

# Internet of Things: Architecture and Programming

## Demo Exercise 6

In this demonstration, you will use Esp8266 Thing Dev ([www.sparkfun.com/products/13711](http://www.sparkfun.com/products/13711)) to send data to the cloud. With a proper SDK, you can easily send data to the cloud for further processing. In this demo, we will use Microsoft Azure, because of available SDK.

### **AWS:**

Unfortunately, I could not find a good SDK for the AWS (Amazon cloud) except "aws-sdk-arduino-esp8266" library. You can find the sample code here: <https://github.com/heskew/things-aws-iot-soil-monitor/blob/master/sensor/sensor.ino>

There is a problem with this library: after each publish, the esp8266 restarts, probably because of memory leak. Note: bonus point for a group who find a SDK for AWS without any issue.

### **Azure:**

Azure gives each new user a free account with 200\$ initial credit that can be used in the first month. Azure, like AWS, has many services, e.g. IoT hub, storage, and virtual machine. In addition, it provides Command Line Interface (CLI), which makes it much easier to configure your cloud. At first, you can learn how to use azure services through its website, then try the CLI.

For this demo, you will send a sensor data (your choice) from your ESP8266 thing Dev to the cloud (azure IoT hub), and either store it or display it using a web app (your choice). Using azure IoT hub, you can connect your IoT device to the cloud, set up credential for each device, and forward the messages to other azure services. You can follow the steps below, preferably in the provided order.

#### **Step 0: Set up your cloud**

you can follow the official Azure tutorial to send data from ESP8266 Thing Dev to the cloud <https://docs.microsoft.com/es-es/samples/azure-samples/iot-hub-sparkfun-thingdev-client-app/sparkfun-esp8266-client/>

#### **Step 1: compile your code**

Compilation error? you might get some compilation errors because of the incompatibility of libraries. On my machine, I have been able to compile the code using ESP8266 (boards manager) v. 2.4.0, AzureIoTHub/AzureIoTUtility/AzureIoTProtocol\_MQTT v. 1.0.45, and ArduinoJson v. 5.11.2.

you can hard code the credential information, such as wifi SSID and password, and device connection string, instead of getting it as the input. In this case, you can void the whole "readCredentials" function in credentials.ino.

## Step 2: Register your device in cloud

you have to register your device in iot hub, under “IoT devices”, by providing the “device ID”. (In the code provided by aforementioned tutorial, you can find device id in config.h. you might want to change it). Then, you can generate the “connection string” by clicking on the your device ID in iot hub “IoT devices”. You have to use the connection string in your device code (app.ino).

## Step 3: set up connection string in your code

set the connection string in your code, and compile it again.

## Step 4: Upload and Debug

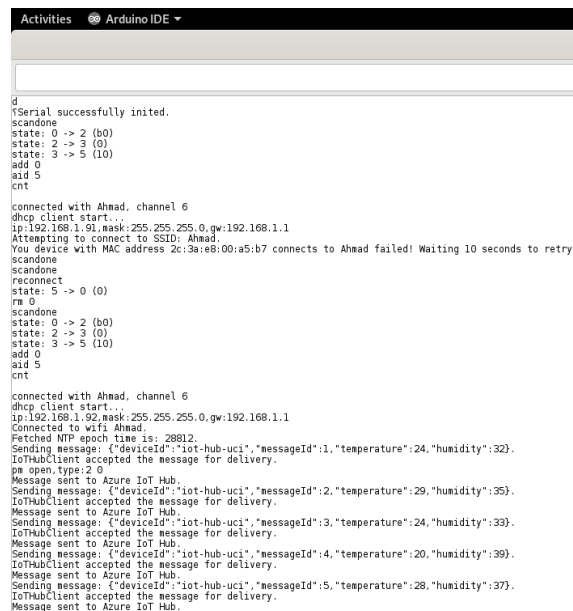
upload your code and debug it.

In setup(), in app.ino, there are 3 main functions.

```
InitWifi();    //connect to WiFi
initTime();    //sync the time with ntp
initSensor();  // initialize the sensor
//It is also set up the connection to the cloud using the provided connection string (from step 3).
```

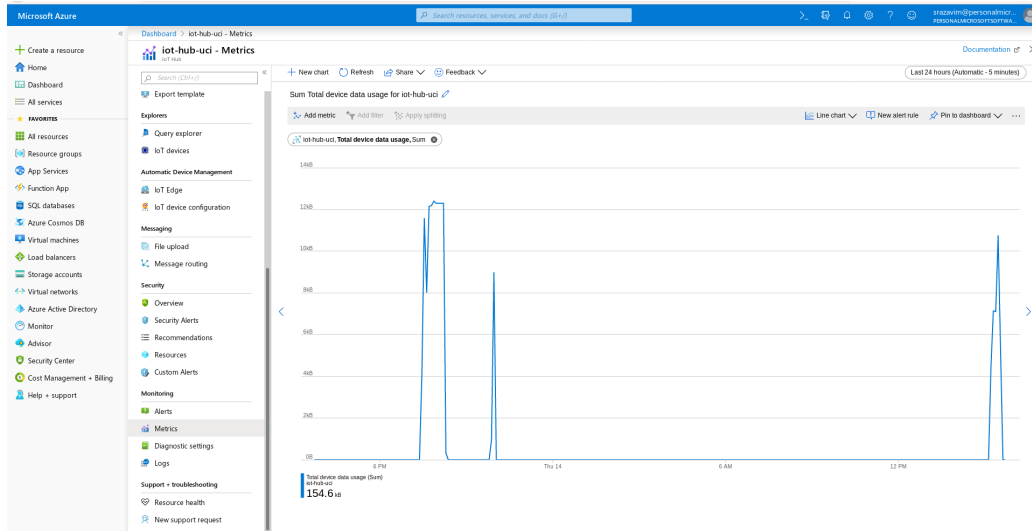
in the serial monitor, you can see if any of these steps fail, and fix the problem.

If everything is fine, you will see something like this in your serial monitor:



```
d
Serial successfully initied.
scandone
state: 0 -> 2 (b0)
state: 2 -> 3 (0)
state: 3 -> 5 (10)
add 0
aid 5
cnt
connected with Ahmad, channel 6
dhcp client start...
ip:192.168.1.92,mask:255.255.255.0,gw:192.168.1.1
Attempting to connect to SSID: Ahmad.
You device with MAC address 2c:3a:e8:00:a5:b7 connects to Ahmad failed! Waiting 10 seconds to retry.
scandone
reconnect
state: 5 -> 0 (0)
rm 0
Scandone
state: 0 -> 2 (b0)
state: 2 -> 3 (0)
state: 3 -> 5 (10)
add 0
aid 5
cnt
connected with Ahmad, channel 6
dhcp client start...
ip:192.168.1.92,mask:255.255.255.0,gw:192.168.1.1
Connected to wifi Ahmad.
Fetched NTP epoch time is: 28812.
Sending message: {"deviceId":"iot-hub-uci","messageId":1,"temperature":24,"humidity":32}.
IoTHubClient accepted the message for delivery.
rm open, type: 2 0
Message sent to Azure IoT Hub.
Sending message: {"deviceId":"iot-hub-uci","messageId":2,"temperature":29,"humidity":35}.
IoTHubClient accepted the message for delivery.
Message sent to Azure IoT Hub.
Sending message: {"deviceId":"iot-hub-uci","messageId":3,"temperature":24,"humidity":33}.
IoTHubClient accepted the message for delivery.
Message sent to Azure IoT Hub.
Sending message: {"deviceId":"iot-hub-uci","messageId":4,"temperature":20,"humidity":39}.
IoTHubClient accepted the message for delivery.
Message sent to Azure IoT Hub.
Sending message: {"deviceId":"iot-hub-uci","messageId":5,"temperature":28,"humidity":37}.
IoTHubClient accepted the message for delivery.
Message sent to Azure IoT Hub.
```

you can check if you are receiving any data in your cloud by clicking on “metrics” under “monitoring” in your azure IoT hub.



You can easily set up your Azure IoT hub using Azure CLI.

<code>az extension add --name azure-cli-iot-ext</code>	add iot extension to your Azure CLI
<code>az group create --name MyResourceGroup --location eastus</code>	add new resource group
<code>az iot hub create --resource-group MyResourceGroup --name {YourIoTHubName}</code>	add new IoT hub
<code>az iot hub device-identity create --device-id {your_device_name} --hub-name {hub name}</code>	Add your_device_name to your "hub name"
<code>az iot hub device-identity show-connection-string --hub-name {YourIoTHubName} --device-id MyNodeDevice --output table</code>	Show the connection string

more documents about Azure CLI:

<https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-create-using-cli>

<https://docs.microsoft.com/en-us/azure/iot-hub/tutorial-connectivity>

<https://docs.microsoft.com/en-us/azure/iot-hub/quickstart-send-telemetry-node#prerequisites>

After making sure that you can send data to the cloud, you can either store it or visualize it.

For visualization, you can use azure web app. For this purpose, you might follow this tutorial:

<https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-live-data-visualization-in-web-apps>

At the end, you can see the sensor data in your browser.

**Note:** Although you have 12 month free account, your initial credit might run out because some of Azure services are not free. You can disable those services just after using them. You may check the bill daily to make sure of it.