# Lab 6: Setting up OM2M and Communication with ESP32

Spring-2025  
February 20, 2025

## Student Details

Name: Samyak Jain, Rishabh Goyal  
Room No.: N125  
Table No.: 42

## AIM

To set up OM2M, communicate with an ESP32, and implement data exchange between the ESP32 and an OM2M server for remote monitoring of CPU and RAM usage.

## Software Tools Required

- Postman REST-API Client  
- OM2M Server  
- Java JDK 1.8  
- Python  
- Jupyter Notebook

## Hardware Tools Required

- 1 ESP32  
- 1 LED  
- 2 Laptops  
- Stable WiFi connection

## Experiment 1: Setting up Postman, OM2M Server, and Jupyter Notebook

### Procedure

1. Postman Installation:  
 - Download and install Postman.  
 - Import the OM2M REST API collection for testing.

2. OM2M Server Setup:  
 - Download and extract OM2M.  
 - Modify config.ini to set the port to 5089.  
 - Start the OM2M server and log in via http://127.0.0.1:5089/webpage.

3. Python and Jupyter Notebook Setup:  
 - Install Python and Jupyter Notebook.  
 - Set up a virtual environment and install required dependencies.

Creating the OM2M Resource tree using postman

1. Go to the Postman then click Workspaces then press Import then Upload Files and select the JSON files named as ‘OM2M REST APIs.postman collection.json’
2. Click on Import. We can see the request collections below the OM2M Rest APIs
3. Before using the collection make sure the OM2M server is running and available at http://127.0.0.1: 5089/webpage. The request can be sent by pressing the “Send” button in postman, after selecting any request.
4. Now send each of the following commands to construct the resource tree

* Creating an Application Entity: Select Post Create Application Entity Press Send
* Creating a container for device called Node-1: Select Post Create Node-1 container Press Send
* Creating a Descriptor container: Select Post create descriptor cnt Press Send
* Creating a Data container: Select Post Create Data Container Press Send
* Creating descriptor content instance: Select Post create descriptor cin Press Send
* Creating data content instance 1 & 2: Select Post Create Content instance 1/2 Press Send
* Retrieving the latest content instance: Select GET latest Content instance Press Send
* Retrieving the oldest content instance: Select GET oldest Content instance Press Send
* Retrieving all content instances: Select GET all data of a container Press Send
* Updating number of attributes inside data container: Select PUT update cnt attrs Send.

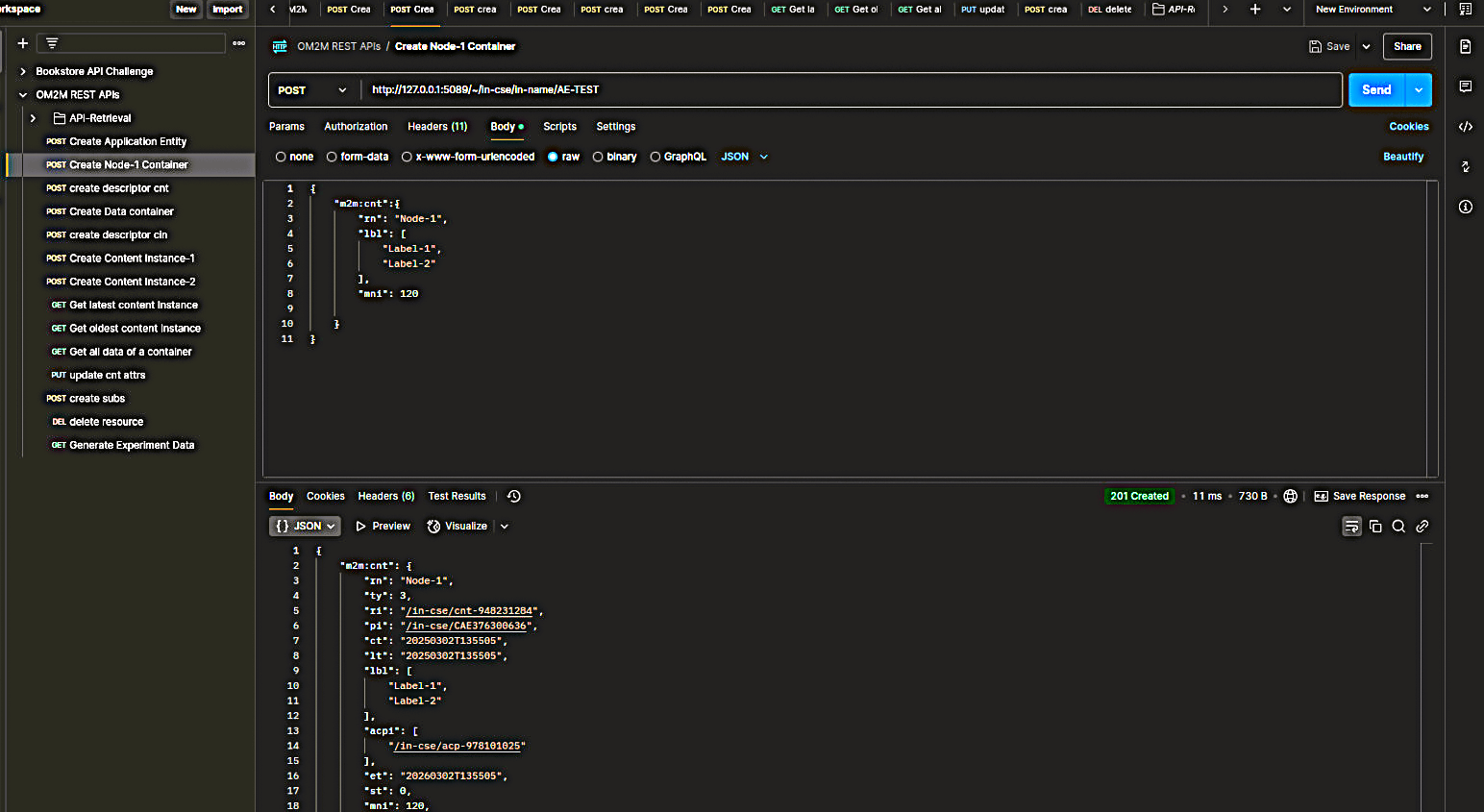
### Observations

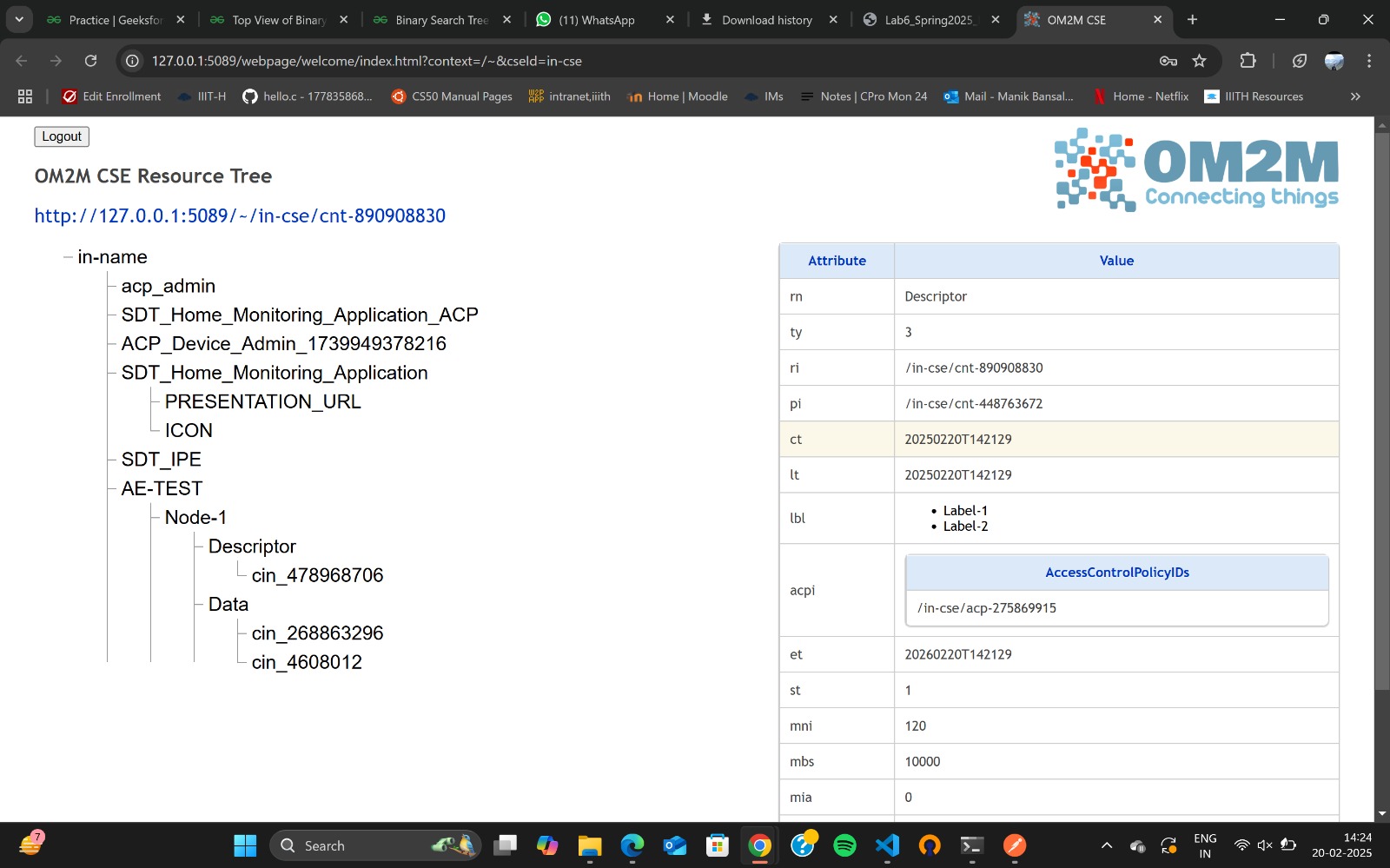
- The OM2M resource tree was successfully created.  
- Postman successfully sent API requests to OM2M.  
- Jupyter Notebook was successfully configured.

- resource tree was formed

### Results/Conclusions

OM2M and Postman were set up successfully, allowing device resource management through API requests.





## Experiment 2: Publishing CPU and RAM Usage Data to OM2M

### Procedure

1. Using Jupyter Notebook as a Mock Device:  
 - Launch Jupyter Notebook and run "Mock Device CPU Usage.ipynb".  
 - Publish CPU and RAM usage data to OM2M.

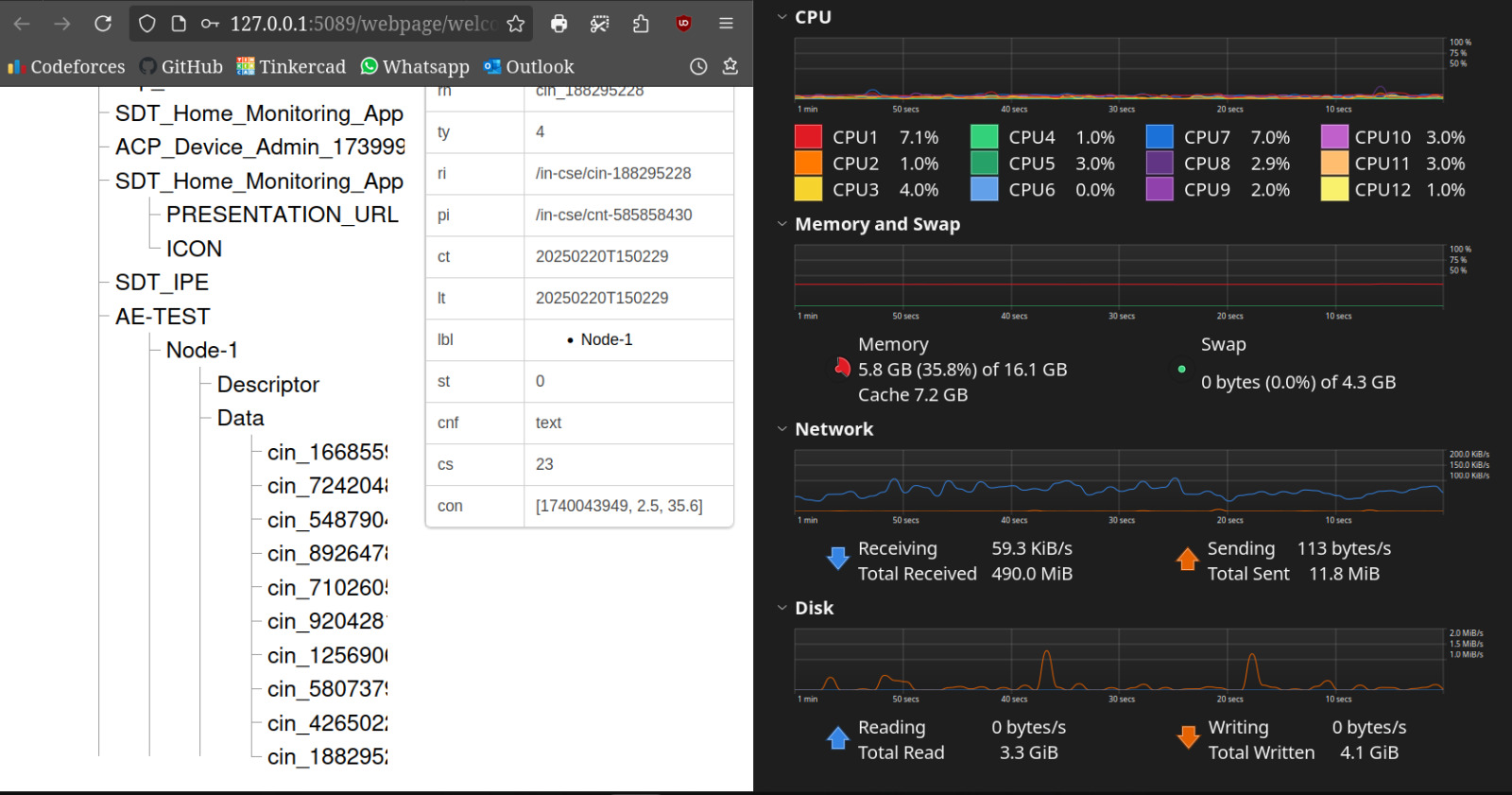
2. OM2M Resource Tree Updates:  
 - Send API requests to create an application entity and data container.  
 - Retrieve and visualize stored data.

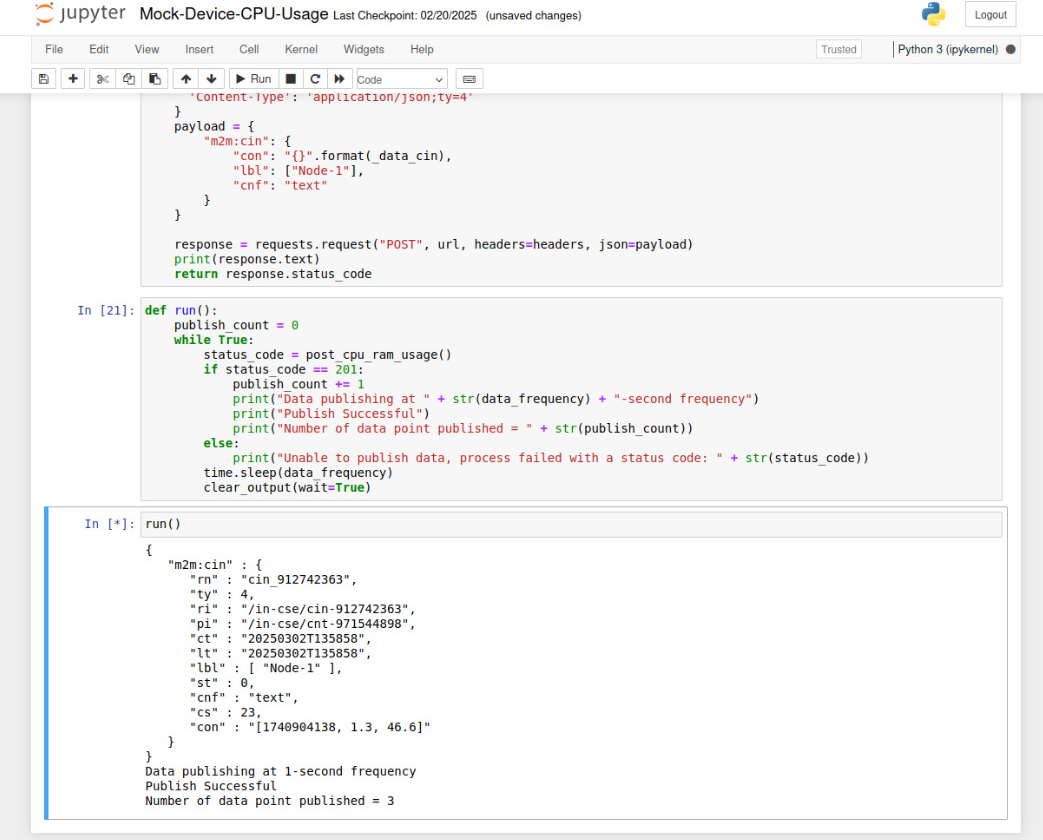
### Observations

- CPU and RAM usage data was successfully published to OM2M.  
- Data updates appeared in the OM2M resource tree.

### Results/Conclusions

The system successfully simulated data publishing from a device to OM2M.





## Experiment 3: Implementing a Remote CPU and RAM Usage Monitor

### Procedure

1. ESP32 Data Retrieval from OM2M:  
 - ESP32 fetches the latest CPU and RAM usage data via an HTTP GET request.  
 - Data is printed to the serial monitor.

2. LED Blinking Based on CPU/RAM Threshold:  
 - If CPU or RAM usage exceeds a defined threshold, the LED blinks rapidly.  
 - The LED stops blinking once values return to a safe range.

### Observations

- ESP32 successfully retrieved data from OM2M.  
- LED responded correctly to threshold exceedances.

### Results/Conclusions

The ESP32 successfully monitored and reacted to CPU and RAM usage changes.

