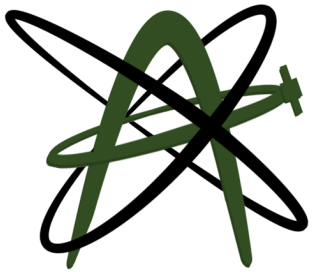
Satellite Ground Station



Critical Design Review

Group Composition

- Andrew Keller - Base Station Hardware
- Ranek Kiil - Server and Software
- Jacob Ortt - Satellite Communications
- Bryson Peeters - Project Lead

Motivation

Automated and integrated solution to track satellite and communicate with it.

- Point antenna at satellite
- Tune radio to satellite’s frequency
- Store and decode signals from space
- Send commands to satellite when it passes
- Provide public interface to stored data

Hardware Overview

- 70cm Yagi-Uda Antenna
- Circular Polarization
- Half-Duplex Communication

- Kenwood TS-2000
- Filters signals from Antenna
- Sends broadcast signal

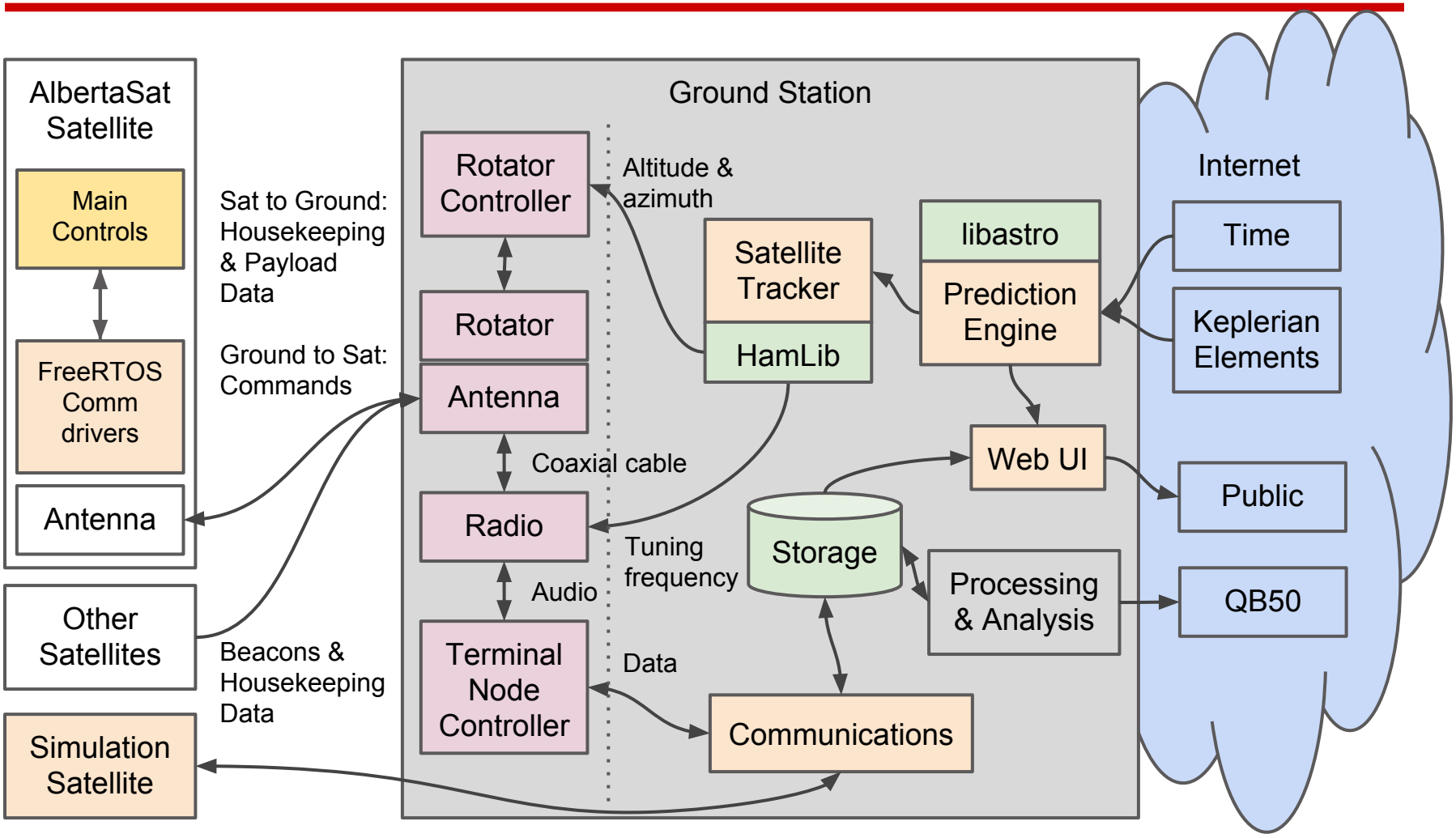
- Terminal Node Controller
- Modulates outgoing data
- Demodulates incoming data

- Linux based server terminal
- Accessible from Internet
- Implements Software Component

- Alfa MD-01 Controller
- Controls the speed of the Rotor and alignment of the Antenna.

- AlfaSpid BIG RAS/HR
- Rotates the Antenna to follow the satellite orbit.

Unified Overview



Challenges / Design Calculations

- Doppler effect on Tx and Rx frequencies
- Interfacing between software components of the ground station and satellite itself.
- Testing with temporary, non-ideal installation.
- Continuous rotation of the rotor is impractical so intervals must be timed to catch the rising edges of the satellite transmission.
- Understanding antenna polarization

Code Example

```
import numpy
import ephem
import datetime
import requests
import dateutil

def loadTLE(data):
    """ Loads a TLE file and creates a list of satellites."""
    f = data.splitlines()

    # Chunk by three-line sets
    satellites = [f[i:i+3] for i in range(0, len(f), 3)]

    # Iterate through and add to dictionary
    satellite_list = {}
    for satellite_lines in satellites:
        satellite_lines = map(str, satellite_lines)
        satellite_list[satellite_lines[0].strip()] = satellite_lines

    return satellite_list
```

Code Example

```
# Define rooftop at the University of Alberta Campus
rooftop = ephem.Observer()

# These figures currently approximate CCIS Observatory
rooftop.lat = numpy.deg2rad(53.528)
rooftop.lon = numpy.deg2rad(-113.527)
rooftop.elevation = 700

# Calculations are relative to today
rooftop.date = datetime.datetime.utcnow()

# Load Keplerian Elements
tle_data = requests.get("http://www.celestrak.com/NORAD/elements/cubesat.txt")
satellites = loadTLE(tle_data.text)
satellite_lines = satellites["QB50P2"]
satellite = ephem.readtle(*satellite_lines)
```

Code Example

```
# Calculate the next pass
info = rooftop.next_pass(satellite)
rise_time = info[0].datetime()
transit_time = info[2].datetime()
set_time = info[4].datetime()
pass_time = (set_time - rise_time).seconds
lt = ephem.localtime(info[2])


print("Next rise at %s, set at %s, transit at %s, pass duration %s seconds." % (rise_time, set_time, transit_time,
pass_time))


# Make a range of times every second for the duration of the pass
times  = [rise_time - datetime.timedelta(seconds=5) + datetime.timedelta(seconds=x) for x in range(0, pass_time +
10)]


# Compute satellite locations at each datetime
altitudes, azimuths = [], []
for date in times:
    rooftop.date = date
    satellite.compute(rooftop)
    altitudes.append(numpy.rad2deg(satellite.alt))
    azimuths.append(numpy.rad2deg(satellite.az))
```

Test Plan

- Communication protocol tested using simple RS-232
- Weather satellites for testing of tracking
- Two “precursor” cube-sat satellites for testing receipt of beacons and housekeeping data
- ISS for testing transmission

Feature Priority

- Prediction Engine
- Rotator Controls
- Communications Link (Ground Station)
- Data Processing and Analysis
- Web UI

Optional:

- Communications Link (Satellite)
- Communications with other university satellites in the program

Questions?