CoAP based IoT data transfer from a Raspberry Pi to Cloud

Thomas Lee Scott
School of Computing, Mathematics & Digital Technology
Manchester Metropolitan University
Manchester, United Kingdom
tomscott292@gmail.com

Amna Eleyan
School of Computing, Mathematics & Digital Technology
Manchester Metropolitan University
Manchester, United Kingdom
A.Eleyan@mmu.ac.uk

***Abstract*—This paper describes the development of an Internet of Things (IoT) monitoring system using ThingsBoard IoT platform. ThingsBoard is an open source software tool, which is used to collect, monitor and visualise streams of data received in real-time by sensor devices. The platform can be hosted in the cloud and provides Message Queuing Telemetry Transport (MQTT), The Constrained Application Protocol (CoAP) and HyperText Transfer Protocol (HTTP) protocols support. MQTT and HTTP protocols have mostly been used to develop various IoT systems. However, this paper investigates the use of the CoAP in transmitting sensor data to the cloud. It aims to explore how CoAP fits into the IoT ecosystem and what advantages it offers over other IoT protocols. A CoAP–based IoT architecture is proposed using a Raspberry Pi (RPi) and sensors acting as IoT endpoints. These endpoints will poll sensors (e.g. temperature and humidity) and using CoAP will send the latest data formatted as JavaScript Object Notation (JSON) to the ThingsBoard cloud endpoint at regular intervals. ThingsBoard can create real-time IoT Dashboards for sensors data visualization and share it with users.**

***Index Terms*—Internet of Things, CoAP, M2M, constrained devices, Raspberry Pi board.**

I. Introduction

The reduced cost of low powered small devices, such as the RPi, has made it more accessible to create bespoke systems. This combined with the increasing popularity of home automation allows for these devices to be used in the Internet of Things (IoT).

The IoT can be viewed as a large distributed network consisting of highly dynamic devices [1]. Small low powered "smart" devices can connect and communicate with one another. Some of these devices can contain or communicate with sensors that record real-world data. This data can then be transmitted to other devices allowing them to trigger actions. In this way groups of smart devices can be used to improve day to day situations such as automated houses (thermostats and heating etc.), security and improved monitoring.

The Raspberry Pi [2] is a credit card sized computer developed by the Raspberry Pi Foundation. The RPi's ability to act as a GNU/Linux server and the interfacing services provided by its general purpose I/O pins make it a popular choice of hardware for IoT applications [3].

With 48% of the UK market considering their smartphone as the most important device for Internet access [4], allowing users to use their handheld devices to view and manage their data has become increasingly necessary. Cloud platforms that allow access from any device go a long way to solving this problem. Storing sensor data in the cloud allows for easy access to users from any device as well as allowing for scalable storage.

As these devices are limited in computing power, it is important that the devices communicate efficiently. This paper explores the use of CoAP as a protocol to transmit sensor data from a small, low powered device (RPi) to send sensor data to the cloud.

In this system, a sensor will be attached to the RPi; the RPi will be responsible for taking the data from the sensor and then using the CoAP protocol to transmit this data to the cloud platform.

A. Motivation

With the increasing adoption of IoT systems and technology in day to day life, it becomes increasingly important for the devices to be able to operate at peak efficiency. As there are numerous technologies and standards that allow different IoT devices to communicate, this paper investigates the use of CoAP in order to determine: 1) how CoAP transfers data from an IoT device to another CoAP node. 2) how a RPi can be used as a flexible platform to provide sensor data. 3) how this data can be sent to a cloud platform.

B. Related Work and Contribution

Rode *et al.* [5] carries out a similar investigation, using a RPi device as an IoT node connected to sensors. The RPi collects the data from the sensors and then transmits the data to a HTTP server. The HTTP server acts as a stand in for a cloud platform and recieves the sensor data and displays it to the user. Rode *et al.* [5] proposes using the MQTT protocol to transmit the sensor data from the RPi to the HTTP server. MQTT is a popular IoT protocol developed to specialise in the transfer of data from Wireless Sensor Networks (WSNs) [6]. MQTT works on a publish/subscribe model where the RPi acts a *publisher*, publishing the sensor data to the broker and the HTTP server acts as a *subscriber*. The MQTT broker is responsible for coordinating subscribers to the data and subscribers will usually have to contact the broker explicitly

in order to subscribe [6]. This contrasts to the approach taken in this proposal using CoAP, where each node in the CoAP network acts as both a server and a client in a more traditional HTTP model and nodes within the infrastructure will communicate with one another directly.

Jassas *et al.* [7] has used a RPi connected to sensors to measure patients' body temperature and transmit this data wirelessly to the cloud. In that paper, the data was transmitted to an Amazon Web Services (AWS) cloud computing platform. There the data was stored, mined in order to make decisions, and displayed to the user allowing the data to be updated and reviewed. The data was transmitted from the RPi to the AWS server using Secure Socket Layer (SSL). The development of specialised protocols for constrained devices, such as CoAP could allow these health monitoring RPis to save power, save network bandwidth and potentially receive more readings to process.

Lee *et al.* [8] proposes a RPi combined with a DHT22 sensor to measure the indoor temperature in real-time. This data was transmitted using HTTP to a Representational State Transfer (REST) Application programming interface (API), where the temperature was stored in a database. These temperatures were then used to inform an application replicating the actions of an air conditioner. The use of a RESTful API in this paper would allow the project to easily be adapted to using CoAP to replace HTTP.

Most of the above related IoT architectures are deployed using either MQTT or HTTP. However, this paper proposes a CoAP–based IoT architecture, which uses CoAP to transmit sensor data from RPi to ThingsBoard IoT cloud platform to monitor and visualise sensor devices and share it with users.

II. Background

A. Internet of Things

The Internet of Things (IoT) is an umbrella term used to describe physical 'smart' devices equipped with telecommunication interfaces, connected to one another via the Internet [9]. Whereas the Internet traditionally connected computers, the embedding of electronics into physical objects has allowed the Internet to expand [1]. These devices can contain sensors which will produce data and in some cases, these devices can be controlled remotely. The combination of these devices in a network, especially when the actions of one device are informed by the data from another device, is the foundation of IoT [10]. IoT systems can impact in many areas such as home automation, where devices can work together to automate heating and security aspects of the home [8], and medicine where devices can be used as monitors to provide real-time information about patient health [3].

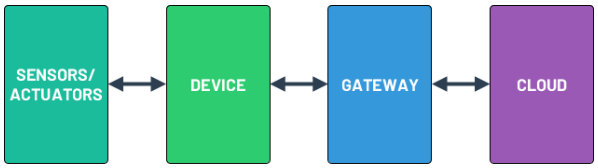

Figure 1: Basic IoT Architecture.

Although IoT lacks a standardized architecture approved by an authorized body [10], Figure 1 illustrates various common components present within any given IoT architecture. The sensor/actuator is the component that will receive data from the real world or perform a physical action. This data is then transmitted to a device which is responsible for the sensor/actuator. The device may be responsible for multiple sensors, as in Jassas *et al.* [7], which connected multiple e-health sensors to a single RPi device. This device will then communicate with a gateway. The gateway is responsible for ensuring that the data is sent to the correct destination. Within CoAP, the gateway would be a router ensuring messages are sent to the correct endpoints. The cloud is the final destination for the sensor data; here, the data will be stored and can be processed for further analysis.

B. The Constrained Application Protocol

The Constrained Application Protocol (CoAP) is a transfer protocol specialised for use with the web, constrained nodes and constrained networks [11]. The protocol is designed for Machine-to-Machine (M2M) applications and is ideally suited for use within the IoT ecosystem. CoAPs features of observable resources, multicasting, M2M discovery make it a better fit for IoT applications than HTTP [12].

CoAP recognises that web services have become dependent on REST architecture and works to implement a subset of REST common with HTTP while optimising for M2M applications [11]. It achieves this by offering built-in discovery, multicast support and asynchronous message exchanges [11].

CoAP uses a compact binary format with a fixed header size of 4 bytes, exchanging messages over User Datagram Protocol (UDP) or Datagram Transport Layer Security (DTLS) to send messages securely. CoAP resources are addressable by Uniform Resource Identifiers (URIs) and can be interacted with through the same methods as HTTP: GET, PUT, POST and DELETE.

With regard to reliability, CoAP offers four types of messages: Confirmable, Non-Confirmable, Acknowledgement and Reset [13]. After a Confirmable request is sent to a CoAP endpoint, the endpoint will respond with an Acknowledgement message. This message can contain the requested data in a 'piggybacked' response. Otherwise, an empty Acknowledgement message is sent and a Confirmable message will be sent once the data is ready. The original requester will then respond with an empty Acknowledgement message to confirm receipt of the data [11].

III. Designed CoAP-based IoT Architecture

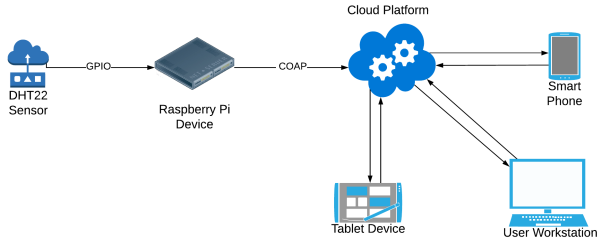

Figure 2: CoAP-based IoT Architecture.

The system shall consist of four main elements: the sensor, the RPi, CoAP and the cloud platform. The sensor will collect the data and pass this to the RPi. The RPi will then be responsible for manipulating the data into a suitable format for transmission via CoAP. The implementation of CoAP will communicate with the cloud platform. The cloud platform will store the data, allowing access to users. The parts of the system are explained below:

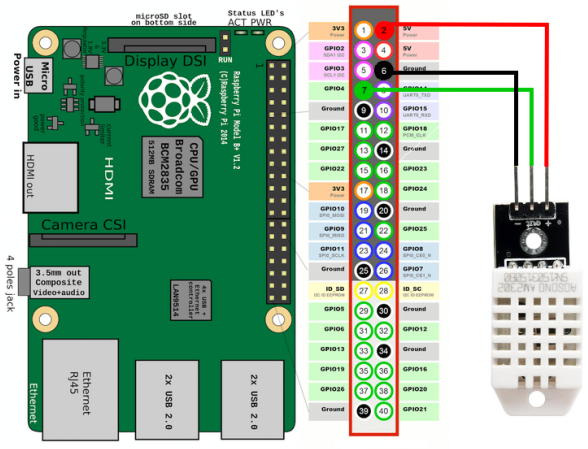

Figure 3: Wiring diagram for connecting the DHT22 sensor to the RPi.

Sensors The DHT22/AM2302 digital relative temperature and humidity sensor will be used to collect temperature and humidity data from the environment. The DHT22 has three pins: One for the 3.3–5.5V power, one for data output and one pin for neutral. The RPi can connect to the DHT22 using the RPi's General Purpose Input Output (GPIO) ports to provide power, a ground connection and an output for the DHT22's data. This connection is shown in Figure 3. The DHT is capable of measuring temperatures in the range of -40 to 80 degrees Celsius and reporting a relative humidity of between 0 to 100% to an accuracy of $\pm 2\%$. A disadvantage of the DHT22 sensor is that it will only report new data once every two seconds, leading to reading being up to two seconds old.

Device The Raspberry Pi 3 Model B is a Linux-based microcomputer. It includes built-in WiFi, GPIO ports, a 1.2GHz Quad-Core processor, MicroSD card slot, memory, video/audio outputs, Ethernet port, and power source [2]. It uses a MicroSD memory card as a boot drive and runs a specialised GNU/Linux distribution named Raspbian [14]. Although the RPi can be attached to a monitor using the built-in HDMI port and controlled with a USB mouse and keyboard, this study will connect to the RPi using SSH. The RPi will run a Python script which will collect the data from the DHT22 sensor and send it to the CoAP endpoint in the cloud.

Gateway As this is a CoAP based architecture and as such mirrors the workings of HTTP the gateway will be the router of the local network. This will be responsible for directing the CoAP messages to the CoAP endpoint identified by the cloud platforms URI.

Cloud This study will make use of an open source cloud platform, ThingsBoard [15]. ThingsBoard is an IoT platform that allows the collection and visualisation of data from IoT devices, a connection from IoT devices using MQTT, CoAP or HTTP, the definition of rules to validate incoming data among other features. The RPi will use the URI provided by the ThingsBoard CoAP API to send the sensor data to. Once ThingsBoard receives the sensor data it will update a dashboard which will show the temperature and humidity data provided by the sensor over time, this dashboard is displayed in Figure 6.

The complete design of this system and how each component will interact is illustrated in Figure 2.

The DHT22 sensor will connect to the RPi using the RPi's on board GPIO ports as shown in Figure 3. The AdaFruit Python DHT library will be used in a Python script to get the sensor data from the DHT22 sensor. The AdaFruit Python DHT library is a library that provides methods to interact with DHT sensors connected to the RPi's GPIO pins. A CoAP endpoint will need to be created on the RPi in order to send the sensor data to the cloud. Using the CoAPthon Python library the script will create a CoAP endpoint on the RPi. The CoAPthon library is a Python implementation of the CoAP protocol.

```
1:  procedure Get sensor data
2:      while running do
3:          Get data from sensor;
4:          if data is returned then
5:              Format data into JSON object;
6:              Create CoAP POST message;
7:              Send CoAP message;
8:          else
9:              Wait interval time and continue;
10:         end if
11:     end while
12: end procedure
```

Figure 4: Get sensor data algorithm

This endpoint will act as an interface for the DHT22 sensor.

At intervals, the script will use the AdaFruit DHT library [16] methods to retrieve the DHT22 sensor's current temperature and humidity readings. This data will then be formatted into JSON in order to be transmitted. Then using the CoAPthon [17], [18] library a CoAP message will be created with the sensors JSON data as the payload. This loop will then repeat while the RPi is active, this process is formalised in Figure 4. This message will then be sent over UDP to a CoAP URI hosted by the cloud platform as illustrated in Figure 5.

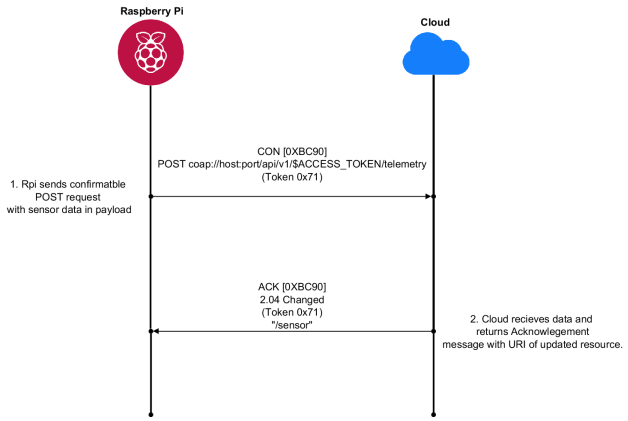

Figure 5: Diagram showing the communication between RPi and the Cloud.

The cloud's responsibility will be to receive the POST requests from the RPi containing the JSON data, format it and display it to the user, who will be accessing the cloud platform via HTTP. Once the ThingsBoard platform has received the request it should send an Acknowledgement message to the CoAP endpoint, containing a 2.01 (Created) Response Code or a 2.04 (Changed) Response Code and the URI of the created / updated resource [11].

IV. CoAP-based IoT Architecture Implementation

To implement the proposed architecture, first, the RPi was loaded with a headless distribution of the Raspbian [14] operating system. Raspbian is a specialised GNU/Linux distribution based on Debian, optimized for the RPi. It comes with over 35,000 packages including the Python programming language. Python will be used to create and run the script that comprises the implementation of the client side system. After setting up the RPi and connecting the DHT22 sensor to the RPi's GPIO pins as per Figure 3, a Python script was constructed to collect the sensor data and, using CoAP, send the data to the ThingsBoard [15] endpoint.

The CoAP client is created using the Python package, CoAPthon [18]. This package contains a `HelperClient` class that takes a CoAP path and port upon initialisation. An object is created using the path to the ThingsBoard telemetry endpoint and the default CoAP port of 5683. This CoAP client object will be used to send the POST message containing the data to the ThingsBoard cloud.

To retrieve the readings from the DHT22 sensor, first, the Adafruit DHT22 library [16] was imported to the script. This library contains a `read_retry()` method that will attempt to read the temperature and humidity data from the DHT22 sensor and return values as floating point decimals; if no reading is available it will try again up to a specified number of retries, defaulting to fifteen. This method is shown in Listing 2.

```
1          {
2              'temperature': 22.6,
3              'humidity': 30.05
4          }
```

Listing 1: Sensor data formatted to JSON for message payload.

```
def get_sensor_data():
    # Try to grab a sensor reading.  Use the
    ↪  read_retry method
    # which will retry up to 15 times to get a
    ↪  sensor reading
    # (waiting 2 seconds between each retry).
    humidity, temperature =
    ↪  Adafruit_DHT.read_retry(_SENSOR,
    ↪  _GPIO_PIN)

    return SensorData(humidity, temperature)
```

Listing 2: Method for getting data from DHT22 sensor.

Once the sensor data is returned, the payload is constructed. The payload consists of the temperature and humidity data formatted into a JSON object, shown in Listing 1. This data is converted using the built-in Python JSON library.

```
def send_data(sensor_data, client, path):
    # Format the data to JSON to be sent
    payload = sensor_data.as_json()
    # Send POST message with payload to endpoint
    response = client.post(path, payload)
    # Print response from cloud to console
    print(response.pretty_print())
```

Listing 3: Method for sending formatted data to the cloud.

With the formatted payload, the CoAP message is sent to the ThingsBoard endpoint; the responding message from the ThingsBoard node is printed to the console. The Python procedure that sends the data to the ThingsBoard endpoint can be seen in Listing 3. Here, the script will wait a specified amount of time and then will repeat the process from retrieving the data. This process is shown in Figure 4.

Once the ThingsBoard cloud receives the data, the data is processed through the ThingsBoard cloud's rules engine. For this study, no extra rules have been applied to the data handling. The default handling of data sent to the telemetry endpoint is to identify the device that is sending the data, store the values and update any dashboards associated with that device. The device is identified by the device access token sent as part of the request. An image of the ThingsBoard dashboard for this study is shown in Figure 6. These dashboards are customisable and allow for the data to be shown in realtime or show a selection of historical results. The dashboard in Figure 6 has four display widgets: one showing the last temperature reading, one showing all the latest temperature

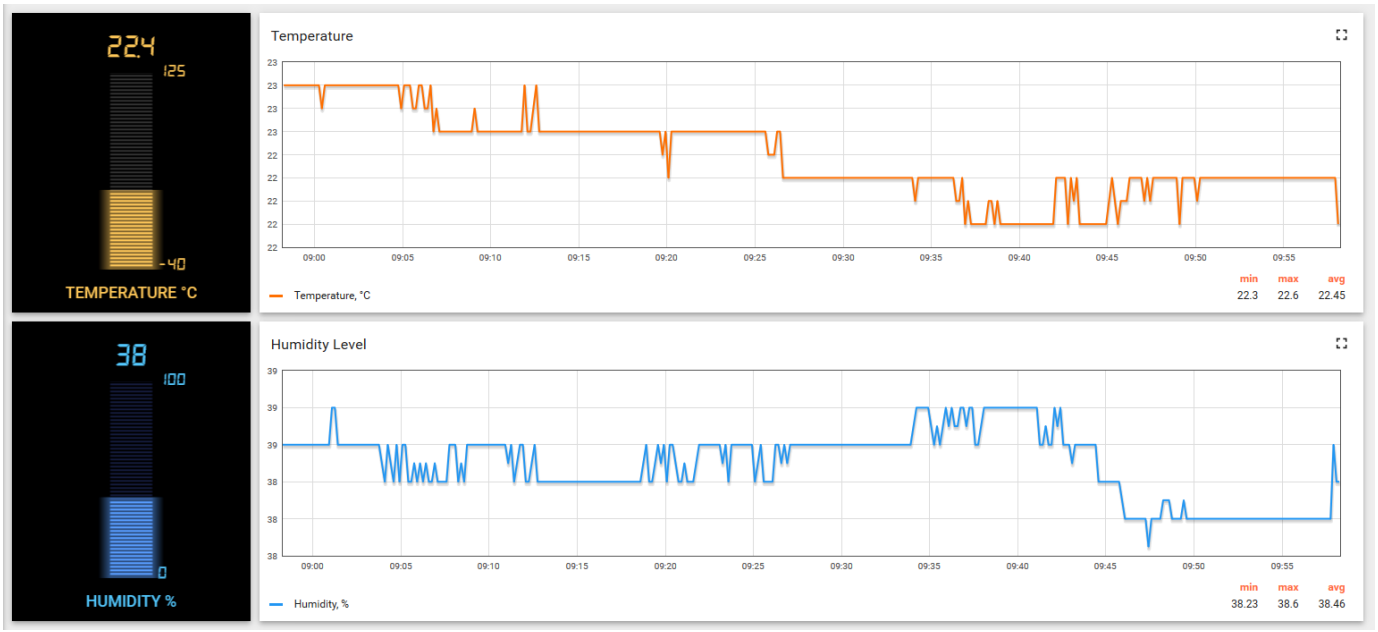

Figure 6: ThingsBoard dashboard created in implementation showing temperature and humidity data.

readings for the previous hour, one showing the last humidity reading and one showing all humidity readings in the last hour.

V. Conclusion and Future Work

The CoAP–based IoT architecture is proposed using a Raspberry Pi, temperature and humidity sensors and ThingsBoard IoT platform to monitor and visualise sensors data. The CoAP protocol is used to send the temperature and humidity data formatted as JavaScript Object Notation (JSON) to the ThingsBoard cloud endpoint at regular intervals. ThingsBoard is deployed to monitor and visualise data by creating IoT Dashboards and updating in real-time.

The RPi has been connected to a DHT22 temperature and humidity sensor using the RPis GPIO pins. A Python script running on the RPi uses an external library provided by AdaFruit to poll the sensor at intervals and obtain the current temperature and humidity. Using the CoAPthon library, a CoAP client is created and used to send a CoAP POST message containing the sensor data to a ThingsBoard telemetry endpoint. This data is then displayed to the user in a dashboard.

Further work can be done with the data once it has been sent to the cloud. The ThingsBoard platform offers many options regarding actions performed, based on data received from the IoT devices. Setting up alerts when data falls outside of expected ranges, and sending messages to other IoT devices, as a result, would allow this study to be applied to other uses such as smart home automation. It would also be of interest to connect an actuator to the RPi and see how this could be manipulated based on CoAP messages sent from the cloud.

References

[1] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, Sep. 1, 2012.

[2] R. Pi, "Model B," *Raspberrypi. org. Saatavissa: https://www. raspberrypi. org/products/raspberry-pi-3-model-b/. Hakupäivä*, vol. 6, p. 3, 2018.

[3] R. Kumar and M. P. Rajasekaran, "An IoT based patient monitoring system using raspberry Pi," in *2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16)*, Jan. 2016, pp. 1–4.

[4] Ofcom. (Aug. 1, 2018). The Communications Market 2018: Narrative report.

[5] S. D. Rode, S. Sagrolikar, and M. S. Kulkarni, "IOT based Raspberry PI home Automation Using Cloud," vol. 3, no. 2, p. 5, 2017.

[6] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks," in *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, Jan. 2008, pp. 791–798.

[7] M. S. Jassas, A. A. Qasem, and Q. H. Mahmoud, "A smart system connecting e-health sensors and the cloud," in *2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE)*, May 2015, pp. 712–716.

[8] C. Lee, S. Park, Y. Jung, Y. Lee, and M. Mathews, "Internet of Things: Technology to Enable the Elderly," in

2018 Second IEEE International Conference on Robotic Computing (IRC), Jan. 2018, pp. 358–362.

[9] M. Centenaro, L. Vangelista, A. Zanella, and M. Zorzi, "Long-range communications in unlicensed bands: The rising stars in the IoT and smart city scenarios," *IEEE Wireless Communications*, vol. 23, no. 5, pp. 60–67, Oct. 2016.

[10] R. Minerva, A. Biru, and D. Rotondi, "Towards a definition of the Internet of Things (IoT)," *IEEE Internet Initiative*, vol. 1, pp. 1–86, 2015.

[11] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC Editor, RFC7252, Jun. 2014.

[12] M. Kovatsch, M. Lanter, and Z. Shelby, "Californium: Scalable cloud services for the Internet of Things with CoAP," in *2014 International Conference on the Internet of Things (IOT)*, Oct. 2014, pp. 1–6.

[13] P. Bellavista and A. Zanni, "Towards better scalability for IoT-cloud interactions via combined exploitation of MQTT and CoAP," in *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a Better Tomorrow (RTSI)*, Bologna, Italy: IEEE, Sep. 2016, pp. 1–6.

[14] Raspbian, *Raspbian GNU/Linux 9.6 (stretch)*, version 9.6, Raspbian, Nov. 2018. [Online]. Available: https://www.raspbian.org (visited on 12/20/2018).

[15] ThingsBoard, Inc., *ThingsBoard*, version 2.2.0, ThingsBoard, Inc., Nov. 29, 2018. [Online]. Available: https://thingsboard.io (visited on 12/20/2018).

[16] Adafruit, *Adafruit_Python_DHT*, version 1.4.0, Adafruit, Nov. 7, 2018. [Online]. Available: https://github.com/adafruit/Adafruit_Python_DHT (visited on 12/20/2018).

[17] G. Tanganelli, C. Vallati, and E. Mingozzi, "CoAPthon: Easy development of CoAP-based IoT applications with Python," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, Dec. 2015, pp. 63–68.

[18] G. Tanganelli, *CoAPthon3*, version 1.0.1, Jan. 25, 2018. [Online]. Available: https://github.com/Tanganelli/CoAPthon3 (visited on 12/20/2018).