# ONEM2M Lab Session

**IIIT-Hyderabad**

**9 and 11, April 2019**

## CONTENTS:

## Pre-requisites:

**NOTE : INSTALLATION AND SETTING UP TAKES UP LOT OF TIME. IT IS ADVISED TO DO ALL THE REQUIRED INSTALLATIONS PRIOR ATTENDING THE LAB.**

1. Software:
    a. Python IDE with
        i. Requests Library(installation process)
    b. Arduino IDE
        i. With ESP8266 boards installed (installation process)
        ii. Required Library files: (How to install library file in Arduino IDE)
            1. SoftwareSerial library
            2. ESP8266WiFi library
            3. ArduinoJson v5.13.5 library
    c. OneM2M server(installation process)

2. Hardware:
   Special kits will be provided for this lab.

3. [Optional] Basic understanding of python, JSON and IP protocols

4. Download code from this Code repo link
    a. **ESP8266-RX** : ESP8266 code for receiving side
    b. **ESP8266-TX** : ESP8266 code for sending side
    c. **OneM2M.py** : This is the OneM2M library code used for this lab
    d. **Test.py** : Run this code first to ensure communication with server.
    e. **Create_tree.py** : Use this file to create the OneM2M resource tree.
    f. **Lab.py** : You will write logic that reads the tapping and actuate LEDs
    g. **SingleDoubleTap.bin** : Bin code for NUCLEO board.

## Lab guidelines:

1. All laptops are supposed to be connected to this **Wi-Fi Network SSID : "ASUS"**(not ASUS_5G) and **Password : "onem2mlab"** as the OneM2M CSE(server) is hosted on this network.
   To access the Resource tree, **enter this url in your browser** : **192.168.1.233:8080/webpage**.
   **OneM2M Credentials - username : admin, password : admin**

2. Then, we will build an OneM2M resource tree, register entities, create containers, etc… which are required for this IoT experiment.
   Here, you will use the python code provided. This code needs to be filled with the payload data and type fields.

3. Then we will latch-up the hardware. For this experiment, Nucleo F401RE along with ESP8266 are required
   Overall, students are supposed to switch on remotely placed LEDs that is connected to an receiver ESP8266 when the sensor shield is tapped.
   This process will be discussed in detailed during the lab session

4. At the end of this lab, every student would be comfortable with the idea of integrating OneM2M framework into IoT projects/products by using API's for:
   a. Resource Creation,
   b. Grouping,
   c. Filtering through labels
   d. Reading from the tree
   e. Writing to the tree

## Experiment:

Let’s breakdown this idea, as discussed in the earlier section, into milestones.

**Agenda:**
**- To switch on remote LEDs numbered 1 and 3 when Nucleo F401RE is tapped once.**
**- And To switch on remote LEDs numbered 2 and 4 when Nucleo F401RE is tapped twice.**

We use ESP8266, a low cost microcontroller with TCP/IP stack,
- One at transmitter side that can talk to the Nucleo board and POST to the OneM2M Resource tree (Code for this ESP is given in the code repo as ESP8266-TX)
- Second at receiver side that scans the resource tree and switch on those specific LEDs.
(Code for this ESP is given in the code repo as ESP8266-RX)

We then start with building the OneM2M resource tree.

1. **Build an OneM2M Resource tree**
    a. Registering entities
        i. Use the python script ***“create_tree.py”***, create 4 AEs in this format: **led-<*table number*>-<*LED_number*>** *for 4 LEDs.* *(Eg: If your table number is 3, then the AE names you should create are :led-3-1, led-3-2, led-3-3, led-3-4)*
        Remember to pass these 3 labels while creating each AE:
            1. LED(for all AEs)
            2. LED number(1,2,3,4)
            3. Is it even or odd?(even,odd) [case sensitive]
            4. Table number
        *ii.* *In the same script, also create an AE with the name* ***tappingEvent-<table number>.*** This will store the the single/double tap events.
        *(Eg: If your table number is 3, then the name should be tappingEvent-3)*
        Remember to pass these labels while creating each tappingEvent AE:
            1. Table number

b. Creating containers and container instances
    i. Create a container named DATA under each AE.
    ii. This will create CNTs in the tree.
c. Build the URL of the container and post a sample value to the above URL and visualize in the tree
d. Reading content from resources using GET
    i. Send a GET request to an AE’s DATA container to get the latest data by appending “/la” to its URI.
e. [OPTIONAL]Filtering using labels
    i. Now try to get the URI using filtering methods using labels as mentioned earlier.
f. [OPTIONAL]Grouping
    i. Create a group AE by grouping all,even and odd LEDs. Follow this syntax:
       LED-GROUP-<table number>-<all/even/odd>
       (Eg: “LED-GROUP-3-even” for even LEDs).
       This way, it will then be possible to control multiple LEDs simultaneously.

**List of tree parameters for your understanding:**

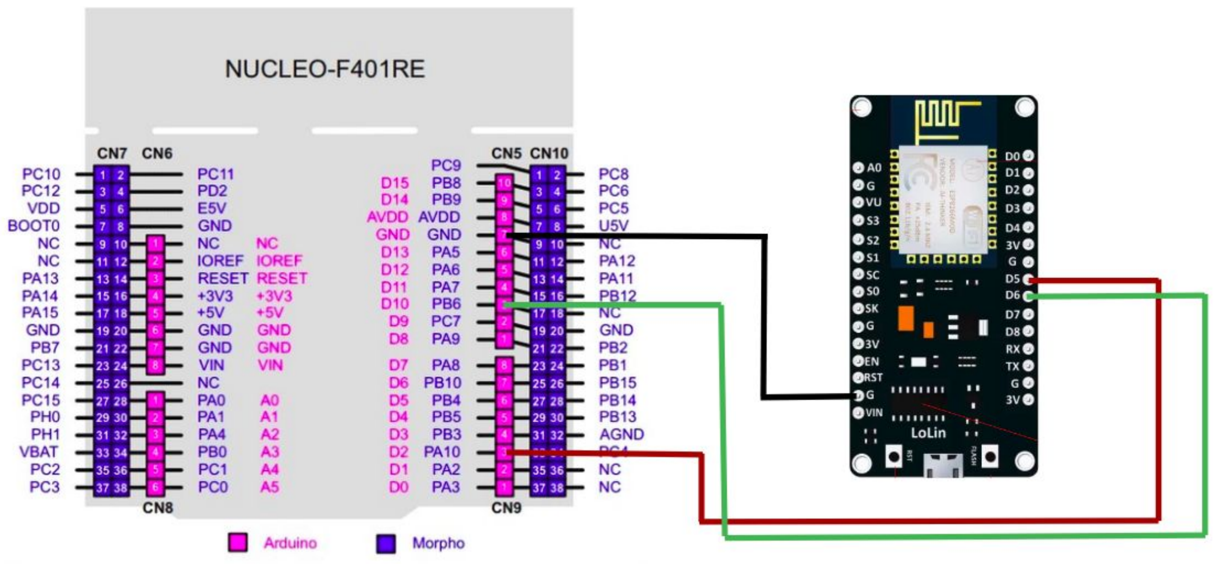
- ae: Application Entity(Sensor/actuators)
- cnt: Container(For holding various kinds of data under the same AE)
- cin: Content Instance(For holding various instances of the same data type)
- sub: Subscription
- rn: Resource Name
- ty: Type
- ri: Resource ID
- pi: Parent Id
- Acpi: Access Control Policies IDs
- uril: URI List
- ct: Creation Time
- et: Expiration Time
- lt: Last Modified Time
- lbl: Label
- cnf: Content Format
- con: Content
- mni: Maximum Number of Instance
- api: Application Id
- poa: Point of Access
- rr: Request Reachability
- sur: Subscription URI

2. **Transmitter side:**

Once you have built the OneM2M resource tree, it’s time to latch up the hardware.

a. **H/w setup**

**USB CABLE ON THIS SIDE**



S/w:

b. **S/w setup:**

i. Code to detect the tap(single/double) event on Nucleo board is given which sends data to ESP8266 via UART.

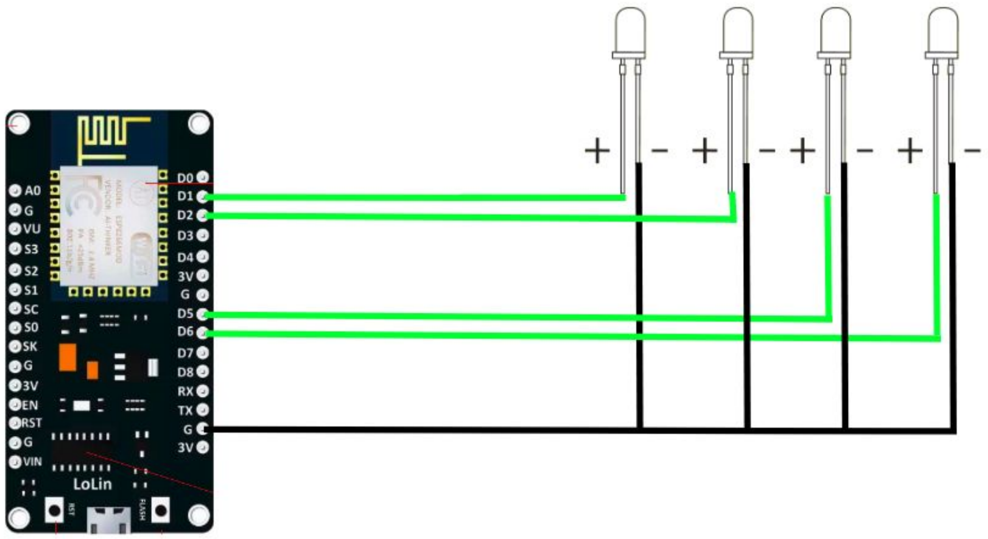
ii. The code in folder ‘ESP8266-TX’ reads from Nucleo and will POST this data in the **tappingEvent**’s DATA container. This code is already written for you.
You have to do only one change before uploading via arduino IDE:
   1. Update your Lab table number

iii. On your laptop, write a python script in **“lab.py”** that scans the **tappingEvent**’s DATA container periodically and get the latest data on how the board is tapped.

iv. If it is tapped once, you need to switch the odd LEDs ON on the RX-ESP8266. If tapped twice, switch on the even LEDs
Based on the type of tapping, update the specific LED containers that should glow.

## 3. Receiver side:

a. **H/w setup**



b. **S/w setup:**

i. The code on ESP8266-RX scans the OneM2M resource tree periodically by sending GET requests to URIs to get the latest values for its LEDs.

ii. This code is provided. Just change the table number before uploading it to the board.

iii. Based on this value, the LED will either turn on/turn off.

## Some useful Links:

- OneM2M
- OneM2M Installation Instructions
- https://wiki.eclipse.org/OM2M/one/REST_API
- Theory
- Dashboard for visualizing data