IoT exercises – Week 9

This week, we will start using the NodeMCU/ESP8266 to for MQTT based communication. In the lecture we talked about the TCP bsed MQTT. Particularly, we will code on the V3.1.1, which is an ISO and OASIS standard. For more details about the protocol, please refer to the detailed documents of this standard (http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html). The mqtt module will be applied.

The official documentation is provided here again for your reference. Please remember, when you are building your own IoT project in the future, always refer to the bespoke built-in modules and read their documentation first, which will be helpful!!!

https://nodemcu.readthedocs.io/en/master/

The details of mqtt module can be found in

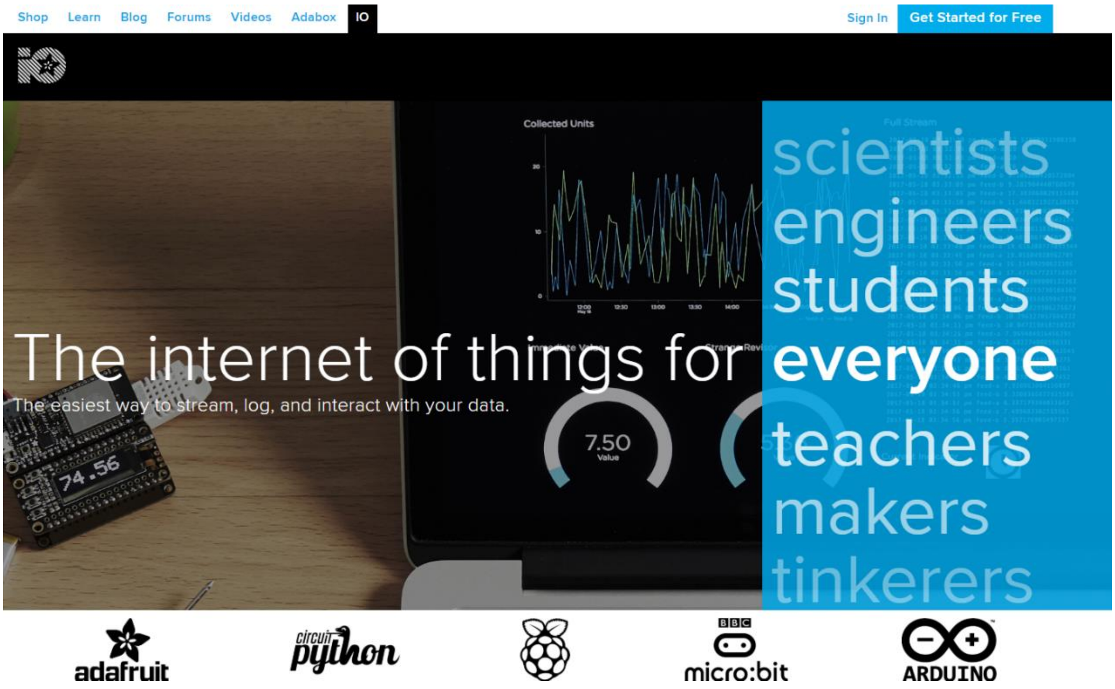
https://nodemcu.readthedocs.io/en/master/modules/MQTT/

This is extremely important for you to go on further with your IoT applications.

More public brokers are available: https://github.com/mqtt/mqtt.github.io/wiki/public_brokers

Exercise 0:

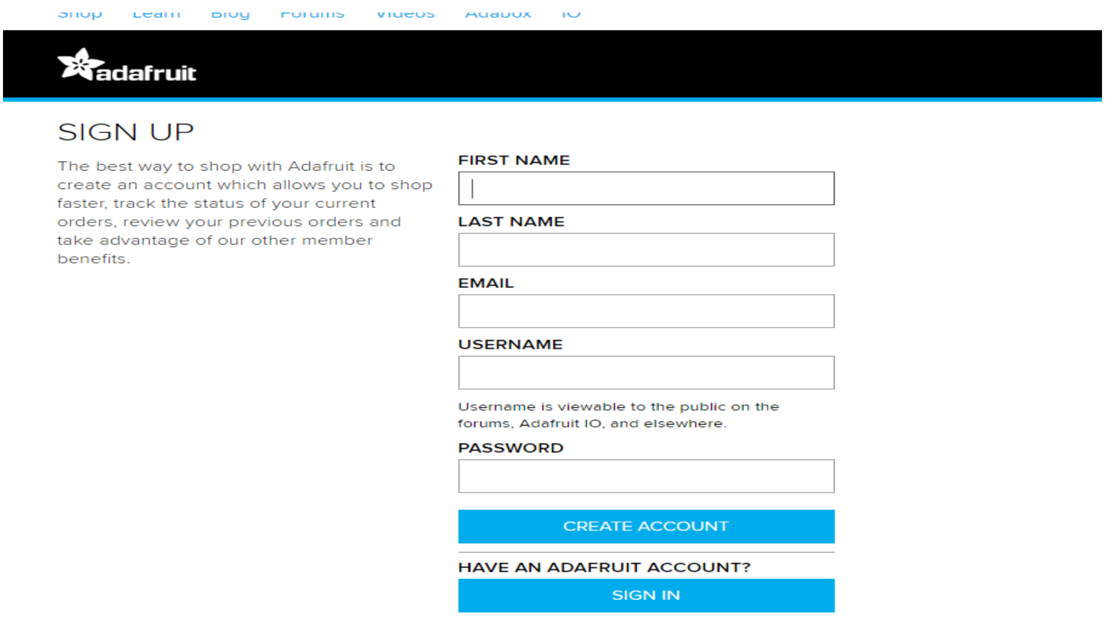
In our following exercises, the bespoke platform of https://io.adafruit.com/ with a user-friendly dashboard **will be adopted as the broker**. There are also more available brokers like https://iot.eclipse.org/getting-started/. Now we set up the **adafruit** one as an example.



The internet of things for everyone

The easiest way to stream, log, and interact with your data.

1st. Sign up for the adafruit account.



SIGN UP

The best way to shop with Adafruit is to create an account which allows you to shop faster, track the status of your current orders, review your previous orders and take advantage of our other member benefits.

FIRST NAME

LAST NAME

EMAIL

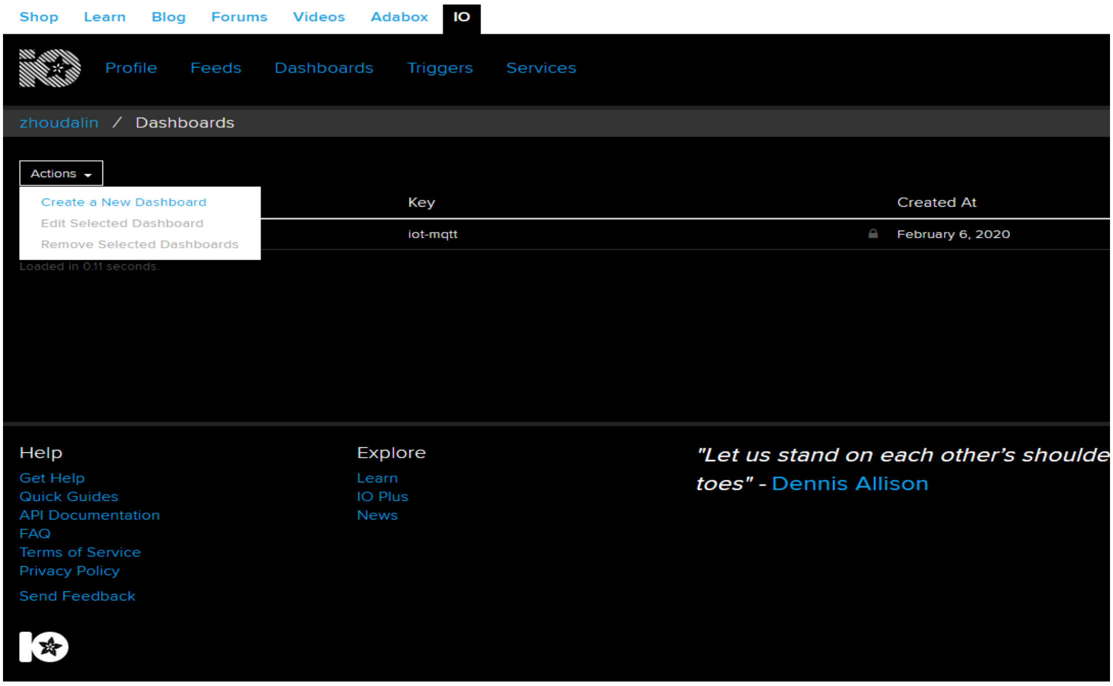
USERNAME

Username is viewable to the public on the forums, Adafruit IO, and elsewhere.

PASSWORD

HAVE AN ADAFRUIT ACCOUNT?

2nd. Log into IO and create your dashboard.

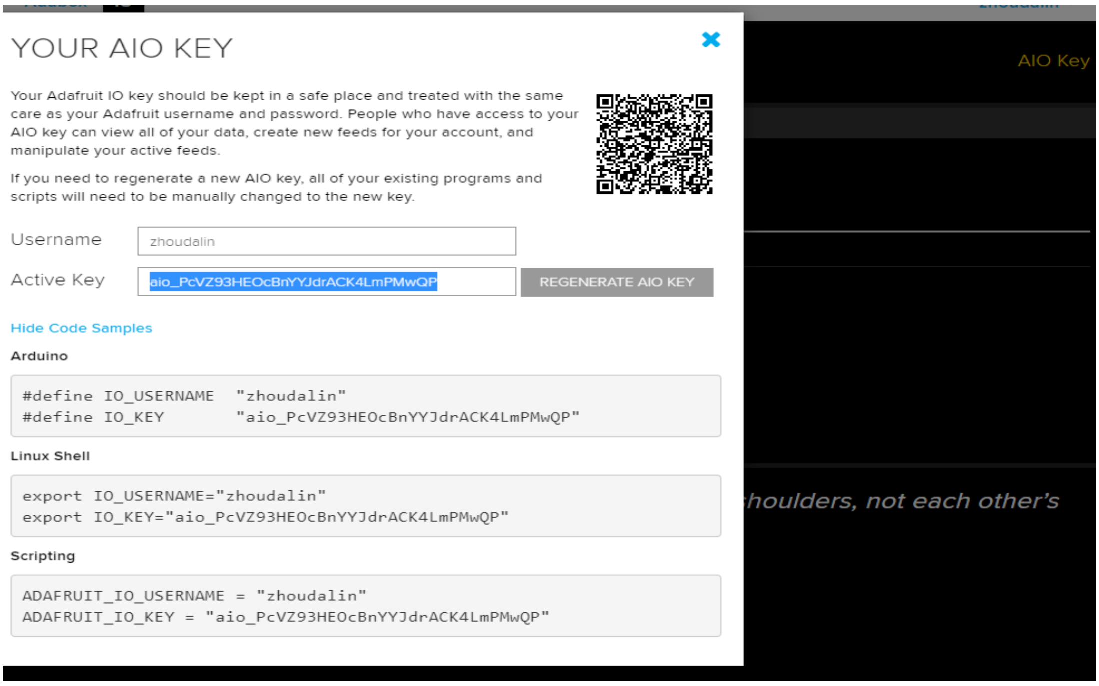


Key	Created At
iot-mqtt	February 6, 2020

Loaded in 0.11 seconds.

"Let us stand on each other's shoulde toes" - Dennis Allison

3rd.Check your username and key for later logging through MQTT.



YOUR AIO KEY

Your Adafruit IO key should be kept in a safe place and treated with the same care as your Adafruit username and password. People who have access to your AIO key can view all of your data, create new feeds for your account, and manipulate your active feeds.

If you need to regenerate a new AIO key, all of your existing programs and scripts will need to be manually changed to the new key.

Username: zhoudalin

Active Key: aio_PcVZ93HEOcBnYYJdrACK4LmPMwQP

Arduino

```
#define IO_USERNAME  "zhoudalin"
#define IO_KEY       "aio_PcVZ93HEOcBnYYJdrACK4LmPMwQP"
```

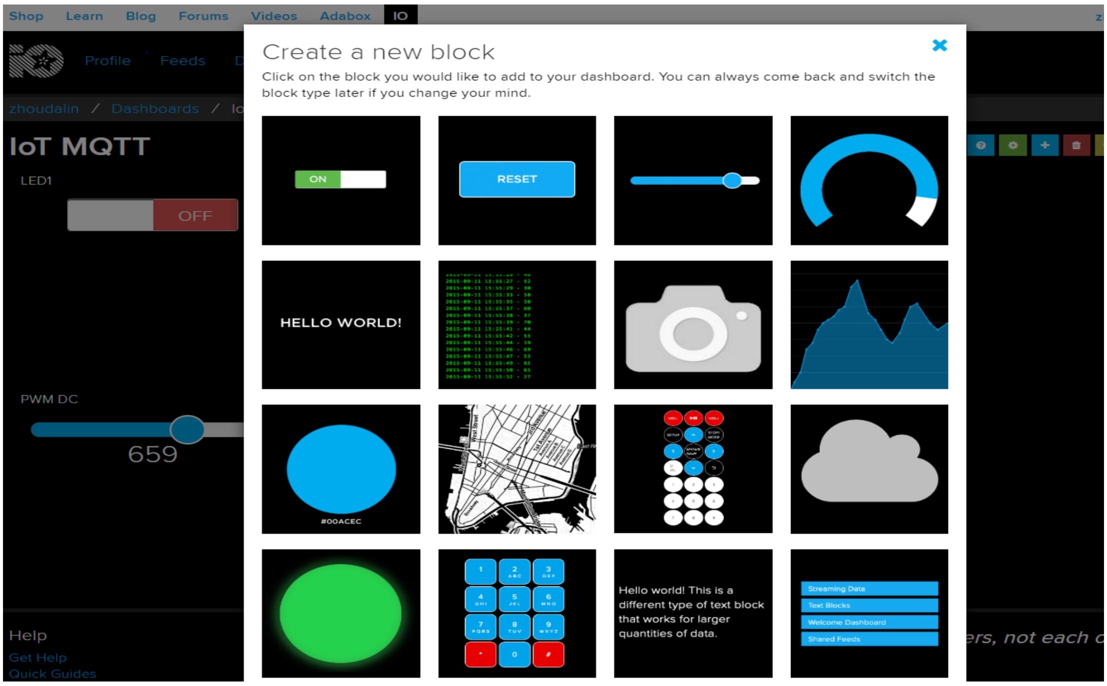
Linux Shell

```
export IO_USERNAME="zhoudalin"
export IO_KEY="aio_PcVZ93HEOcBnYYJdrACK4LmPMwQP"
```

Scripting

```
ADAFRUIT_IO_USERNAME = "zhoudalin"
ADAFRUIT_IO_KEY = "aio_PcVZ93HEOcBnYYJdrACK4LmPMwQP"
```

4th.Create the blocks that you need. Please do review their functions and descriptions.

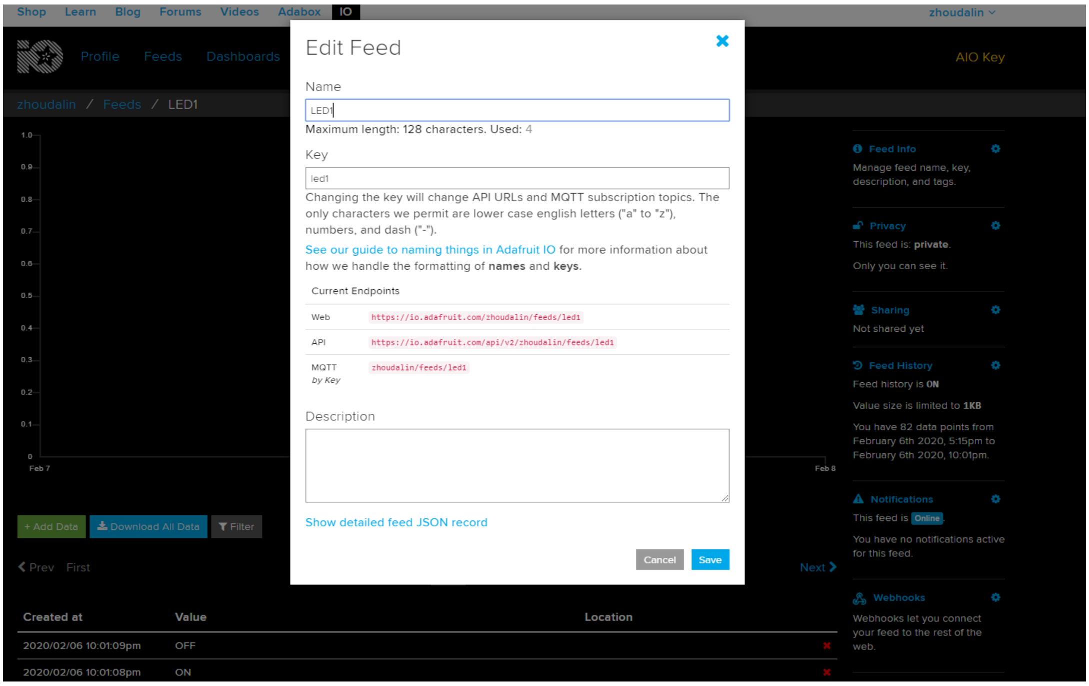


Create a new block

Click on the block you would like to add to your dashboard. You can always come back and switch the block type later if you change your mind.

Hello world! This is a different type of text block that works for larger quantities of data.

5th.Check the feeds info of your blocks for later communication (subscribe and publish) in MQTT.



zhoudalin / Feeds / LED1

Edit Feed

Name

LED1

Maximum length: 128 characters. Used: 4

Key

led1

Changing the key will change API URLs and MQTT subscription topics. The only characters we permit are lower case english letters ("a" to "z"), numbers, and dash ("-").

See our guide to naming things in Adafruit IO for more information about how we handle the formatting of **names** and **keys**.

Current Endpoints	
Web	`https://io.adafruit.com/zhoudalin/feeds/led1`
API	`https://io.adafruit.com/api/v2/zhoudalin/feeds/led1`
MQTT *by Key*	`zhoudalin/feeds/led1`

Description

Show detailed feed JSON record

Cancel Save

Feed Info

Manage feed name, key, description, and tags.

Privacy

This feed is: **private**.

Only you can see it.

Sharing

Not shared yet

Feed History

Feed history is **ON**

Value size is limited to **1KB**

You have 82 data points from February 6th 2020, 5:15pm to February 6th 2020, 10:01pm.

Notifications

This feed is Online.

You have no notifications active for this feed.

Webhooks

Webhooks let you connect your feed to the rest of the web.

Created at	Value	Location
2020/02/06 10:01:09pm	OFF	
2020/02/06 10:01:08pm	ON	

Exercise 1:

In this exercise, you will need to create a MQTT based client and communicate with the **broker** (in MQTT, different from the CoAP, a broker is expected to act as the server).

```
====================================================================

--connect your NodeMCU/ESP8266 first to the Internet using its station mode

HOST="io.adafruit.com" --adafruit host

PORT=1883 --1883 or 8883(1883 for default TCP, 8883 for encrypted SSL or other ways)

ADAFRUIT_IO_USERNAME="zhoudalin"--put your own username here

ADAFRUIT_IO_KEY="aio_PcVZ93HEOcBnYYJdrACK4LmPMwQP"--put your own io_key here

-- init mqtt client with logins, keepalive timer 300 seconds

m=mqtt.Client("Client1",300,ADAFRUIT_IO_USERNAME,ADAFRUIT_IO_KEY)

-- setup Last Will and Testament (optional)

-- Broker will publish a message with qos = 1, retain = 0, data = "offline"

-- to topic "/lwt" if client does not send keepalive packet

m:lwt("/lwt","Now offline",1,0)

--on different event "connect","offline","message",...

m:on("connect",function(client)
print("Client connected")
            print("MQTT client connected to"..HOST)
            client:subscribe(SUBSCRIBE_TOPIC,1,function(client)
                  print("Subscribe successfully")
                  end)
      end)
m:on("offline",function(client)
            print("Client offline")
end)
m:connect(HOST,PORT,false,false,function(conn) end,function(conn,reason)
            print("Fail! Failed reason is: "..reason)
end)
```

===

Exercise 2:

Recall the ON/OFF control we have implemented on LED. In this exercise, Use the MQTT based communication to subscribe to a broker to control the ON/OFF of LED.

===

```
SUBSCRIBE_TOPIC="zhoudalin/feeds/led1" -- put your topic of subscribe shown on the IoT platform/broker site
m:on("connect",function(client)
            print("Client connected")
            print("MQTT client connected to"..HOST)
            client:subscribe(SUBSCRIBE_TOPIC,1,function(client)
                  print("Subscribe successfully")
                  end)
end)
m:on("message",function(client,topic,data)
         print(topic .. ":" )
```

```
      if data ~= nil then
        print(data)
        LEDOnOff=YourOwnFunc_here(data) – please convert the read data into 0/1 for control
        if LEDOnOff==1 or LEDOnOff==0 then
            gpio.write(pinLED,LEDOnOff)
        end
      end
end)
```

===

Exercise 3:

Recall the PWM control we have implemented on LED. In this exercise, Use the MQTT based communication to subscribe to a broker for the adjustable control of LED. You only need to slightly change your code for Exercise 2.

You might have found a pitfall here that the subscription is not always successful, sometimes not conducted. Please think of a way to solve it.

Exercise 4:

Recall the ADC reading we have implemented with photoresistor and pot resistor. In this exercise, Use the MQTT based communication to publish the current ADC input to the broker.

===

```
function pubADC(client)
      mytimerPublish = tmr.create()
      mytimerPublish:register(2000,1,function()
      client:publish(PUBLISH_TOPIC,tostring(adcV),1,0,function(client)
      print("ADC reading sent: ",adcV)
          end)
      end)
      mytimerPublish:start()
```

```
end
```

===

Exercise 5:

Now you have implemented both the subscription and publishing through one MQTT based client and communicate with the broker. Combine Exercise 3 and 4 to create a bidirectional communication for PWM control and display.

Exercise 6:

Now you have implemented a whole MQTT based client and communicate with the broker. Recall the variable resistor and long-pushed button based control. Taking 2 NodeMCU/ESP8266 as 2 clients with the same broker, use one's sensory data to control the other one's LED.

(Optional Challenge) Exercise 7:

In this exercise, you are expected to build you own MQTT based broker.