Solar Capstone Build Instructions

2.5 Build Instructions

2.5.1 Introduction

The software system project we decided to work on this semester involves four solar panels located on top of the L building of Humber College North Campus. The purpose of this project was to interact and store the gathered data from all four solar panels into a database. Then, with the retrieved information we intend to display it on our mobile and web application. The data we will be retrieving from the solar panels consist of the power, daily and total yield energies. These four solar panels are manufactured from three different companies. For example, the PV1 and PV4 hardware was manufactured by Sunny Webbox, PV2 was manufactured by Envoy Communications Gateway and PV3 manufactured by Outback. The hardware for our project was already provided and installed for us so our main focus was to create the software aspect. At the end of this project we want to be able to retrieve the pushed data on our firebase. This data will be displayed onto our mobile and web applications for the Humber community and users to observe how solar panels work, and how much solar energy has been collected and expended.

2.5.2 System Diagram

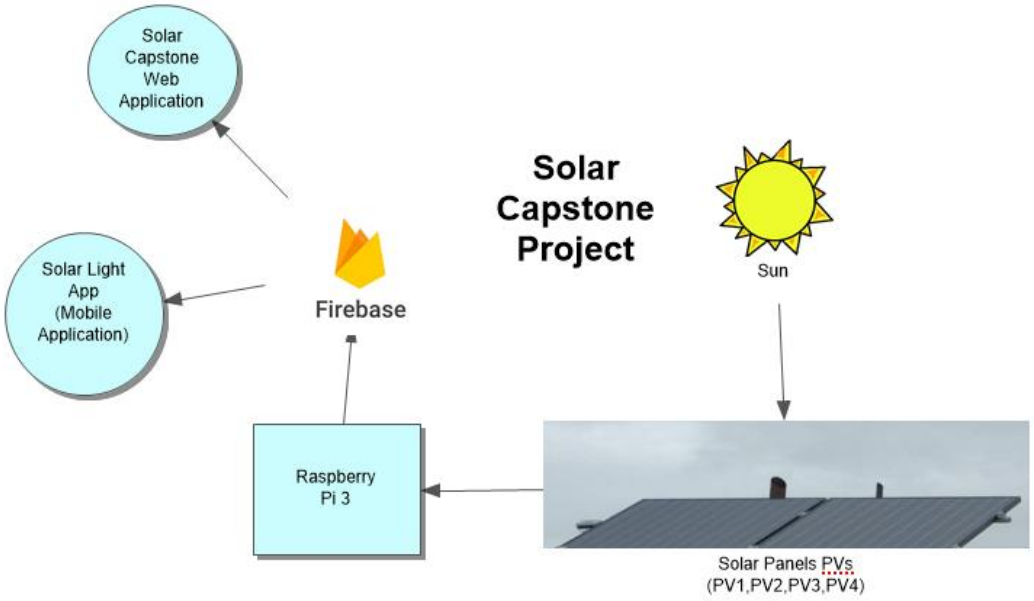

Image 2.5.2a: Solar Capstone Project System Diagram

2.5.3 Bill of Materials/Budget

The materials needed to replicate the Solar Capstone Project are solar panels and a Raspberry Pi. The solar panels were provided by the school and we bought the Raspberry Pi from Amazon. The solar panels were manufactured by the companies Envoy Communications Gateway, Sunny Webbox and Outback Power.

1. Raspberry Pi 3 (CanaKit Starter Kit): CAD $99.99

   Link: https://www.amazon.ca/CanaKit-Raspberry-Complete-Starter-Kit/dp/B01CCF6V3A/

2. Solar Panel PVs (Provided):

PV1 and PV4: Envoy Communications

PV2: Envoy Communications Gateway

PV3: Outback Power

2.5.4 Time Commitment

This project can be completed with a group of three people or individually. However, if completed in a group the tasks can be divided and completed quickly compared to individually. This project can be completed in the span of 13 weeks where each task is divided weekly. For example, one week we will write python code to retrieve specific data and another week we can write code to push the retrieved data into a database. At the end of the project we will present our completed application which will be able to successfully retrieve pushed data.

2.5.5 Power up

If you are booting up the Raspberry Pi for the first time, insert the micro SD card to the SD card reader from the Raspberry Pi package and make sure that the correct operating system is pre-installed. If it is not, the OS can be downloaded from this link: https://www.raspberrypi.org/downloads/noobs/. Once the OS has been installed on the Raspberry Pi, insert the micro SD card into the Raspberry Pi and power it. After doing this, the Raspberry Pi should then boot up. Before going further, users will be required to set up the operating system and configure their Raspberry Pi settings and Wi-Fi.

2.5.6 Software Assembly

The data came from four different sources and so, we needed to pick a database which would store the data under one platform. As a result, we decided to use Firebase as it's free to use and can be easily accessed. Additionally, we have experience with Firebase

from CENG319: Software Projects. We also had to decide the platform we wanted to use to create the mobile and web application. We choose Android Studio to create our mobile application and Notepad++ to create the web application. The web application is created using HTML and JavaScript which will be locally hosted on the PC.

Furthermore, a Python script is required to retrieve and push data into our Firebase. The python code we created will retrieve the solar panel data from the solar panel websites hosted on Humber's network and store the data on Firebase. The code can be downloaded here: https://minhaskamal.github.io/DownGit/#/home?url=https:%2F%2Fgithub.com%2FRaphaelNajera%2FSunlight_Sensor%2Ftree%2Fmaster%2Fdocumentation%2FCENG355%20Solar%20Capstone%2Ffirmware

The steps to run the Solar Capstone python code on the raspberry pi:

1. Power up the raspberry pi and set up VNC server on the raspberry pi
2. On the PC, download and run VNC viewer. With VNC viewer you can remotely connect to the raspberry pi
3. On the raspberry pi before running the code, you have to set the default python to python 3 and install the following modules which is required to run the code.

   On the terminal type the following code to set the default python version to python 3.5.3.

   nano ~/.bashrc

When you open ~/.bashrc, add new alias to change the default python executable.

```
alias python='/usr/bin/python3'
```

After you added the new alias, save the file, and enter this command.

```
. ~/.bashrc
```

Next enter the following command to install the modules.

pip install --upgrade setuptools

sudo apt-get install libxml2-dev libxslt-dev python-dev

sudo apt-get install python3-lxml python-lxml

pip install beautifulsoup4

pip install pyrebase

The following module are required because with beaultifulsoup 4, it has the function to allow the code to read the html and retrieve the data from the PV's website. The pyrebase allow us to push the data that the code retrieved from the website to the firebase.

4. Once you got the module installed, you can now run the code. The code will loop every 30 minutes.

```
python Solar_Capstone_PV_v4.py
```

2.5.7 Database

Create a Firebase account using Google Mail to access exclusively to your database. Then we had to create the tables for each solar panel PV and test the code we wrote with placeholder values. Next, we wrote the python code that is going to be used to get

and retrieve the data from firebase. This python code will execute every 5 minutes to fetch the latest results and push it into firebase. We also need to retrieve the epoch time to be able to delete this data after 30 days. Since we have a limited amount of storage this is the current solution we have come up with. The epoch time will not be displayed to the user it is for the programmer's use.

2.5.8 Mobile Application

Android Studio was the software we decided to use to make the counterpart of the mobile application. Our app connects to Google's Firebase where it will retrieve existing live data ran 24/7 on an external device running a python script. Our application is called "Solar Light" because it our project revolves around collecting solar energy produced from the sun. Once the app is created we then setup the app to connect our firebase by clicking Tools, and then Firebase. On the firebase assistant we clicked on analytics and then log an analytics event. We followed the steps such as login to the firebase, so the app can connect to the firebase and then adding the code to read from the firebase. Then we create four different screens for each of the four PV's by creating a navigation drawer for the solar PV's. Each of the PV's will display which PV you are looking at, the current date, power, daily yield, and total yield. It will also show a log of the pervious data that it retrieved from the firebase. It will also show total daily yield each week and month. The app will include a main screen which will display more information about the solar panels and the importance of using solar energy. On the action bar, it will display the settings, link to the Humber website and about information.

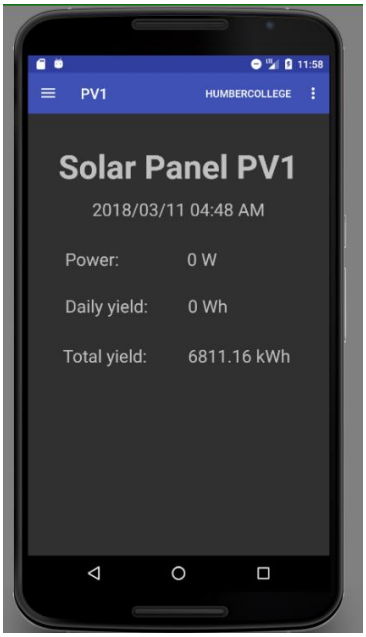

Image 2.5.8a: Screenshot of PV1 on Solar Light app

Link to the GitHub Solar Light application:

https://github.com/RaphaelNajera/Solar-Capstone-App

2.5.9 Web Application

The web interface was simply created using Notepad++ where one HTML document linked to four separate HTML forms. The HTML linked to four different frames and consisted of different panes that displayed four different sections. It is chronologically ordered so that users have a good idea of which solar panel is which. In each of these frames it will display the current PV panel and latest data retrieved from the last query. In other words, it would be the latest data retrieved from the database and placed under the <table> tag. Additionally, it will retrieve an array of data which is simply the history of all the fields. As such this allows the user to observe how much data has been collected every 30 minutes or so. This HTML can be hosted locally or through a domain service such as GitHub and will be able to be open on any browsing platform.

Link to the GitHub Solar Capstone Web application:

https://github.com/j-liang/solarcapstone_web

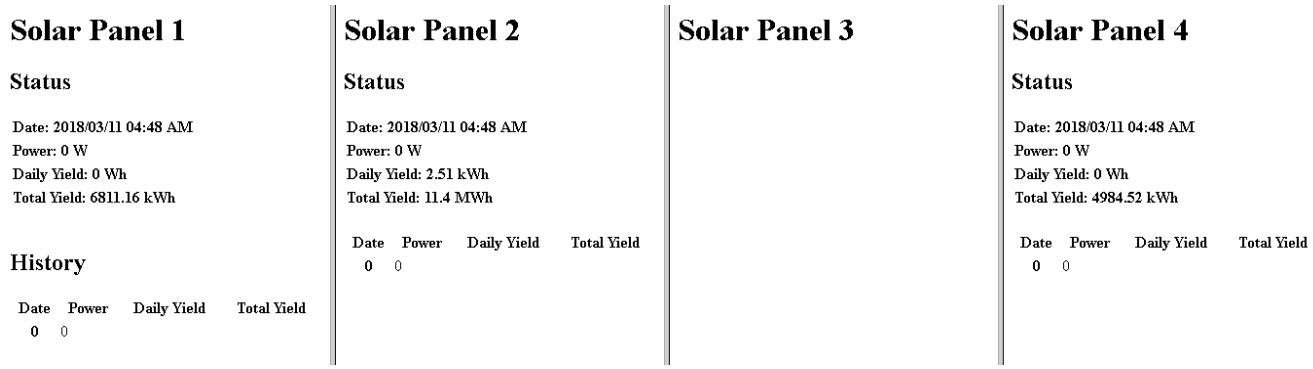

Image 2.5.9a: Screenshot of the Web application

2.5.10 Unit Testing

We did a lot of debugging and testing in order to check if specific tags were found with certain data. In order to achieve this, we wrote a python script to complete this task then we defined functions to push data into a pre-made firebase. However, prior to running the script we have to ensure that the raspberry pi is set up with installed python functions in the terminal which is explained under software assembly. Once the data has been stored it can be retrieved and be listed into our mobile and web applications. The web application can be ran on any Internet browser and it will retrieve the required information.

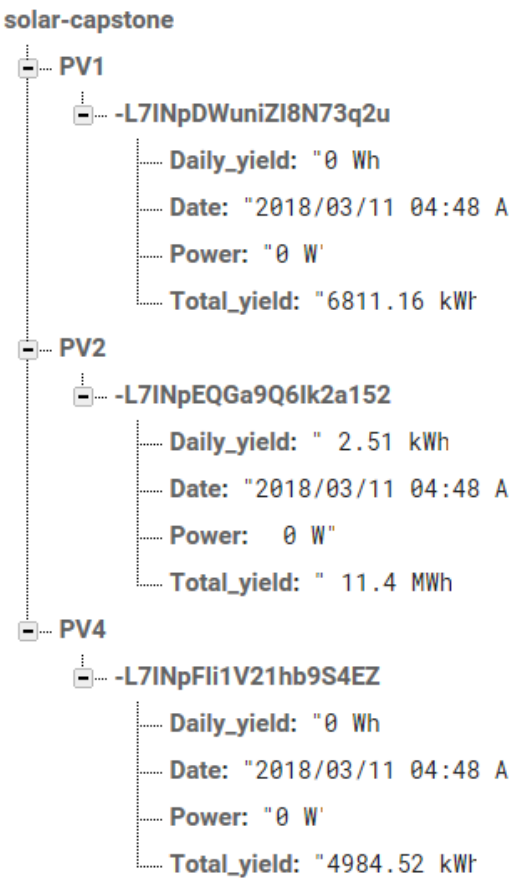

Image 2.5.10a: Testing data on the firebase

2.5.11 Production Testing

Once we successfully ran the python code on the raspberry pi, it will display the name of the solar panel PV's, epoch time, date, power, daily yield and total yield information from each of the solar panel. Our python code will be running 24/7 which the code will keep looping so that as it progresses the status will change. This will allow our mobile application and web application to display date, power, daily yield, and total yield from each of the solar panel with the current and updated status.

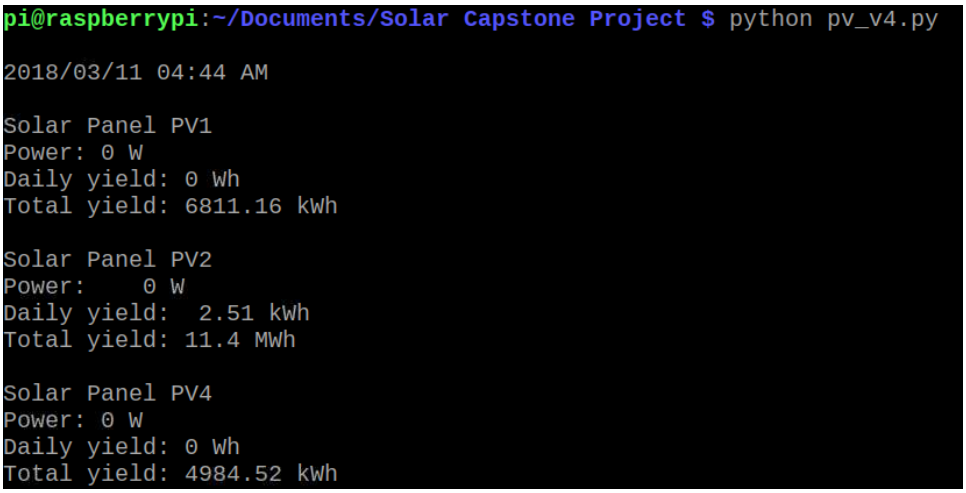

Image 2.5.11a: Screenshot of the solar capstone code running