e-Yantra Summer Internship-2015

IoT Connected valves for irrigation of greenhouse

QUICK INSTALLATION GUIDE

Contents

1 Hardware

1.1 ESP8266 WiFi module

ESP8266 is a low power WIFI module which is very much suitable for our irrigation application. Some of the important features of the ESP8266 are:

- Its an ultra low power module which works on 3.3V. This is ideal for IoT applications
- It has high on-chip integration
- It has GPIO support for the application devices that you want to control
- It has a 32 bit CPU
- It is very small in size
- It is a low cost module
- It can act as a standalone device without the need of any external controller

We chose to go with ESP8266 over CC3200 because the ESP module had all the functionalities that we needed and was cheaper, lighter and better for IoT applications.

1.2 IDE for ESP8266

1. ESP8266 comes with the NodeMCU firmware loaded and it accepts LUA script which can be loaded into the board using ESplorer IDE, but the use of this IDE lacks proper documentation.
2. Download ESplorer from the website ESPlorer
3. LUA is an embeddable, fast, powerful and light script. This is useful in data description and has procedural syntax. It also has some other important features like: Automatic memory management and Garbage collection. Here's LUA API
4. One other way of loading code into the ESP is using the Arduino IDE.This directly loads code into the EEPROM and the NodeMCU firmware can be loaded back in the ESP if needed. Using arduino
5. Clone arduino IDE fro ESP-Arduino Github repository and follow the installation procedure. Arduino. **But we prefer to inbuilt nodeMCU with ESPlorer IDE.**

1.3 Installation procedure:

- Download nodeMCU latest firmware from the link above
- Download ESPTOOL
- Move to the directory containing esptool and execute the following command

  ```
  sudo python esptool.py --port /dev/ttyUSB0 write_flash 0x00000
  The_Path_To_The_NodeMCU_Firmware.bin
  ```

- Make sure you are connected to the right port and change can access it. Else change ownership using

  ```
  Sudo chown $username /dev/ttyUSB*
  ```

note: Make sure GPIO 0 is connected to ground while flashing

1.4 ESPlorer IDE

- The ESPlorer IDE is a great tool to work with the ESP8266 making the loading of code into the ESP an easy task
- Download ESPlorer here ESPlorer
- ESPlorer is a platform that helps you load lua scripts into the ESP. follow this for more help webserver_example
- For the GPIO mapping refer this gpio_map
- Refer this for some more documentation regarding nodeMCU nodeMCU

1.5 Troubleshooting

- ESPlorer needs JAVA 7 or higher. install JAVA 8 for ubuntu by following these steps JAVA 8. After downloading ESPlorer make your way to the folder in which ESPlorer.jar exists and run this command in the terminal >`java -jar ESPlorer.jar` and your done
- esptool depends on pySerial for serial communication with the target device.

If you choose to install esptool system-wide by running `python setup.py install`, then this will be taken care of automatically. If not using `setup.py`, then you'll have to install pySerial manually by running something like

`pip install pyserial`, `easy_install pyserial` or `apt-get install python-serial` depending on your platform. The official pySerial installation instructions are here). This utility actually have a user interface! It uses Argparse and is rather self-documenting. Try running `esptool -h`. Or hack the script to your hearts content. The serial port is selected using the `-p` option, like `-p /dev/ttyUSB0` (on unixen like Linux and OSX) or `-p COM1` (on Windows). The perhaps not so obvious corner case here is when you run esptool in Cygwin on Windows, where you have to convert the Windows-style name into an Unix-style path (`COM1` -> `/dev/ttyS0`, and so on).

2 Server Installation

2.1 Ubuntu

1. Have Ubuntu image in pen-drive or a cd
2. Take care to disable secure boot before installation
3. After installation if windows doesn't show in the boot options then repair the GRUB from UBUNTU using *boot repair*.Have boot repair in a disc or pen-drive.
4. If you use face further problems use GRUB customizer from UBUNTU.Install Grub customizer from software centre.
5. Once Windows shows in boot options then change the Boot preference order from windows by going to the bios setup.
6. In Sony systems while booting use *ASSIST* key

for further information look up

Ubuntu Insatllation

2.2 Installation of Apache2, PHP and MySQL (LAMP-server)

<u>About LAMP</u>

LAMP stack is a group of open source software used to get web servers up and running. The acronym stands for Linux, Apache, MySQL, and PHP. Since the virtual private server is already running Ubuntu/openSuse, the linux part is taken care of. Here is how to install the rest.

2.2.1 Install Apache

To install apache, open terminal and type in these commands:

For Ubuntu

```
sudo apt-get update
sudo apt-get install apache2
```

For openSuse

```
sudo zypper in apache2
```

Firewall Adjustments

In openSuse, by default the firewall configuration blocks all traffic coming on port 80 to your machine. So if you need to allow access so that the web server can be accessed from within a LAN we need to fine tune the firewall configuration. The below step needs to be performed as root user. The supplied configurations are called apache2 and apache2-ssl. They can be enabled via YaST, by adding them to FW_CONFIGURATIONS_EXT in the following path /etc/sysconfig/SuSEfirewall2

```
sysconf_addword /etc/sysconfig/SuSEfirewall2 FW_CONFIGURATIONS_EXT
apache2 sysconf_addword /etc/sysconfig/SuSEfirewall2 FW_CONFIGURATIONS_EXT
apache2-ssl rcSuSEfirewall2 restart
```

Starting Server

Start the server and configure it to automatically start at boot time.

```
rcapache2 start chkconfig -a apache2
```

2.2.2 Install MySQL

MySQL is a powerful database management system used for organizing and retrieving data

To install MySQL, open terminal and type in these commands:

For Ubuntu

sudo apt-get install mysql-server libapache2-mod-auth-mysql php5-mysql

For openSuse

```
sudo zypper install mysql-server
```

To start the server

```
sudo systemctl start mysql.service
```

2.2.3 Install PHP

PHP is an open source web scripting language that is widely use to build dynamic webpages.

To install PHP, open terminal and type in this command.

For Ubuntu

```
sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt
```

For openSuse

```
sudo zypper in php5 php5-mysql apache2-mod_php5
```

Installing phpMyAdmin

For Ubuntu

```
sudo apt-get install phpMyAdmin
```
For OpenSuse
```
sudo zypper in phpMyAdmin
```

Finally restart apache so that all of the changes take effect:
For Ubuntu
```
sudo service apache2 restart
```
For openSuse
```
sudo systemctl start apache2.service
```

3 MOSCA broker

3.1 Broker and nodejs

Based on their use, there are two widely available MQTT brokers available on the web.

1. Mosquitto
2. Mosca

We chose Mosca because it is:

1. MQTT 3.1 compliant.
2. supporting various storage options for QoS 1 offline packets, and subscriptions.
3. As fast as it is possible.
4. Usable inside ANY other Node.js app.

Installation on a OpenSuse system (version 13.1)
Part 1 First install nodejs and npm if both are absent in your system.
Nodejs

```
sudo zypper install nodejs
```

npm

```
sudo zypper install npm
```

If everything goes right (no errors) then proceed to second part.

Part 2
Installation of Mosca

```
npm install Mosca
```

If everything goes write then you are good to go.

To start the broker >`mosca -v --host yourhostipaddress |bunyan`
Installation on an Ubuntu system (version 14.4)
Part 1 First install nodejs and npm if both are absent in your system.
Nodejs

```
sudo apt-get update
```

```
sudo apt-get install nodejs
```

```
sudo ln -s /usr/bin/nodejs /usr/bin/node
```

npm

```
sudo apt-get install npm
```

If everything goes right (no errors) then proceed to second part.

Part 2
Installation of Mosca

```
sudo npm install debug
```

`sudo npm install mosca bunyan -g` //for installing mosca globally

```
sudo npm install daemon
```

If everything goes write then you are good to go.

To start the broker >`mosca -v --host yourhostipaddress |bunyan`

3.2 Testing the broker

First to test the mqtt broker, we need to install a mqtt client on the system.

```
sudo npm install mqtt -g
```

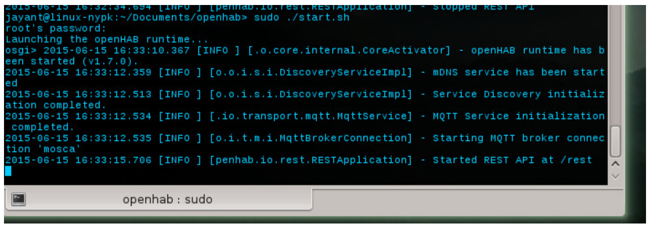
Second, creating scripts for testing
Script for starting MQTT broker service, **mosca-app.js**
To start the service type >`node mosca-app.js`

Script for publishing, **client-pub.js**
Open a new Terminal and Execute above MQTT Subscription Client client-sub.js >`node client-pub.js`

Script for subscribing, **client-sub.js**
Now open a differnent Terminal and Execute above MQTT Subscription Client client-pub.js >`node client-sub.js`

Successful Output From client-sub.js terminal

```
Hello!
```



```
2015-06-15 16:32:34.694 [INFO ] [penhab.io.rest.RESTApplication] - Stopped REST API
jayant@linux-nypk:~/Documents/openhab> sudo ./start.sh
root's password:
Launching the openHAB runtime...
osgi> 2015-06-15 16:33:10.367 [INFO ] [.o.core.internal.CoreActivator] - openHAB runtime has b
een started (v1.7.0).
2015-06-15 16:33:12.359 [INFO ] [o.o.i.s.i.DiscoveryServiceImpl] - mDNS service has been start
ed
2015-06-15 16:33:12.513 [INFO ] [o.o.i.s.i.DiscoveryServiceImpl] - Service Discovery initializ
ation completed.
2015-06-15 16:33:12.534 [INFO ] [.io.transport.mqtt.MqttService] - MQTT Service initialization
 completed.
2015-06-15 16:33:12.535 [INFO ] [o.i.t.m.i.MqttBrokerConnection] - Starting MQTT broker connec
tion 'mosca'
2015-06-15 16:33:15.706 [INFO ] [penhab.io.rest.RESTApplication] - Started REST API at /rest
```

Figure 1: MOSCA start

3.3 Troubleshooting

If while installing mosca there are errors in fetching the links from server, do any of the following

- Check that nodejs version is not ‘pre’, if it is then update it to a stable non pre version.
- reinstalling nodejs and npm in root mode.
- restart your system.

While executing those scripts, if any of the nodejs script file shows error **Mqtt module not found**, then copy the scripts to a folder inside mosca directory, and try executing from there.

For **openSuse** users, open firewall in yast2 GUI. Then go to allowed service, then go to to advance option, there enter 1883 inside tcp port.

After successfully installing MOSCA broker on the system, you are now good to go on next step.

3.4 Running MOSCA

3.4.1 Configuring IP on which MOSCA has to be run.

*on the terminal type >`mosca -help`, it will display all the handles which mosca currently supports we are going to use handles, **-v, –host, | bunyan**. To start the server,on the terminal type >`mosca -v --host 'ur ip' |bunyan`.*
Server testing can be done via two codes included in the software folder.

NOTE: for openSUSE users, make sure that entry for tcp port 1883 is in the firewall for inbound connection.

- to run the code, first run client-sub.js on a different terminal, and then run client-pub.js on a second terminal.
- message from the topic subscribed will be printed where client-sub.js has been run.

3.4.2 Interfacing MOSCA and ESP

- Install MOSCA on your system
- Setup the server using *mosca -v –host $hostname | bunyan*
- Burn the LUA scripts on the ESP8266
- send subscribe request from your terminal using >`node client-sub.js` (go to the apprpriate folder first)
- send publish command from the terminal >`node client-pub.js`
- see the sent data on your terminal with confirm message

3.5 Troubleshooting

3.5.1 ESP8266

- take care to use the delays at appropriate places so as to send out publish and subscribe requests
- LUA syntax must be kept in mind.
- Use >`tmr.alarm` instead of >`tmr.delay` wherever possible

3.5.2 MOSCA server

- The local IP must be changed in the the ESP and also the >`client-pub.js` and >`client-sub.js` scrits
- The topic must be the same in the >`client-pub.js` and subscribe code in the ESP

4 Setting up the website

UI part consist of PHP core, MySQL and phpMQTT library(SSKAJE)

4.1 Database

Table Description Create database named 'iot', and above tables with below mentioned format.
Database name: iot
Tables in database:

- devices
- groups
- sensors
- tasks

Device table
Name: Devices

Table 1: Devices table

FIELD	TYPE	NULL	KEY	Default	EXTRA
id	int(255)	NO	PRI	NULL	auto_increment
name	varchar(255)	YES		NULL	
macid	varchar(30)	NO		NULL	
group	varchar(255)	YES		NULL	
status	int(1)	YES		NULL	
battery	int(1)	YES		NULL	
timestamp		NO		CURRENT_TIMESTAMP	
action	varchar(255)	YES		NULL	

Group Table
Name: groups

Table 2: Groups table

FIELD	TYPE	NULL	KEY	Default	EXTRA
id	int(10)	NO	PRI	NULL	auto_increment
name	varchar(255)	NO		NULL	

Sensor Table
Name: sensors

Table 3: Sensors Table

FIELD	TYPE	NULL	KEY	Default	EXTRA
id	int(10)	NO	PRI	NULL	auto_increment
name	varchar(255)	NO		NULL	

Tasks Table

Name: tasks

Table 4: Tasks Table

FIELD	TYPE	NULL	KEY	Default	EXTRA
id	int(10)	NO	PRI		auto_increment
item	varchar(255)	NO		NULL	
start	int(4)	YES		NULL	
stop	int(4)	YES		NULL	
action	int(1)	YES			

4.2 PHP core

After setting up the database and its tables, copy IOT folder inside Software folder of the repo to your htdocs folder(linux). There are few files which needs to be run in background for data gathering, such as device discovvery, battery status and moisture value.

1. Device discovery –> `subscribe.php`
2. Valve and other sensors scheduling –> `tasks.php`
3. Listening for battery status –> `battery.php`
4. Listening for Moisture value –> `moisture.php`

Tasks.php is executed every minute using cron and above other files are run continuously in php cli mode.

To use cron feature for tasks.php, go to terminal and type

```
* * * * * /usr/bin/php destinationtoabovephpfiles/tasks.php
```

To use cli mode, go to terminal and type

```
/usr/bin/php destinationtoabovephpfiles
```

Main php files are:

`index.php`-for managinng the manual control of the sensors
`devices.php`-for seeing devicesśtatus and other values including battery status
`time.php`-for scheduling automated tasks
`manage.php`-for adding types of sensors, groups and newly discovered devices

MQTT Library used

Uses php library **SSKAJE MQTT** for communicating with MOSCA broker.

SSKAJE MQTT
GITHUB

About the website/Screenshots