**Software and Hardware Report**

**for**

**Weather Station**

**Version 1.0 approved**

**Prepared by Brian Luna, Mario Diaz, Mathew Apanovich, Johnathan VanBibber, and Ashley Williams**

**CIS4935.700F18 AstroThunder**

**11/20/2018**

**Functionality**

The weather station project required both hardware and software features to be implemented in order to complete the requirements and additional features. The weather station site charts data on temperature, humidity, wind speed, wind direction, and what the weather currently looks like outside all on the same page within the final project. The line charts can be hovered over to see the exact value of the points, and all related data for the most recent reading is listed up at the top below the menu bar and label stating when the most recent reading occurred. The menu bar at the very top can direct users to past data readings from the charts listed above, to a predictive forecast for the next 12 hours, or an 8-day forecast. While all of this is going on it is being shown using the current weather from a location selected by the user using a city, ZIP code or set of coordinates.

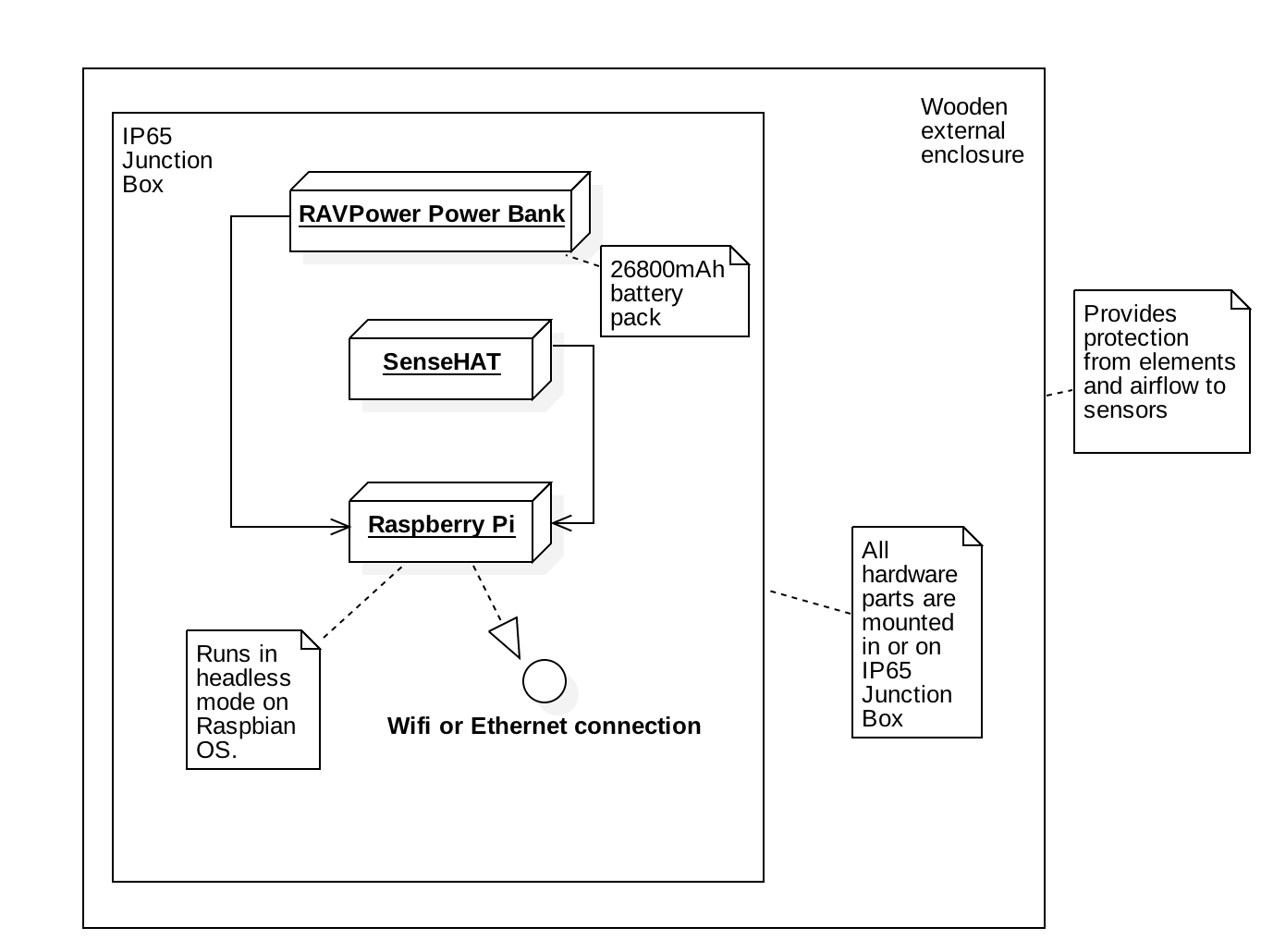
The user can see all the hardware that is used and the set up for the weather station within its case, as well as additional information about the project on the main page of the site label. Links to a Contact Us and documentation page are also provided.

**Original plan changes**

Originally the site’s charting was all going to be done with an API linked to a single python file getting readings from the Pi for temperature, humidity, and pressure. Hoping to fully flesh out the details more and customize it to the groups liking, it was instead changed to break down each component further. In order to perform the individual measurements a different hardware approach was needed.

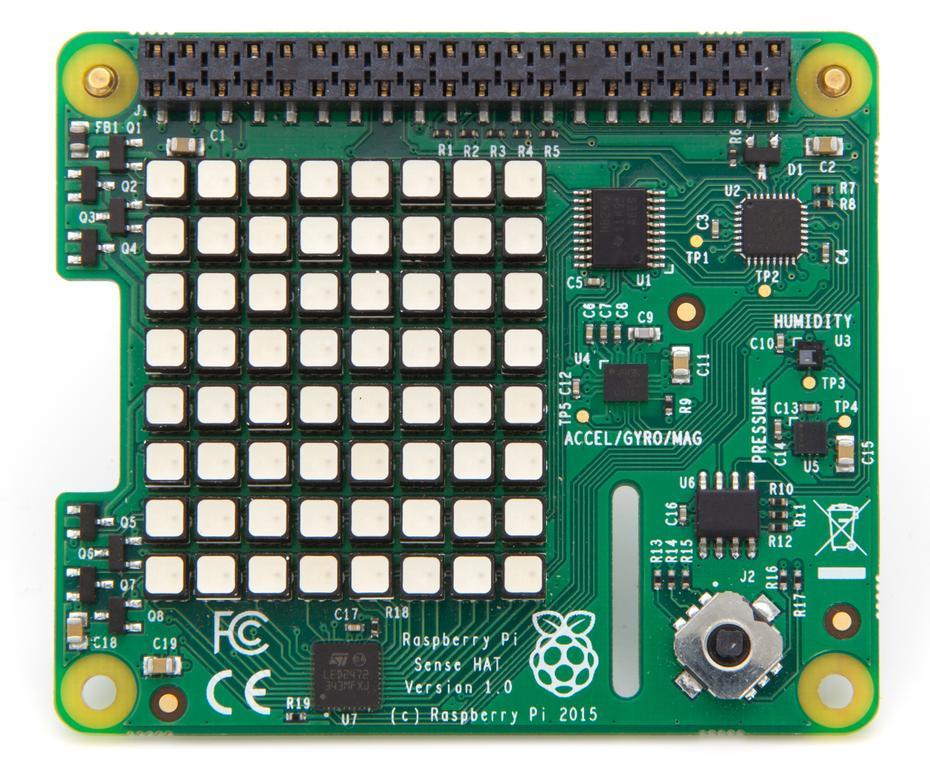
The second iteration of the plan included using a custom weather PC board from BC-Robotics which included a RJ11 port and analog to digital converter that was needed for the wind anemometer and wind vane. The assembly for this iteration included soldering the components together which allows more room for error and the use of imported Adafruit libraries. The code for the BC-Robotics guide [("Raspberry Pi Weather Station - Part 1")](https://www.bc-robotics.com/tutorials/raspberry-pi-weather-station-part-1/) on the project was used for testing this iteration. The wind anemometer was not able to be detected by the Raspberry Pi possibly due to faulty weather PC board or solder connection. Due to the replacement parts only being available from a single vendor and on backorder, a third option was chosen.

The last and final version of the project included the Pi itself, the Sense HAT, python code to get readings, a connected mysql database with all associated data, and a series of HTML documents for different pages that were all hosted on the USF FTP server with charts created by the group through the use of the Google Chart API.

**Hardware**

**Raspberry Pi Model 3 B+ **

**SenseHAT**



**Ribbon Cable\ **

For the hardware aspects of the weather station the main component was a Raspberry Pi model 3 B+, 64GB micro sd-card, and associated Sense HAT. The hardware setup provided a challenge as the Sense HAT sensor was set up to be located directly over the Raspberry Pi CPU causing temperature reading to go up. In order to mitigate this issue the sensor was moved further away from the CPU to allow more airflow with a ribbon cable. Any further differences between the standard area temperature and the temperature from the heat of the CPU were identified and calibrated accordingly. All the hardware components were placed into an enclosure to ensure no damage would be done to them from any weather the system would face while gathering data. This added another layer of difficulty as the case had to be waterproof, but also allow enough air flow to get to the Raspberry Pi and Sense HAT so that the readings would not be thrown off. In order to make the Raspberry Pi more accessible and safer in inclement weather while in the case, it was moved over to battery power, so it could be located away from any power source in a more remote area should the team ever desire.

**PICTURE OF PI ASSEMBLED**

****

Initially a wind vane was to be used to gather information on the wind speed and direction, however several challenges were introduced with this as the Raspberry Pi did not have the required connections. When additional hardware was introduced to get the required port to plug in the wind vane, the recording instrument did not function, causing the group

to move this feature over to an API called Open Weather.

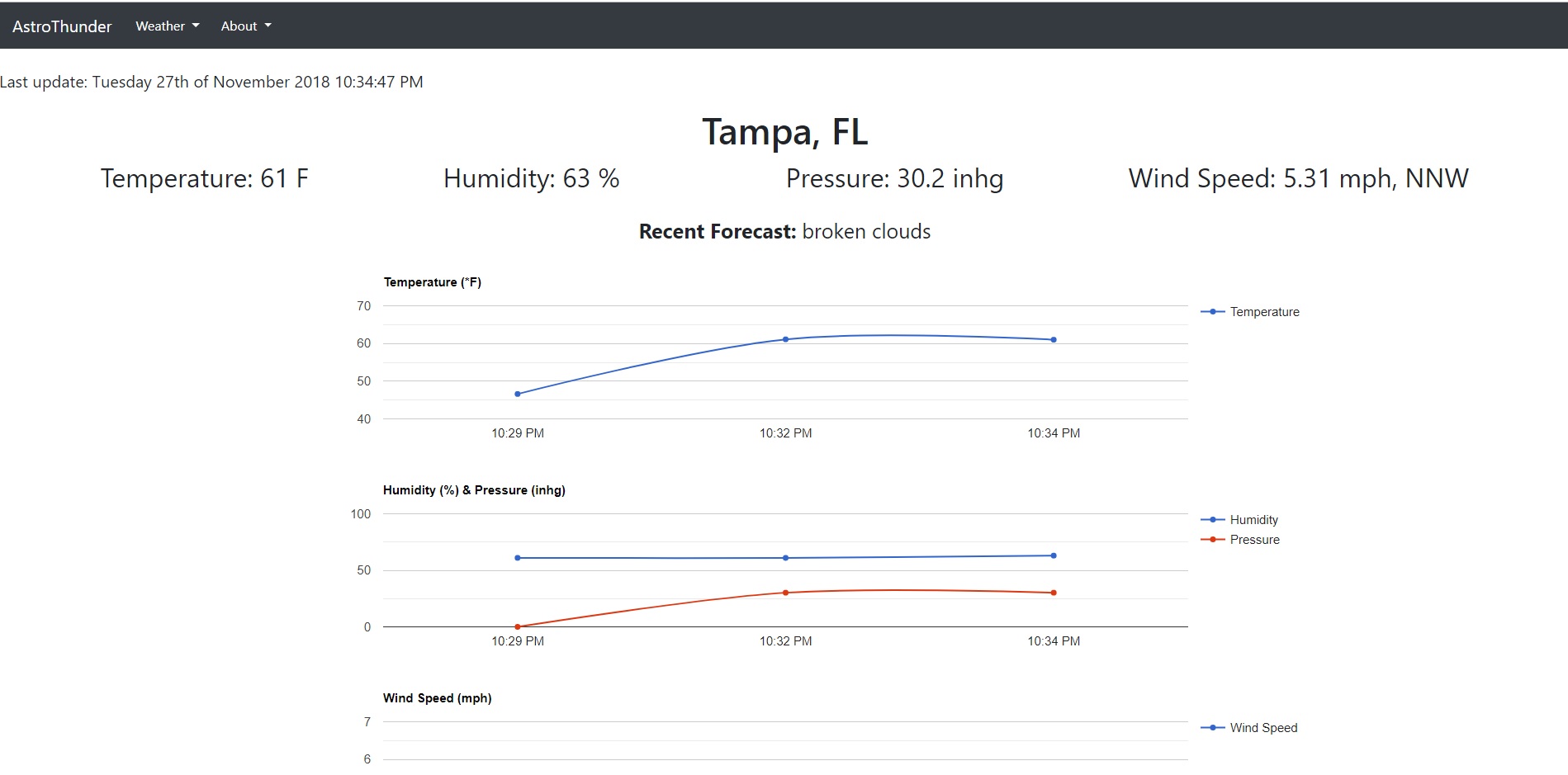
**Software**

The Raspberry Pi was given a base operating system image of Raspbian stretch with desktop and recommended software. Once the OS was installed the Pi was set up as a LAMP server (Linux, Apache, MySQL, PHP). The web app phpMyAdmin ([phpmyadmin.net](https://www.phpmyadmin.net)) is used to administer the MySQL databases.

The initial setup of the Raspberry Pi proved to have some difficulties, as we were attempting to follow a guide for the install and upgrade of most of the packages, some of which were out of order or not completing properly. Several times during the setup we had to backtrack to start the install process over fresh to get the most recent version that was needed by the python code we were currently creating. There were also additional packages that needed to be installed in order to get the full functionality out of our code and have all the packages available that were not listed in this guide that we were unaware of at the start, causing us to go back once again and modify the updates and installs as the python code was being written (**Citation of guide**).

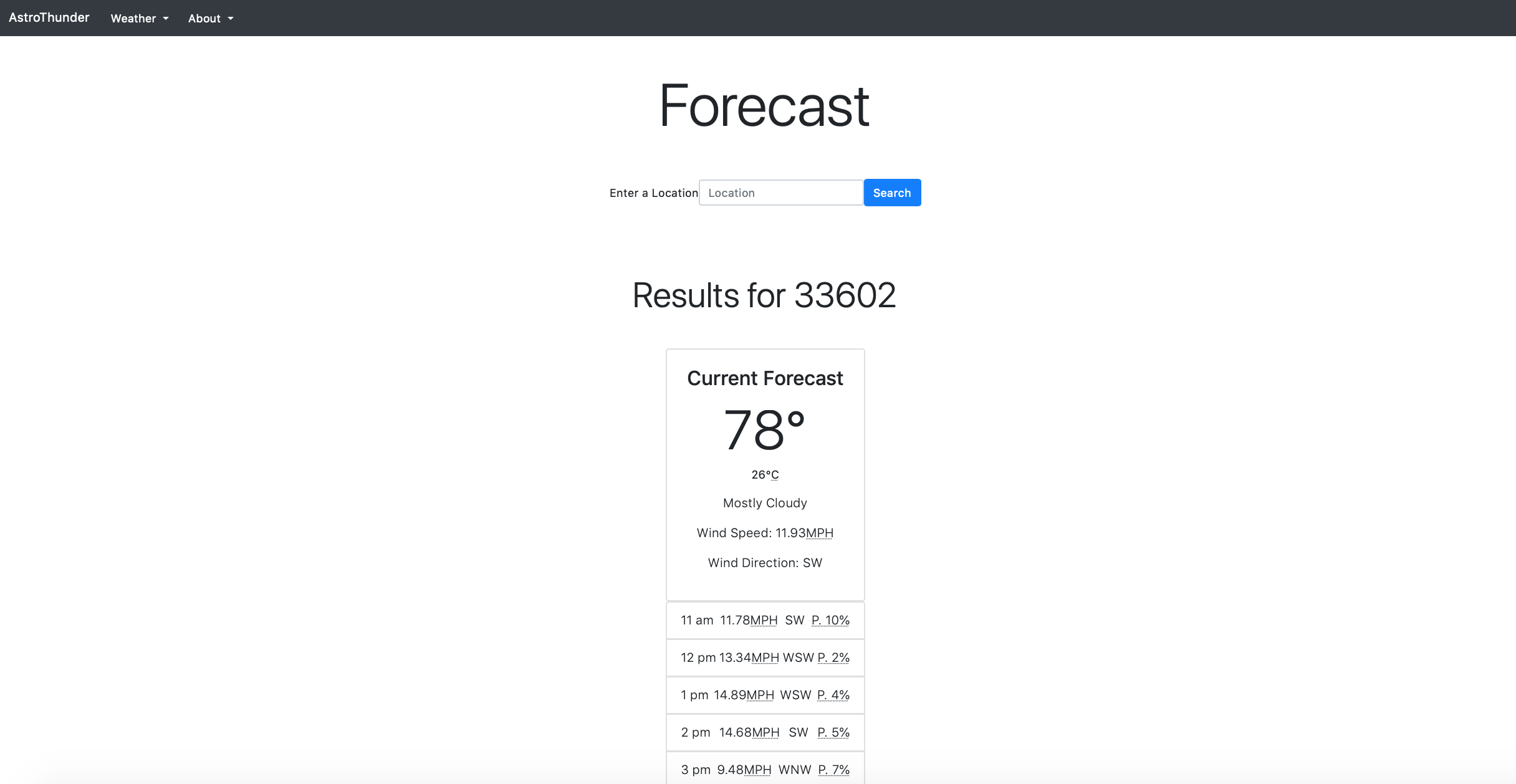
To implement the Sense HAT’s features, the code to gather the weather information for temperature, CPU temp, pressure, and humidity are all done in the same python code. This same code also sends the data to a text file and database. The text file is used to ensure the code is properly working and also gives a backup and reference should anything go wrong in future charting or loss of data. The code allows the Raspberry Pi and Sense HAT to sleep for 5 minutes before making another reading and sending it out again.

The database used for the project contained all data for time, temperature, pressure, wind speed, wind direction, and what the weather appears like in the area ( ie. overcast clouds, sunny, etc.) with the latter 3 being pulled from the Open Weather Map API.

In order to chart the data, HTML code was written to include Google Chart features and PHP code that would link the current database made from the python code to the charts and additional data ([developers.google.com](https://developers.google.com/maps/documentation/geocoding/start)). Within these sections of PHP, SQL queries were used to only pull the data that was needed for each individual chart, fetching it and setting it to the rows value. An auto-refresher is implemented to reload the page every 5 minutes if the user is idle on the page to give them the most recent and accurate data.****

In the case of the Raspberry Pi shutting down due to power issues, when the Pi powers back up it runs a batch code to ensure all required programs start back up so it may resume functionality and the database and site are able to update once again. The python code reboots using the built in crontab ([raspberrypi.org](https://www.raspberrypi.org/documentation/linux/usage/cron.md)).

Within the HTML for the site, users have access to reach the charts and additional information. One challenge faced with the HTML code was combining code from two different people in the group and both using different versions of bootstrap. Errors were thrown and formatting was off until solely bootstrap version 4.1.3 was implemented for both the menu bars, pictures, and forecast formatting. All data within the forecast menu item is used by integrating the site with the Dark Sky’s Weather API ([darksky.net](https://darksky.net/dev)) to show the data over the next 8 days with an expansion on data over the next 12 hours and uses Google’s Geocoding API ([developers.google.com](https://developers.google.com/maps/documentation/geocoding/start)) in order to allow the user to input an address, city, or ZIP code in the search bar to retrieve the weather forecast. For the added forecast functionality a tutorial is followed ([CodeTime.io](https://codetime.io/)). The website main page, forecast page, contact page, and the testing doc page are all hosted on USF’s FTP server, with the weather station page utilizing no-ip.

****

**Citations**

“Raspberry Pi Weather Station - Part 1.” *BC Robotics*, [www.bc-robotics.com/tutorials/raspberry-pi-weather-station-part-1/](http://www.bc-robotics.com/tutorials/raspberry-pi-weather-station-part-1/).

*Dark Sky - Saturday, Sep 22nd, 2018*, <https://darksky.net/dev>.

*Google, Geocoding,* <https://developers.google.com/maps/documentation/geocoding/start>.

*Google, Charts,* https://developers.google.com/chart/.

“Scheduling Tasks with Cron.” *Rotate Display 90º? - Raspberry Pi Forums*, [www.raspberrypi.org/documentation/linux/usage/cron.md](https://www.raspberrypi.org/documentation/linux/usage/cron.md).

*CodeTime, Working with PHP,* [*https://codetime.io/series/using-the-darksky-api-with-php*](https://codetime.io/series/using-the-darksky-api-with-php)

contributors, phpMyAdmin. “PhpMyAdmin.” *PhpMyAdmin*, [www.phpmyadmin.net/](http://www.phpmyadmin.net/).