

UNIVERSITY OF PISA  
Department of Information Engineering

Lino Moises Mediavilla Ponce

# Motion classification application on an Arduino board supporting multiple IoT protocols

Summer School on Enabling Technologies for IIoT

Reviewers: Prof. Sergio Saponara  
Prof. Giuliano Manara

Pisa 2022

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Theoretical Background</b>	<b>3</b>
2.1	MQTT . . . . .	3
2.2	CoAP . . . . .	4
2.3	Web of Things . . . . .	4
<b>3</b>	<b>Implementation</b>	<b>5</b>
3.1	Arduino Application . . . . .	5
3.2	MQTT . . . . .	5
3.3	CoAP . . . . .	6
3.4	Web of Things . . . . .	6
<b>4</b>	<b>Conclusion</b>	<b>8</b>
	<b>References</b>	<b>9</b>

# 1 Introduction

The objective is to develop a motion classification application on an Arduino board and to expose its functionality by:

1. Sending events through an MQTT topic
2. Sending events with Coap protocol
3. Applying the Web of Things paradigm

The motivating lectures within the IIoT Summer School were those of professors Mingozi (Web of Things) and Vallati (Cloud Integration). The rest of the report is structured as follows: first a brief theoretical background about MQTT, CoAP and Web of Things is presented. Then, the instantiation of each of this protocols in the application are described.

## 2 Theoretical Background

### 2.1 MQTT

The Message Queue Telemetry Transport protocol is designed for resource-constrained devices. It uses a publish-subscribe architecture based on topics, and like HTTP, it relies on TCP and IP, but it's designed to have relatively lower overhead [3].

The essential components of an mqtt-based system are a broker, zero or more subscribers and zero or more topics. Figure 1 shows a typical architecture where the data sources are sensors and the basic mqtt components are complemented by a cloud analytics platform.

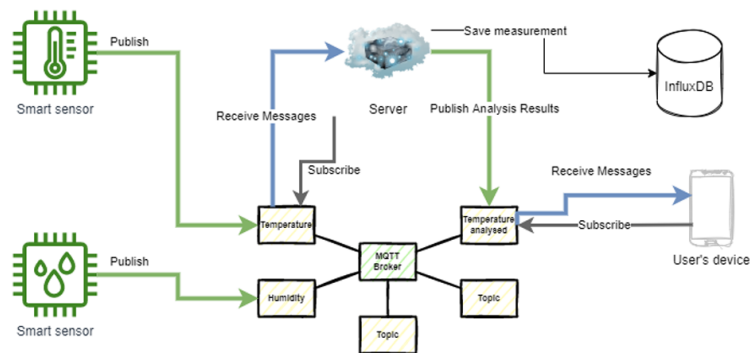


Figure 1. Example of MQTT architecture [1]

## 2.2 CoAP

The Constrained Application Protocol has also been developed as a low-overhead protocol for communication on constrained devices. It supports a request-response model like HTTP, and it is based on the REST architectural style. Additionally, it also supports a publish-subscribe model by allowing clients to subscribe to *resources* [3].

Unlike MQTT, which is based on TCP, CoAP is based on the User Datagram Protocol (UDP) [3]. Figure 2 shows the structure of CoAP requests within a typical client-server interaction.

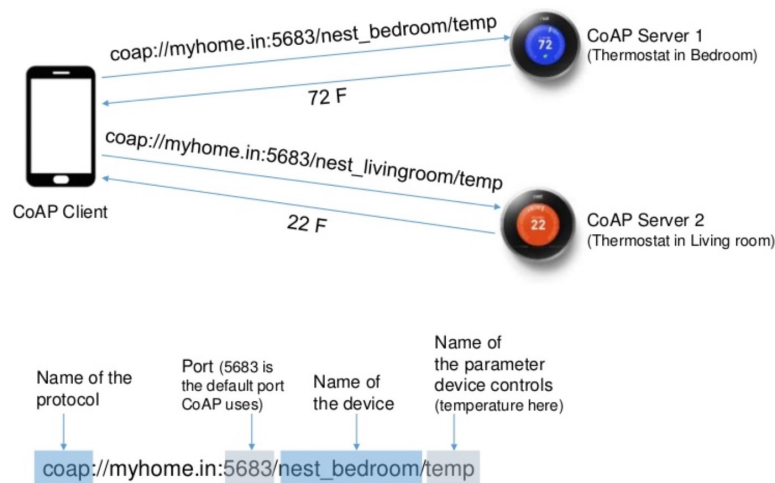


Figure 2. Example of CoAP requests [2]

## 2.3 Web of Things

The Web of Things concept emerges from applying Web Technologies to the Internet of Things to access information and services of physical objects. Each of the interconnected physical objects possesses a digital counterpart, i.e. a *Web Thing*, which exposes the functionality as a RESTful API and is accessed via HTTP [4]. *Web Things* usually have a HTML or JSON representation and an OWL-based semantic description.

WebThings<sup>1</sup> is an open-source implementation of the Web of Things, originally developed by the Mozilla Foundation. It offers a software distribution for smart home gateways, which can be deployed on Raspberry Pi boards and is also available as a docker image. Moreover, it also offers the WebThings framework, which is a collection of re-usable software components available in popular programming languages such as Java, Javascript, Python and also for Arduino boards.

<sup>1</sup><https://webthings.io>

## 3 Implementation

### 3.1 Arduino Application

The movement classification application was deployed on an Arduino Nano rp2040 Connect board<sup>2</sup>, in the form of arduino sketches (one per communication protocol).

The main functionality of the application is to take an interval of samples from the Inertial Measurement Unit and output the label of the current movement that the board is experiencing, i.e. stationary, walking, jogging, biking, or driving. The classification is performed in the IMU's own Machine Learning core, using a pre-trained model<sup>3</sup>.

The complete source code of the application has been made publicly available on github<sup>4</sup>.

### 3.2 MQTT

The Arduino sketch makes use of the ArduinoMqttClient<sup>5</sup> library. On the other hand, the broker is an instance of Eclipse Mosquitto<sup>6</sup>, an open-source Message Broker, that is configured to run as a service.

Figure 3 shows the current movement status being sent through an MQTT topic to the broker running on my personal DigitalOcean cloud server.

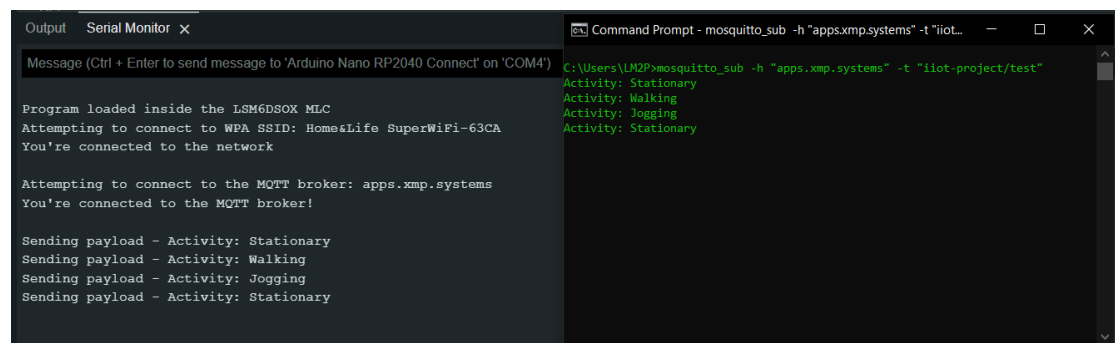


Figure 3. MQTT demo

<sup>2</sup><https://docs.arduino.cc/hardware/nano-rp2040-connect>

<sup>3</sup><https://docs.arduino.cc/tutorials/nano-rp2040-connect/rp2040-imu-advanced>

<sup>4</sup><https://github.com/linomp/rp2040-webthing>

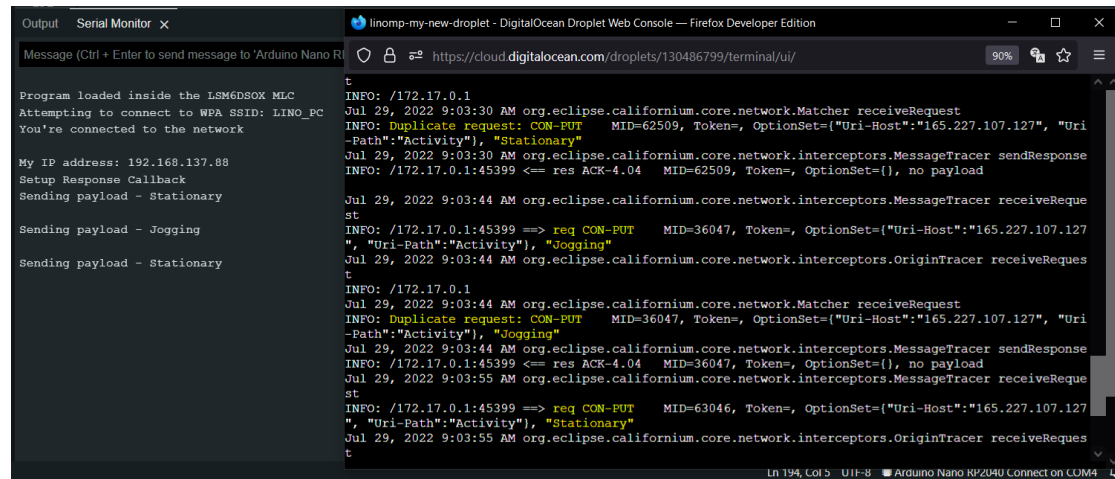
<sup>5</sup><https://www.arduino.cc/reference/en/libraries/arduinomqttclient/>

<sup>6</sup><https://github.com/eclipse/mosquitto>

### 3.3 CoAP

The CoAP simple library<sup>7</sup> was used in the Arduino sketch to implement the client-side. The server, instead, is an instance of Eclipse Californium<sup>8</sup>, an open-source Java implementation of CoAP for IoT Cloud services. It has been conveniently wrapped as a docker image<sup>9</sup>.

Figure 4 shows the current movement status being sent as a CoAP PUT request to the CoAP test server running on my personal DigitalOcean cloud server.



The screenshot shows a terminal window with two panes. The left pane, titled 'Serial Monitor', displays the output of an Arduino Nano connected to an LSM6DSOX MLC. It shows the program loading, network connection attempts, IP address (192.168.137.88), and the sending of payloads for 'Stationary', 'Jogging', and 'Stationary' movements. The right pane shows the logs of a DigitalOcean droplet running Eclipse Californium. It displays CoAP messages including 'Duplicate request: CON-PUT' and 'req CON-PUT' with details like MID, Token, and OptionSet, along with acknowledgments (ACK) and response status (st).

Figure 4. CoAP demo

### 3.4 Web of Things

For the Arduino side, it was necessary to use the library provided by the WebThings Framework<sup>10</sup>. The different movement statuses are exposed as "ThingProperties", which are set to **On** or **Off** depending on which type of movement the board is experiencing.

The Gateway firmware<sup>11</sup> provided by WebThings was installed on a Raspberry Pi 3 Model B+, and after setting it up it was possible to discover the Arduino board as a WebThing, along with its functionality, by simply connecting both boards to the same network. Figure 5 shows the full hardware setup, i.e. the Arduino board and the Raspberry Pi where the gateway firmware was installed.

<sup>7</sup><https://www.arduino.cc/reference/en/libraries/coap-simple-library/>

<sup>8</sup><https://github.com/eclipse-californium/californium>

<sup>9</sup><https://github.com/Pixep/coap-testserver-docker>

<sup>10</sup><https://webthings.io/framework/>

<sup>11</sup><https://webthings.io/gateway/>

Figure 6 shows the WebThings Gateway UI and the representation of the "WebThing" implemented by the application running on the Arduino board, whereas Figure 7 shows the current movement status being sent to a WebThings gateway running on a Raspberry Pi on the local network.

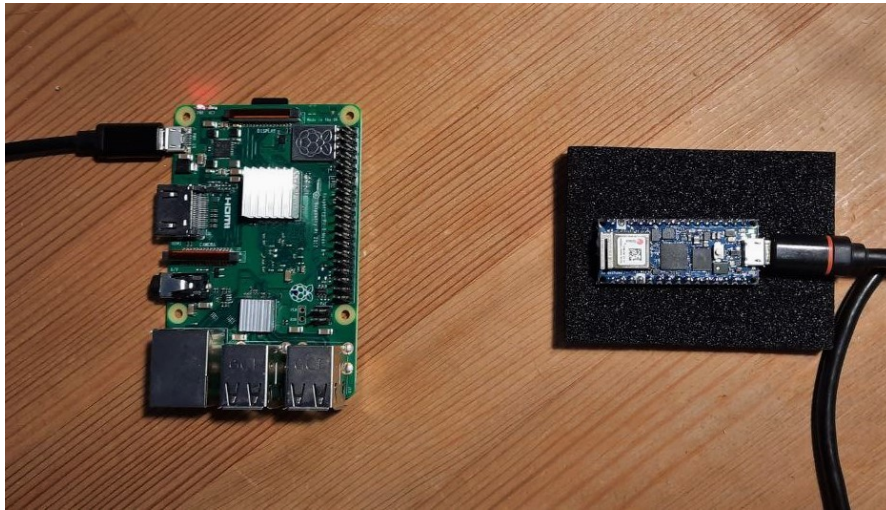


Figure 5. Hardware Setup

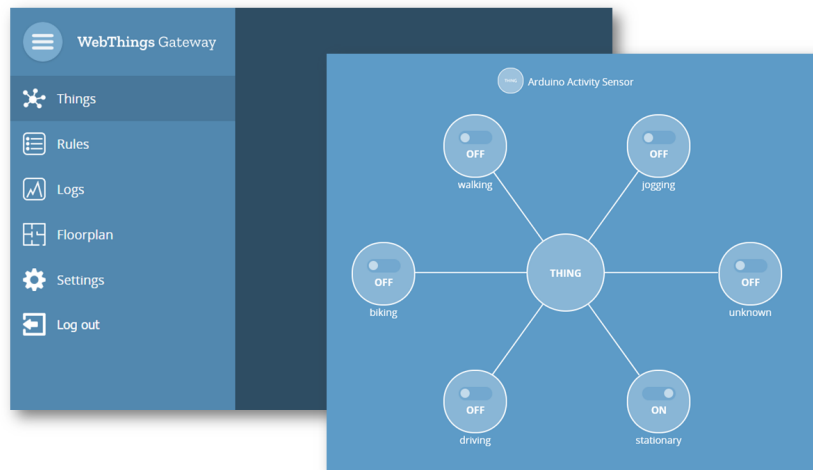


Figure 6. WebThings UI



Figure 7. WebThings demo

## 4 Conclusion

In the present work, a motion classification application was developed, which exposes its functionality via three types of communication protocols studied in the 2022 edition of the IIoT Summer School. Namely: MQTT, CoAP and Web of Things.

The developer experience and robustness of the system varied from one communication protocol to the other, with MQTT being the most straightforward one to set-up and debug, and the WebThings method being the most challenging one to bring up, since it presented several challenges during set-up, and the automatic discovery feature did not always work smoothly.

Nonetheless, this project provided valuable hands-on experience in micro-controller programming, IoT protocols, and server configuration & management.



## References

- [1] Influx Data. Python MQTT Tutorial: Store IoT Metrics with InfluxDB, May 2022.
- [2] Devopedia. Constrained Application Protocol, July 2019.
- [3] Dinesh Thangavel, Xiaoping Ma, Alvin Valera, Hwee-Xian Tan, and Colin Keng-Yan Tan. Performance evaluation of MQTT and CoAP via a common middleware. In *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 1–6, Singapore, April 2014. IEEE.
- [4] Nguyen Khoi Tran, Quan Z. Sheng, Muhammad Ali Babar, and Lina Yao. Searching the Web of Things: State of the Art, Challenges, and Solutions. *ACM Computing Surveys*, 50(4):1–34, July 2018.