CubeSat Design Review

GonkBots // April 17, 2022

Outline of Design Review

1. SMART Requirements
2. Concept of Operations
3. Software Architecture
4. Mechanical Layout
5. Link Budget
6. Power Budget
7. Mass Budget and Bill of Materials
8. Data Budget
9. Integration and Test Plan

SMART Requirements

Goal: Identify colored pieces of plastic in an ocean from a birds-eye view (4 weeks)

Week 1: Basic Subsystem Function

- Create functions for plastic location
- Test ACDS code
- Ensure data accuracy

Week 2: Communications

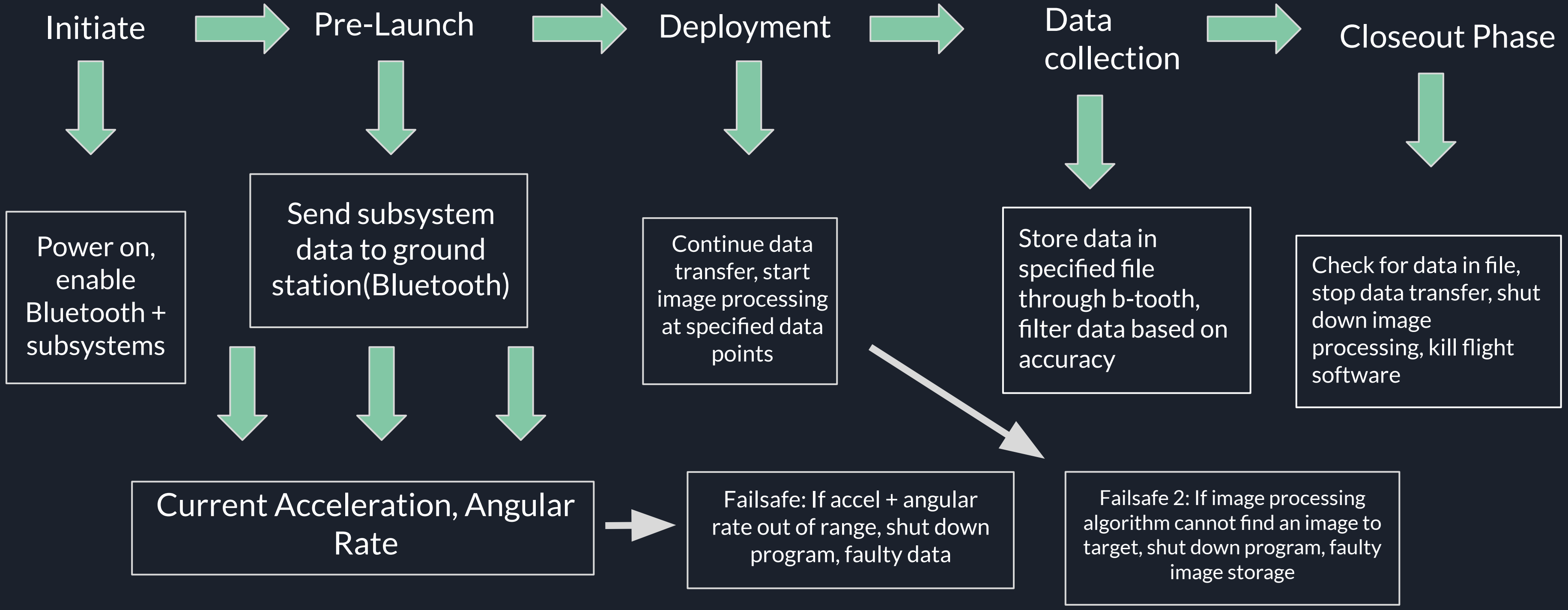
- Finalize decision on ground station laptop
- Test Pi Bluetooth, data transfer
- Ensure data transfer rates constant throughout deployment

Week 3: Testing

- Run through full deployment procedures
- Test CubeSat in different environments, tweak code + design to match

Week 4: Finalize

Concept of Operations



Initiate

Pre-Launch

Deployment

Data collection

Closeout Phase

Power on, enable Bluetooth + subsystems

Send subsystem data to ground station(Bluetooth)

Continue data transfer, start image processing at specified data points

Store data in specified file through b-tooth, filter data based on accuracy

Check for data in file, stop data transfer, shut down image processing, kill flight software

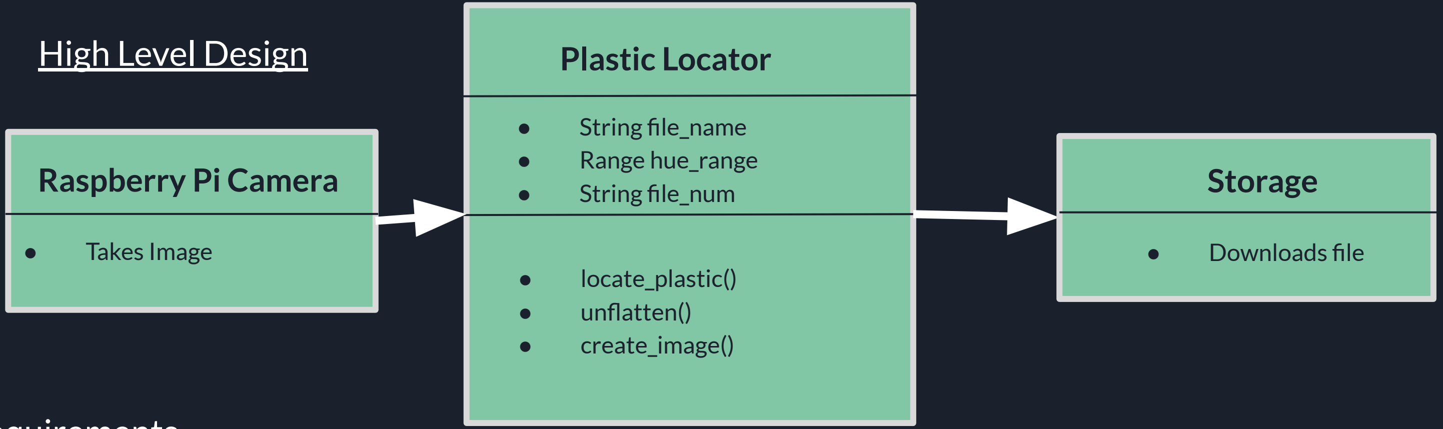
Current Acceleration, Angular Rate

Failsafe: If accel + angular rate out of range, shut down program, faulty data

Failsafe 2: If image processing algorithm cannot find an image to target, shut down program, faulty image storage

Software Architecture

High Level Design



Raspberry Pi Camera

- Takes Image

Plastic Locator

- String file_name
- Range hue_range
- String file_num

- locate_plastic()
- unflatten()
- create_image()

Storage

- Downloads file

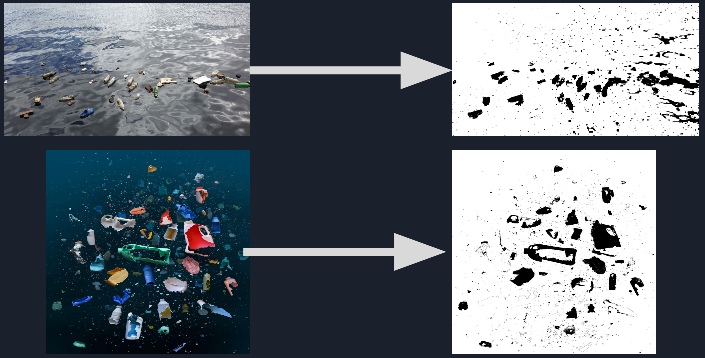
Software Requirements

- Identify colored plastics in the ocean
- Take pictures from Raspberry Pi Camera

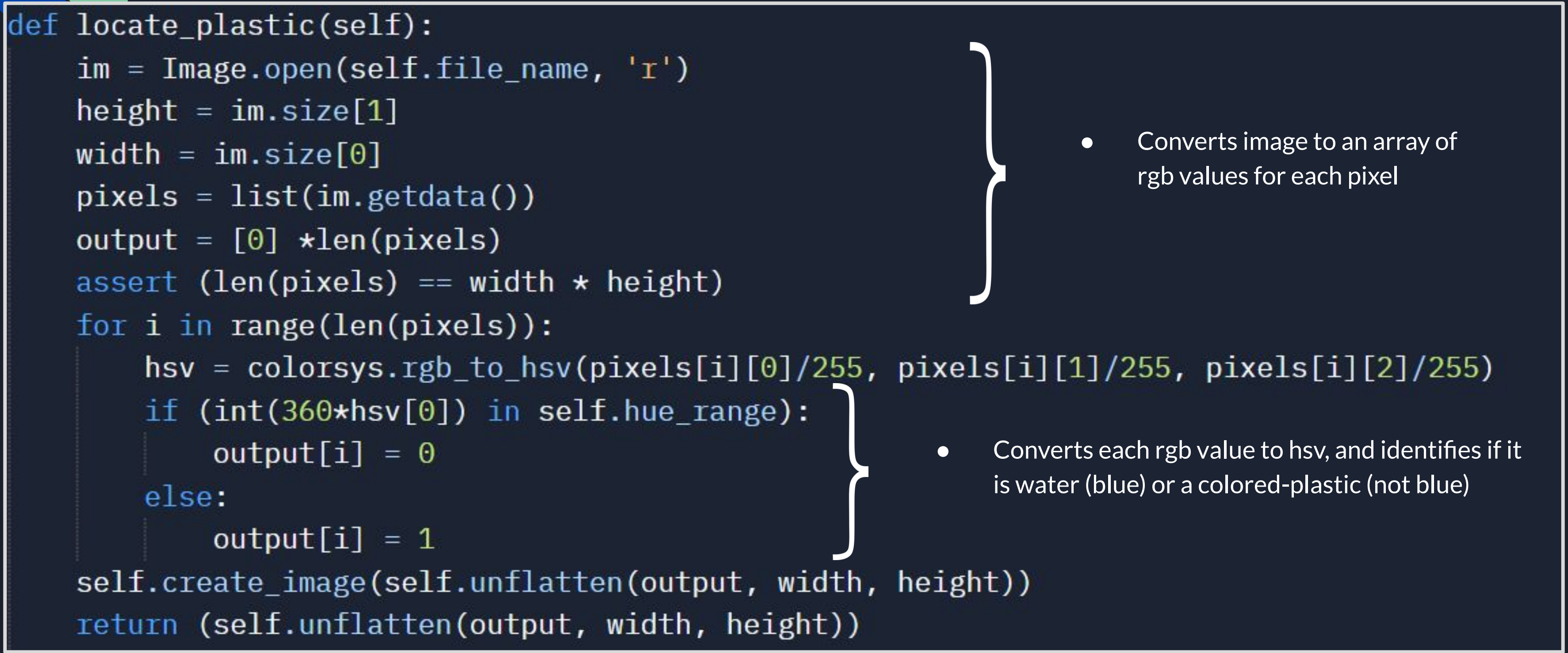
Plans for SW Development

- Integrate plastic locator with ADCS and Flight Software

Sample Tests



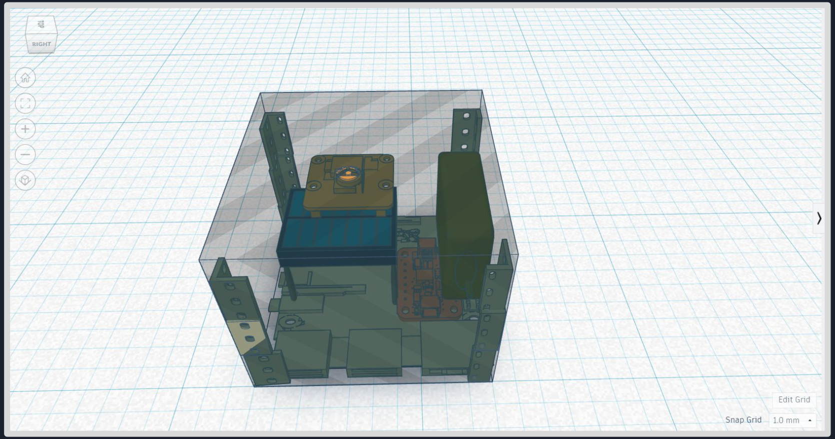
locate_plastic() Code



```python
def locate_plastic(self):
    im = Image.open(self.file_name, 'r')
    height = im.size[1]
    width = im.size[0]
    pixels = list(im.getdata())
    output = [0] *len(pixels)
    assert (len(pixels) == width * height)
    for i in range(len(pixels)):
        hsv = colorsys.rgb_to_hsv(pixels[i][0]/255, pixels[i][1]/255, pixels[i][2]/255)
        if (int(360*hsv[0]) in self.hue_range):
            output[i] = 0
        else:
            output[i] = 1
    self.create_image(self.unflatten(output, width, height))
    return (self.unflatten(output, width, height))
```

- Converts image to an array of rgb values for each pixel
- Converts each rgb value to hsv, and identifies if it is water (blue) or a colored-plastic (not blue)

Mechanical Layout



Snap Grid 1.0 mm

Basic CubeSat plan. Components should include most acrylic pieces, corner rails, and hardware that interacts with the pi.

Simple, elegant design, prioritizing ease of access to hardware and lasting battery life.

Link Budget

Frequency: 2.4 GHz = 2400000000 Hz
Orbit: 1.3 m
Bandwidth: 125 MHz = 125000000 Hz

Ground Antenna (School WiFi)

- **Gain:** 0.12 dBi
- **Transmit Power:** 100 mW = 0.1 W = -10 dBW
- **Noise Temperature:** 290 K

Satellite Antenna (Raspberry Pi 4b)

- **Gain:** 0 dBi
- **Transmit Power:** 10 mW = -20 dBW
- **Noise Temperature:** 320K

Uplink

Uplink EIRP: -9.88 dBW
Free Space Path Loss: 42.325 dB
Received Power: -52.205 dBW
Noise Power: -122.579 dBW
Signal to Noise Ratio: -174.784 dB

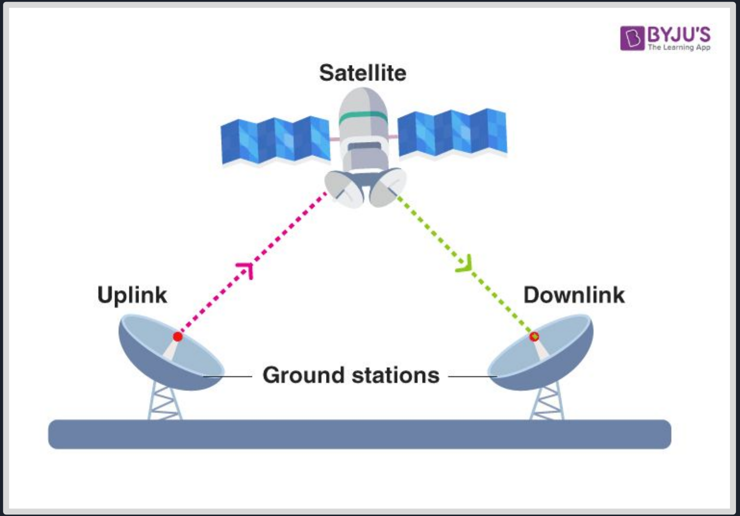
Downlink

Downlink EIRP: 0
Free Space Path Loss: 42.325 dB
Received Power: -42.205 dBW
Noise Power: -123.007 dBW
Signal to Noise Ratio: -165.212 dB

Link Budget

As satellite moves through orbit:

- Strength of link will vary, fluctuates based on position
- Radius (r) used in calculation for Free Space Path Loss
  - As radius increases, more likely that data will be lost
- Since orbit is circular, link will end up at the same strength at every revolution
- Approximately 90 Mbps



Power Budget

Powered Components:

- Raspberry Pi: 2.29 W
- Camera/IMU: (Running Pi W - Idle Pi W) = 2.485W-2.29W = 0.195W

Power Producers:

- Solar Panel (Faulty): +2.3V, 0.6mA, 0.00138W
- Solar Panel: +6V, 0.1A, 0.6W
- Fully charged battery: 10,000 mAh

Power Lab Observations:

- Solar panel not able to power pi + imu system alone
- Solar panel failed to provide current when dealing with high load
- Battery was able to provide power for a long time (-1% after 6-7 min)

Mass Budget and Bill of Materials

Item	Est Amount	Total Mass (g)
Raspberry Pi & power supply	1	46
SD Card	1	1
IMU	1	3
Camera & Case	1	6
Jumper Cables	1	2.3
Solar Panel	1	9.5

Item	Est Amount	Total Mass (g)
Velcro	2	12.2
USB Cable	1	19.7
Battery	1	146.0
Acrylic Panels	4	70.2
Acrylic side brackets	10	22.1
Screws	10	16.6

Mass Budget and Bill of Materials

Item	Est Amount	Total Mass (g)
Aluminum corner rails	4	32.0
USB Power Meter	1	12.3
SD card reader	1	14.1

Total mass: 413 grams

Data Budget

Data Transfer in our CubeSat

- IMU Accel data + Velocity
- Image files => Pi Storage(Bluetooth)
  - Process of picture capture, storage, analysis takes 3-4 sec/img
  - Data transfer takes 0.05 Seconds

Out of range/Faulty data filtered out

Influence on ConOps

- Allows for fast, accurate data collection
- Failsafe functions to stop faulty data

Integration and Test Plan

- **TESTING**
  - 5 Images with colored plastics in body of water
  - Analyze effectiveness & accuracy of program
  - Collaborate & brainstorm creative ideas

- **INTEGRATION**
  - Improve ability to detect clear/blue plastic
  - Increase scope of plastic detection
  - Identify differences between plastic and other objects in the water

Summary

- Simple design to accomplish complex tasks
- Mechanical design with room for trial and error
- Presented issues and ways to overcome them

Team GonkBots

Intro & SMART - Stef Linden

ConOps - Jonathan Wang

Systems & Software Architecture - Charles Tang

Mechanical Layout & Summary - Shivnath Shankar

Link Budget - Alex Wang

Power Budget - Divyasiddha Shivashok

Mass Budget and Bill of Materials - Isabella Palit

Data Budget - Mihir Tatavarti

Integration and Test Plan - Christopher Wang