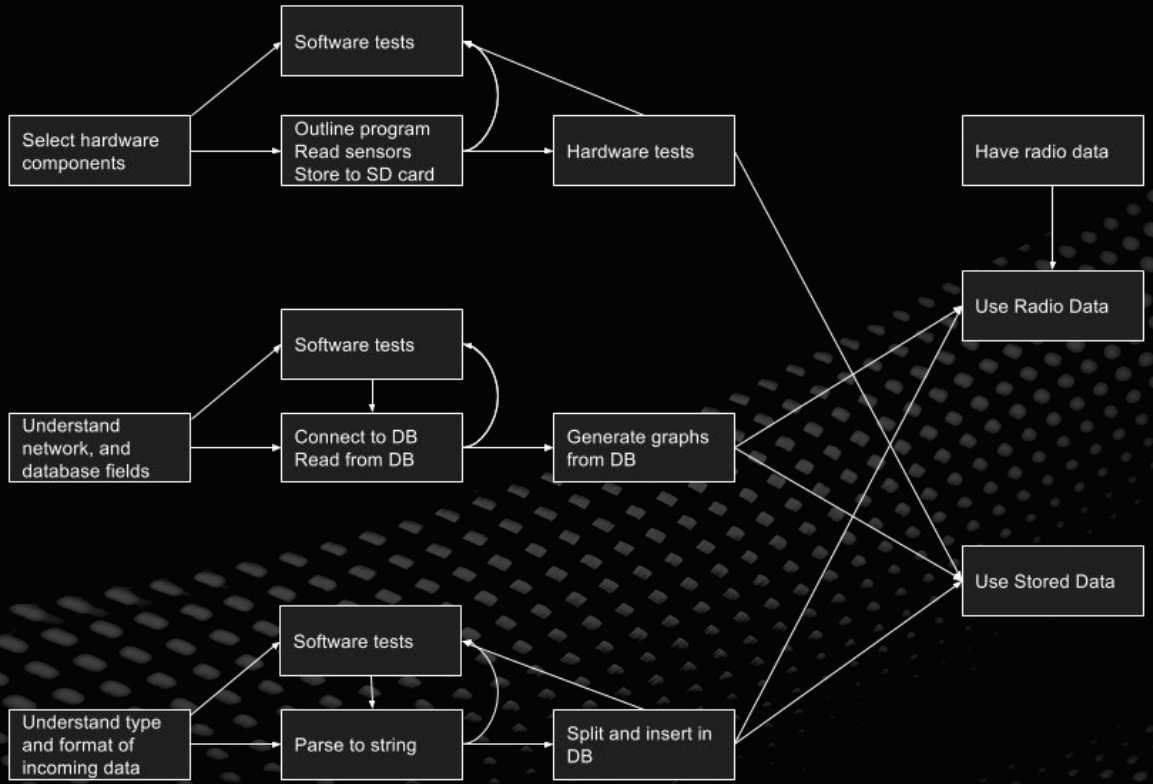
Overview

1. Project Goals
2. Development Roadmap
3. Progress and Problems
    - Ground Station
    - Database
    - Data Parsing
    - Payload Avionics
    - Client Side Display
    - Avionics Testing
    - Rocket Avionics
    - Networking & Hosting
    - Display Testing

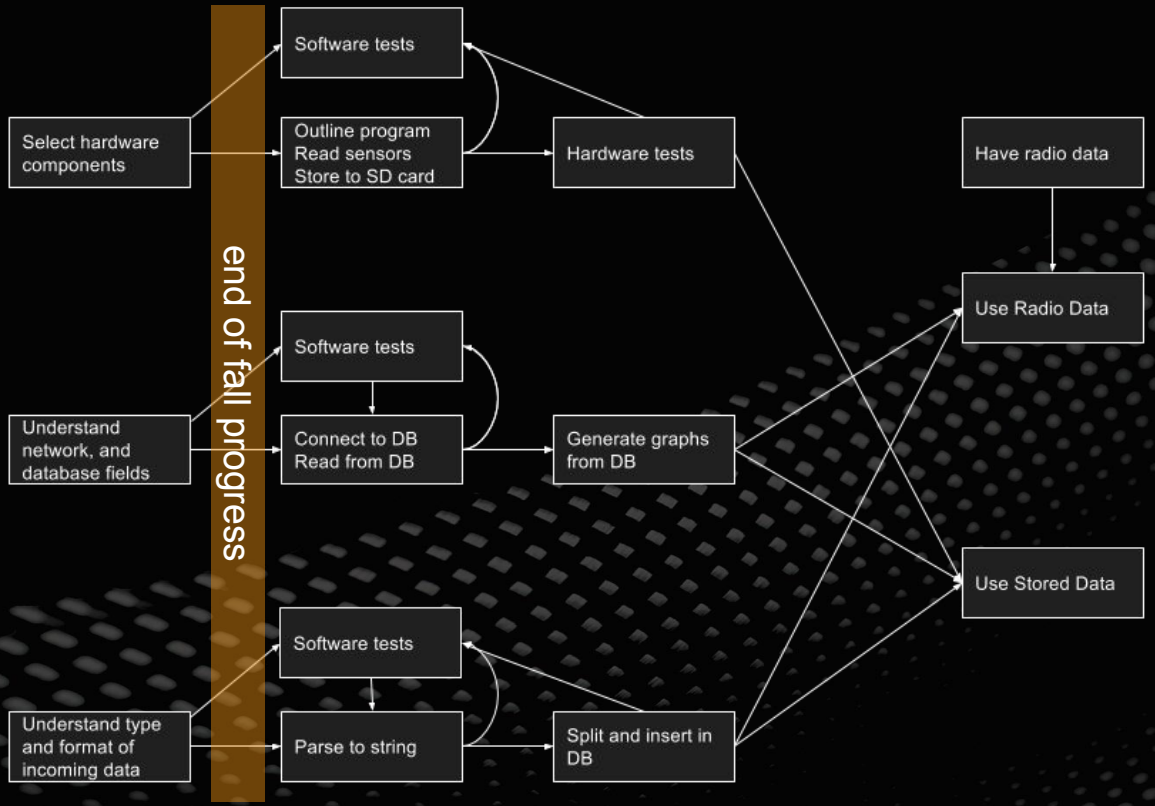
Project Goals

- Support the team and competition goals.
- Write avionics software for the rocket.
- Write software to display flight data.

Development Roadmap



Development Roadmap



Ground Station

Design Goals:

- Work without external power or networking.
- Portable, can function in a desert environment.

Tasks:

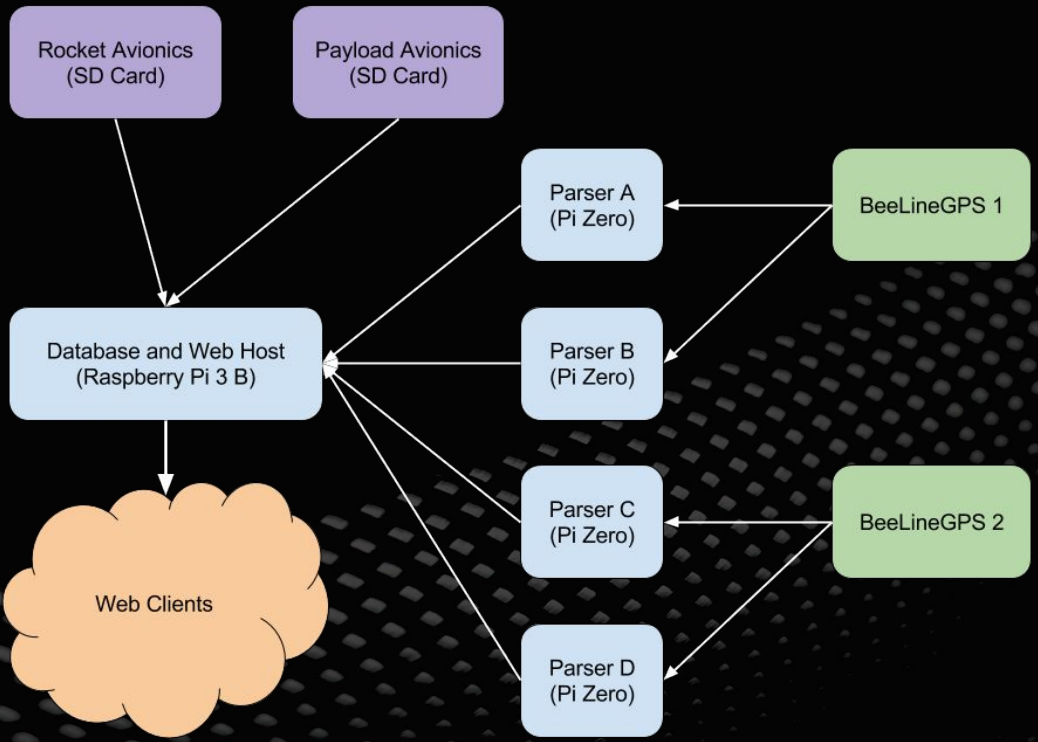
- Receive four radio signals containing live telemetry.
- Import flight data from onboard instruments using SD card.
- Store flight data in a database.
- Serve a WiFi network and web page to display flight data.

Ground Station

Problems:

1. Which microcomputers should we use?
   Decided on Raspberry Pi Zero's for parsers,
   and a Raspberry Pi 3 for the database/web host.
2. Which battery can support our computers, and for how long?
   Decided to use a 22,000 mAh USB power bank.
3. How can we protect the electronics?
   Decided to use a hardshell project case.

Flow of Information



Database

Design Goals:

- Store flight data from multiple sources.
- Allow queries from multiple clients.

Tasks:

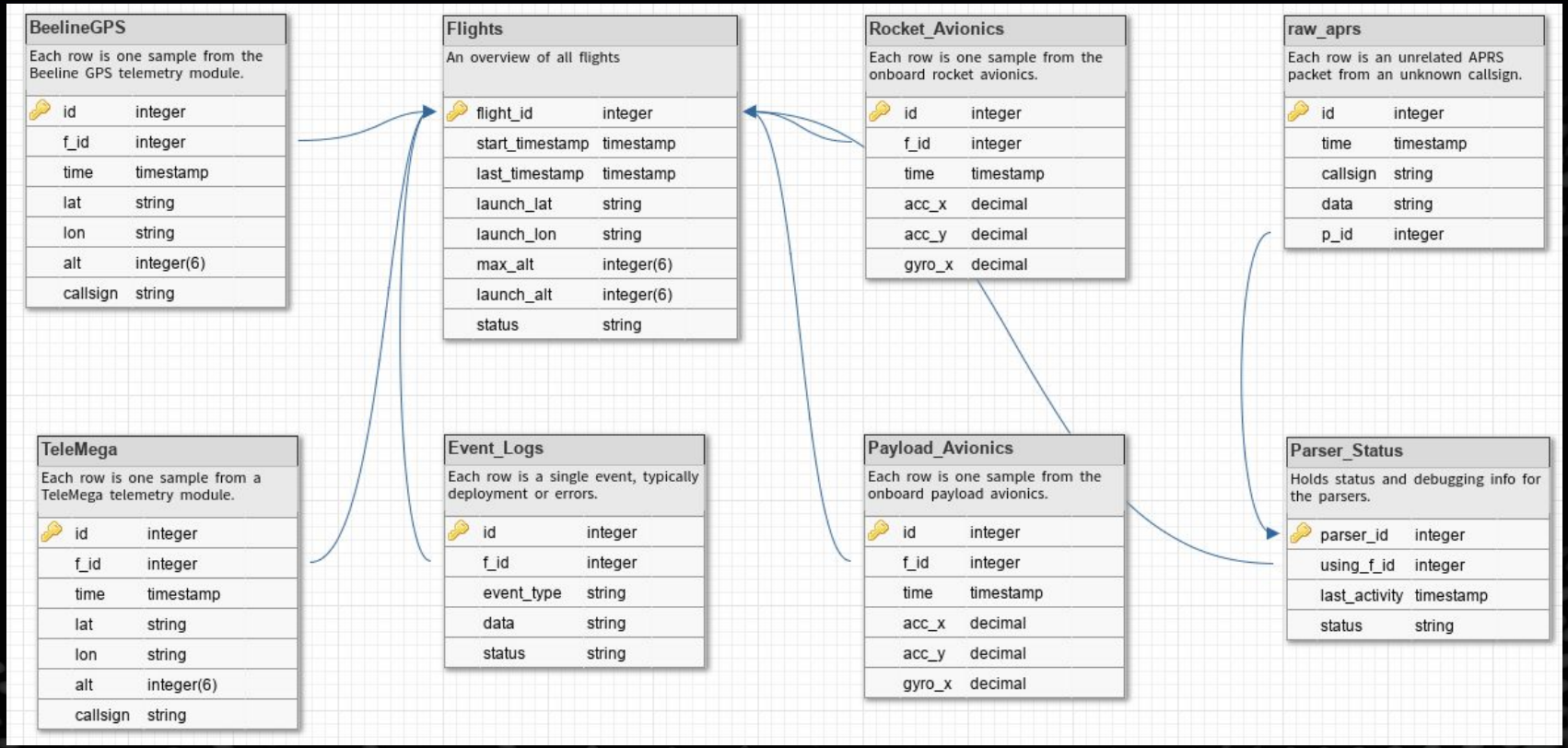
- Create a reasonable set of tables to store data.
- Think about what information will be useful.
- Find a way to relate data between sources.

Database

Problems:

1. Data from different sources will contain different fields.
   Database will contain one table per source.
2. How will we know which records are related?
   New flights are created by parsers.
   Assign a unique Flight_ID to each flight.
3. Data from different sources needs to be related.
   Use Flight_ID as a foreign key in related tables.

Database


BeelineGPS
Each row is one sample from the Beeline GPS telemetry module.
id integer
f_id integer
time timestamp
lat string
lon string
alt integer(6)
callsign string
Flights
An overview of all flights
flight_id integer
start_timestamp timestamp
last_timestamp timestamp
launch_lat string
launch_lon string
max_alt integer(6)
launch_alt integer(6)
status string
Rocket_Avionics
Each row is one sample from the onboard rocket avionics.
id integer
f_id integer
time timestamp
acc_x decimal
acc_y decimal
gyro_x decimal
raw_aprs
Each row is an unrelated APRS packet from an unknown callsign.
id integer
time timestamp
callsign string
data string
p_id integer
TeleMega
Each row is one sample from a TeleMega telemetry module.
id integer
f_id integer
time timestamp
lat string
lon string
alt integer(6)
callsign string
Event_Logs
Each row is a single event, typically deployment or errors.
id integer
f_id integer
event_type string
data string
status string
Payload_Avionics
Each row is one sample from the onboard payload avionics.
id integer
f_id integer
time timestamp
acc_x decimal
acc_y decimal
gyro_x decimal
Parser_Status
Holds status and debugging info for the parsers.
parser_id integer
using_f_id integer
last_activity timestamp
status string

Data Parsing

Design Goals:

- Receive multiple radio signals for live telemetry.
- Receive flight data stored on SD cards.
- Make flight data available for display.

Tasks:

- Each parser computer receives one radio signal as audio source.
- Parse audio for text strings using Direwolf software.
- Import files from SD cards and parse them for text strings.
- Validate text string and insert into database on network.

Data Parsing

Problems:

1. Incoming radio packets may be from other senders.
   Use a callsign field to verify source.
2. Data sources may be corrupt or contain invalid data.
   Use checksums and validate fields before inserting into DB.
3. Data from one flight may be received several hours or days apart.
   When importing from files, prompt for flight ID.

Payload Avionics

Design Goals:

- Record flight data using several electronic sensors.
- Control a propeller to reduce net acceleration.

Tasks:

- Record from several sensors including acceleration, roll/pitch/yaw, temperature, humidity, air pressure, etc.
- Detect and log moment of ejection.
- Control propeller to create zero gravity environment.

Payload Avionics

Problems:

1. We don’t have direct control over the sensor choices.
   Do our best to work with the ECE team who does.
2. We may not have early access to flight electronics.
   We may have to build our own copy of their PCB.
3. How fast should we spin the propeller?
   We’ll use a closed PID loop to read acceleration, predict a new prop speed, and adjust over time.

Display - Client Side

Design Goals:

- Allow users to view both stored and live flight data.
- Create a web browser GUI to organize and display flight data.
- Supported by browsers on cell phones, tablets, and laptops.

Tasks:

- Query database for flight data.
- Display flight data in interesting and useful ways.
- Build dynamic graphs when live telemetry is available.

Display - Client Side

Problems:

1. How do we generate dynamic graphs from a database?
   Using a javascript library called CanvasJS.
2. How will the graphs look good on cell phones AND laptops?
   We're still working on this question.

Avionics - Reading Sensors

Design Goals:

- Read from available flight sensors.
- Reuse sensor code between Rocket and Payload Avionics.

Tasks:

- Check sensor status / handle errors.
- Write functions to read from each type of sensor.
- Convert measurement unit if necessary.

Avionics - Reading Sensors

Problems:

1. Which sensors are available?
   The ECE team has final say over sensor models.
2. What sensors are available for testing?
   We can borrow sensors from the ECE team or purchase.
3. How do we read from the sensors?
   All sensors will return digital values.
   Most sensors will be I2C, but some may be SPI.

Avionics Testing

Design Goals:

- Create a test suite for avionics software.
- Ensure flight events trigger on time.

Tasks:

- Write unit tests for all functions.
- Simulate launches using test data from previous flights.

Avionics Testing

Problems:

1. How can we introduce instrument failures?

   Use robustness testing.

2. How can we simulate a flight from beginning to end?

   Use flight data from last year's team, or our test launch.

3. How can we test hardware?

   We will need to test ECE hardware, and can build our own

Rocket Avionics

Design Goals:

- Record flight data using several electronic sensors.
- Use sensor data to detect and record flight events.

Tasks:

- Record from several sensors including acceleration, roll/pitch/yaw, temperature, humidity, air pressure, etc.
- Detect and log moment of apogee.
- Detect and log separation and parachute deployments.

Rocket Avionics

Problems:

1. We don’t have direct control over the sensor choices.
   Do our best to work with the ECE team who does.
2. We may not have early access to flight electronics.
   We may have to build our own copy of their PCB.

Networking & Hosting

Design Goals:

- Connect parser computers to database.
- Allow users to connect via WiFi and view Display page.

Tasks:

- Create network between Pi Zeros and Pi 3 using USB OTG.
- Create WiFi network from Pi 3.
- Create web host on Pi 3 that serves Display page.

Networking & Hosting

Problems:

1. How do we create a network over USB?
   The Pi Zero's are capable and there are tutorials available.
2. How do we serve WiFi from the Pi 3?
   Possibly using the built-in WiFi card or external router.
3. How do we serve a webpage from the Pi 3?
   Using either Apache or NGINX.

Display Testing

Design Goals:

- Ensure the graphs and information is accurate.
- Ensure the page is viewable on multiple platforms.

Tasks:

- Write unit tests for all reasonable functions.
- Simulate inputs using test and flight data.

Display Testing

Problems:

1. How do you test websites?
   We're not sure yet.
   Learn about industry testing procedures.
2. How can we ensure the page is viewable on target platforms?
   Use a variety of devices to view the pages.

Week 2

- Problem Statement Started
- Met with Nancy
- Requirements

Week 3

- Met with TA
- Met with ECE Team
- Meeting with 100k Mentors

Week 4

- Looking for Code from Previous Year

Week 5

- Requirements Document
- CanvasJS
- ECE Telemetry Unit Choice

Weeks 6-7

- Tech Review Started and Broken Up

Week 8

- CanvasJS Licensing Issue
- Finishing Tech Review

Week 9

- ECE Team will Write their Own Code
- Design Document Responsibilities
- Problem with Triggering Flight Events

Week 10

- Design Document and Progress Report
- Nancy Approved of Using Altus Metrum