BRADLEY University

Laboratory Notebook

BEMOSS and Its Enhanced Applications

Brian Lauer

blauer@mail.bradley.edu

Beginning March 13, 2018

Contents

Monday, May 06, 2019

I emailed Mr. Mattus asking him whether he made any progress on finding a laptop that can be used to demonstrate the installation of BEMOSS.

Thursday, May 09, 2019

I picked up a department laptop from Mr. Mattus today. He cleared the partition completely, so no time had to be spent removing a previous operating system from the machine. Then, I installed Ubuntu 16.04.6 LTS on the system with a bootable USB flash from the link provided on the BEMOSS installation guide.

Wednesday, May 22, 2019

As recommended by Dr. Miah, I worked on running BEMOSS on the previous team's laptop. By running `./startBEMOSS_GUI.sh` inside the directory `/home/bemoss/BEMOSS3.5/GUI`, I was able to start up the BEMOSS Launcher Wizard. By selecting Run BEMOSS in the TKinter GUI the server was started, and I was able to connect to the local web server at `localhost:8082`.. At this point, the software was able to detect the WeMo Insight Switch and thus control the Philips Hue bulb connected to it. I was not able to control the motor due to time constraints, but I will do so soon.

I did some more research on the features that BEMOSS offers at https://github.com/bemoss/BEMOSS3.5/wiki/BEMOSS-Features including the ability to provide local and remote monitoring and security.

Thursday, May 23, 2019

Friday, May 24, 2019

The hierarchy of BEMOSS was researched today with the motivation of understanding the software better in the Developer Resources. The first layer consists of the UI and User Management which reside in the central server. Here admins can manage different nodes on the network and either deny or accept requests from users. Layer 2 is the BEMOSS Application and Data Management Layer which allows developers to create custom applications for target devices that can be added to the UI. It may be interesting to explore this feature once I have gotten a better idea of what sort of original contributions I would like to make to the project. A motivation for this part of BEMOSS is to integrate web services like IFTTT (IF This Then That) which may be interesting to use. The third layer is the operating system and framework layer consisting of the agent platform VOLTTRON developed by the Pacific Northwest National Laboratory. Six different agents perform various different tasks such as detecting new devices (lighting/plug load controllers) on the network and monitoring them to ensure they are running properly. Layer 4 is the BEMOSS connectivity layer that handles the communication between the operating system layer and the physical hardware devices. This is where support is extended to different communication technologies like Wi-Fi, Ethernet, and Serial(RS-485). Each device supported by BEMOSS has an API translator needed to handle the differences in device attributes.

Monday, May 27, 2019

No work was done due to Memorial Day.

Tuesday, May 28, 2019

The outline for the presentation on May 31 follows:

- Introduction
- Applications of BEMOSS
- Hardware/software needed to install BEMOSS
- Immediate future work

The following questions are answered to determine what needs to be added to the presentation:

- What is BEMOSS? BEMOSS or Building Energy Management Open Source Software is an agent-based software platform engineered to allow small- and medium-sized commercial buildings to more seamlessly integrate equipment designed for sensing and control. This software can allow building owners and engineers to manage building energy use better by monitoring different load control devices such as lighting loads, plug loads, and HVAC controllers.
- How can BEMOSS be applied to the real world?
- What hardware and software is needed to install BEMOSS?
- What kind of future work is available to be implemented with this software?

After working on the presentation, I worked in the lab to document the toggling of the WeMo insight switch with BEMOSS. Most of this information was gathered from https://github.com/bemoss/BEMOSS3.5/wiki/User-Guide-for-BEMOSS-UI. Once the BEMOSS server has started, type localhost:8082 in the web browser to go to the BEMOSS Web UI. The username is 'admin' and the password is the one set during installation. To discover the switch, click the "Discover New Devices" tab in the left navigation bar. Under the "All Plug Load Controllers" menu, select either "All Plug Load Controllers" or "Belkin International Inc. Insight." Click "Discover Selected Devices" to complete the process. The number of discovered devices will appear on the Discover New Devices tab. If only one device has been discovered, a 1 will appear next to the name Discover/Manage. On the Discover/Manage page, approve the device by setting the approval status to "Approved" then select "Save Changes to Plugload Controllers." Navigate to the tab NODE1. Select "View All" under "Plugload", then select the WeMo smart plug icon to change the status of the plug and view the power consumption.

Wednesday, May 29, 2019

I spent the first few hours of the day reading through [**?**]. I worked on researching the applications of BEMOSS and the introduction for the May 31st presentation. The future work still needs to be researched and added. Also, I would like to add some pictures to the slides to help the audience members gain a better visual understanding. Ideally, the presentation should be finished tomorrow morning, so I can have more time to practice. I practiced the presentation at the end of the day today without everything completed which isolated my knowledge gaps and gave me a better idea of what I should work on. To provide better flow between slides I must find ways to transition well between them.

Thursday, May 30, 2019

More work was done on the presentation.

Friday, May 31, 2019

Work on presentation and meeting with other members of the Robotics and Mechatronics (RAM) group.

Monday, June 3, 2019

I read through [**?**] and [**?**] to obtain more ideas on original contributions I can make to the project. One thing I found in [**?**] is the use of an induction motor in McNeese State University's microgrid. With a variable frequency drive, this could be integrated with BEMOSS to control different types of industrial loads. One problem is the high price tag on both. In [**?**], a Particle Photon board was used to control the brightness of fluorescent lighting via step-dim ballasts. The Raspberry Pi is definitely a better option here than the Photon board as the previous senior project group used an RPi in their project.

Tuesday, June 4, 2019

Today, I attempted to install BEMOSS on my Ubuntu laptop. After running `./startBEMOSSGUI_.sh` in the GUI directory, I encountered some problems. The following errors were thrown:

```
Traceback (most recent call last):
    File "Web_Server/run/defaultDB.py", line 91, in <module>
        admin = User.objects.get(username='admin')
    File "/home/ramgroup/BEMOSS3.5/env/local/lib/python2.7/
    site-packages/django/db/models/manager.py",
    line 85, in manager_method
        return getattr(self.get_queryset(), name)(*args, **kwargs)
    File "/home/ramgroup/BEMOSS3.5/env/local/lib/
    python2.7/site-packages/django/db/models/query.py",
    line 379, in get
        self.model._meta.object._name
django.contrib.auth.models.DoesNotExist: User matching query does not exist.
OS settings imported

Traceback (most recent call last):
    File "bemoss_lib/databases/cassandraAPI/initialize.py",line 186 in
<module>
        init()
    File "bemoss_lib/databases/cassandraAPI/initialize.py", line 99, in init
        casYamlFile = open(settings.PROJECT_DIR+"/cassandra/conf/cassandra.yaml",'r')
IOError: [Errno 2] No such file or directory:
'/home/ramgroup/BEMOSS3.5/cassandra/conf/cassandra.yaml'
```

I was able to eliminate the first error by using 'admin' for the Django superuser rather than the default 'ramgroup'. I tried to eliminate the second error by deleting BEMOSS3.5 from my home directory and recloning; however, the same problem persisted. I found a directory named `/BEMOSS3.5/cassandra` on the previous group's laptop which is not being created when I run BEMOSS on my machine. This directory contains the file "cassandra.yaml" which the file "initialize.py" is attempting to access. This leads me to believing that there is some issue with the creation of the directory.

Monday, June 10, 2019

More work was done to install BEMOSS. I emailed one of the members of the previous senior project named Bob about the error.

Tuesday, June 11, 2019

After looking through some of the files in `/BEMOSS3.5/GUI/GUI.py`, I found the line of code preventing the installation of BEMOSS which is 108:
`bemoss_is_installed = os.path.isdir(ui_path) and os.path.isdir(cassandra_path) and os.path.isdir(env_path)`
Since the cassandra directory is non-existent, the expression `os.path.isdir(cassandra_path)` evaluates as `False`.

After some further searching I found that the BEMOSS is failing to download and install the cassandra database due to a dead link in the shell script
`/BEMOSS3.5/GUI/bemoss_install_v3.5.sh`
When line 48:
`wget http://downloads.datastax.com/community/dsc-cassandra-3.0.9-bin.tar.gz`
is run, a "404 not found" error is generated by the server. The URL was entered into a web browser and it was found that the requested URL was not found on the server.

As a possible solution, I used a different URL to download the cassandra database in
`/BEMOSS3.5/GUI/bemoss_install_v3.5.sh`
After adding a comment on line 48, the previously mentioned URL on line 49 is changed to
`https://archive.apache.org/dist/cassandra/3.0.9/`
`apache-cassandra-3.0.9-bin.tar.gz`
Lines 49-53 were changed to

```
wget https://archive.apache.org/dist/cassandra/3.0.9/
apache-cassandra-3.0.9-bin.tar.gz
tar -xzf apache-cassandra-3.0.9-bin.tar.gz
sudo rm apache-cassandra-3.0.9-bin.tar.gz
sudo rm -rf cassandra/
sudo mv apache-cassandra-3.0.9 cassandra
```

After this, BEMOSS was successfully installed. The post-installation instructions were followed on the BEMOSS wiki, but, at the end, errors were still being thrown while attempting to get the web server up and running.

Wednesday, June 12, 2019

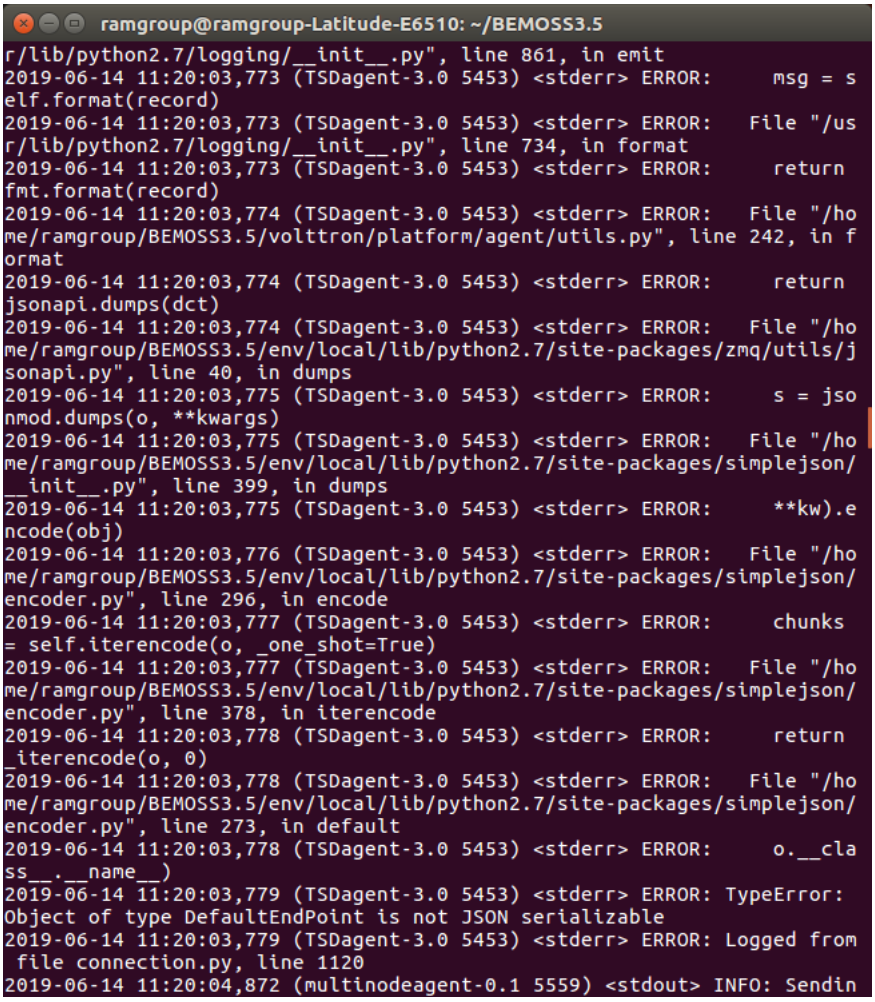
After viewing the issue on the BEMOSS repo: https://github.com/bemoss/BEMOSS3.5/issues/47, I found that the IP address in `parent_ip.txt` did not match the IP of my system, so I changed this to the correct IP. This corrected the problem and the BEMOSS web server was able to boot successfully. Note this text file is only created after BEMOSS is run.

Thursday, June 13, 2019

In the lab, I worked on getting BEMOSS up and running. I mistakenly used the wired connection at first when attempting to run the BEMOSS server but decided to connect to the wireless network ECE-Robotics1 as the Raspberry Pi controlling the motor uses this network. I was able to login as admin into BEMOSS but experienced a problem when attempting to connect to the WeMo Insight switch. When I attempt to navigate to the plug load page to control the WeMo switch the page does not load. It is unclear whether this is an issue with the Insight switch or with BEMOSS itself. I also tried working with the WeMo plug on the previous group's Ubuntu laptop but ran into the same issue leading me to believe it is possibly an issue with the WeMo switch.

Friday, June 14, 2019

More work was done at the beginning of the day to help identify and fix the problem of the Plugload page not loading. After one attempt the page eventually loaded but took a great deal of time. It was finally discovered that the laptop must be connected to the wired network as well as ECE-Robotics1 in order to function properly. Without a wired connection, the PC is unable to connect to the Internet which causes errors. However, although the software was working properly errors were reported by the TSDagent. These are captured in the figure below.



I decided to try the same setup on the previous group's machine to see if I would receive the same problem. After running the software on the previous group's machine, I received the same errors with the TSDagent thus concluding that the proposed solution mentioned on page 31 is not a complete one. I will need to email Ashraf with the details on this.

In the meantime, I will attempt to get the DC motor running with the BEMOSS software. After modifying the file permissions of three of `shell_control.sh` using
`chmod u+x shell_control.sh`
I was able to identify and control the motor using `pyshell_control.py` which rotates the motor counter clockwise then clockwise. Soon I need to start creating the GUI that will show all devices on the network and enable the ability to control them.

Monday, June 17, 2019

Work on the presentation slides for June 21 was done. Further research on the Beamer class was conducted to add more detail to the presentation.

Tuesday, June 18, 2019

The presentation slides are almost complete at this point. A few more captions need to be added to the figures and sources must be added to the bibliography.

Wednesday, June 19, 2019

Work on the presentation was continued and uploaded to github. A few additions may need to be made as I was not able to get a full 10 minutes out of it. Tonight, the presentation will be practiced and polished, so that I am ready to go by Friday. This time I need to know exactly what I am saying before going in so I can avoid any pauses.

A page on wikipedia on computer networks was read:
`https://en.wikipedia.org/wiki/Computer_network`
to better understand what is going on with this project. Knowledge of the python Tk interface must be obtained to build the GUI due June 21.

Thursday, June 20, 2019

Had a short meeting with Dr. Miah in the lab. Here is what needs to be done:

- Add progress and plan to presentation.
- IoT discovery and control GUI and BEMOSS plugload icon must be implemented before June 28. Conference paper must be completed and submitted before June 28. However, this is not likely to be finished by then as little to no progress has been made on the GUI or BEMOSS motor integration.
- Need to start recording hours when working with the DC motor so I can get paid. I need to talk with Mrs. Polen to get an account setup with Bradley.
- I need to start thinking of a device to implement in BEMOSS. Otherwise, this project will not be successful without an original contribution. Thus, a lot of research must be done.
- Agent-based architecture will need to be researched by reading some research papers.
- Need to send Ashraf an email asking if he has made any progress on the project.

  All the scripts written by Reece and Bob to control the motor were understood except for `XBEETEST.py` on the Raspberry Pi as I still need to do research on the XBee modules if I am to use them in the project.

Friday, June 21, 2019

To better understand how to create the GUI to control the devices in the lab, I read up on the documentation for PyGTK at https://python-gtk-3-tutorial.readthedocs.io/en/latest/layout.html#.

Sunday, June 23, 2019

Work was done on researching a device to integrate within BEMOSS. Here are some possible ideas:

- Ultrasonic Range Finding module - hcsr04
  I have one of these and have been programming it some and could potentially have an interesting application for IoT. However, this does seem rather simple and would not likely take long to fully implement.
- Digital Multimeter
- Accelerometer
- Gyroscope
- Dust Sensor `https://www.waveshare.com/dust-sensor.htm`
- PM2.5 Particle Sensor `https://www.cytron.io/p-honeywell-pm2.5-particle-sensor-module`
  Looking at this module it uses a two wire UART output so it would likely be very easy to interface with the Raspberry Pi

Monday, June 24, 2019

While building the GUI, I came to realize that Gtk is simply too complex for me, so I decided to change tkinter which is a bit simpler. At this point I have finished the GUI and simply need to connect the callbacks to the events using tk.widget.bind(event,callback). Once this has been done I will be ready to move on to implementing the logic to control the wemo switch and motor within the tkinter application.

Tuesday, June 25, 2019

To better understand lower level networking concepts, I will use the python module `socket` to ping addresses on the network and resolve their hostnames. It may save time to use the `nmap` command used by the previous group; however, I would like to build a system from scratch completely in python. As the motor needs to be implemented within BEMOSS as soon as possible, I will work on this first and determine how to use the wemo switch later. I will base some of my work off [**?**]. However, after working for some time I found that the program I was attempting to write was rather inefficient and using the nmap command will be much more faster. Thus I have decided to use the scripts written by Bob and Reece. I was able to write a single python script using the socket module to parse through all hosts on the network 'ECE-Robotics1' and place them in a list along with their respective IP addresses.

Wednesday, June 26, 2019

I was able to succesfully add the Raspberry Pi and Wemo switch names to the listbox; however if the button 'Discover IoT Devices' is pressed continually, devices will continually be added to the list. Right now I need to determine how to initialize the Raspberry Pi by sshing into it. Then, after selecting the toogle button, I must figure out how to remotely send commands to the device to turn it on or off without having to reconnect. After some tinkering I found that I can simply just ssh into the pi and run a script that simply turns the moltor on when the button is toggled on and off when the button is toggled off. Thus no initializatin will be needed.

I need to write scripts to perform the following operations:

- Use nmap to the scan wifi credentials
- Place the credentials into text file
- Scan for the IP addresses and place them into a text file
- Assign the address read from the file to a variable and use this to remotely login to the device (for the RPi)

However, I have almost no knowledge of bash so this will take some researching. I found a way of storing the first IP address in `IPAddresses.txt` in a variable that can be used to call the python scripts running on the pi to control the motor. Each time the toggle button is pressed, the text file is read from which is inefficient. More ideally I would like to implement some feature where the IP addresses of the devices are stored in a place that can be accessible to any shell script within the directory. I would like to do this later on; however, I need to move on to working the Wemo switch's API into this application.

To perform the implementation of the WeMo switch, I will need to read through the documentation provided by the BEMOSS team and some of the python code to find exactly what code needs to be written to create a fully functional system. This code is very complicated thus it may take a great deal of time to work through.

I found a url on the bemoss website: `www.bemoss.org/api-interface-wemo-smart-plug/` that explains how some of the code works. The switch uses the upnp (Universal Plug and Play) protocol.

Thursday, June 27, 2019

A github gist was located that contains a script to control a wemo device: https://gist.github.com/pruppert/af7d38cb7b7ca75584ef. I was able to successfully control the wemo switch with this. This will be helpful in understanding the code provided in the bemoss repo and on the bemoss website. The url to send commands to the switch is `http://192.168.1.112:49153/upnp/control/basicevent1`
I attempted to add this url into the method `getDeviceStatus`; however when I run the code I receive a 500 response (internal server error). After running an nmap scan with the port scan enabled I found that the wemo switch uses both ports 53 and 49153 for communication, so the cause of the problem here is unknown. Later, I determined the problem is that no XML was being sent in the body of the POST request to the device. I simply copied the XML from the python script I found online to control the switch and modified the code from the bemoss website. This is the function used to turn on the Insight switch.

```
def turnOnSwitch():
    header = {
        'Content-Type': 'text/xml; charset="utf-8"',
        'SOAPACTION': '"urn:Belkin:service:basicevent:1#SetBinaryState"'
    }
    body='<?xml version="1.0" encoding="utf-8"?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<s:Body><u:SetBinaryState xmlns:u="urn:Belkin:service:basicevent:1">
<BinaryState>1</BinaryState>
</u:SetBinaryState></s:Body></s:Envelope>'
    controlUrl='http://192.168.1.112:49153/upnp/control/basicevent1'
    response = requests.post(controlUrl, body, headers=header)
    del response
```

To understand how this code works more research will need to be done as I know very little about XML.

Friday, June 28, 2019

Layer 1 (UI layer) of the BEMOSS hierarchy was researched for a few hours to try and understand how new devices are to be added to BEMOSS. Details such as how the Model-Message-View-Template works and the UI project structure are provided. However, this doesn't really provide any details on what happens when an element in the UI is selected such as pressing a button to toggle a device on and how the message flows down to the device. In other words, I'm not able to understand the chain between the UI layer and the device as this is not documented on the BEMOSS wiki. A single file I found that could be possibly helpful is `/BEMOSS3.5/Web_Server/webapps/device/templates/plugload/plugload.html`. This will require some digging to understand the html, css, and jQuery as I have little to no experience in any of these. In addition, I may need to research the agent based system to understand how these agents interact with the devices' APIs.

Monday, July 1, 2019

To understand how the device discovery agent works, I studied each line of `BEMOSS3.5/Agents/DeviceDiscoveryAgent/devicediscovery/agent.py` carefully. In `BEMOSS3.5/BEMOSS_lib/db_helper.py` a class named db_connection is defined with method database_connect that reads the system's ip address from `parent_ip.txt` and passes it as an argument into the method psycopg2.connect in order to connect to the PostgreSQL database named 'bemossdb'.