

A Generalized Open Source Platform for Building Energy Management

Brian Lauer
Advisor: Dr. Suruz Miah

Department of Electrical and Computer Engineering
Bradley University
1501 W. Bradley Avenue
Peoria, IL, 61625, USA

Friday, April 3, 2020

Outline

- 1 Objectives
- 2 ESP8266 MCU Board
- 3 ESP8266 MCU Board
- 4 WeMo Switch Discovery and Control
 - Explanation of Discovery
- 5 WeMo Switch Control
 - Explanation of Control
- 6 Front End Work
- 7 Front End Work
- 8 Objectives for Coming Weeks

Objectives

- Write networking code for communicating with the DC motor and WeMo switch
- Build simple login page

ESP8266

- Made the decision to control DC motor with only a ESP8266 Node MCU board
- Plan to swap out Raspberry Pi and XBee module



Figure: ESP8266 Node MCU Lua V3 courtesy of ebay.com

- Wrote a primitive Python TCP/IP web server to run on the board
- File must be named `main.py` to allow execution on start up
- Continually accepts connections from clients connected to the same network (`server_socket.accept()`)
- Continually accept commands from remote clients (`client_socket.recv(BUFFER_SIZE)`)
- 'ON' toggles a GPIO pin on while 'OFF' toggles the GPIO pin off
- PWM value can be set on a separate pin using the format 'PWM: value' where value is an integer
- Could be useful in controlling the speed of the motor

WeMo Switch Discovery and Control

Explanation of Discovery

- Gained a better understanding of the code in the BEMOSS repository for controlling the WeMo switch
- SSDP (Simple Service Discovery Protocol) utilized for discovery of the switch and other UPNP (Universal Plug and Play) devices
- UDP socket is created and a HTTP request is sent over UDP to address "239.255.255.250", 1900
- HTTP method: 'M-SEARCH', url: '*', version: HTTP/1.1

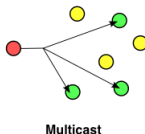


Figure: Courtesy of
<https://williamboles.me/discovering-whats-out-there-with-ssdp/>

WeMo Switch Discovery and Control

Explanation of Discovery

- Headers of multicast http request include
- HOST - where the message will be sent ("239.255.255.250", 1900)
- MAN - message type which is always ssdp:discover
- ST - search type, in the case of WeMo switch is "upnp:rootdevice"
- MX - time (seconds) a root device can take before responding

```
M-SEARCH * HTTP/1.1  
HOST: 239.255.255.250:1900  
MAN: ssdp:discover  
ST: upnp:rootdevice  
MX: 3
```

Figure: Request used in WeMo API in BEMOSS

WeMo Switch Discovery and Control

Explanation of Control

- Responses returned by available UPNP devices contain a header location which specifies location of the XML file listing services available and metadata
- WeMo switch contains file called `setup.xml`
- XML file is requested and parsed in Python using `xml.dom.minidom` module
- Tags like `modelName`, `manufacturer`, `serialNumber` and `deviceType` can be used for further identifying the device

WeMo Switch Discovery and Control

Explanation of Control

- Once address of device is found request can be made
- Body must be created with binary state and or brightness depending on model of switch



```
Open ▼  *Untitled Document 1  Save  ≡  
<?xml version='1.0' encoding='utf-8'?>  
<s:Envelope xmlns:s='https://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>str(int(status))</BinaryState>  
    .....</u:SetBinaryState>  
  ....</s:Body>  
</s:Envelope>
```

Figure: Body of request

Front End Work

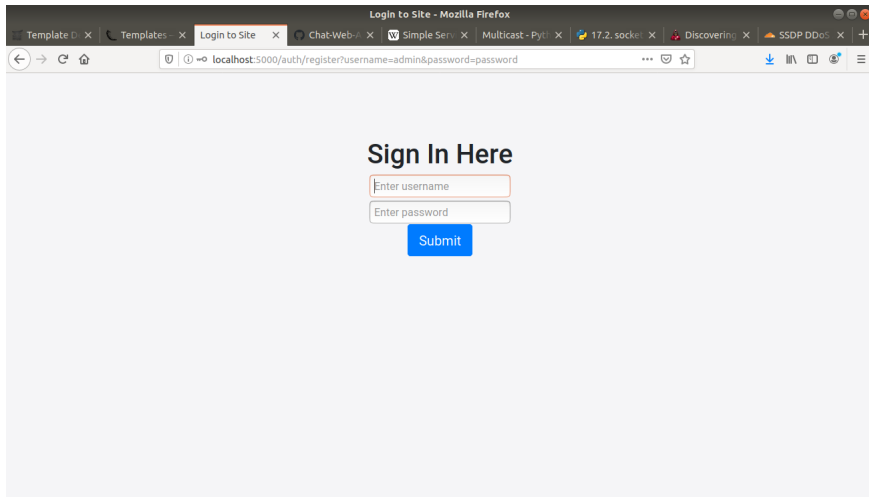


Figure: Prototype login page for BEMS

Front End Work

```
ramgroup@ramgroup-Latitude-E6510: ~/myproject
File Edit View Search Terminal Tabs Help
ramgroup@ramgroup-Latitude-E6510: ~/... x ramgroup@ramgroup-Latitude-E6510: ~/... x
No hello
127.0.0.1 - - [03/Apr/2020 14:29:47] "GET /auth/register?username=&password= HTTP/1.1" 200 -
No hello
127.0.0.1 - - [03/Apr/2020 14:29:51] "GET /auth/register?username=&password= HTTP/1.1" 200 -
127.0.0.1 - - [03/Apr/2020 14:29:51] "GET /static/css/bootstrap.min.css HTTP/1.1" 304 -
No hello
127.0.0.1 - - [03/Apr/2020 14:29:51] "GET /auth/register?username=&password= HTTP/1.1" 200 -
No hello
127.0.0.1 - - [03/Apr/2020 14:30:10] "GET /auth/register?username=asd&password=asd HTTP/1.1" 200 -
No hello
127.0.0.1 - - [03/Apr/2020 14:44:01] "GET /auth/register?username=asd&password=asd HTTP/1.1" 200 -
No hello
127.0.0.1 - - [03/Apr/2020 15:21:08] "GET /auth/register?username=blauer&password=asdf HTTP/1.1" 200 -
No hello
127.0.0.1 - - [03/Apr/2020 15:21:46] "GET /auth/register?username=admin&password=password HTTP/1.1" 200 -
```

Figure: Flask debug console capture

Objectives for Coming Weeks

- Create plan for software development
- Setup SQLite database for users and devices
- Fix problem with login page
- Read papers on agent based architecture

Any Questions?