IOT RESEARCH FOR NGDS

1. Introduction

NextGen Flight Test network is combination of wired and wireless sensors. The Acquisition device collects data from multiple sensors and convert it into DAR format for further data calculation, storage and analysis. As part of this IOT research implemented a solution for collecting data from different wireless sensors that can be managed/ configured through Management Server and the data from sensors can be converted into DAR format for further analysis.

2. About CoAP and LwM2M

3.1 Constrained Application Protocol (CoAP) for IOT

IoT needs to integrate various sensors, computer and communication equipment, which are using different communication protocols. Wireless Protocols are mainly used in three layers, which are PHY/MAC layer, Network/Communication layer and Application layer. Lightweight protocol

CoAP is intended to be used and considered as a replacement of HTTP for being an IoT application Layer protocol. It is simple, low overhead protocol designed for environments like low end microcontrollers and constrained networks with critical bandwidth and high error rate.

CoAP is based on Message Request/Response Model.

It uses DTLS when considering security, namely integrity, authentication and confidentiality.

3.2 Lightweight Machine to Machine

LwM2M is a system standard developed by the Open Mobile Alliance for remote device management in the Internet of Things and other Machine-to-Machine applications. It aims to develop a fast deployable client-server specification to provide machine to machine services.

It is designed to be transported either over UDP secured with DTLS in IP networks, or over SMS directly in cellular phone networks. The application data is encapsulated using the Constrained Application Protocol (CoAP).

Market Motivation
It provides both efficient device management as well as security workflow for IOT applications, making it especially suitable for use in constrained networks.

LwM2M Architecture

The OMA Lightweight M2M Enabler, consisting of a LWM2M Client (M2M device) and a LWM2M Server (M2M service/platform/application), employs a client-server architecture plus CoAP with UDP/SMS transport bindings as shown in Fig below. The enabler is targeted, in

particular, at constrained devices, e.g. devices with low-power microcontrollers and small amounts of Flash and RAM over networks requiring efficient bandwidth usage. At the same time, LWM2M can also be utilized with more powerful embedded devices that benefit from efficient communication.

The LWM2M Enabler defines the application layer communication protocol between a server and a client. The LWM2M Server is typically located in a private or public data center and can be hosted by the M2M Service Provider, Network Service Provider or Application Service Provider. The LWM2M Client resides on the device and is typically integrated as a software library or a built-in function of a module or device.

Following four logical interfaces are defined between the server and client:

1. **Bootstrap**: allows LWM2M Bootstrap Server to manage the keying, access control and configuration of a device to enroll with a LWM2M Server.

2. **Device Discovery and Registration**: allows an LWM2M Client device let the LWM2M Server know its existence and register its capability.

3. **Device Management and Service Enablement**: allows the LWM2M Server to perform device management and M2M service enablement by sending operations to the Client and to get corresponding responses from the LWM2M Client.

4. **Information Reporting**: allows the LWM2M Client to report resource information to the LWM2M Server; can be triggered periodically or by events.

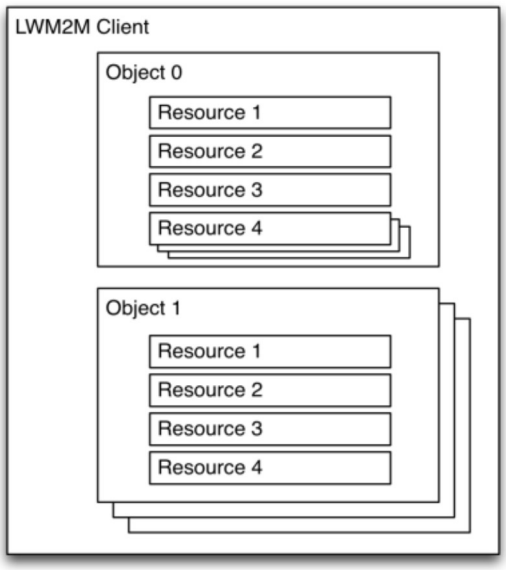
LWM2M communication model is based on simple COAP methods such as GET, PUT, POST, and DELETE with bindings over UDP or SMS as transport layer. The binary encoded message overheads will only be a few bytes and the flat, simple objects with uniform URI across devices makes the protocol best suited for constrained device connectivity and easy management.

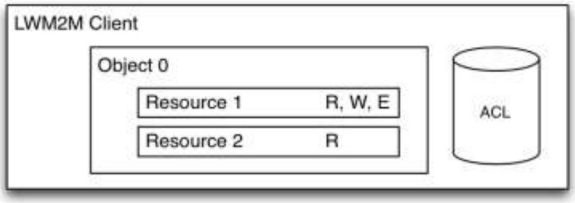
Object Model in LwM2M

A LWM2M Client has one or more Object Instances. An Object is a collection of Resources. A Resource is an atomic piece of information that can be Read, Written or Executed.
Objects and Resources are identified by a 16-bit Integer, Instances by an 8-bit Integer.
Objects/Resources are accessed with simple.

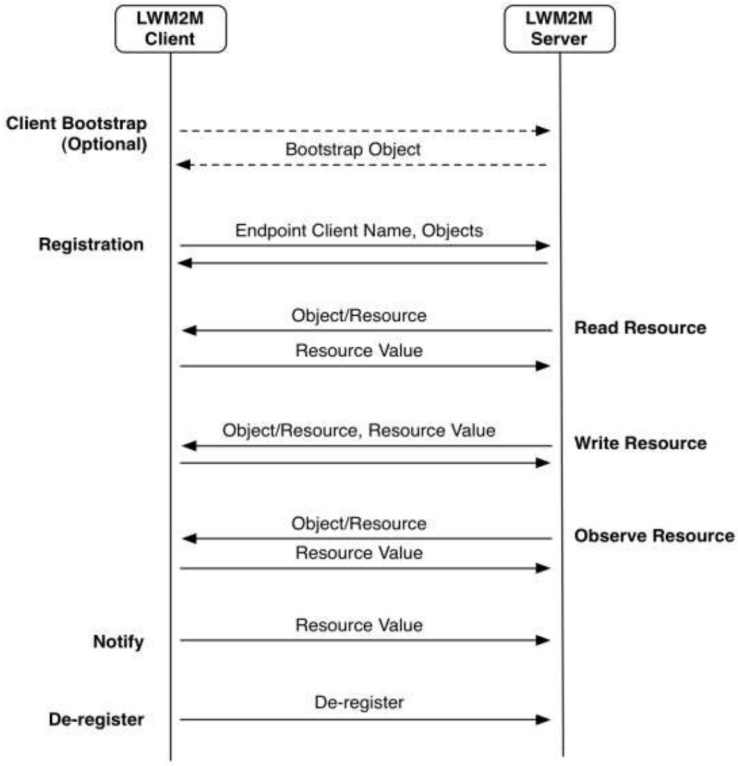
URIs: **/{Object ID}/{Object Instance}/{Resource ID}**

Example: /3/0/1 =
3 = Device Object,
0 = Object Instance #0,
1 = Manufacturer Resource





Example for LwM2M Interface Flow

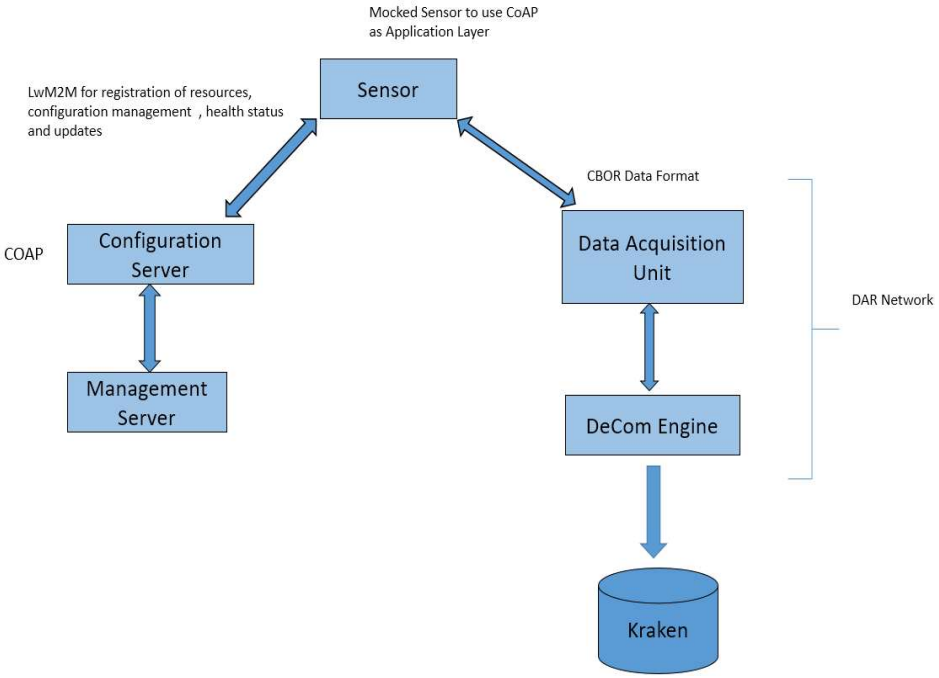


3. Prototype to show functionality of CoAP and LwM2M

Here Management Server and Simulated Sensor is being developed in Python to show the functionality of CoAP and LwM2M functionality. Both Server and Client are on local machine.

Packages Used: Txthings, Twisted, Kivy

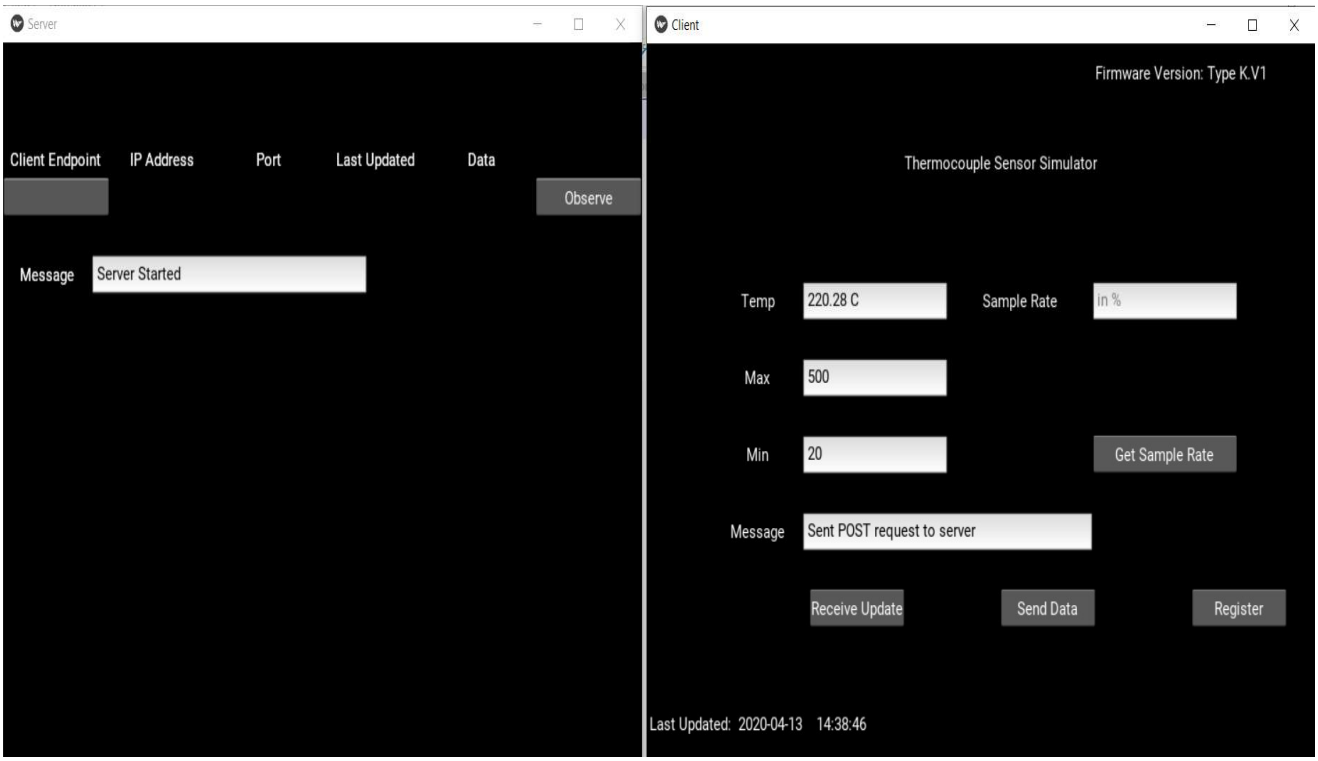
Git: https://git.web.boeing.com/ngds/iot_prototype1



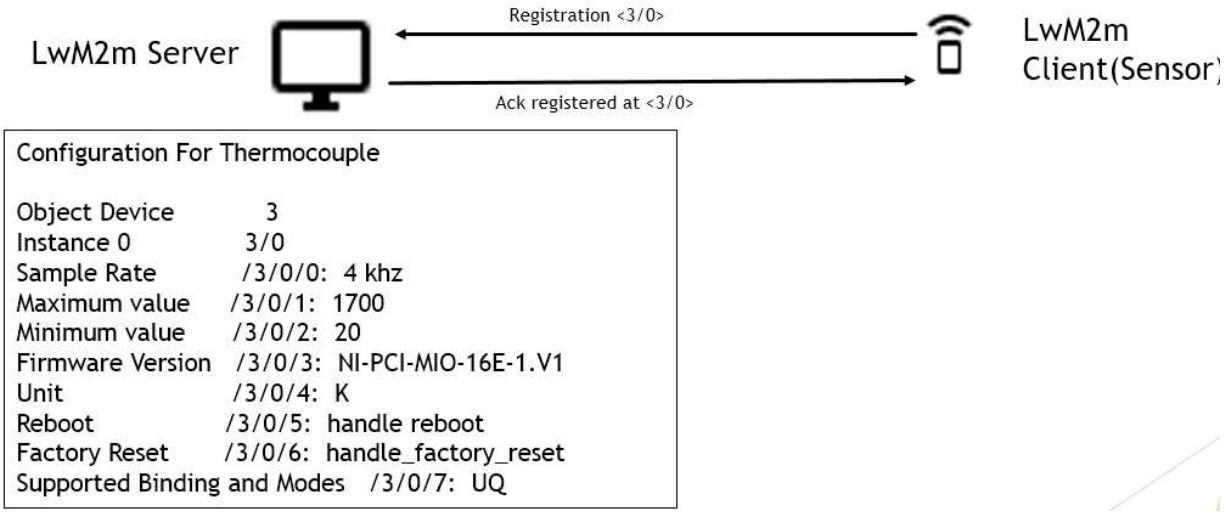
Following Functionalities Implemented in Prototype

1) **Register**: Client send Register request to server and server sends acknowledgement that client is added.
2) **Send data**: Client can updates whenever data is changed.
3) **Receive Update**: Server sends configuration changes to client for example update Firmware version.
4) **Observe**: Functionality on server side to observe a sensor data.

Instructions to install Python libraries and run below prototype are mentioned here: [\\fil-as-01\BTnE-India\5.I&DS\IDS India\100-Work\IOT\2. WIP\Lwm2mClient-Server]

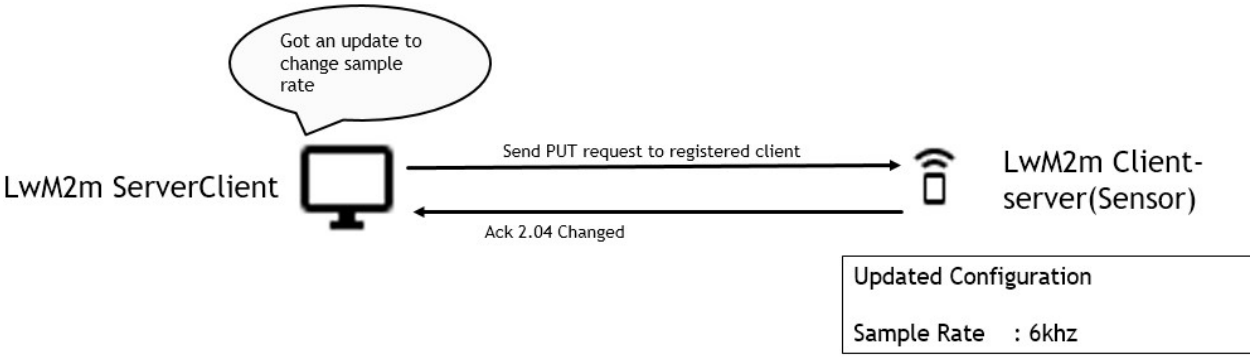


Use Case 1:



Client registers and sends configuration to Server

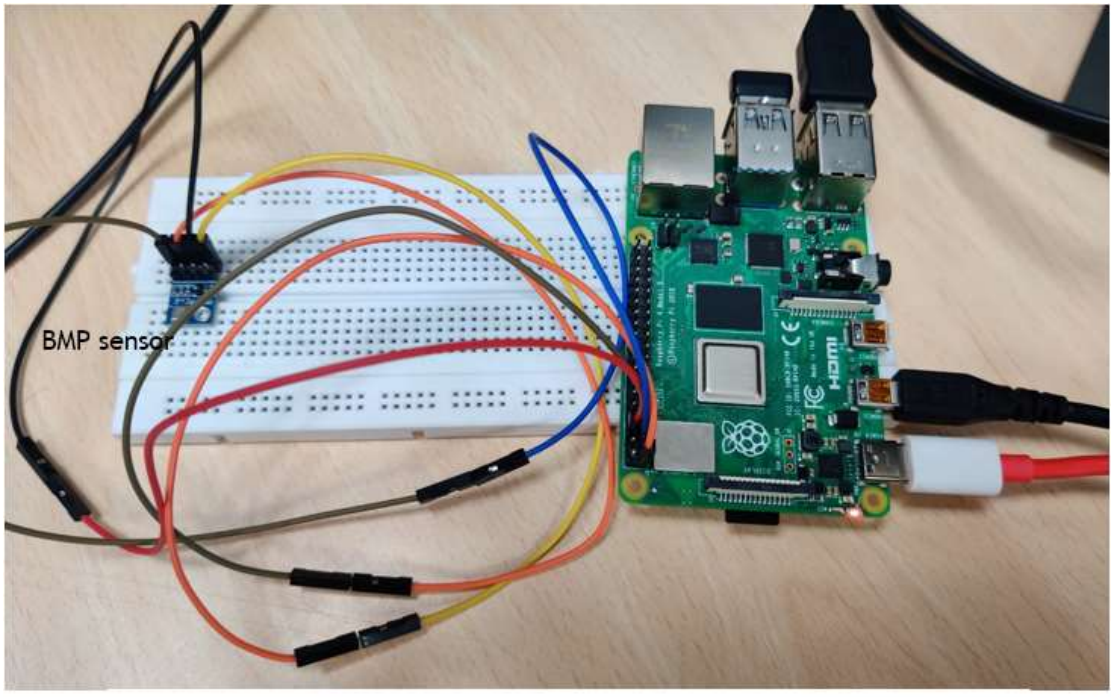
Use Case 2:



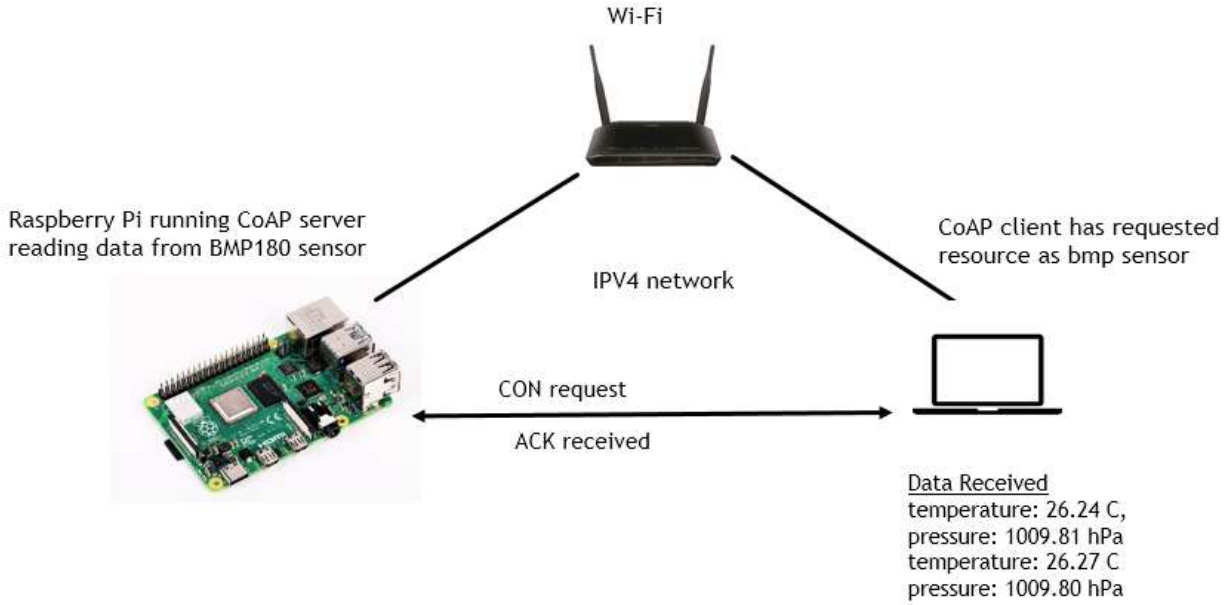
Server sends configuration update to Client to change Sample Rate

5. Raspberry Pi and Bmp180 sensor

Raspberry Pi4B and Bmp180 sensor is being used to collect real-time data. On Raspberry Pi install bmp180 library and add CoAP Server on Raspberry Pi and CoAP client on local machine in same network.



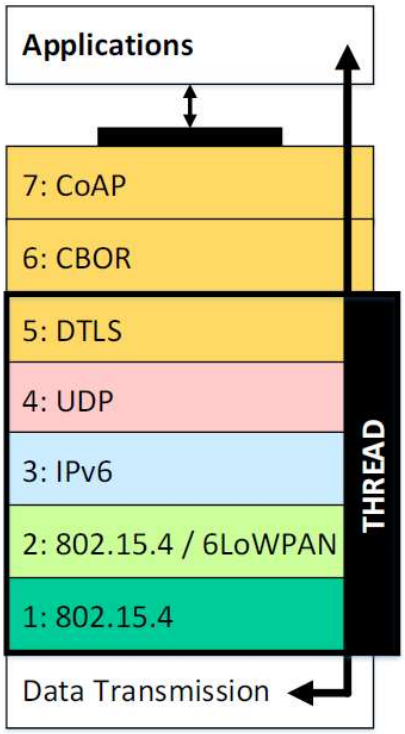
Bmp180 Sensor connected to Raspberry Pi board



Communication between Raspberry Pi with sensor data and Local host

6. OpenThread API for IOT

- Thread is an IPv6-based networking protocol designed for low-power Internet of Things devices in an IEEE 802.15.4-2006 wireless mesh network, commonly called a Wireless Personal Area Network (WPAN).
- Thread is a mesh implementation of 802.15.4 which defines the lower network portion of the stack spanning from layer 1 to layer 5.
- The benefit of Thread over other alternatives is that it includes IP addressing, thus making constrained devices interoperable and compatible with other IP networks.
- The networking layer enables open application protocols, including the use of Restful APIs and web services, such as CoAP.



As part of IOT OpenThread environment is being set up using docker image available on OpenThread website that provides simulation for thread nodes and thread network with CoAP implemented as an Application layer on Raspberry Pi and collected data from BMP180 sensor in Wi-Fi network.

Simulation Link: https://codelabs.developers.google.com/codelabs/openthread-simulation/#6

Demo: \\fil-as-01\BTnE-India\5.I&DS\IDS India\100-Work\IOT\2. WIP\OpenThread Demo for IOT.pptx

7. Roadmap for Further Development

For further enhancement to IOT development Simulated Sensor Acquisition Node (SSUA) and Simulated Sensor Interface Unit (SSUI) needs to be developed.

SSUA is a simulated sensor node web Interface that implements CoAP to add the sensors available in network.

SSUI is Interface that implements CoAP and LwM2M protocol for acquiring data and managing device configuration in network. It is based on eDAU framework that handles the DAR integration in existing framework.

Below is the URL to access further details

\\NOS\Data\I_and_DS_India\IoT\References\IoT for DAR specification.pdf