



Laboratory Notebook

Development of an Intelligent Building Energy Management System

Brian Lauer and Elliot Watkins

blauer@mail.bradley.edu, ejwatkins@mail.bradley.edu

Beginning September 14, 2020

Contents

Monday, September 14, 2020	4
1 Meeting Minutes	4
2 Homework	4
Monday, September 21, 2020	5
1 Meeting Minutes	5
Tuesday, September 22, 2020	7
1 Lab work	7
Thursday, September 24, 2020	9
Monday, September 28, 2020	10
1 Meeting Minutes	10
Tuesday, September 29, 2020	11
1 Lab work	11
Thursday, October 1, 2020	12
1 Lab work	12
Monday, October 5, 2020	14
1 Meeting	14

r.

Monday, September 14, 2020

1 Meeting Minutes

BL

Today, in the meeting with Dr. Miah, we kicked off the project involving development of a building energy management platform. A Github repository was created for the project which will also be paired with a corresponding Google Drive titled "seniorProject1-2020-21". On Tuesdays and Thursdays, we will have lab time from 8am to 11am and weekly meetings with the advisor from 4pm to 5pm on Mondays this semester. The lab times are devoted specifically for work rather than research or documentation. Before coming to lab, work must be cut out for both of us.

2 Homework

BL

This document was created and pushed to the Github repository for both of us to use. For better portability, one of the laptops previously used for research on this project was wiped clean and LUbuntu was installed. As of now, most of the code developed for the platform was done on a desktop PC dual-booted with Ubuntu Linux and Windows 10. To be able to move between lab and home easily, a more ideal situation is to use a dedicated laptop with Linux installed. Because the USB drive used to install Linux was previously formatted with Ubuntu 18.04, the diskpart utility was used with the following commands:

```
select disk 1
clean
create partition primary
format quick
```

The Startup Disk Creator GUI program on Linux was utilized to flash the LUbuntu 64-bit desktop iso file on to the USB-thumb drive. The commands `sudo apt-get update` and `sudo apt-get upgrade` were run on the laptop to perform necessary upgrades and updates. Lastly, `sudo apt-get install git` was used to install git. Another package installed was `textlive-full` to compile latex with all the necessary packages.

Monday, September 21, 2020

1 Meeting Minutes

BL

In the meeting with Dr. Miah, Dr. Miah pointed out that templates are available in the Google Drive for the lab notebook and other deliverables which we did not realize. In the System Level Requirements document (the first deliverable) we must replace the current block diagram with the drawing of the house connected to a microgrid. Inside, the home will be a block of the BEMS core which will connect to various IoT devices in the building. The controllable loads in the building including devices like lights, appliances (including dishwashers, washing machines), and smart plugs. Examples of uncontrollable loads include desktop PCs, laptops, thermistors, and microwaves. In a potential future research project, connectivity with a microgrid may be possible, so powerlines representing the main grid will be connecting the house to a microgrid through a PCC (Point of Common Coupling). In the current version of the diagram, the file is simply too large as it contains many images from the Internet. This will be redrawn in inkscape with custom figures to reduce the file size.

The final outcome of the project will be

- Laptop running in the IoT lab or office
- 2-3 devices in office/lab can be controlled successfully
- System should recognize the devices immediately upon connection to the network
- Algorithm implemented to help improve energy efficiency

As a part of the first deliverable, three modes of operation will be declared including

- Device search mode
- Device operation mode
- Third mode to be determined

The sections of the deliverable will be

- Problem Statement
- System Architecture

- Block Diagram
- Modes of Operation

Another point brought up is whether an energy efficient algorithm should be included in the software. Dr. Miah said yes. This will possibly help to boost the complexity of the project and improve the usefulness of the platform. This algorithm will have to be determined.

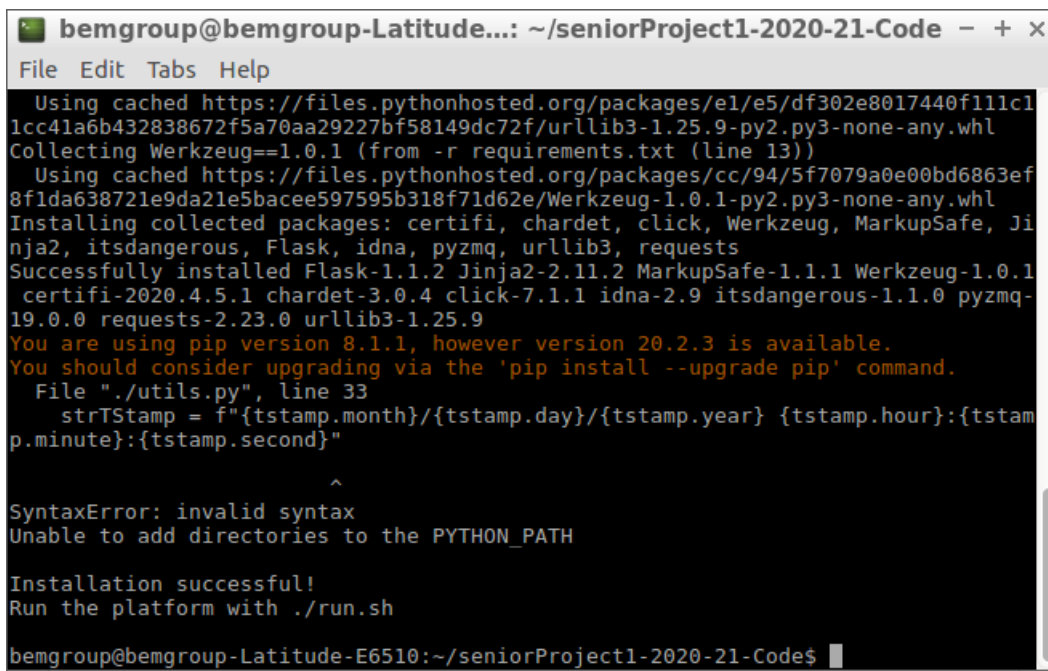
As of today, we should both be writing code on Tuesdays and Thursdays or possibly on our own, and we can both use code written for the research project. The split should be 50/50 so a Gantt chart will need to be made.

Tuesday, September 22, 2020

1 Lab work

BL

Elliot and I met over Zoom to work on getting a new Github repo setup for the senior project code specifically. Although Dr. Miah mentioned in the meeting that the code should be on Github, we simply will periodically upload the zipped Github project to the Drive to the implementation/ folder. I copied over the files from the research to the new Github repo cloned locally on my Linux laptop. After running the `install.sh` script, the installation failed as the Linux laptop I am using is running Python3.5 rather than Python3.6 which is required for f-string support. An example of a line that failed is shown in the figure below:



```

bemgroup@bemgroup-Latitude...: ~/seniorProject1-2020-21-Code - + x
File Edit Tabs Help
Using cached https://files.pythonhosted.org/packages/e1/e5/df302e8017440f111c1
1cc41a6b432838672f5a70aa29227bf58149dc72f/urllib3-1.25.9-py2.py3-none-any.whl
Collecting Werkzeug==1.0.1 (from -r requirements.txt (line 13))
Using cached https://files.pythonhosted.org/packages/cc/94/5f7079a0e00bd6863ef
8f1da638721e9da21e5bacee597595b318f71d62e/Werkzeug-1.0.1-py2.py3-none-any.whl
Installing collected packages: certifi, chardet, click, Werkzeug, MarkupSafe, Ji
nja2, itsdangerous, Flask, idna, pyzmq, urllib3, requests
Successfully installed Flask-1.1.2 Jinja2-2.11.2 MarkupSafe-1.1.1 Werkzeug-1.0.1
certifi-2020.4.5.1 chardet-3.0.4 click-7.1.1 idna-2.9 itsdangerous-1.1.0 pyzmq-
19.0.0 requests-2.23.0 urllib3-1.25.9
You are using pip version 8.1.1, however version 20.2.3 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
File "./utils.py", line 33
    strTimestamp = f"{tstamp.month}/{tstamp.day}/{tstamp.year} {tstamp.hour}:{tstam
p.minute}:{tstamp.second}"
                    ^
SyntaxError: invalid syntax
Unable to add directories to the PYTHON_PATH

Installation successful!
Run the platform with ./run.sh

bemgroup@bemgroup-Latitude-E6510:~/seniorProject1-2020-21-Code$

```

Figure 1: *fstring error in utils.py*

A simple solution to this problem was simply using string concatenation:

```
strTimestamp = str(tstamp.month) + "/" + str(tstamp.day) + "/" + str(tstamp.year) + " "
```

```
strTStamp += str(tstamp.hour) + ":" + str(tstamp.minute) + ":" + str(tstamp.second)
```

Further changes were made in the `ControlAgent.py` and `DiscoveryAgent.py` files. In particular, the line in `subscribe` to connect to the server acting as the central exchange to process publish/subscribe messages between the web server backend and agents listening for subscriptions was altered to support string concatenation. After these alterations to the source code were made, the platform was successfully started. However, it is clear that no messages published to the server were received by both either the `ControlAgent` or `DiscoveryAgent` which will need to be debugged in the next lab or possibly before. This is evident as the terminals showing the debug output of each agent not indicating any messages received. Messages should be published via the `publish` function in the file `PROJECT_DIR/WebServer/pubsub.py`. All these problems must be fixed to allow the `ActiveDevices` page to discover and connect to devices.

Thursday, September 24, 2020

BL

Today, I debugged some of the problems with the ControlAgent agent not receiving messages. The agent was run in isolation and tested with the publish function in the pubsub.py file. The agent was able to successfully receive messages. However, this was not tested while the platform was running which will need to be done in the next lab session.

I (Elliot) started to create a device driver for the BeagleBone Blue based off of the WeMo Switch driver. In doing so, I also started researching ways to control individual pins via wifi, starting with the GPIO.

Monday, September 28, 2020

1 Meeting Minutes

EW

Agenda:

How to auto-detect devices on the network (interrupt or polling)? We were thinking of constantly pinging the network for supported devices in an infinite loop. Is there a way for a device to send a request to the server once it has joined the network?

We would like to add support for a smart power meter that can connect to WiFi, so power data for the entire building/house can be shown in the UI. What would you recommend in terms of a cost-effective solution? I found a device called the Sense Energy Monitor that can send data over wifi to an mobile or web app that comes with the device. The device itself connects to the electrical panel of a house. However, it is quite expensive at a price of 299 dollars.

Does the platform still need to be agent-based like BEMOSS?

Discussion: For auto-detecting, we need to research how to create an interrupt. IF that turns out to be impossible, we will try to replicate Windows trying to find bluetooth devices. It will be a polling process that is explicitly demanded by the user.

MATLAB can be used to measure power usage. Simscape is an addon that includes a power sensor module. After checking online, we also discovered it is possible to connect MATLAB to a website and transfer data. With this, we should be able to plot power usage on the UI.

Finally, Dr. Miah informed us that the project will indeed be agent-based.

Helpful Links:

<https://drive.google.com/file/d/1jx7SldW3zNLdA5WBNl89LUBIKoKQuEMI/view?usp=sharing>

<https://www.codeguru.com/> <https://smile.amazon.com/Bluetooth-Voltmeter-Multimeter-Resistance-Impedance/dp/B07PZRSYXD/>

Tuesday, September 29, 2020

1 Lab work

BL

Today, work was done on getting the linux laptop connected to the "AERO" network in the senior project lab, so that experiments can be done properly with the WeMo Switch and eventually Beaglebone Blue. Because this was not working, Elliot set up a hotspot on his laptop which was tethered through BUSecure. Secondly, Elliot paired the WeMo Switch with the WiFi network in the lab with his phone as I did not remember the password to my WeMo account, so a new one was made. A couple of bugs were still present in the current version of the software. One of them was causing the ControlAgent to crash everytime a publish message was attempted to be sent through interprocess communication. We found the problem was due to the following error

```
can't convert type 'dict' to 'str' implicitly
```

This was fixed by using `json.dumps` in the `pubsub.publish` function. At the end of the lab session, we were able to succesfully toggle the WeMo Switch on and off through our platform. One of the things Elliot pointed out was the fact that the checkbox toggle button for turning the device on and off is not set to the proper position when the page is loaded as it is always in the off position. We decided this feature will need to be added soon as it had not been added.

Thursday, October 1, 2020

1 Lab work

BL

We decided to write down the trace of commands/function calls that occur from the top layer (UI layer) all the way down to the bottom layer (device API layer) to turn on/off the Switch:

- Main active devices page: `PROJECT_DIR/WebServer/templates/active_devices.html`
- JS code running on `active_device.js` calls `$.ajax()` which sends an AJAX request to the url `'/active_devices/ajax/setDeviceStatus'`
- Sending a POST request to URL `'/active_devices/ajax/setDeviceStatus'` calls `sendDeviceStatusToControlAgent()`
- In subscribe of `PROJECT_DIR/AgentPlatform/ControlAgent.py` a call to `processMethod()` occurs which calls `setDeviceStatus()`
- `setDeviceStatus()` calls `WeMoAPI.setState`
- `PROJECT_DIR/DeviceDrivers/WeMoAPI.py` `setState()` function sends an XML SOAP request over HTTP to the WeMo Switch

In order to display the correct device status on the web server via the `getDeviceStatus()` method in the `ControlAgent` class:

- `getState` will retrieve the status using XML SOAP requests
- `renderActiveDevices()` will be called in `active_devices.py`. We would like to eventually poll for a state change (Not necessarily this function). Also, we want to use the most recent timeseries data from a table.
- First we will need to use `getState()`, but eventually we want to use a data table. Also, we need to research interrupts.

I suggested installing the Cassandra database in the software as a first step towards getting the status updated on the web page as it can store time series data easily. The instructions for install Cassandra are listed below taken from ¹

¹<https://cassandra.apache.org/download/>

1. `echo "deb https://downloads.apache.org/cassandra/debian 311x main" | sudo tee -a /etc/apt/sources.list.d/cassandra`
2. `curl https://downloads.apache.org/cassandra/KEYS | sudo apt-key add -`
3. `sudo apt-get update`

We encountered the following error which was documented on the site

```
GPG error: http://www.apache.org 311x InRelease:
The following signatures couldn't be verified
because the public key is not available: NO_PUBKEY: E91...
```

The PUBKEY is extracted from the error message itself and must be added with `sudo apt-key adv -keyserver pool.sks-keyservers.net -recv-key E91..`

Lastly the software was installed with `sudo apt-get install cassandra`. To interface with the software with Python, we decided to use the DataStax Python driver for Apache Cassandra. It was installed with the following commands

1. `source venv/bin/activate`
2. `pip3 install cassandra-driver`

Monday, October 5, 2020

1 Meeting

BL

Agenda:

- What is a reliable and efficient way to obtain the status of the WeMo and update the checkbox on the web page? This is needed in case someone turns the WeMo on and off manually.
- Is a Gantt chart required for the first deliverable?
- Could you review what we have written so far for the first deliverable?
- We are considering integrating the DC motor with the software through the BeagleBone Blue. As an extension from the project in 2019 would we be able to add a speed control algorithm to use PWM to control motor speed? If so, should we develop our own or use the one from your mechatronics textbook?