Aurora Solar Coding Challenge

Michael Quint
11/29/17

Part 1 Bonus:

a) The total amount of energy produced by the entire solar installation is 63,819.1 kWh. Using my setup, getting this information was simple. I typed in the device name "Plant" and channel name "GriEgyTot" and clicked fetch data point. I could have also clicked fetch component list first, then clicked on "Plant" and then "GriEgyTot" to get the same data.
b) The total amount of energy produced by each inverter is also easily gathered with my web application. Simply type in E-Total to the channel input and leave the device input empty. This will automatically search for that channel information throughout all devices.
Device SI5048EH:1260016316 produced 32,283.0 kWh and device SI5048EH:1260016365 produced 31,536.1 kWh. You can gain further insight by searching for E-Total-In and deducing for each inverter, it is about 40% higher than Energy Total.
c) Finally, to get the average charge for each battery in the system I ran Fetch Data Point again with an empty device field and put "BatSoc" as the channel. This returned the percent charge of both batteries in the system which average out to 65.9% charge. If I were building this application further, I would create a button to automatically get the average for the user.

Part 2:

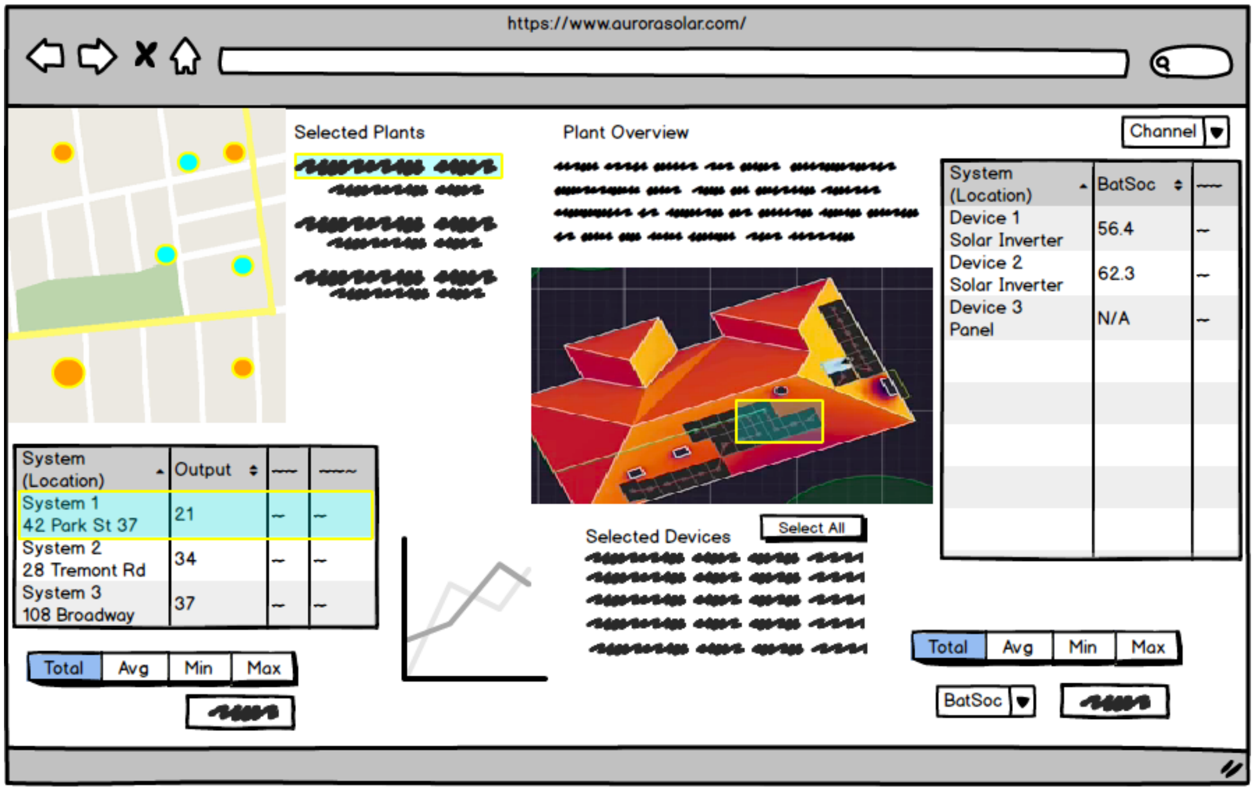
Solar installation monitoring is an important tool to add value to systems by analyzing parameters and tracking output. The given API endpoints hold important information about a system. The entire plant itself holds data about its total output as well as all of the individual data points for each of its devices. There is a lot of information to keep track of for each project, and if 10,000 new projects are created every week, there needs to be an efficient way to monitor all systems.

Monitoring software will need to be user friendly and easy to use since installers will want quick access to the information they seek. The software should show image representations of the setup and all of the devices in use. The user should be able to filter by device type or select which individual panels or inverters they want to analyze. Then they should instantly see all of the most relevant information sorted by device. For example, if the installer selects all solar panels, they should be able to see all of the relevant parameters with values. First, the average, min, and max values will be given, and the user can drill down into each individual device to see the specific information. This will work the same on the entire plant level and installers can see the system output for each individual system, or aggregate data for multiple systems they own given filtering parameters.

I would design this system in JavaScript using the React.js framework. React is powerful because it creates a single page application that loads only the components it needs for each action. For example, when looking into a given solar plant, the displayed data will change with each parameter selection, but the rest of the page will remain the same. Using React will allow the page to quickly change without doing a full reload which creates a better user experience. I would keep the backend similar to its current status as a mix between Rails using PostgreSQL and Python hosted on EC2 machines. Python is suited well for computations and data analysis which would be necessary as the system scales up. PostgreSQL scales well and can handle a large amount of data. It is also free and supports JSON which will be used in the application.

When creating my API end-points I would make sure that the database serves up only the information I seek for a specific request. For example, in this challenge when you make a request to "/plant_overview", it returns a list of all of the channels within each device. In a large plant, this one data object would be massive. The data is also redundant because many devices share the same or similar list of channels. The schema would have a section for all plants, with limited information about each plant. When requested, more information about each plant will be loaded such as all of the devices. All of the devices would hold reference to their channels, held in a different slice of state. Finally, the data points can live in the schema in multiple ways depending on their use. Flexibility is important. For example, if you want to request data on one attribute for each solar panel, their needs to be a better way than looking into each individual panel's information.

Part 3: User Interface Mockup



System (Location)	Output	~~	~~
System 1 42 Park St 37	21	–	–
System 2 28 Tremont Rd	34	–	–
System 3 108 Broadway	37	–	–

System (Location)	BatSoc	~~
Device 1 Solar Inverter	56.4	–
Device 2 Solar Inverter	62.3	–
Device 3 Panel	N/A	–

I did not include annotations in the mockup but I will describe my thinking a bit here. A user selects which plants to monitor from a map or a list view (not shown). They can then view aggregate data about all of the systems. Clicking on the graph will bring them to a page with more information on trends over time and deeper analysis of the total multiple plants system. When the user clicks on a single plant, the plant overview will appear with detailed information on the single plant. The user can choose specific devices to see more information and filter by channel in addition to getting the total, average, min, or max for each data attribute.