FlyThePi

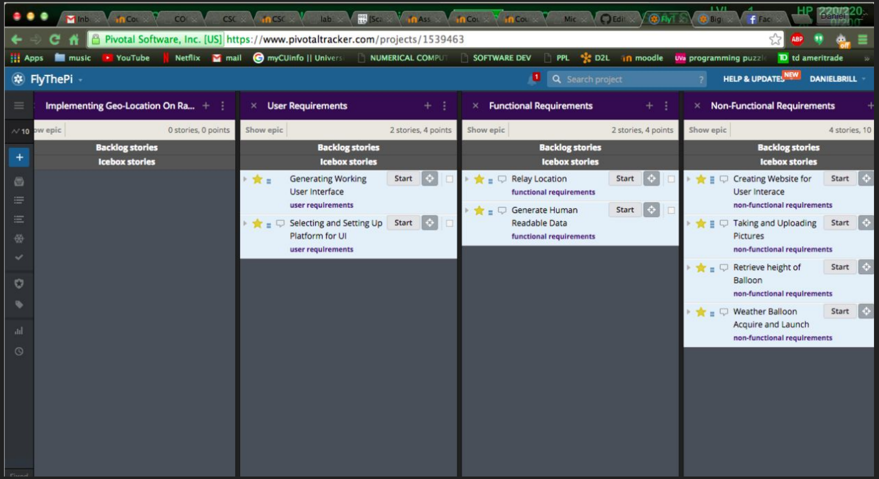
Sarah Godine, Dan Bye, Danny Brill, Bryan Hawthorne,
Lucy Wilkinson

Pivotal Tracker

Purpose: This was a means for us to track our progress with.

Use: We discovered that in order to use this you had to pay a monthly fee which did not fit the scope of the project. We decided to no longer use this platform.

Rating: 1



https://www.pivotaltracker.com/projects/1539463

FlyThePi

Implementing Geo-Location On Ra...
ow epic 0 stories, 0 points
Backlog stories
Icebox stories

User Requirements
Show epic 2 stories, 4 points
Backlog stories
Icebox stories
Generating Working User Interface — Start
user requirements
Selecting and Setting Up Platform for UI — Start
user requirements

Functional Requirements
Show epic 2 stories, 4 points
Backlog stories
Icebox stories
Relay Location — Start
functional requirements
Generate Human Readable Data — Start
functional requirements

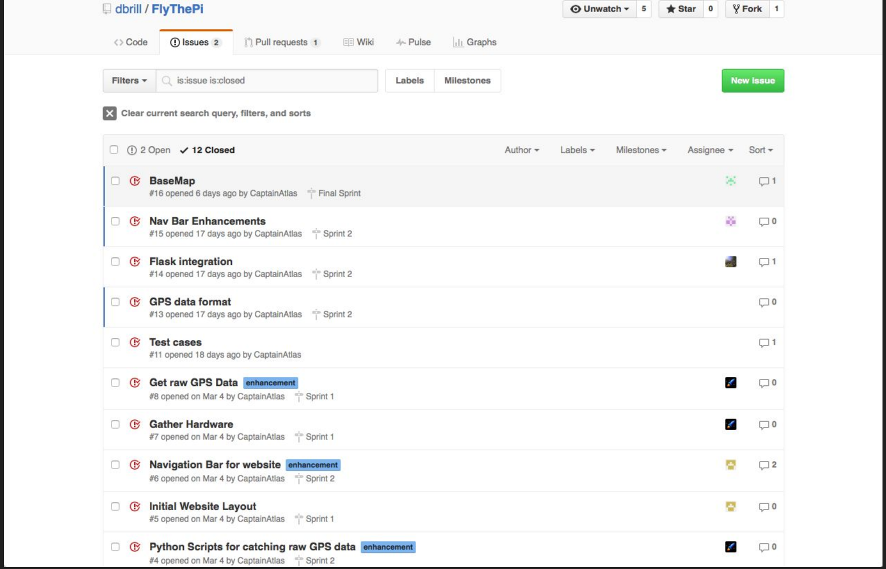
Non-Functional Requirements
Show epic 4 stories, 10
Backlog stories
Icebox stories
Creating Website for User Interace — Start
non-functional requirements
Taking and Uploading Pictures — Start
non-functional requirements
Retrieve height of Balloon — Start
non-functional requirements
Weather Balloon Acquire and Launch — Start
non-functional requirements

GitHub Issue Tracker

Purpose: This was a means for us to track our progress with.

Use: After discovering that Pivotal Tracker was not a viable option, we tried the GitHub Issue Tracker. This proved to be very functional and helped us track our project thoroughly.

Rating: 3.5



Issue	Details	Milestone	Comments
BaseMap	#16 opened 6 days ago by CaptainAtlas	Final Sprint	1
Nav Bar Enhancements	#15 opened 17 days ago by CaptainAtlas	Sprint 2	0
Flask integration	#14 opened 17 days ago by CaptainAtlas	Sprint 2	1
GPS data format	#13 opened 17 days ago by CaptainAtlas	Sprint 2	0
Test cases	#11 opened 18 days ago by CaptainAtlas		1
Get raw GPS Data enhancement	#8 opened on Mar 4 by CaptainAtlas	Sprint 1	0
Gather Hardware	#7 opened on Mar 4 by CaptainAtlas	Sprint 1	0
Navigation Bar for website enhancement	#6 opened on Mar 4 by CaptainAtlas	Sprint 2	2
Initial Website Layout	#5 opened on Mar 4 by CaptainAtlas	Sprint 1	0
Python Scripts for catching raw GPS data enhancement	#4 opened on Mar 4 by CaptainAtlas	Sprint 2	0

Methodologies

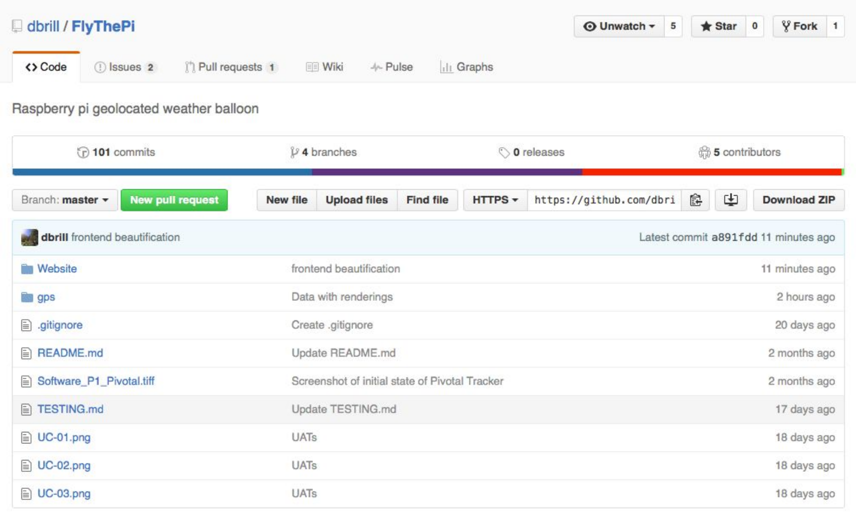
- Agile
  - We developed our project iteratively in sprints, adjusting goals to ensure we still had a working deliverable.
- Pair Programming
  - We paired up and worked on parts of the project.
- Peer Code Reviews
  - We went over each others code to ensure it was legible and functioning as intended.

GitHub Repository

Purpose: This was a place for to store all of the code we did.

Use: Each team member would upload their code to our flythepi repository. This way we had it all in one place and it was each for each member to use. This was very useful for us and a key part in bringing the project together.

Rating: 5



Raspberry pi geolocated weather balloon

101 commits · 4 branches · 0 releases · 5 contributors

https://github.com/dbri

dbrill frontend beautification · Latest commit a891fdd 11 minutes ago

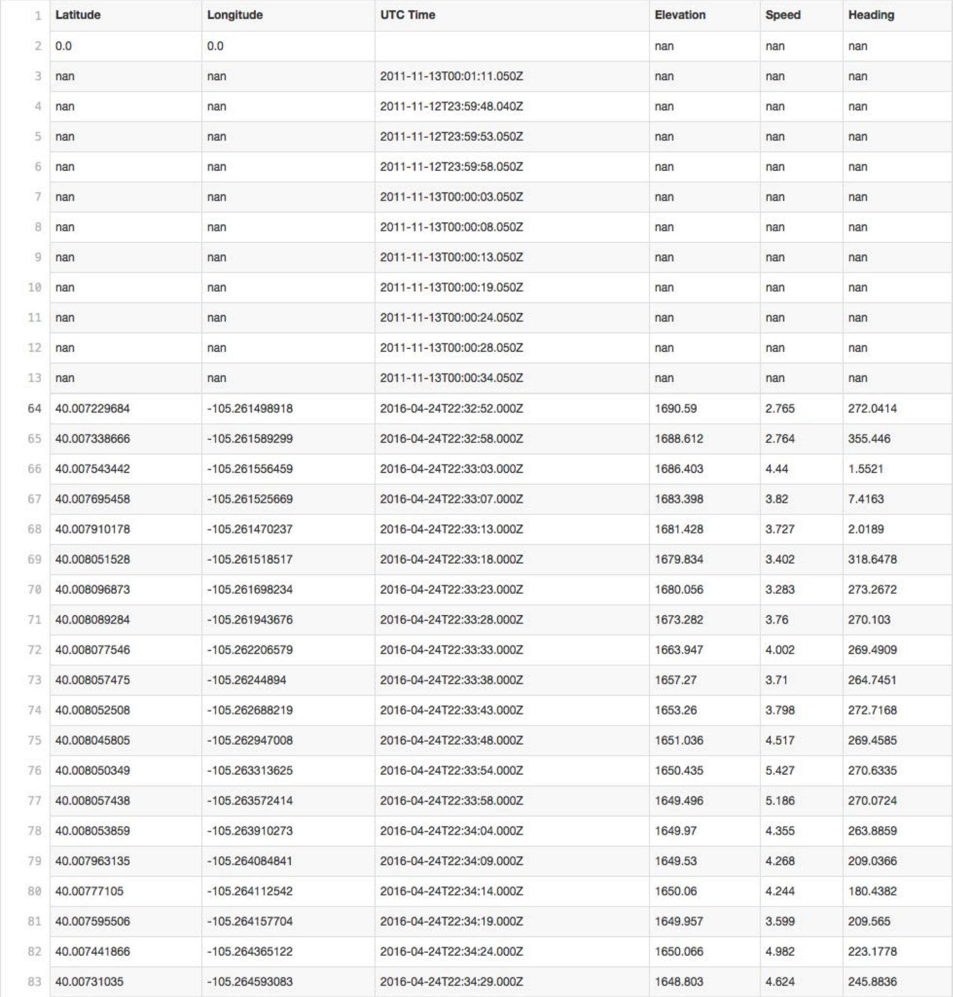
Website	frontend beautification	11 minutes ago
gps	Data with renderings	2 hours ago
.gitignore	Create .gitignore	20 days ago
README.md	Update README.md	2 months ago
Software_P1_Pivotal.tiff	Screenshot of initial state of Pivotal Tracker	2 months ago
TESTING.md	Update TESTING.md	17 days ago
UC-01.png	UATs	18 days ago
UC-02.png	UATs	18 days ago
UC-03.png	UATs	18 days ago

CSV Files

Purpose: We used a local store as our form of database.

Use: We used these files to extract the data from the RaspberryPi. We then used the file to upload it onto our webpage and make it into something usable. This was a key part in our project because without it we would not have any of our GPS information.

Rating: 4



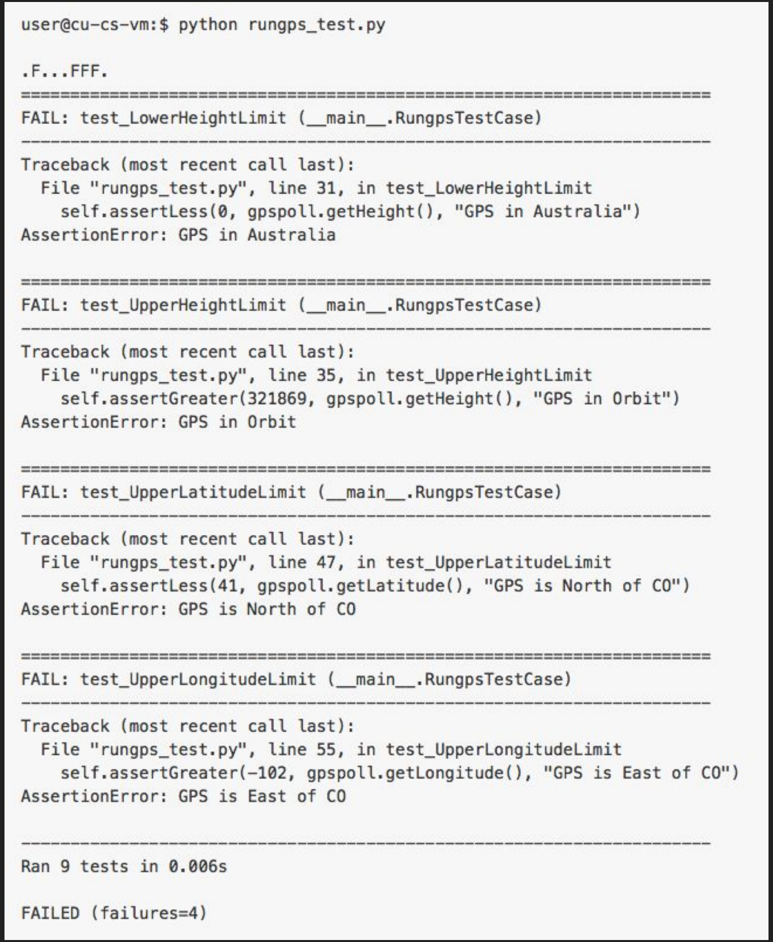
1	Latitude	Longitude	UTC Time	Elevation	Speed	Heading
2	0.0	0.0		nan	nan	nan
3	nan	nan	2011-11-13T00:01:11.050Z	nan	nan	nan
4	nan	nan	2011-11-12T23:59:48.040Z	nan	nan	nan
5	nan	nan	2011-11-12T23:59:53.050Z	nan	nan	nan
6	nan	nan	2011-11-12T23:59:58.050Z	nan	nan	nan
7	nan	nan	2011-11-13T00:00:03.050Z	nan	nan	nan
8	nan	nan	2011-11-13T00:00:08.050Z	nan	nan	nan
9	nan	nan	2011-11-13T00:00:13.050Z	nan	nan	nan
10	nan	nan	2011-11-13T00:00:19.050Z	nan	nan	nan
11	nan	nan	2011-11-13T00:00:24.050Z	nan	nan	nan
12	nan	nan	2011-11-13T00:00:28.050Z	nan	nan	nan
13	nan	nan	2011-11-13T00:00:34.050Z	nan	nan	nan
64	40.007229684	-105.261498918	2016-04-24T22:32:52.000Z	1690.59	2.765	272.0414
65	40.007338666	-105.261589299	2016-04-24T22:32:58.000Z	1688.612	2.764	355.446
66	40.007543442	-105.261556459	2016-04-24T22:33:03.000Z	1686.403	4.44	1.5521
67	40.007695458	-105.261525669	2016-04-24T22:33:07.000Z	1683.398	3.82	7.4163
68	40.007910178	-105.261470237	2016-04-24T22:33:13.000Z	1681.428	3.727	2.0189
69	40.008051528	-105.261518517	2016-04-24T22:33:18.000Z	1679.834	3.402	318.6478
70	40.008096873	-105.261698234	2016-04-24T22:33:23.000Z	1680.056	3.283	273.2672
71	40.008089284	-105.261943676	2016-04-24T22:33:28.000Z	1673.282	3.76	270.103
72	40.008077546	-105.262206579	2016-04-24T22:33:33.000Z	1663.947	4.002	269.4909
73	40.008057475	-105.26244894	2016-04-24T22:33:38.000Z	1657.27	3.71	264.7451
74	40.008052508	-105.262688219	2016-04-24T22:33:43.000Z	1653.26	3.798	272.7168
75	40.008045805	-105.262947008	2016-04-24T22:33:48.000Z	1651.036	4.517	269.4585
76	40.008050349	-105.263313625	2016-04-24T22:33:54.000Z	1650.435	5.427	270.6335
77	40.008057438	-105.263572414	2016-04-24T22:33:58.000Z	1649.496	5.186	270.0724
78	40.008053859	-105.263910273	2016-04-24T22:34:04.000Z	1649.97	4.355	263.8859
79	40.007963135	-105.264084841	2016-04-24T22:34:09.000Z	1649.53	4.268	209.0366
80	40.00777105	-105.264112542	2016-04-24T22:34:14.000Z	1650.06	4.244	180.4382
81	40.007595506	-105.264157704	2016-04-24T22:34:19.000Z	1649.957	3.599	209.565
82	40.007441866	-105.264365122	2016-04-24T22:34:24.000Z	1650.066	4.982	223.1778
83	40.00731035	-105.264593083	2016-04-24T22:34:29.000Z	1648.803	4.624	245.8836

Python Unit Testing

Purpose: This was method in which we tested the GPS in our project.

Use: We used a python unit test module to handle the automated testing. In order to do this, you had to run the command python rungps_tesy.py. For that command to work you had to make sure rungps.py was in you directory.

Rating: 4



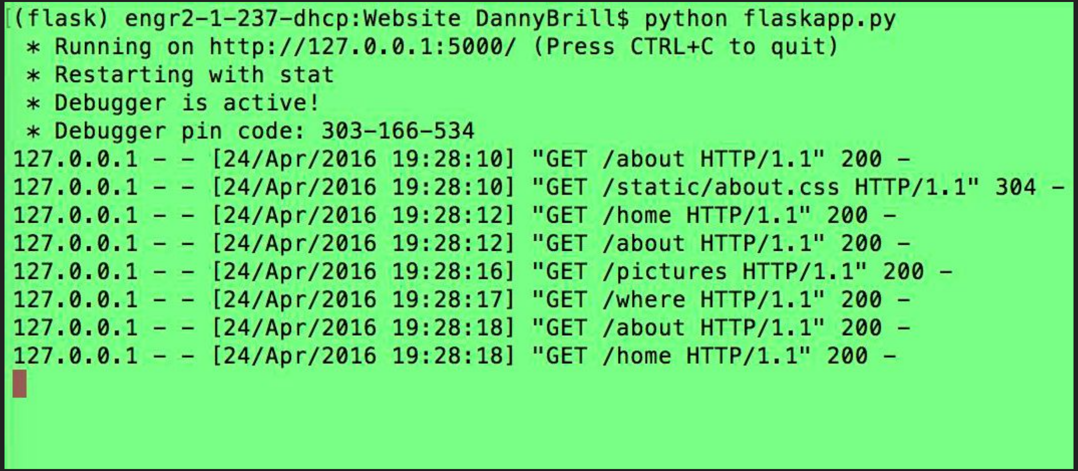
```
user@cu-cs-vm:$ python rungps_test.py

.F...FFF.
======================================================================
FAIL: test_LowerHeightLimit (__main__.RungpsTestCase)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "rungps_test.py", line 31, in test_LowerHeightLimit
    self.assertLess(0, gpspoll.getHeight(), "GPS in Australia")
AssertionError: GPS in Australia

======================================================================
FAIL: test_UpperHeightLimit (__main__.RungpsTestCase)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "rungps_test.py", line 35, in test_UpperHeightLimit
    self.assertGreater(321869, gpspoll.getHeight(), "GPS in Orbit")
AssertionError: GPS in Orbit

======================================================================
FAIL: test_UpperLatitudeLimit (__main__.RungpsTestCase)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "rungps_test.py", line 47, in test_UpperLatitudeLimit
    self.assertLess(41, gpspoll.getLatitude(), "GPS is North of CO")
AssertionError: GPS is North of CO

======================================================================
FAIL: test_UpperLongitudeLimit (__main__.RungpsTestCase)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "rungps_test.py", line 55, in test_UpperLongitudeLimit
    self.assertGreater(-102, gpspoll.getLongitude(), "GPS is East of CO")
AssertionError: GPS is East of CO

----------------------------------------------------------------------
Ran 9 tests in 0.006s

FAILED (failures=4)
```

Flask

Purpose: We used the python flask framework to host our website.

Use: We used Flask to host the website. This was how we put everything together and ran it. This was a key part in website development.

Rating: 4



```
(flask) engr2-1-237-dhcp:Website DannyBrill$ python flaskapp.py
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger pin code: 303-166-534
127.0.0.1 - - [24/Apr/2016 19:28:10] "GET /about HTTP/1.1" 200 -
127.0.0.1 - - [24/Apr/2016 19:28:10] "GET /static/about.css HTTP/1.1" 304 -
127.0.0.1 - - [24/Apr/2016 19:28:12] "GET /home HTTP/1.1" 200 -
127.0.0.1 - - [24/Apr/2016 19:28:12] "GET /about HTTP/1.1" 200 -
127.0.0.1 - - [24/Apr/2016 19:28:16] "GET /pictures HTTP/1.1" 200 -
127.0.0.1 - - [24/Apr/2016 19:28:17] "GET /where HTTP/1.1" 200 -
127.0.0.1 - - [24/Apr/2016 19:28:18] "GET /about HTTP/1.1" 200 -
127.0.0.1 - - [24/Apr/2016 19:28:18] "GET /home HTTP/1.1" 200 -
```

Harnessing GPS data

- We used GPSD to interface with the antenna module on our Pi and actually retrieve and record the data

- Used BaseMap to render visualizations of where our Pi traveled based on the GPS data stored in the CSV file.

Raspberry Pi

Purpose: This was how we obtained all of the GPS information.

Use: This was the main part of our project. It retrieved raw GPS data and stored it into a CSV file.

Rating: 5

Challenges

- Initially we struggled with where our files should be located; whether it be on the Pi, on a hosted store like heroku/github, or on a local machine.

- Retrieving data from the Pi and formatting it into our csv turned out to be a lot more time consuming than anticipated. Because the Pi’s buffer would fill up, we had to employ threading to ensure that no GPS data was lost when writing to the csv.

Demo