ECE498: Senior Capstone Project I
Project Proposal

Project Title: BEMOSS

Reece Bachman Robert O'Malley and Jordan Ingram
Advisor: Dr. Suruz Miah

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

Table of Contents

1 Introduction

As smart power grids become more prevalent and data on smart building energy usage becomes more available, the opportunity for energy efficiency increases drastically. Data collected from smart buildings allows a neural network to control the workings of a building in an efficient manner. We propose to expand on the loads that can be controlled through BEMOSS by enabling control of a DC motor. Building energy management open source software (BEMOSS) is an internet of things (IoT) software that was developed at Virginia Tech. This proposed control of a DC motor through an IoT software presents the opportunity to close blinds, open barriers, and move objects throughout the building. The ability to open and close blinds building wide permits the ability to regulate the interior temperature of a building with a considerably lower power consumption. For instance, closing blinds during a hot day will naturally cool a room while opening blinds during a cold day will naturally heat the room. This added ability to control the interior environment will reduce the energy consumed by a heating ventilation air conditioning (HVAC) system.

2 Background Study

Authors in [1] proposed a energy management system based around Zigbees and PLCs. Han's system would place a energy measurement and communication unit in each outlet and light in the consumer's home. The energy usage will be measured and the data collected will be sent to a home server run on a Zigbee. These Zigbees will analyze the data and give feedback to the user on how to better manage their energy usage. Renewable energy will be connected to a PLC to allow the use of renewables as they need to converted to AC. The home server on the Zigbee will predict the amount of energy that will be obtained by renewables by accessing weather data and automatically adjust the user's device schedule.

In [2] the authors once again proposed sensors in all outlets and lights to measure the energy usage in homes. Renewable energy sources, like solar panels and wind energy, are connected to an inverter and battery system, to allow the storage of excess power. Collotta connects the internet of things devices using Bluetooth rather than WiFi, due to the lower power consumption. The system checks the current time and cross checks it with the peak time for energy consumption by the user's energy provider. If it is during peak times, the system checks the battery system for excess stored energy. When the user is trying to use energy during a peak time and without a battery charge the system warns the user, but allows the user to ignore these warnings.

In [3] the energy system uses two model predictive controllers (MPC), one for the building's HVAC system and one for the system's battery power flow. The building's energy management system predicts the temperature of each room separately using sensors and a Kalman filter for robustness. The battery system put in place by Mantovani can run on two modes, one to minimize energy cost and another to minimize power flow. The building is also c-onnected to wind turbines and photo-voltaic cells

and predicts the energy produced using a simplified model.

The authors in [4] propose a Internet of Energy network rather than an IoT network. An IoE combines IoT and smart grid technology using four key components: Energy Router, Storage System, renewable sources, and plug-and-play appliances. The IoE allows for an easier way to produce a net zero energy building, that produces as much or more energy than it consumes. The energy router consists of a solid-state-transformer, grid control, and communication meant for data management. The storage system like batteries, reduce the stress on the power grid and lower voltage fluctuation. Renewable sources reduce carbon emmisions, however they reduce harmonics that need to be handled with additional hardware. Lastly the plug-and-play appliances are the devices that the end-user uses in a home.

In [5] the system heavily interfaces with the users' smartphones as a way to monitor the building occupants. Since smartphones almost all have a way to track GPS, the system tracks the users' location and send it to the building's server. The building's server is broken up into a number of subservers to handle an individual room's needs. By tracking location the building can do such things like pre-heating or pre-cooling a room before the user is even in the building. All the subservers are connected to the main server which is connected to cloud storage which is used for hosting the large amount of data and handling the intense computations.

3 Functional Requirements

3.1 System Architecture

For our project, we consider a main BEMOSS command station that will be used to transmit our commands via a TCP/IP network. From there, a central node will collect the commands from the TCP/IP network and execute control algorithms to command our remote nodes. Radio frequency (RF) communication will be employed to communicate between the central node and the remote agents. This is expressed in Figure 1

3.2 Block Diagrams

Our proposed BEMOSS capable device is a DC motor. This DC motor will be controlled via a slave XBee. A python script running on a single board computer will send remote AT commands through a coordinator XBee to the remote XBee. As shown in Figure 2, the XBee coordinator communicates motor direction and PWM data to the remote XBee attached to a motor driver. This XBee communication will be also be used to transmit motor position from the remote motor back to the central node. This bidirectional data exchange will provide a closed-loop system feedback to precisely control motor movement. The data exchange between the slave XBee and the motor driver (MD) will require logic converters. A TXS0108E 8 channel logic converter will be employed to convert between the 3.3v logic of the XBee module and the 5v logic of the motor driver and corresponding encoder output channels.

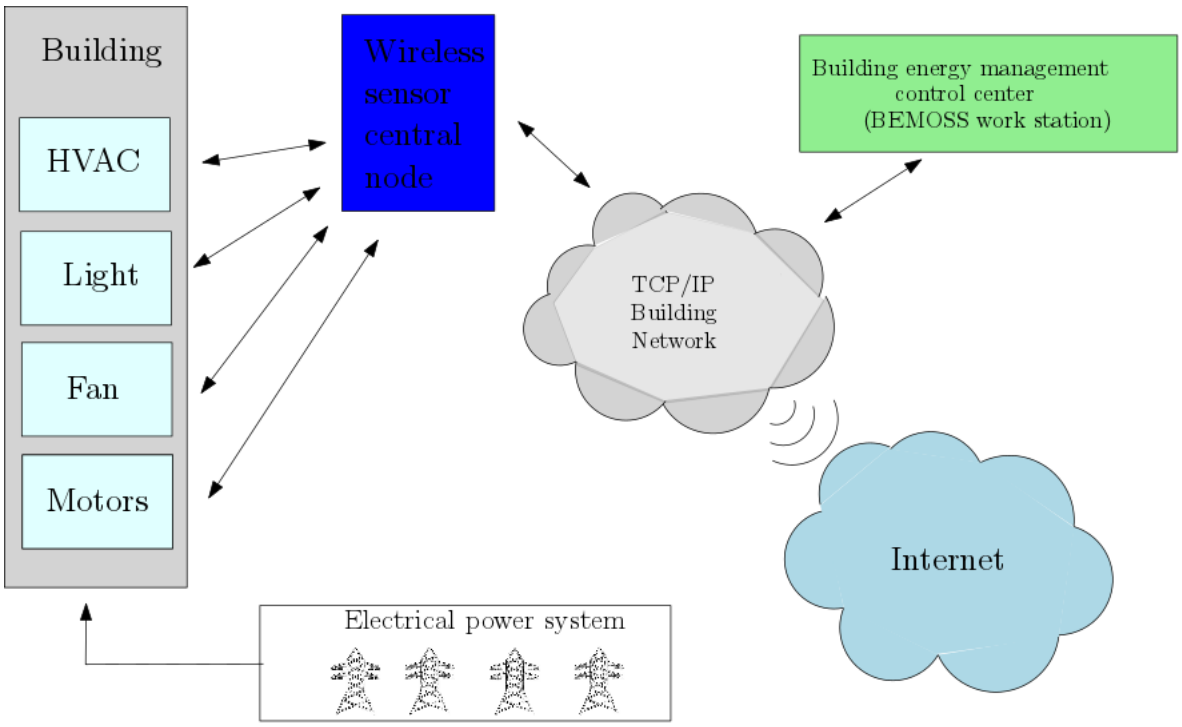


Figure 1: High Level Diagram of System Architecture

3.3 Specifications

For our project, we have three main specifications that we will be implemented. The first is that we will be introducing a new supported device within BEMOSS. This device will work with BEMOSS without any configuration changes within the software. The second is to get BEMOSS to work on a new device such as a Raspberry Pi. This is a beneficial portion of our project as it presents the ability to have multiple nodes for different BEMOSS environments. The final specification is to develop an algorithm for BEMOSS energy control. We will determine the amount of energy consumed in an environment without an algorithm employed, how much energy is consumed with BEMOSS running, and how much energy is consumed with the developed algorithm employed within BEMOSS.

4 Preliminary Work

After we familiarized ourselves with the architecture and working of BEMOSS, we downloaded and installed the necessary software for the project. To do so, we set up a virtual machine with Ubuntu Linux 16.04. We also did research on what BEMOSS has been used for in the past. This research included who works with the software as well as what it can currently do.

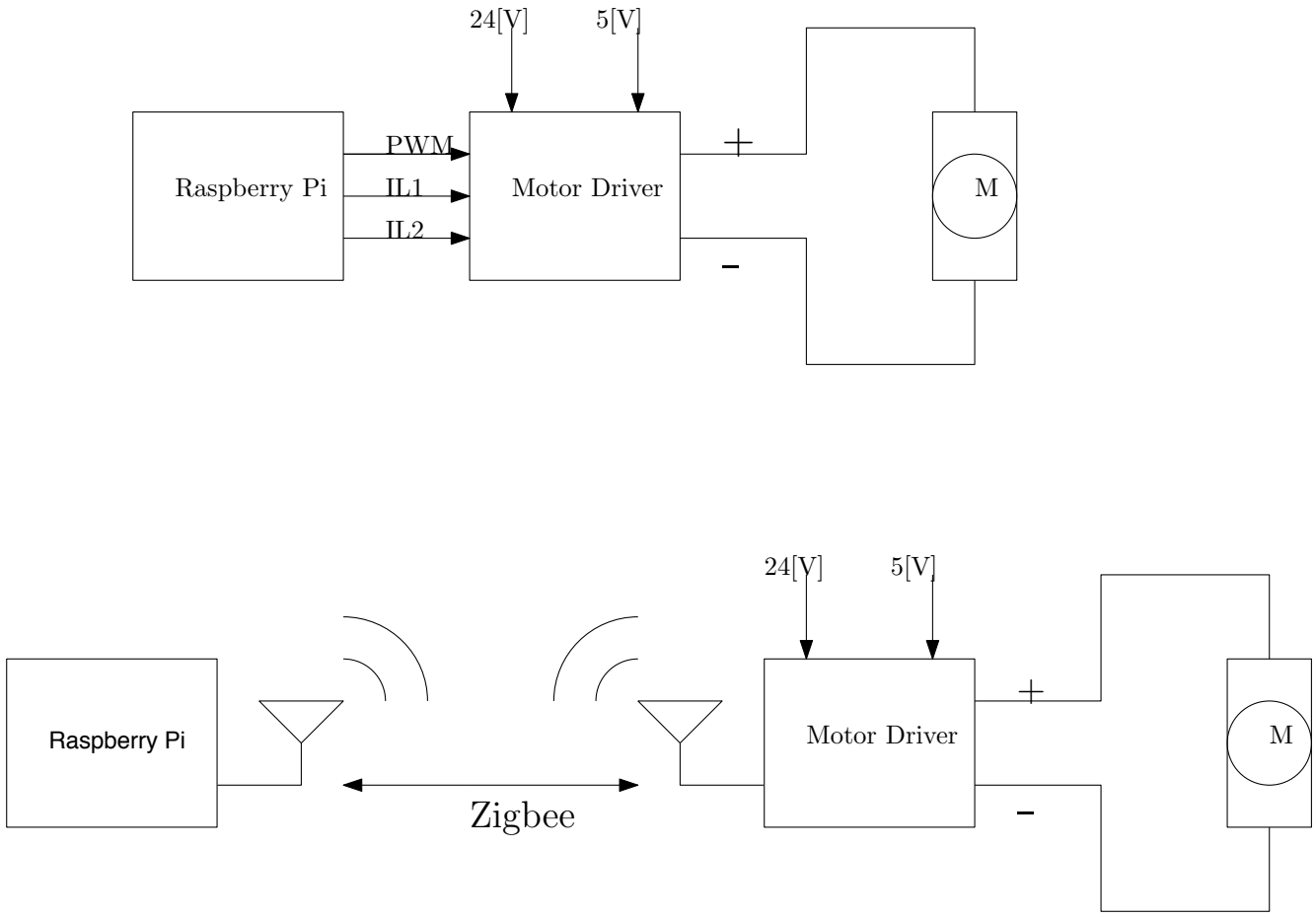


Figure 2: Raspberry Pi- Motor Driver Communication

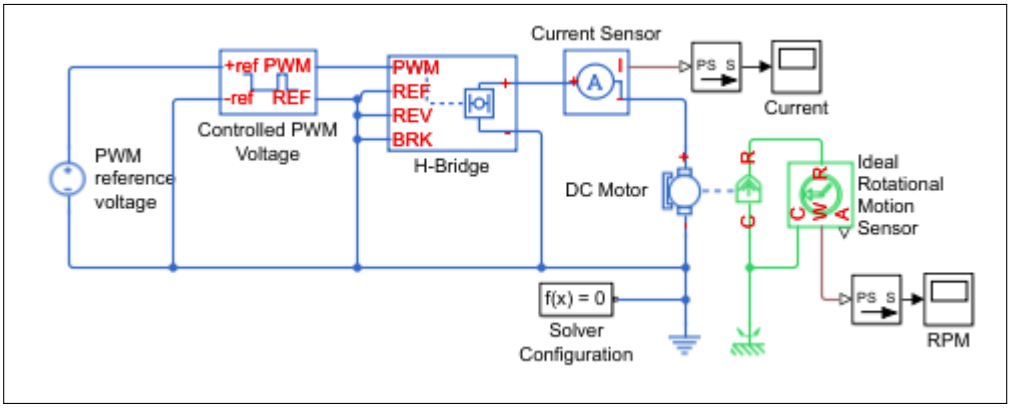


Figure 3: Simulink Model of Motor with H Bridge Control

4.1 Modelling

We began by creating a model for our motor schematic using Simscape. This model uses a pwm signal coming from a power source that is fed into a H-bridge. This signal is then run through a motor. Power is calculated from this model for later use when designing a neural network algorithm to minimize the cost function. Next we began to create a model for a HVAC system for a house. This model consists of a thermostat, heater and thermal loads. To begin a simple model was used to control the temperature. This controller only checks for the current internal temperature of the house and compares it with the reference temperature point. To build on to this model we followed the works of [6][7] to develop an adaptive control model for N-number of rooms. The control algorithm used for this model is based on a state-space model using the simple system of equations

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$$

From [6] the system of equations for the single room case is as follows:

$$\frac{\mathrm{d}\mathbf{T}^e}{\mathrm{dt}} = \begin{bmatrix} \dot{T}_1^e \\ \dot{T}_2^e \end{bmatrix} = \begin{bmatrix} -\frac{u_{cc}A_{cc}}{M_{cc}C_v} & \frac{Q_w\rho_w C_{\rho w}}{M_{cc}C_v} \\ 0 & -\frac{Q_w\rho_w C_{pw}+U_tA_t}{V_t\rho_w C_{pw}} \end{bmatrix} \begin{bmatrix} T_1^e \\ T_2^e \end{bmatrix} + \begin{bmatrix} \frac{U_{cc}A_{cc}}{M_{cc}C_v}T_{amb} - \frac{Q_w\rho_w C_{pw}}{M_{cc}C_v}T_{wo} \\ \frac{U_tA_t}{V_t\rho_w C_{pw}}T_{amb} + \frac{Q_w\rho_w C_{pw}}{V_t\rho_w C_{pw}}T_{wo} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{15000}{V_t\rho_w C_{pw}} \end{bmatrix}\chi + \begin{bmatrix} (\frac{\rho_a C_{pa}}{M_{cc}C_v}Q_{a1} - \frac{U_{cc}A_{cc}}{M_{cc}C_v}) \\ 0 \end{bmatrix}$$

Using the specifications from [6] the model becomes:

$$\begin{cases} \dot{T}_1^e = -.02815T_1^e + 1.2073T_2^e - 0.02815T_{z1} + 0.0005Q_{a1} + 3.932Q_{a1}T_{z1} - 3.932Q_{a1}T_{z1} + 6.0420 \\ \dot{T}_2^e = -0.0007T_2^e + 0.006\chi + .0192 \\ \dot{T}_{z1} = -0.0006T_{z1} - 0.1144T_{z1}Q_{a1} + 0.1144T_1^eQ_{a1} + 0.021 \end{cases}$$

This system is controlled by the inputs Q_{a1} and χ. This system is a linear time

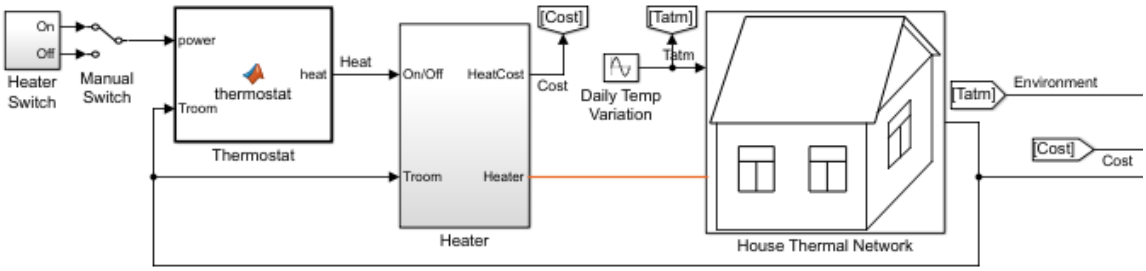


Figure 4: Simulink House HVAC Model

variant system and therefore needs to be solved using a ordinary differential equation (ode) solver.

To model this system we chose to use Simulink as it will allow us to easily set up the system and has a built in ode solver and will allow us to more easily connect the house model with the models of our other devices for when we develop a robust energy management system.

4.2 Design

In the fully functioning system, a computer running linux will act as the main BEMOSS control center. From there, a Raspberry Pi 3B microcomputer[1] will act as a central node by receiving the commands from the TCP/IP network and transmitting the necessary information to the remote motors over a Zigbee network. In this central node, numerous python scripts will be employed to transmit AT commands to the Xbee network and to employ control algorithms to accurately control and monitor the remote motors. The coordinator XBee will command the remote xBees to toggle 2 of their pins depending on the desired rotation direction of the motor's shaft (as commanded from the BEMOSS control center). These two pins will be connected to a L298N motor driver which will power and switch the polarity of the motor. When either of the remote pins switch high, relays will be activated to supply 5v to the encoder. This power will only be available when the motor is being commanded to turn as to prevent energy waste. The encoder output channels will pass through a TXS0108E logic level converter to transmit the positional data through the remote xBee back to the coordinator XBee. This data will be used to provide a closed loop feedback of the motor's position. The data will be processed on the central node's computer and will influence the precise transmission of the remote motor shut off commands after a certain position has been achieved.

[1] `https://raspberrypi.org`

4.3 Experimental Activities

Thus far, our work has been oriented towards crafting a useful system by creating a standard functioning product and continually expanding on it's abilities and practicality. Our original design used a Raspberry Pi to send data to a H-Bridge that controlled the rotational direction and speed of rotation of a motor. From there, we expanded the ability of this system by creating a central node that can communicate between numerous remote motors. We did this by incorporating a Zigbee network into our system. We initially used XCTU to initialize the XBee modules and send AT commands through a USB port to a coordinator XBee. From there, the command would be received from remote XBee modules and control a remote motor. After we successfully transmitted commands from XCTU to rotate the motor shaft in either direction, we aimed to make the product more practical. We proceeded to create an interface between the serial port of a Raspberry Pi and the coordinator XBee as to eliminate the need for human interaction in XCTU and allow the ability for additional automation between BEMOSS commands and motor rotation. This Zigbee/ Raspberry Pi interface allows for a single internet connection to control a building's worth of remote blinds and doors. In order to experiment, we created python scripts that send AT commands through the Zigbee network that control the speed of a remote motor and change rotational direction of a remote motor. We are currently expanding on the capabilities of the system by creating a closed-loop feedback system to monitor the position of remote motors. We are currently waiting for an order of relays and logic level converters that will allow for a translation between the 3.3v logic level of XBee modules and the 5v logic level of the encoders and H-Bridges that are employed on each remote motor. When this system is implemented, a command to, say, close curtains will initiate the transmission of a remote AT command to rotate a remote motor in the required direction and monitor the positional data until a certain position is reached. The positional data will only be monitored and transmitted while the motor is activated. This feature is added to prevent unnecessary energy consumption by encoders.

We have also been working with already supported BEMOSS devices to research how the software works. We installed BEMOSS on a laptop running Ubuntu 16.04 and controlled a WeMo plug. In order to get BEMOSS to work properly we started on a virtual machine on Bob's laptop. Doing this allowed the server to start but we were not able to find the WeMo plug. The reason we could not find the WeMo plug was because the network settings within a virtual machine are not the same as the network settings on a host computer. The virtual machine attempts to obtain its own ip from a router but the router might not allow this depending on the WiFi card. This was a problem until we were able to obtain a laptop from Mr. Mattus. Once we had the laptop we were able to wipe the hard drive to install Linux Ubuntu16.04. This allowed us to work with BEMOSS and the supported devices as intended. We can conclude from this that virtual boxes cause a lot of issues when the connection to the router needs to be final. Bob worked through this for a while and found out that the bridged adapter does not always work. In testing, we discovered that there were a few issues found on the newer versions of Oracle VM VirtualBox and pertaining

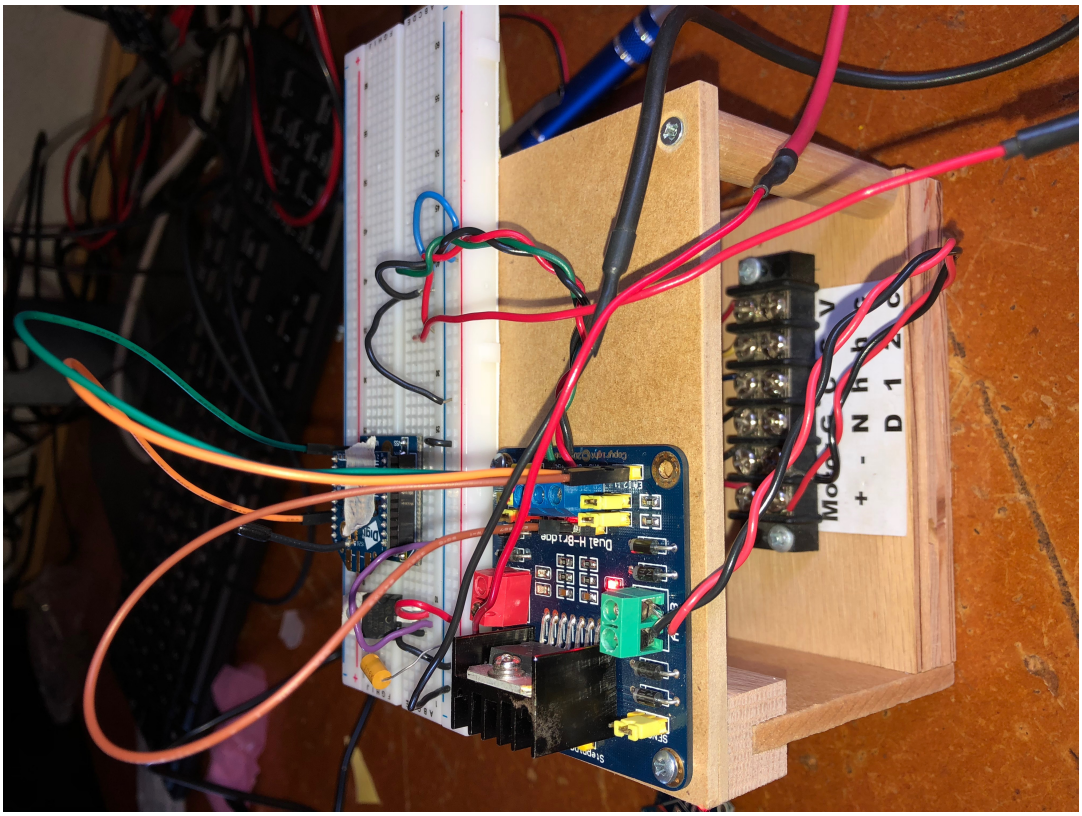
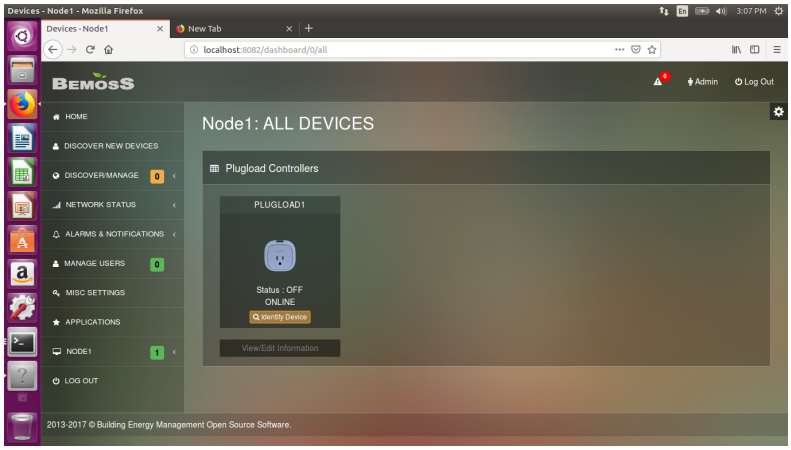
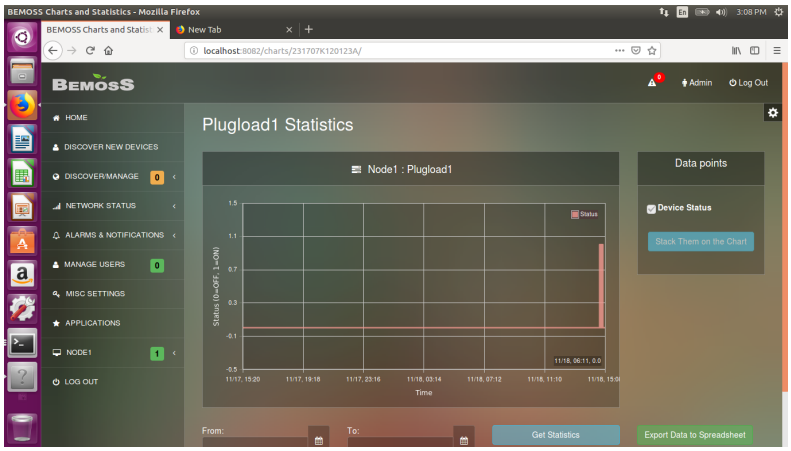


Figure 5: Remote Motor Circuit

to the Bridge network. We had hoped that this would solve our problems but after a week or so of testing realized that this was not a possible solution for our issue. (write about what we've done, what didnt work, what you did to make it work, and what we can conclude now that we have it done)



The device also gives us some data that we will be using in the future. BEMOSS outputs the power usage of each device and allows the user to create a schedule for each device. These features will be utilized in the future to create the neural network algorithm by providing energy cost and by allowing us to reduce the usage of each component by controlling them.



5 Parts List

Our project consists of some easily accessible components. The linux command center is ran as a parallel operating system on our laptops. On top of a laptop, we utilized 1 Raspberry Pi, 2 XBees, 2 xBee to USB explorers, a L298N motor driver, a Pittman 24v motor,2 breadboards, a TXS0108E logic converter, and 2 3v relays. Most of these parts were acquired in the engineering building for free. We did purchase a few BEMOSS supported plugs to test the functionality of our BEMOSS system before we began exploring new load options and capabilities with our motor. The list of purchased parts can be seen in Table 1.

Part	Price	Quantity
XBee Interface Board	$5.31	1
8 Channel Logic Converter	$5.99	1
3.3 V Relay Board	$11.90	1
Buck/ Boost Converter	$9.99	1
WeMo Smart Plug	$ 27.90	2

Table 1: List of ordered parts

References

[1] J. Han, C. Choi, W. Park, I. Lee, and S. Kim, "Smart home energy management system including renewable energy based on zigbee and plc," *IEEE Transactions on Consumer Electronics*, vol. 60, no. 2, pp. 198–202, May 2014.

[2] M. Collotta and G. Pau, "A novel energy management approach for smart homes using bluetooth low energy," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 12, pp. 2988–2996, Dec 2015.

[3] G. Mantovani, G. T. Costanzo, M. Marinelli, and L. Ferrarini, "Experimental validation of energy resources integration in microgrids via distributed predictive control," *IEEE Transactions on Energy Conversion*, vol. 29, no. 4, pp. 1018–1025, Dec 2014.

[4] M. A. Hannan, M. Faisal, P. J. Ker, L. H. Mun, K. Parvin, T. M. I. Mahlia, and F. Blaabjerg, "A review of internet of energy based building energy management systems: Issues and recommendations," *IEEE Access*, vol. 6, pp. 38 997–39 014, 2018.

[5] J. Pan, R. Jain, S. Paul, T. Vu, A. Saifullah, and M. Sha, "An internet of things framework for smart energy in buildings: Designs, prototype, and experiments," *IEEE Internet of Things Journal*, vol. 2, no. 6, pp. 527–537, Dec 2015.

[6] V. Reppa, P. Papadopoulos, M. M. Polycarpou, and C. G. Panayiotou, "A distributed architecture for hvac sensor fault detection and isolation," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 4, pp. 1323–1337, July 2015.

[7] N. K. Dhar, N. K. Verma, and L. Behera, "Adaptive critic-based event-triggered control for hvac system," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 1, pp. 178–188, Jan 2018.