ECE498: Senior Capstone Project I

Project Proposal

Project Title: Development of an Intelligent Building Energy Management Platform

Brian Lauer and Elliot Watkins
Advisor: Dr. Suruz Miah

Electrical and Computer Engineering Department
Caterpillar College of Engineering and Technology
Bradley University

Table of Contents

1 Introduction

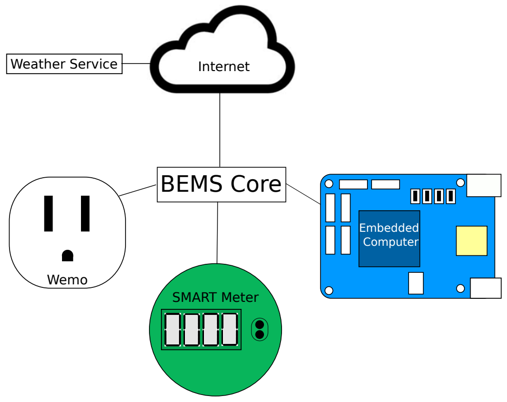


Figure 1: High Level Architecture of the Proposed System

The BEMS Core will interface with as many devices as it has support for. By the end of this semester, we plan to have support for the Wemo Insight Switch and the BeagleBone Blue. The user will be able to control these devices directly from the Web Server. Additionally, we will try to simulate a microgrid with MATLAB Simscape. Another feature we would like to add (if time allows) is a weather notification. The BEMS core will ping an online weather service routinely and notify the user of any storms likely to cause power outage in the forecast. To simulate a building HVAC system, a Simulink model will be created, and the Linear Quadratic Regulator (LQR) based algorithm will be used to regulate the building temperature to optimize both user comfort and energy usage.

2 Background Study

An existing BEMS described in [1] incorporates its own sensors for temperature, humidity, luminosity, air quality, and motion. The data from these sensors is then fed to the BEMS core where decisions are made for the HVAC system and lighting of a building. The sensor data is transferred to the BEMS though the IoT, like our own BEMS core.

One of the works in the literature discusses Model Predictive Control (MPC) which is a modern process control algorithm capable of taking into account current and previous time values to improve building automation [2]. A multilevel hierarchy is presented to split control of a building into two levels: the energy supply level and user level. Example usage of this control scheme provided are temperature control of a building and its corresponding zones and interaction with a smart grid. Studies were performed on a commercial building with the building's energy supply system and hierarchy model predictive controller implemented in Matlab.

In [3], the authors describe similar functional requirements to our own BEMS Core. They periodically gather power consumption and device status data and send that data to a database. We are doing exactly what they describe using the Apache

Cassandra database. The authors also use an "analytics [sic] engine to process [the data] and generate reports, graphs, and charts." Our system only generates a power consumption plot for a given day, but we are still meeting this same requirement - just on a smaller scale. The communication with devices and power consumption reports are to be accessible through a web-based application that is easy to use. Again, our implementation is simple and very intuitive for the end-user. Finally, these authors mention a user-login feature. They explain that their system would include multiple levels of accessibility based on the user privileges. Our system does not currently have a user-login feature. However, in its current state, the BEMS Core simply uses a router to access devices that are also connected to the router. Since this is a closed system, users need only protect their router with a secure password.

A control scheme in [4] is presented to manage a photovoltaic array and battery energy storage system. Tests were conducted in a shopping mall with an electronic load to emulate the power consumption of the building. Potentially, a similar technique could be used to model the energy demand of a house in Simulink or Simscape model. Through their scheme, an intelligent BEMS is used to collect measurable data like power, voltage, and current and stored for offline analysis later. Their platform collects data every five minutes compared to the 20 second polling rate currently configured in our platform. A control algorithm was presented proving that grid energy is consumed more heavily during periods of low power demand rather than PV power.

3 Functional Requirements

The minimum viable product of this project will be to connect to a Wemo Insight Smart Plug and a BeagleBone Blue. The only functionality needed on the BeagleBone Blue will be PWM input to the motor drivers. Additionally, we will be implementing a database to track power usage on both devices. To simulate power usage of an entire house, we will use MATLAB simscape to develop a microgrid and power meter.

4 System Architecture

4.1 Inputs

This project will include input from users, smart power meter, devices, and a weather service. Users will be able to adjust the operation of a device (turn up the temperature of an AC unit, for instance) or turn a device on or off completely. Smart power meters will be able to connect the platform and give real-time updates of the building's overall power usage. We would also like to look into meters for individual appliances like AC units and furnaces. The connected devices themselves will communicate to the web server their current status when needed. Finally, our platform will regularly receive weather updates from a weather service and warn the user about potential energy loss due to storms.

4.2 BEMS Core

A high level view of the operation of the software platform is shown in Figure 2. Incorporated into the platform will be a web server developed in Python with the Flask web framework. All external and internal HTTP (Hyper Text Transfer Protocol) requests will be processed through this web server in order to store device or user information in the SQLite database. An external request is defined as a request made by a user or device to, for example, update information in the database with a series of SQLite queries. Internal requests are made by other entities or processes running in parallel on the platform including software agents. Similar to BEMOSS, these agents will help to communicate with various devices supported by the platform like broadcasting a discovery service, sending control commands to devices, and regularly requesting data from devices. The platform will be developed and deployed on a Linux machine to encourage support for deployment on SBCs (Single Board Computers) like the Raspberry Pi or BeagleBone Blue to be conveniently stored in a secure location of a building or home like a server cabinet for instance.

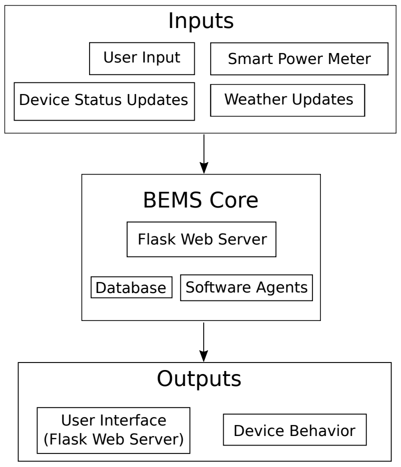


Figure 2: High-level Functional Block Diagram

The SQLite database will be designed to store device information like the device IP address, MAC address, manufacturer name, and device name. A feature will be implemented to allow for customizing the device name. For example, a WeMo Switch connected to the building or home LAN (Local Area Network) could be renamed to "Lamp" or "Light". Secondly, a feature will be added to specify the room or location of each device within the building or home. This will aid any users of the software in distinguishing devices from each other.

Functionality will be available in the final version of the platform to store time-series data like on/off status and power data in a database using Apache Cassandra, an open-source NoSQL database management system. This offers a convenient way to log data from device readings in real time. In the future, machine learning algorithms can be used alongside this database to help optimize energy usage.

A further addition to the software will be added in the form of support for Simscape models, an addon for Simulink to model electromechanical systems. Matlab scripts will be associated with each of these models to collect power data from various electrical models created with Simscape. Using TCP/IP, Matlab will send data from the associated models over the network to the platform to be stored in the Cassandra database.

4.3 Outputs

The BEMS Core will output all data from devices, smart power meter, and weather updates to the web server. The user will then be able to peruse all available data on any smart device connected to the network.

5 Preliminary Work

Prior to the start of the capstone project, work had been completed on the platform based off BEMOSS [5]. Most of the software architecture was heavily derived from the previously developed software.

5.1 Software Agent Design

As part of the base requirements of the design, the platform incorporates multiple agents to help facilitate communication between various parts of the software. Some of the current agents which are Python classes in the software include

- Discovery agent - Traverses through available API files and uses the `findDevices` API call to locate devices on the network, later on auto discovery will be implemented to allow devices to be shown in the software whenever a new supported device joins the network

- Control agent - General agent for all supported devices, capable of retrieving and modifying the status of a device available on the network through API calls, supports starting up a separate thread to periodically query the device for information like ON/OFF status and power consumption

A flow chart of the scheme the Discovery agent uses to initialize a device in the platform is shown in Figure 3. To ensure a new device is only stored once in the metadata database, a query is performed to ensure that only one device exists with the recently discovered device's name. Once a new table is created in the Cassandra database for the device which is unique for the specific device, the control agent is notified to start up a new thread to collect device data.

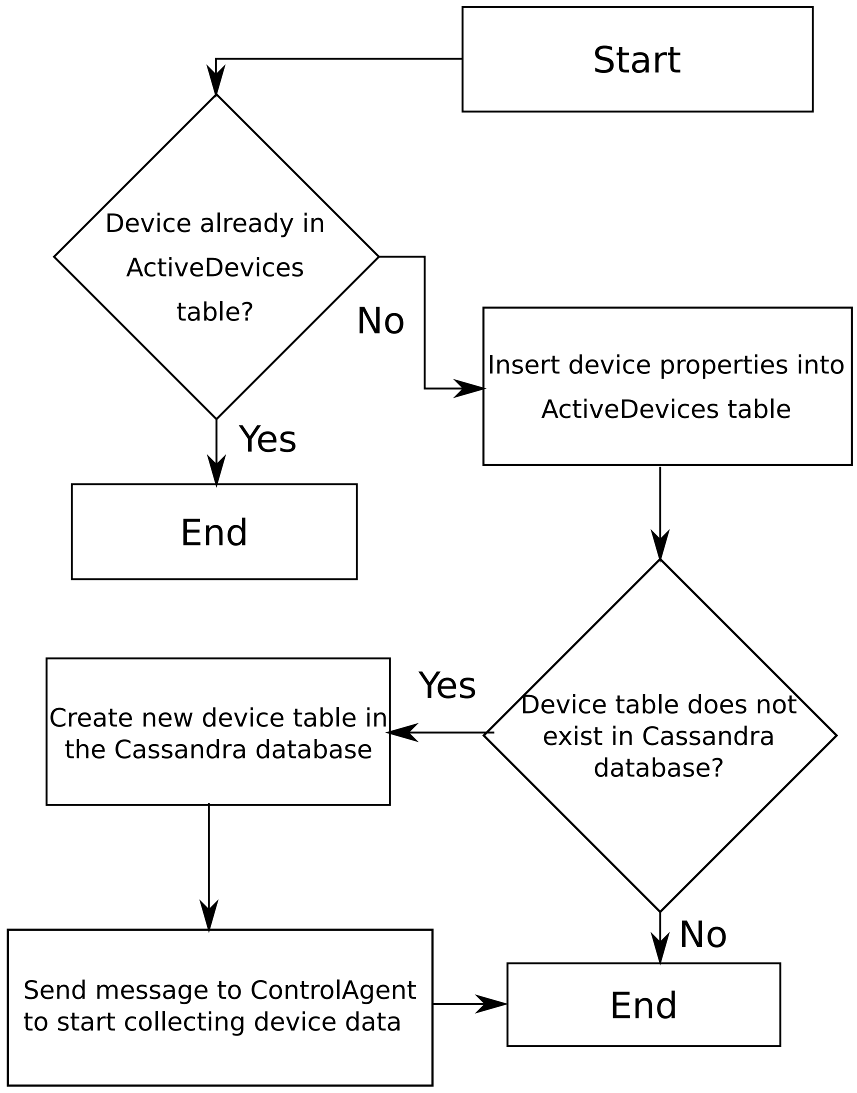

Start
Device already in ActiveDevices table?
No
Yes
Insert device properties into ActiveDevices table
End
Device table does not exist in Cassandra database?
Yes
Create new device table in the Cassandra database
No
Send message to ControlAgent to start collecting device data
End

Figure 3: Flow chart of the `setDeviceToActive` method

A second algorithm that the discovery agent uses to look for devices on the network is shown in Figure 4. The process to search for available devices on the network is executed for each individual device API. Once a device has been found it is then added to the active devices table if it does not already exist.

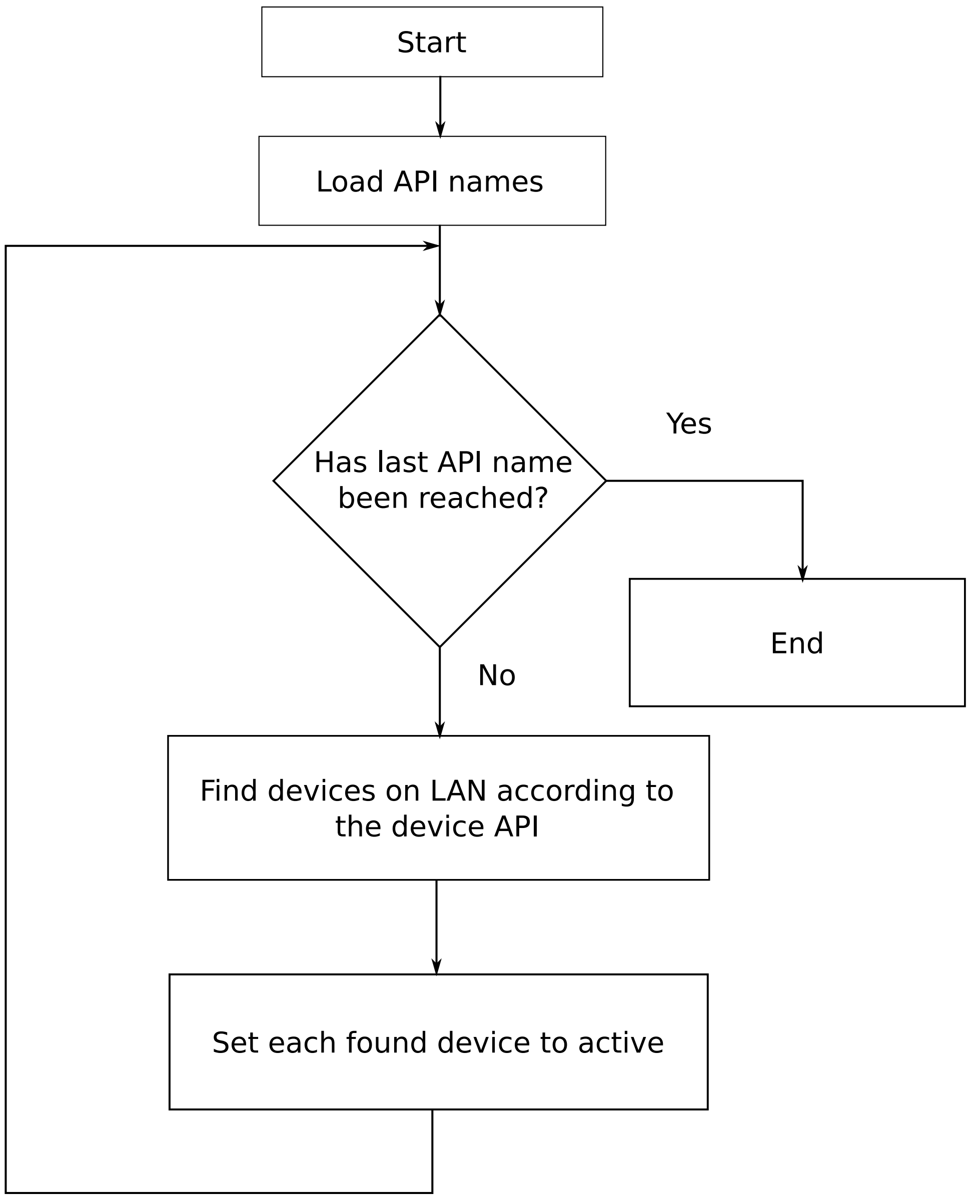

Start
Load API names
Has last API name been reached?
Yes
End
No
Find devices on LAN according to the device API
Set each found device to active

Figure 4: Flow chart of the `searchForDevices` method

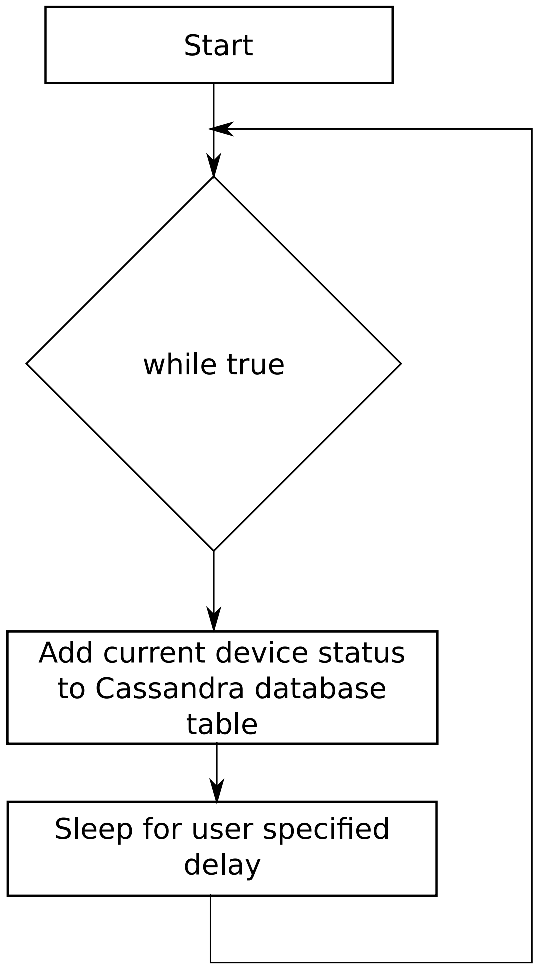

Start
while true
Add current device status to Cassandra database table
Sleep for user specified delay

Figure 5: `ControlAgent.periodicQueryBehavior()` flow chart

More agents will likely be added later to help with supporting devices with behavior differing from that of the WeMo Switch and DC Motor.

5.2 Device APIs

Common APIs are included for each supported device like `setState`, `getState`, `findDevices`, and `findMetadata`. In the WeMo Switch implementation, IP multicasting and Universal Plug and Play are utilized to send out messages to any devices listening on a multicast group with a specific IP address and port number. The location of an XML file is located whenever a Switch has been properly located on the network. This file contains information about the device including the nickname, manufacturer, and IP address as well as other information like Simple Object Access Protocol (SOAP) commands to alter the state of the device like ON/OFF status and commands to retrieve device information. The publish/subscribe model is used for communication between the web server and agents. Each agent supports a separate thread dedicated to subscribing to messages from publishers.

Support for the SQlite database has been successfully added while the Cassandra database support is still being implemented. Also, driver support is still being developed for the Beaglebone Blue and ultimately the DC motor.

5.3 Experimental Setup

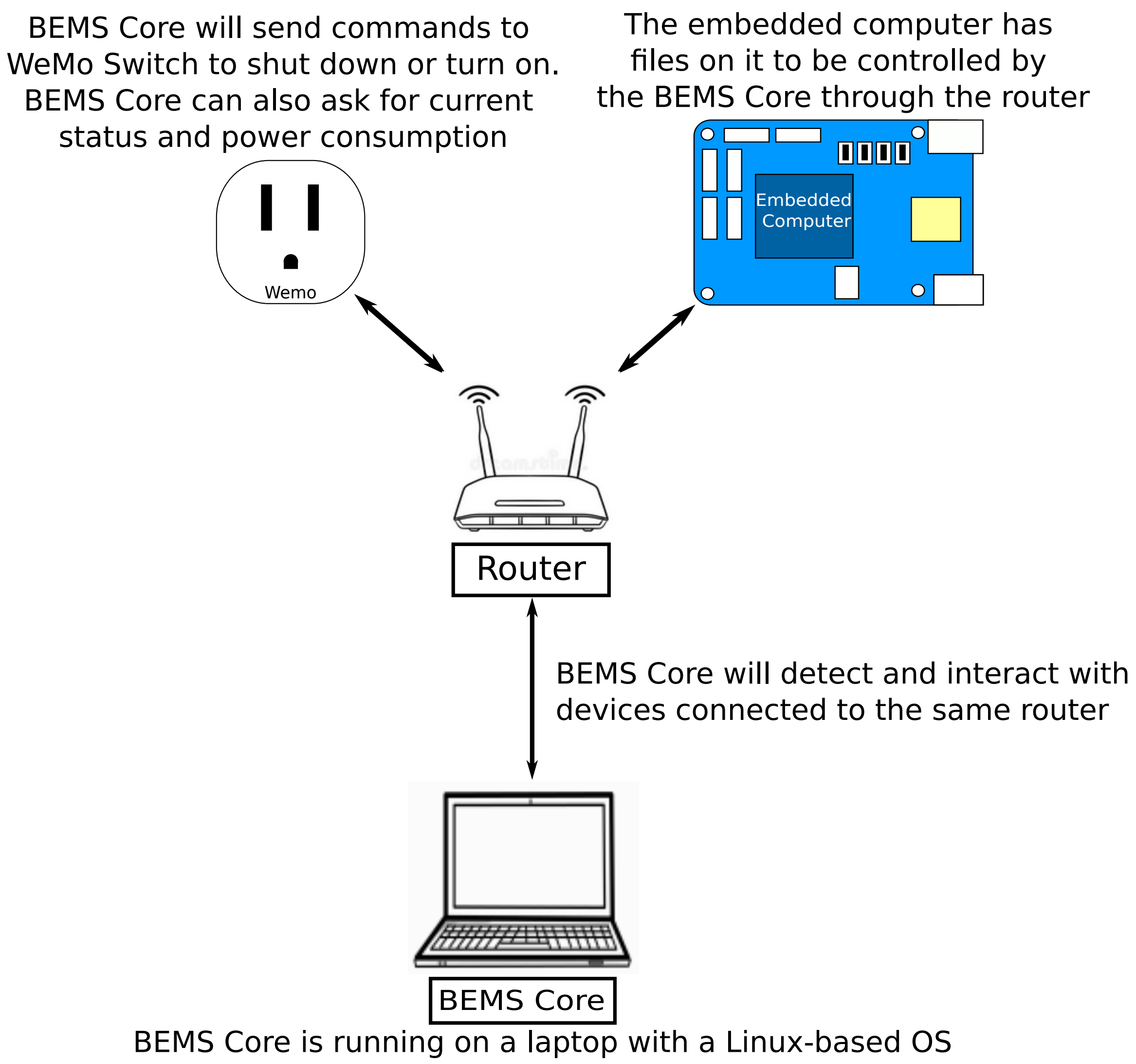
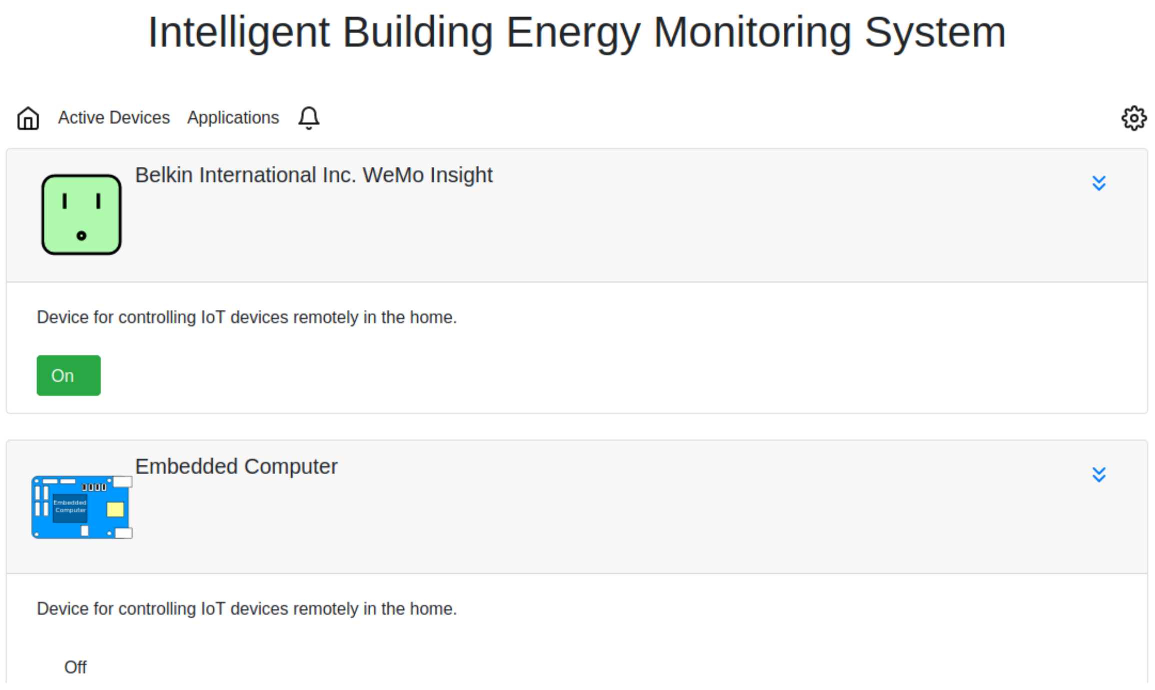

BEMS Core will send commands to WeMo Switch to shut down or turn on. BEMS Core can also ask for current status and power consumption
The embedded computer has files on it to be controlled by the BEMS Core through the router
Wemo
Embedded Computer
Router
BEMS Core will detect and interact with devices connected to the same router
BEMS Core
BEMS Core is running on a laptop with a Linux-based OS

Figure 6: Current lab setup for testing purposes

5.4 Experimental Activities

Up to this point, work has been completed on the web user interface which was developed with the Bootstrap CSS framework and the JQuery Javascript framework. The active device page in Figure **??** displays the current controllable devices in the software. When a request is sent to the active devices URL, the discovery agent will ping all possible supported devices on the network and newly discovered devices


Intelligent Building Energy Monitoring System

Belkin International Inc. WeMo Insight

Device for controlling IoT devices remotely in the home.

Embedded Computer

Device for controlling IoT devices remotely in the home.

Figure 7: Screenshot of Web Server (Active Devices Page)

will appear on the active devices page. Upon pressing the refresh button below the list of controllable devices, all buttons and icons will update with the current status obtained over the network. For example, the refresh button will cause the position of the slide switch shown to change depending on whether the WeMo switch is on or off. The top nav bar will link to other pages including the home page, applications page, notifications and settings page. However, only the active devices page has been implemented.

Figure 8 shows a plot of the Wemo power usage using data we collected during lab hours.

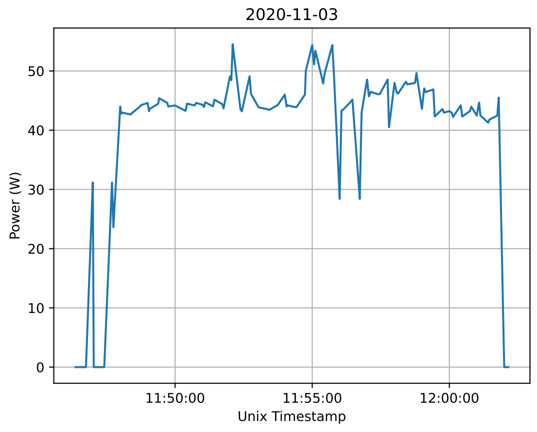


Figure 8: Plot of WeMo Switch power usage 2020-11-03

6 Parts List

- WeMo Insight Smart Plug
- BeagleBone Blue
  - Octavo OSD3358 Microprocessor
  - Wifi/Bluetooth
  - IMU/Barometer
  - Power Regulation and state-of-charge LEDs for a 2-cell LiPo
  - H-Bridges
  - Connectors for 4 DC motors and 8 Servos
- JST-ZH connector (for BeagleBone Blue Motor Connections)
- DC Motor
- ECE department laptop
- ECE department lab router

7 Timeline and Milestones

During the first semester, Fall 2020, work will be done to further polish the WeMo Switch functionality like updating the status and obtaining the power usage. In order to further set up the platform, support for the Cassandra database will be added to store power usage and other device status information. Each entry in the database will be associated with a specific device and timestamp containing the current time and date. To visualize the data, a Matplotlib plot will be generated by reading data from a CSV file generated from the device status information. This plot will be displayed in the UI. DC motor support will be built through Beaglebone Blue connectivity over the TCP/IP bulding network. User login and management will be added to give users a more personalized experience in the platform. In the second semester, Spring 2021, algorithms will be developed to simulate and control both a microgrid and HVAC system along with integration in the platform. Lastly, as a final touch weather service support will be added along with notifications if time allows.

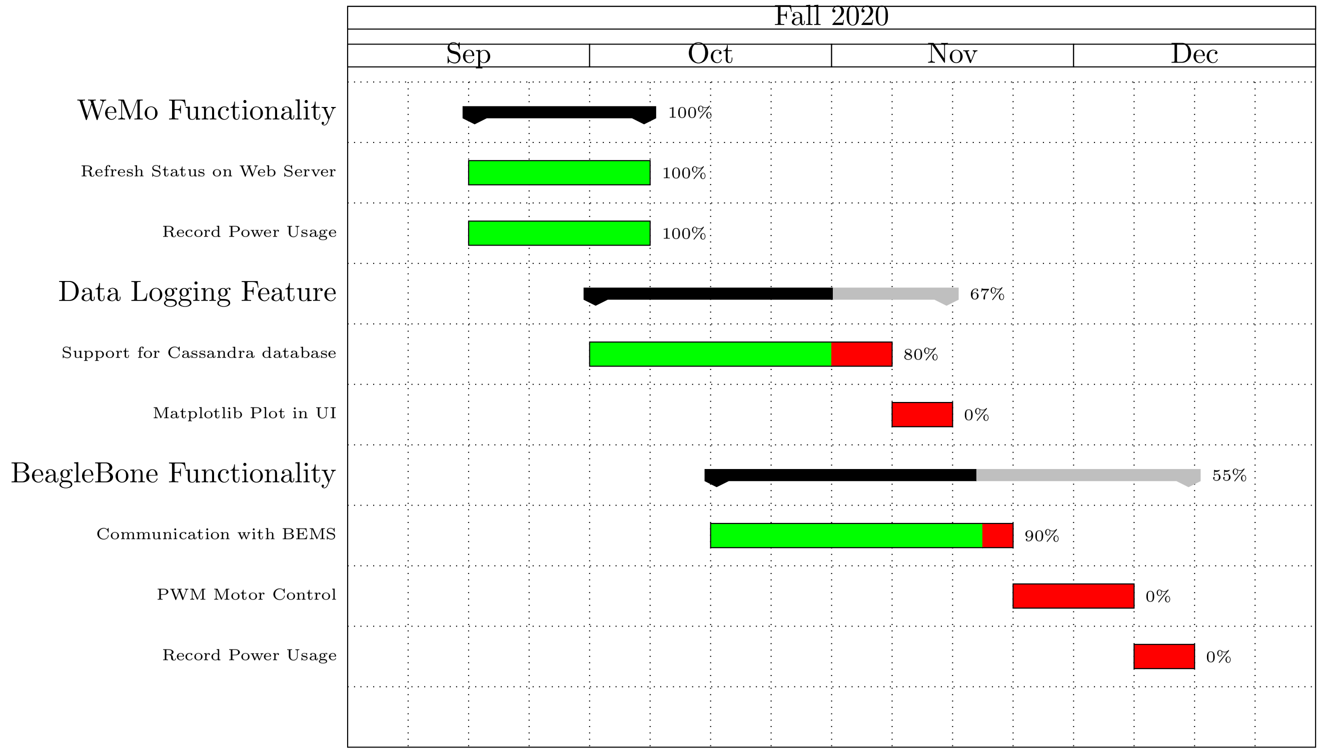


Figure 9: Gantt chart for Fall 2020

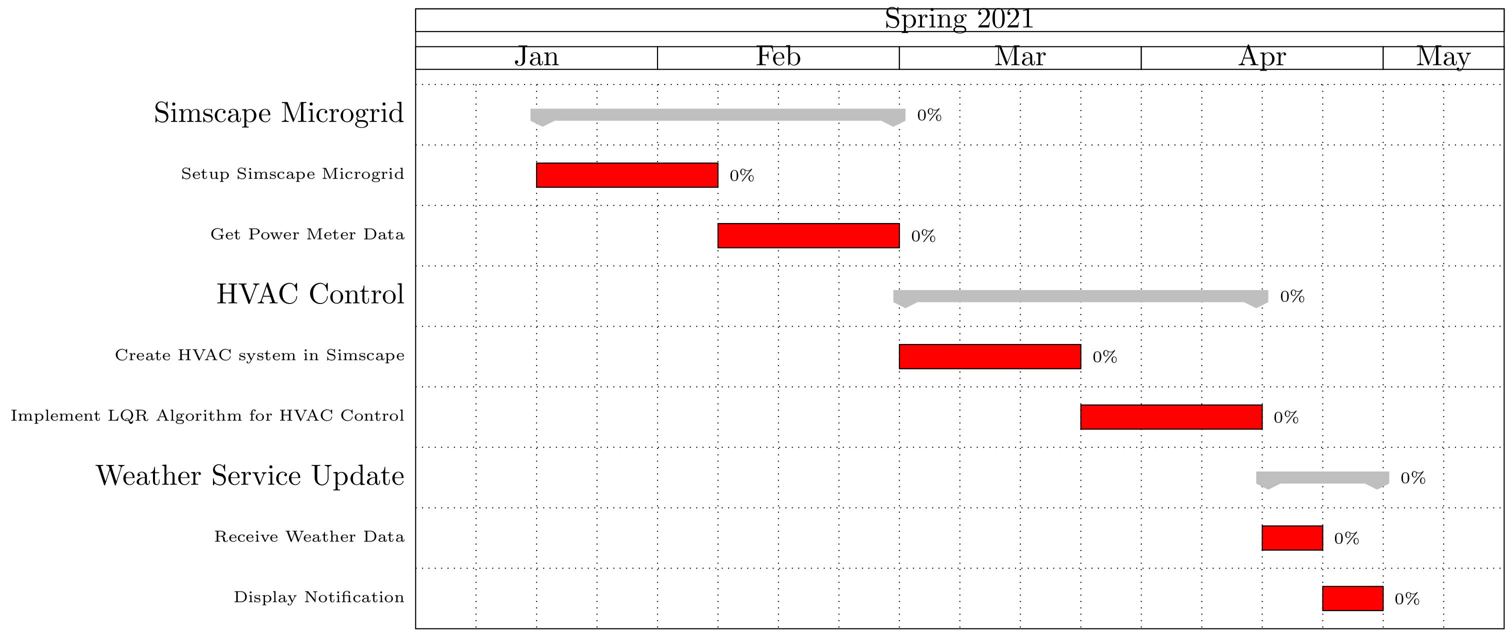


Figure 10: Gantt Chart for Spring 2021

References

[1] B. Mataloto, J. C. Ferreira, and N. Cruz, “Lobems—iot for building and energy management systems,” *Electronics*, vol. 8, no. 7, p. 763, Jul 2019. [Online]. Available: http://dx.doi.org/10.3390/electronics8070763

[2] B. Mayer, M. Killian, and M. Kozek, “Hierarchical model predictive control for sustainable building automation,” *Sustainability*, vol. 9, no. 2, p. 264, Feb 2017. [Online]. Available: http://dx.doi.org/10.3390/su9020264

[3] A. R. Al-Ali, I. A. Zualkernan, M. Rashid, R. Gupta, and M. Alikarar, “A smart home energy management system using iot and big data analytics approach,” *IEEE Transactions on Consumer Electronics*, vol. 63, no. 4, pp. 426–434, 2017. [Online]. Available: https://ieeexplore.ieee.org/document/8494012

[4] G. Barchi, G. Miori, D. Moser, and S. Papantoniou, “A small-scale prototype for the optimization of pv generation and battery storage through the use of a building energy management system,” in *2018 IEEE International Conference on Environment and Electrical Engineering and 2018 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I CPS Europe)*, 2018, pp. 1–5.

[5] V. Tech. (2019) Bemoss3.5. [Online]. Available: https://github.com/bemoss/BEMOSS3.5