Basic Ballooning Modules

Team Members:

- Ben
- Blake
- Forrest
- Segee
- Hummels

Areas of Focus	**Member**
APRS **Description**: We'll need to focus on parsing a stream of APRS data and returning the most recent latitude and longitude in dotted decimal format and altitude in meters. This will use the conversion libraries where necessary. **Initial Deliverable:** For now, if we could get a simple function that is passed a stream location (think stdin file for testing) and can return the appropriate data, we'll be in good shape. **Helpful Stuff:** ● https://github.com/kd7lxl/python-libfap/ - Python bindings for the awesome perl aprs parser	
Elevation **Description:** Along with determining where the balloon will reach altitude 0, it is also important to know where the balloon will actually hit the ground. So, given a coordinate, we should be able to tell where ground level is (in meters). **Initial Deliverable:** Two big parts to this one: 1. Setup a storage system. This could either mean that load all the data into a database of some sort or parse through all the files. In the end, I think we'll find that while a database could mean more setup, it'll be much quicker. Someone can correct me if I'm wrong. 2. Setup a elevation query function. This should take a latitude and longitude as an input and return the elevation of the closest point in it's record. **Helpful Stuff:** ● http://dds.cr.usgs.gov/srtm/version2_1/SRTM1/Region_06/ - Elevation data for our region. ● http://www.numpy.org/ - Use numpy's readfile function	

Wind Data

Description: The driving force behind everything. The data formats for wind data are poorly document and poorly organized. Along with the code setup, a system needs to be setup to reliably retrieve wind data. This includes downloading updated data from NOAA, modifying the files as necessary and creating an easily queryable interface.

Initial Deliverable:

- Test the available interfaces for primarily speed and less ease of use. Our two main options are pygrib and pydap. Pygrib reads directly from grib files, uses little space, but has annyoing data access patterns. Pydap requires the data to be converted (to inefficient formats) and loaded into a server. The data however is much more easily accessed. These two options must be tested for speed.
- Work closely with the predictor engine. Numerous functions will be required but primarily, given a latitude, longitude, and elevation, be able to return the v and w wind speeds.

Helpful Stuff:

Conversion Libraries

Description: We'll potentially be getting data in many contradicting formats. Convert these to some standard (metric) units.

Initial Deliverable:

- Time Conversion (go between APRS local (?) and NOAA UTC)
- Latitude and Longitude (Ensure everything is in dotted decimal)
- Elevation (APRS may be feet, ensure this turns to meters)