Cloud Tracking V2
User Guide

Sponsor
Orlando Utilities Commission (OUC)
Justin Kramer
Rubin York

Project Supervisor
Dr. Mark Steiner

Design Team
William Askew
Clay DiGiorgio
Lars Gustaf Jiborn
Mohab Kellini
Nicholas Moser-Mancewicz
Timothy O'Brien
Gregory Ratz
Justin Zabel

Use

Installation

Jetson Flashing
Jetson Hardware Setup
Camera Connection
Install Requirements
Startup Scripts / Startup Applications

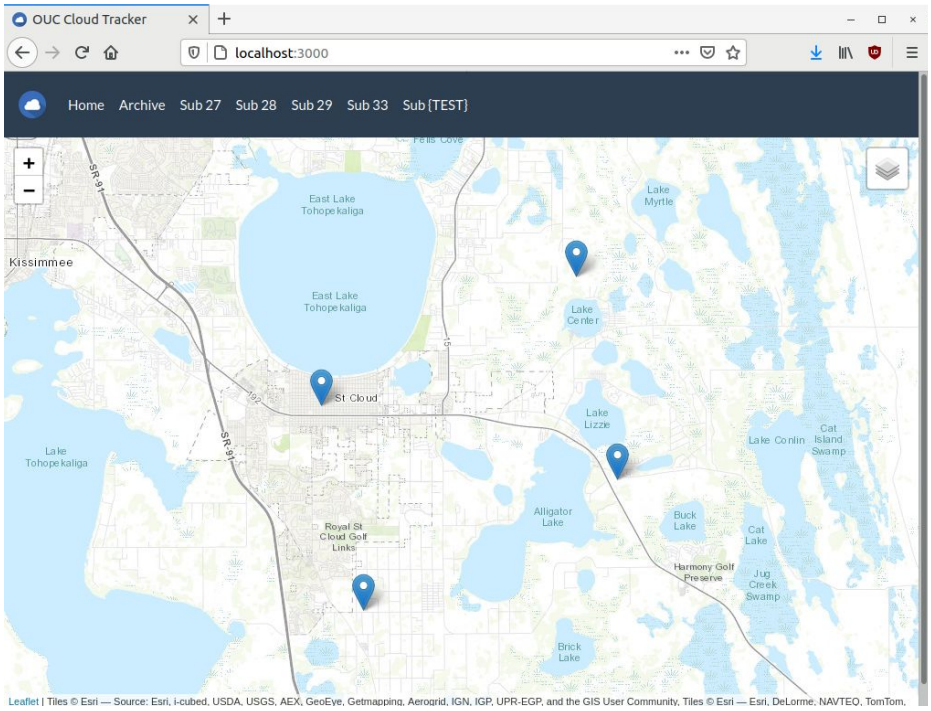
Set up website on custom server
MongoDB on custom server

Table of Contents

The entries in this table of contents are clickable! Give it a try!

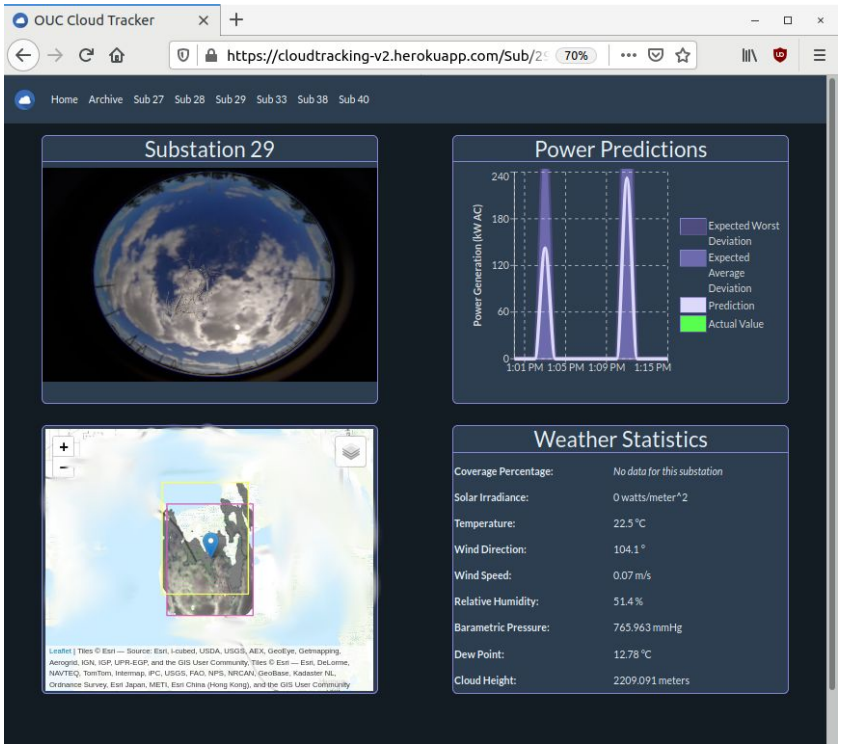
1 - Use

1.1 - Home Page



- The homepage for the website is a map of all substations in the network.
- Click on a marker to visit the homepage for its substation.
- You can also navigate to the station homepages, the archive page, or return to the home page from the navigation bar at the top.

1.2 - Substation Home Pages



Note: the map has been blurred to hide the location of this site.

1.2.1 - Livestream

The top left card of the page

- Displays a live stream of the camera installed onsite.

1.2.2 - Cloud Shadow Map

The bottom left card of the page

- Click the “stacked paper” icon in the top right to toggle different layers of the display
    - for example, you can display only the clouds’ shadows, without the clouds themselves
- Displays a projection of the clouds identified in the livestream onto a map of the site’s location.
- Displays the expected shadows of those clouds as well.
- The green arrows overlaid on the clouds indicate the speed and direction each chunk of cloud is moving in.
- Note: the map takes a few seconds to fully load. If there are no clouds on the map, please leave the page open for a little bit.

1.2.3 - Power Predictions Graph

The top right card of the image

- To hide/show parts of the graph, click the colored boxes in the legend.
- As you leave the page open, Estimated Actual Value readings will accumulate, forming a green line.
    - This will squish the predictions on the right hand side of the graph.
    - If you click the green box, this will be disabled and the predictions will occupy the full graph until the green box is clicked again
- This graph will update once per minute. You will always be able to see the most recent (most accurate) predictions.
- This graph displays 4 separate categories of information (all in kilowats AC)
    - The predicted solar panel power output (white)
    - The average error range for these predictions (light blue)
    - The worst expected error range (dark blue)
    - Estimated actual solar panel output since opening the page (green)

1.2.4 - Weather Stats

The bottom right card of the image

- This card displays the latest weather info for a given substation
- Once per minute, this display will refresh with the latest information

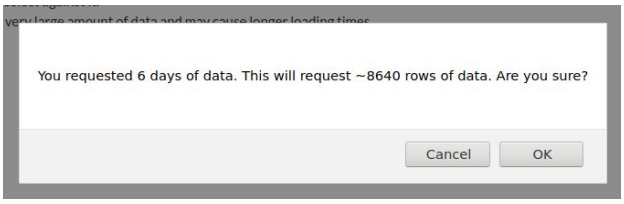
1.3 - Archive Page

1.3.1 - Overview

This page is for viewing stored weather data.

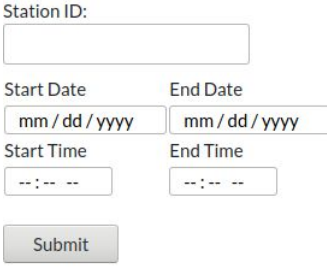
When using this page, you can search for weather data with various constraints.

If your constraints will likely result in a large amount of data being requested, you will be presented with a warning and a prompt to continue or cancel.



1.3.2 - Input Fields

Any and all input fields may be left blank if you want your search to ignore them. For example, if you want all data



from all stations from a particular day, set the Start Date and End Date appropriately, but leave Station ID and Start Time / End Time blank.

Station ID

Put the exact id of the station you want data for here. This is case sensitive, and leaving a space at the end of the ID will also mess it up.

Entering an invalid station ID will cause the search to return no results.

Start Date / End Date

Filling in the Start Date and End Date fields will request only data collected on the day of Start Date, up to the end of End Date.

Leaving End Date blank will cause the server to assume you meant to set it to the current date. As such, filling in End Date is optional.

Note: if Start Date is after the current date or the entered End Date, the server will ignore what you entered in for those fields.

Start Time / End Time

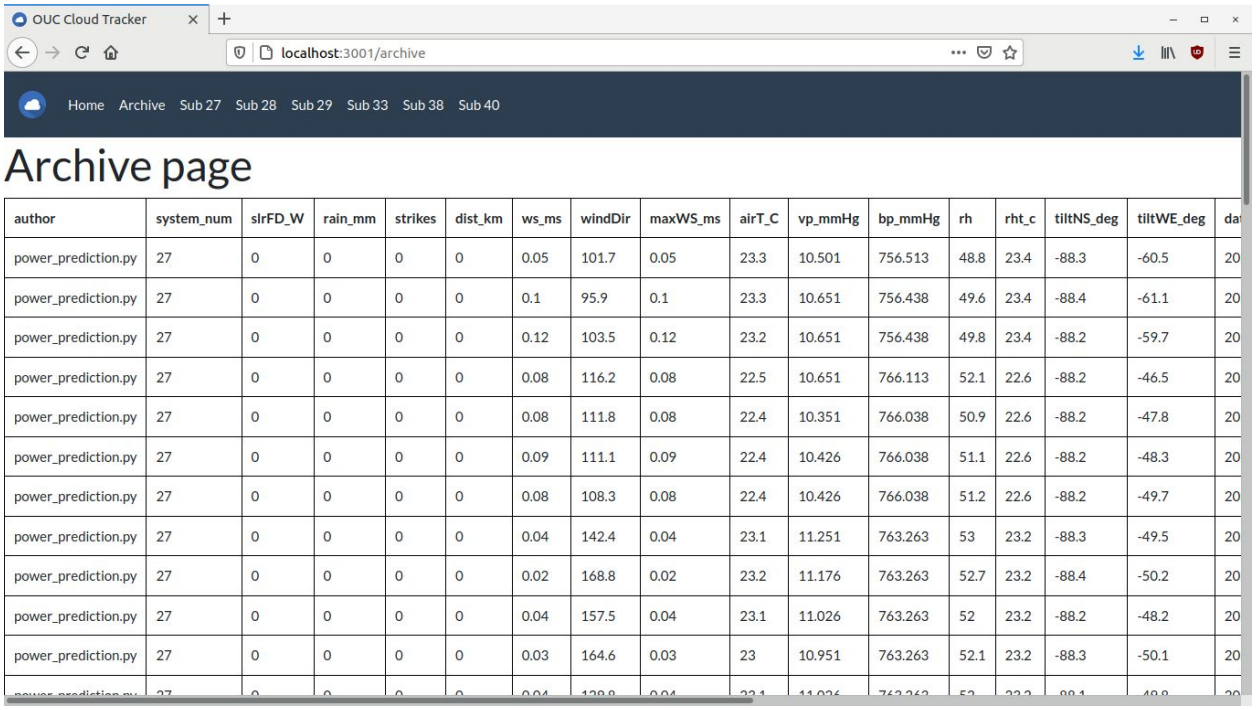
Filling in both of these fields will search only for entries collected at a particular time of day.

For example, if you set Start Time to 1pm and End Time to 3pm, the server will search for all data collected between 1pm and 3pm on the day it was collected. I.e, it will return data from all days, restricted to what was collected between your specified times.

Note: if Start Time is after End Time, the server will ignore what you entered for those fields.

1.3.3 - Viewing Results

You may have to scroll sideways. Each row is one entry recorded, and each column contains a particular datapoint for that entry.



author	system_num	slrFD_W	rain_mm	strikes	dist_km	ws_ms	windDir	maxWS_ms	airT_C	vp_mmHg	bp_mmHg	rh	rht_c	tiltNS_deg	tiltWE_deg	da
power_prediction.py	27	0	0	0	0	0.05	101.7	0.05	23.3	10.501	756.513	48.8	23.4	-88.3	-60.5	20
power_prediction.py	27	0	0	0	0	0.1	95.9	0.1	23.3	10.651	756.438	49.6	23.4	-88.4	-61.1	20
power_prediction.py	27	0	0	0	0	0.12	103.5	0.12	23.2	10.651	756.438	49.8	23.4	-88.2	-59.7	20
power_prediction.py	27	0	0	0	0	0.08	116.2	0.08	22.5	10.651	766.113	52.1	22.6	-88.2	-46.5	20
power_prediction.py	27	0	0	0	0	0.08	111.8	0.08	22.4	10.351	766.038	50.9	22.6	-88.2	-47.8	20
power_prediction.py	27	0	0	0	0	0.09	111.1	0.09	22.4	10.426	766.038	51.1	22.6	-88.2	-48.3	20
power_prediction.py	27	0	0	0	0	0.08	108.3	0.08	22.4	10.426	766.038	51.2	22.6	-88.2	-49.7	20
power_prediction.py	27	0	0	0	0	0.04	142.4	0.04	23.1	11.251	763.263	53	23.2	-88.3	-49.5	20
power_prediction.py	27	0	0	0	0	0.02	168.8	0.02	23.2	11.176	763.263	52.7	23.2	-88.4	-50.2	20
power_prediction.py	27	0	0	0	0	0.04	157.5	0.04	23.1	11.026	763.263	52	23.2	-88.2	-48.2	20
power_prediction.py	27	0	0	0	0	0.03	164.6	0.03	23	10.951	763.263	52.1	23.2	-88.3	-50.1	20

2 - Central Server Setup

2.1 - MongoDB Setup

Note: in this section, anything inside angle brackets <LIKE THIS> should be replaced with an appropriate value. For example, name: “<YOUR NAME HERE>” should look like this name: “Clay” for example.

Note: pretty much everything here is case-sensitive

1. install mongodb on your server
    a. Follow the standard install guides for mongodb. Our server ran Redhat Linux, and mongodb is part of the standard Redhat Repo
2. run the mongodb server
    a. Set up your server to run your database with systemcl
3. create an admin user in the admin table

```
mongo --host <YOUR SERVER’S PUBLIC IP ADDRESS>
use admin
db.createUser(
      {
            user: “<NAME YOUR ADMIN USER>”,
            pwd: “<ADMIN PASSWORD>”,
            roles: [ { role: “userAdminAnyDatabase”, db: “admin” } ]
      }
)
```

4. create a regular user in the table, for the website and jetsons to use
   while still in the mongo terminal and using admin,

```
db.createUser(
      {
            user: “cloud-tracking”,
            pwd: “<NEW PASSWORD>”,
            roles: [ { role: “readWrite”, db: “CloudTrackingData” }
            ]
      }
)
```

5. Test to make sure that your users were successfully created
   exit out of the mongo terminal with Ctrl+C

run the below commands

```
mongo --host <YOUR SERVER'S PUBLIC IP ADDRESS>
use admin
db.getUsers()
```

exit out of the mongo terminal with Ctrl+C

6. set the mongodb to run on your computer's ip address rather than localhost
    a. Option 1

```
mongod --bind_ip localhost,<YOUR SERVER'S HOST NAME>
```

    b. Option 2

```
mongod --bind_ip localhost,<YOUR SERVER'S PUBLIC IP ADDRESS>
```

7. restart the mongodb server
    a. This can be done systemcl, similar to how you set it up in step 2

2.2 - Website Setup

2.2.1 - Initial Setup

1. (optional) set up a hostname for your server
2. Download our code
    a. run the following command

```
git clone https://github.com/Group18CloudPrediction/oucseniordesignv2.git
```

3. Set up the credentials file

```
cd oucseniordesignv2
cd credentials
```

    a. Create a new file with vim, nano, or some other command line text editor
    b. Name the file mongodbCredentials.js
    c. In the file, paste the following line and fill in the appropriate values (this is CaSe SeNSItivE)

```
module.exports = "mongodb://cloud-tracking:<YOUR PASSWORD>@<YOUR SERVER'S
IP
```

```
ADDRESS>:27017/CloudTrackingData?compressors=disabled&gssapiServiceName=mon
godb&retryWrites=true&authSource=admin&w=1";
```

Note: if the above throws an error when you try to start the server, try this:

```
module.exports = "mongodb://cloud-tracking:<YOUR PASSWORD>@<YOUR SERVER'S
IPADDRESS>:27017/CloudTrackingData?gssapiServiceName=mongodb&retryWrites=tr
ue&authSource=admin&w=1";
```

d. Save the file

4. Setting api root address

```
cd ..
```

you should be in the folder oucseniordesignv2 now

```
cd Front_End/src/components/apiCallers
```

a. edit the file _apiRootAddress.js or create it if it doesn't already exist
b. change the line

```
var url = "https://cloudtracking-v2.herokuapp.com";
```

to be

```
var url = "http://<YOUR SERVER'S HOST NAME>";
```

c. If you see a line that says "var heroku = true", it is safe to ignore
d. Note:if you're experiencing issues, try changing this line to

```
var url = "http://<YOUR SERVER'S IP>";
```

5. Setting the port number in server.js

in oucseniordesignv2/server.js, find the line near the top that says

```
var port = process.env.PORT || 3000;
```

replace it with

```
var port = 80;
```

6. allow http through firewall

```
sudo firewall-cmd --add-service http --permanent
sudo firewall-cmd --reload
```

This step and the one above set the webserver to run on http, and allows http requests to be received and sent by the webserver

7. Open the firewall to allow connections to mongodb (this allows the jetsons to communicate with the db)

```
sudo firewall-cmd --add-port=27017/tcp --permanent
sudo firewall-cmd --reload
```

8. set the livestream to go through ws instead of wss
    a. Go to the folder oucseniordesignv2/Front_End/src/components
    b. open SubstationLivestream.js
    c. Change the below line

```
const url_ = "wss://" + server + "/sub-" + this.props.stationID
```

    d. to

```
const url_ = "ws://" + server + "/sub-" + this.props.stationID
```

This step sets the livestream to run over http instead of https. We do this because the server is running on http, not https like heroku, our development server, was

9. run the below command in the oucseniordesignv2/ folder. You may need root privileges (sudo).

```
npm install dependencies
```

10. run the below command in the oucseniordesignv2/Front_End folder. You may need root privileges (sudo).

```
npm install dependencies
```

11. run the below command in the oucseniordesignv2/Front_End folder. You may need root privileges (sudo).

```
npm run build
```

12. Setting up ffmpeg -- this step may be unnecessary, try skipping it. if the live stream doesn't work, stop the web server and do this step
    a. run each command that the second post on this url prompts you to run
        i. https://snapcraft.io/install/ffmpeg/rhel
        ii. if the 3rd command doesn't work, then just skip it
13. Start the web server. See below.

14. Your website can now be visited on your local network by visiting http://<YOUR SERVER'S PUBLIC IP ADDRESS> or http://<YOUR SERVER'S HOST NAME> in your internet browser, such as Chrome or Firefox

2.2.2 - Starting the Web Server

1. run "npm start >> log.txt &" with root privileges ("sudo npm start &" if on linux)
    a. sudo is needed because 80 is a protected port
    b. you should see some text pop up that looks similar to the below:

```
clay@clay-Precision-3520:~/Documents/Class Stuff/senior design/Current
working site/oucseniordesignv2$ sudo npm start >> log.txt &
[1] 17534
clay@clay-Precision-3520:~/Documents/Class Stuff/senior design/Current
working site/oucseniordesignv2$
> oucseniordesignv2@1.0.0 start /home/clay/Documents/Class Stuff/senior
design/Current working site/oucseniordesignv2
> node server.js

App is listening on port 80
MongoDB Connected...
```

    c. Press the Enter / Return key
2. The web server is now running in the background and saving its output to the log file!

2.2.3 - Stopping the Web Server During the Same Terminal Session

1. Run the command `jobs`

```
jobs
```

2. If the server is running, you should see something similar to the below result

```
clay@clay-Precision-3520:~/Documents/Class Stuff/senior
design/Current working site/oucseniordesignv2$ jobs
[1]+  Running       npm start &
```

3. run the command `fg 1`

```
fg 1
```

    a. Note: if the number in the brackets wasn't 1, but was some other number, use that number instead
4. Hit Ctrl+C

2.2.4 - Stopping the Web Server After Having Logged Out

1. First we need to find the process number for the thread that runs the webserver. Run the following command to find that:

```
ps aux
```

You should see output similar to the below:

```
clay@clay-Precision-3520:~$ ps aux
USER         PID %CPU %MEM     VSZ   RSS TTY      STAT START   TIME COMMAND
root           1    0.0  0.0  226040  9632 ?        Ss   09:03   0:19 /sbin/init spla
root           2    0.0  0.0       0     0 ?        S    09:03   0:00 [kthreadd]
root           3    0.0  0.0       0     0 ?        I<   09:03   0:00 [rcu_gp]
...
clay        9820  3.6  0.2 829792 43904 pts/1    Sl   15:49   0:00 npm
clay        9831  0.0  0.0   4632   784 pts/1    S    15:49   0:00 sh -c node serv
clay        9832  6.2  0.3 942792 57688 pts/1    Sl   15:49   0:00 node server.js
...
clay        9843  0.0  0.0  39660  3704 pts/0    R+   15:49   0:00 ps aux
```

2. Find the entry that ends with `npm`
3. Look for its PID value, in this case it's `9820`
4. Execute the following command, replacing <PID number> with the number you found in step 3

```
kill <PID number>
```

3 - New System Installation

3.0 - Setting up the hardware

1. Gather necessary hardware components:
   Geovision Camera
   Polycase Enclosure
   Plywood Base
   Hardware Anchoring Fasteners (screws and zipties)
   Jetson Data Processor
   Power Supply
   18-Gauge Wiring
   Ethernet Cable
   HDMI Cable
   USB-to-ethernet Adapter
   Serial-to-USB Adapter
   Extension Cord
   Terminal Block
   Appropriate sized ring wire crimps

2. Prepare an appropriate sized hole through the wall of the enclosure to allow camera, extension cord and jetson power supply wires to pass through.
3. Prepare the plywood backboard by preparing holes for mounting the board to the corner screw inserts on the enclosure, holes for the zipties to mount the Jetson and holes for mounting the power supply with screws.
4. Prepare 18-Gauge wire by cutting an appropriate length of wire and stripping each end in order to connect ring wire crimps. In the case of the three systems made for this project heat shrink wire crimps were used to ensure a stable connection. It is recommended that a multimeter is used to check for continuity across all prepared wires.
5. Mount Jetson and power supply onto properly sized plywood backboard with respective fasteners. Each component should be mounted sufficiently far enough from each other and from the wall of the enclosure to allow for wire flow as well as properly oriented to make sure wires don’t bend excessively.
6. Strip the three pronged wire of the geovision camera to expose the purple (positive) and black (negative) power wires. Connect these wires to the terminal block. Connect 18-Gauge wire from the terminal block into one of the sets of positive/negative outputs of the power supply ensuring that the polarities match the polarities of the camera's wires on the terminal block.
7. Strip the female end of the extension cord exposing the ground, neutral and positive wires within. In the case of the extension cords used for this project, green was ground, white was neutral and black was positive. Attach wire crimps to each wire and connect each wire to its respective port on the power supply.

8. Connect all necessary auxiliary wiring to the Jetson, including ethernet wire from the camera to the usb-to-ethernet adapter, serial-to-usb for the datalogger, wifi antennas to the jetson and power supply wires to the jetson.
9. Ensure that all wiring is secure and is in proper order and mount the backboard to the enclosure using screw inserts at each corner of the enclosure.

3.1 - Adding a new system to the website

1. Stop the web server (see section 2.2.3)
2. Setting up for the livestream
    a. in ~/server.js, find the initChannels() function
    b. Add a line `createChannel("/sub-<STATION ID>");`
3. Setting up for the cloud-shadow map
    a. in ~/server.js, find the init() function
    b. In ~/api.js add in coverage and shadow functions for the substation, and export.
    c. In ~/map.js import the exported functions
    d. Find the block of code that has a bunch of lines similar to the below, and add the below lines to it, replacing <STATION ID>:

```
client.on('coverage<STATION ID>', (frame) => {
      console.log('coverage<STATION ID> received');
      client.broadcast.emit('coverage<STATION ID>',
      ""+ frame.toString("base64"))
})

client.on('shadow<STATION ID>', (frame) => {
      console.log('shadow<STATION ID> received');
      client.broadcast.emit('shadow<STATION ID>',
      ""+ frame.toString("base64"))
})
```

4. in ~/Front_End/src/components/Navigation.js, find the block of code with many lines similar to the below, and add the below line there

```
<Nav.Link as={NavLink} href="/Sub/{STATION ID}" to="/Sub/{STATION
ID}">{STATION ID}</Nav.Link>
```

5. In ~/Front_End/src/components/map.js, at the top, find the block of code similar to the below and add the below

```
const sub<STATION ID> = [<STATION LAT>, <STATION LONG>]
```

6. In ~/Front_End/src/components/map.js find the block similar to below and add the below

```
else if (this.props.stationID == <STATION ID>) {
      CENTER = sub<STATION ID>
}
```

7. In ~/Front_End/src/components/map.js, find the block of code similar to the below, and add the below block

```
      else if (this.props.stationID == '<STATION ID>') {
            console.log("This is sub <STATION ID>");
            subscribeToCoverage40((err, coverage_img) => {
                  // If already exists, update the coverage image

                  // If Coverage Overlay is available, recompute the bounds
                  // given new CBH
                  if (!(this.coverageOverlay === undefined)) {
                        this.coverageOverlay.setUrl(coverage_img);
                  }


                  console.log("cvg" + coverage_img);
            });

            subscribeToShadow40((err, shadow_img) => {
                  // If already exists, update the shadow image

                  // If Shadow Overlay is available, recompute the bounds
                  // given new CBH
                  if (!(this.shadowOverlay === undefined)) {
                        this.shadowOverlay.setUrl(shadow_img);
                  }
                  console.log("shdw" + shadow_img);
            });
      }
```

8. In ~/Front_End/src/components/combinedMap.js, add the below code

```
const sub<STATION ID> = [<STATION LAT>, <STATION LONG>]
```

9. In ~/Front_End/src/components/combinedMap.js, below code

```
var marker<STATION ID> = L.marker(sub<STATION ID>, {
      draggable: false,       // Make the icon dragable
      title: 'Sub <STATION ID>'
}).on('click', function(e) {window.location = baseURL+"/Sub/<STATION ID>"});
marker<STATION ID>.addTo(this.map)
```

10. in ~/Front_End/, run the below command

```
sudo npm run build
```

11. Start up the web server again. (see section 2.2.2)

3.2 - Setting up a new Jetson system

3.2.1 - Flashing

1. https://developer.download.nvidia.com/embedded/L4T/r27_Release_v1.0/Docs/Jetson_X2_Developer_Kit_User_Guide.pdf?eQEO652ctn-wTY3-2pmD0sqMStfXQ1T8Cy_f1LWcPnbSrDJq6ZdlruGCIwYDDAFeATfdGdUUC2H8kOvg1oz9rGZBZRnRr8fXmk34TLzd46E9WuUB89cAITIfPnhg_qwO6WOQ-kQSA0VyG1EKcbJYYtpmkCFxHIMHrBnChE-pCukHQ1jiaL18qQ

3.2.2 - Downloading the code

1. Download the master branch of the linked github repo. Extract the downloaded zip file to access the code. *Make sure to extract the code to the desktop.*
2. Update pip by running the following commands:

```
sudo apt-get install python3-pip
sudo pip3 install -U pip testresources setuptools==49.6.0
```

3. Next you need to install the required dependencies. Most dependencies can be installed by running the command pip3 install -r requirements.txt.
4. Some dependencies have to be installed separately because of their incompatibility with the Jetson OS. Run the following commands to install the remaining dependencies.

```
sudo apt-get install python3-opencv
pip3 install scipy
pip3 install matplotlib
pip3 install dnspython (add this to requirements.txt)
```

5. Finally, run the following commands to install tensorflow and its dependencies. These commands can be found in full here.

```
sudo apt-get update

sudo apt-get install libhdf5-serial-dev hdf5-tools libhdf5-dev
zlib1g-dev zip libjpeg8-dev liblapack-dev libblas-dev gfortran

sudo pip3 install -U numpy==1.16.1 future==0.18.2 mock==3.0.5
h5py==2.10.0 keras_preprocessing==1.1.1 keras_applications==1.0.8
gast==0.2.2 futures protobuf pybind11

sudo pip3 install --pre --extra-index-url
https://developer.download.nvidia.com/compute/redist/jp/v44
tensorflow
```

6. All required python modules should now be installed. If you attempt to run the scripts and receive a “module not found” error, it means that module is not installed. After installing the module, run the script again. It will continue to throw “module not found” errors until all modules are installed. If all else fails, repeating this process will ensure all modules are eventually installed.
7. Run the following command, it fixes a TLS error which is caused by and issue in the Jetson OS:

```
export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
```

3.2.3 - Setting the code to run on startup

1. Allowing the appropriate files to run
    a. These are the relevant files
        i. power_verification.py
        ii. app.py
        iii. run_power.sh
        iv. run_app.sh
        v. run_stream_ffmpeg.sh
    b. Follow these steps for each of the files listed above

        i. right click on the file and click on properties
        ii. Click on the permissions tab
        iii. check “Allow this file to run as a program”
2. Open the Session and startup window
    a. In the top left, click the blue circle with the funny white logo
    b. hover over “Accessories”
    c. click “Application Finder”
    d. type “startup”
    e. Click “Session and Startup” in the right hand pane
    f. All further instructions for 3.2.3 will be in the “Session and Startup” window
3. database interfacing
    a. Click the add button
    b. Name it “data_collection_and_predictions” (legacy name: “power_verification”)
    c. Set the command to home/Desktop/JetsonCode-maste-run/JetsonCode-master/startup_scripts/run_power.sh
4. main app
    a. Click the add button
    b. name it app.py
    c. Set the command to home/Desktop/JetsonCode-maste-run/JetsonCode-master/startup_scripts/run_app.sh
5. livestream
    a. Click the add button
    b. name it “cloud_livestream” (legacy name: “cloud_tracking”)
    c. Set the command to home/Desktop/JetsonCode-maste-run/JetsonCode-master/startup_scripts/run_stream_ffmpeg.sh

3.2.4 - Connecting the datalogger

1. Connect the datalogger to the Jetson’s mini-usb port by using a mini-usb to usb converter. Connect the serial end of the cable to the datalogger and the usb end to the usb converter.
2. To determine which USB port the datalogger is connected to, run the command: dmesg | grep tty. Unplug the datalogger from the Jetson and plug it back in. You should be able to find which USB port it is connected to near the bottom of the output. It will look something like /dev/ttyUSB*, where * is a number. This might be useful later.
3. Add the current user to the dialout group. If the username is cloud-tracking, type the following command in the terminal: sudo adduser cloud-tracking dialout. Restart the Jetson. Alternatively, you can run the command: sudo chmod 666 dev/ttyUSB0. If you use this command, you will have to rerun this command every time the datalogger is unplugged from the Jetson.

4. If the connected USB port is not dev/ttyUSB0, you will need to modify the code with the correct USB port. Search for all instances of “/dev/tty” in power_verification.py and datalogger_runner.py. Replace each instance with the updated USB port.

3.2.5 – Possible datalogger issues

1. If the datalogger won’t connect, there could be an issue with the pyserial/serial module. Run the following commands:

```
sudo pip3 uninstall pyserial
sudo pip3 uninstall serial
sudo pip3 install pyserial
```

3.2.6 - Connecting the livestream

a. Be sure to install all requirements including ffmpeg as described in the user setup guide.
b. To select an allow the FFMPEG to stream IP address for the camera , go to http://192.168.0.10/ and create a username and password[BR1] . A software update through Geovision made the default p/w and username obsolete. If ever the p/w or username are forgotten a manual reset of the camera can be used to re-create log-in credentials. This can be found in more detail in section 6.3 of the GeEovision user manual. To download the manual go to Geovision Downloads
c. All camera logins are kept by the OUC.
d. Go into Events and Alerts, click RTSP Check Disable Authentication, this allows the ffmpeg livestream to be streamed to our application

e. To stream from the camera to the website log into the camera, and send the livestream to an active substation on the website.
f. Bash this file and replace the 1 at the end of line 11 with sub-{stationID}

EX: .../sub-27

g. https://github.com/Group18CloudPrediction/JetsonCode/blob/master/startup_scripts/run_stream_ffmpeg.sh
    i. Opens an ffmpeg stream, streaming from the camera input to the website's backend
2. Also, you will need to set the server ip on this same line too. Use the IP for your central webserver
   EX, the end of that line should look like this: `https://<WEB SERVER IP>/cloudtrackinglivestream/sub-27`

3.2.6 - Connecting to the database

1. find the creds.py file
2. change the username and the password
3. change the cluster url to `@<YOUR SERVER IP>:27017/CloudTrackingData?compressors=disabled&gssapiServiceName=mongodb&retryWrites=true&authSource=admin&w=1`
    a. If this causes errors, try this instead `@<YOUR SERVER IP>:27017/CloudTrackingData?compressors=disabled&gssapiServiceName=mongodb&retryWrites=true&authSource=admin&w=1`

3.2.7 - Possible power prediction issues

Check out the comments in predict.py and the rest of the documentation for it

3.2.8 - Setting up the config file

set the send to database to true?

1. create config/creds.py
    a. it should look like this

```
base_url = "mongodb://"
separator = ":"
cluster_url = "@<YOUR WEB SERVER IP>:27017/CloudTrackingData?gssapiServiceName=mongodb&retryWrites=true&authSource=admin&w=1"
username = cloud-tracking
password = <YOUR PASSWORD>
```

2. in config/substation_info.py
    a. set id
    b. set LAT and LONG
3. in config/cloud_tracking_config.py
    a. change the below line

```
URL_APP_SERVER = ...
```

b. to this

```
URL_APP_SERVER = 'http://<YOUR SERVER IP>'
```

3.2.9 - Starting the Jetson

Restart the Jetson with this command:

```
sudo reboot
```

The startup scripts should automatically run all the code installed.

Note: when a jetson reboots, it will take 15 minutes to update the power prediction line graph on its web page.