CoAP based IoT data transfer from a Raspberry Pi to Cloud

Thomas Scott

20th November 2018

1 Abstract

This research investigates the use of The Constrained Application Protocol (CoAP) in transmitting sensor data to the cloud. It aims to explore how CoAP fits into the Internet of Things (IoT) ecosystem and what advantages, if any, it offers over other IoT protocols. A framework is proposed using a Raspberry Pi (RPi) and sensor acting as an IoT endpoint. This endpoint will allow for CoAP requests and will poll the sensor and return the latest data as JavaScript Object Notation (JSON). The endpoint will be polled from a cloud service, this service will then display the data to the user.

2 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 IoT.

The IoT can be viewed as a large distributed network comprising of highly dynamic devices (Miorandi et al. 2012). 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 (Pi 2018) 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. (Kumar & Rajasekaran 2016)

With 48% of the UK market considering their smartphone as the most important device for internet access (Ofcom 2018) 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 the 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.

3 The Constrained Application Protocol

The Constrained Application Protocol (CoAP) is a transfer protocol specialised for use with the web, constrained nodes and constrained networks (Shelby et al. 2014). The protocol is designed for Machine-to-Machine (M2M) applications and is ideally suited for use within the IoT ecosystem.

CoAP recognises that web services have become dependant in Representational State Transfer (REST) architecture and works to implement a subset of REST common with HTTP while optimising for M2M applications (Shelby et al. 2014). It achieves this by offering built-in discovery, multicast support and asynchronous message exchanges (Shelby et al. 2014).

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 HyperText Transfer Protocol (HTTP): GET, PUT, POST and DELETE.

CoAPs features of observable resources, multicasting, M2M discovery make it a better fit for IoT applications than HTTP (Kovatsch et al. 2014).

4 Design

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.

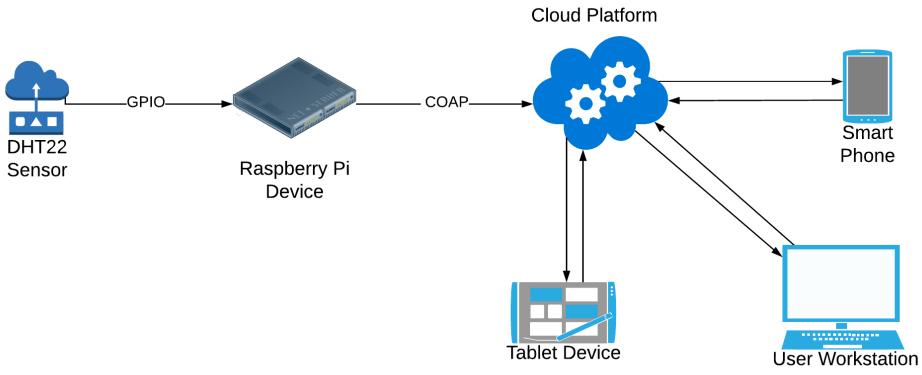

Figure 1: The project infrastructure.

The DHT22 sensor will connect to the RPi using the RPi's on board General Purpose Input Output (GPIO) ports as shown in Figure 2. Using the AdaFruit Python DHT library, a library that provides methods to interact with DHT sensors connected to the RPi's GPIO pins, and the CoAPthon library, a Python implementation of the CoAP protocol, a Python script will create a CoAP endpoint. This endpoint will expose the DHT methods for getting the latest data. When this endpoint receives a request the data will be current temperature and humidity will be retrieved from the sensor. Once this data has been collected it will be formatted in to JSON using Python. This formatted data will then be the payload for the response. The cloud service will be responsible for displaying the data from the sensor and making the request to the RPi to get up to date information. This process is shown in Figure 3.

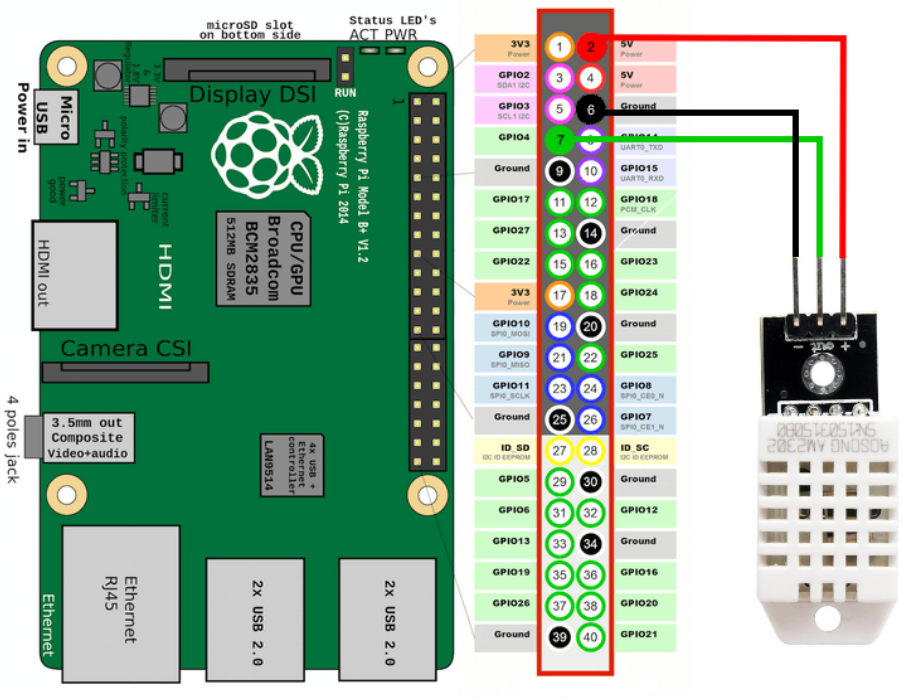

Figure 2: Wiring diagram for connecting the sensor to the RPi.

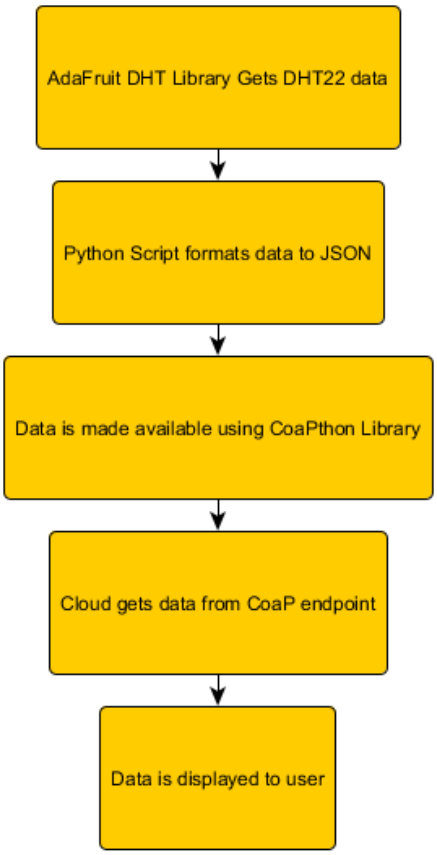

Figure 3: Shows the flow of data from sensor to user.

5 Methodology

The aim of the proposed system is to investigate the implementation of CoAP on a RPi and how CoAP can be used to transmit data to the cloud.

To achieve this a CoAP endpoint will need to be created on the RPi. The RPi will collect sensor data at intervals and store them locally on the device. The RPi will act as a CoAP server that will respond to REST GET requests with the sensor data and the time the reading was taken in a JSON format.

The clouds responsibility will be to send the GET requests to the CoAP endpoint hosted on the RPi and to receive and store the data returned.

The cloud should send a confirmable request to the CoAP endpoint. The CoAP endpoint should then respond with an acknowledgement. If the data is available, the data should be "piggybacked" to this response, as shown in Figure Figure 4, if not the data should be returned in a confirmable message containing the data. In this case the cloud client will then send it's own acknowledgement message to the CoAP endpoint. This is shown in Figure 5.

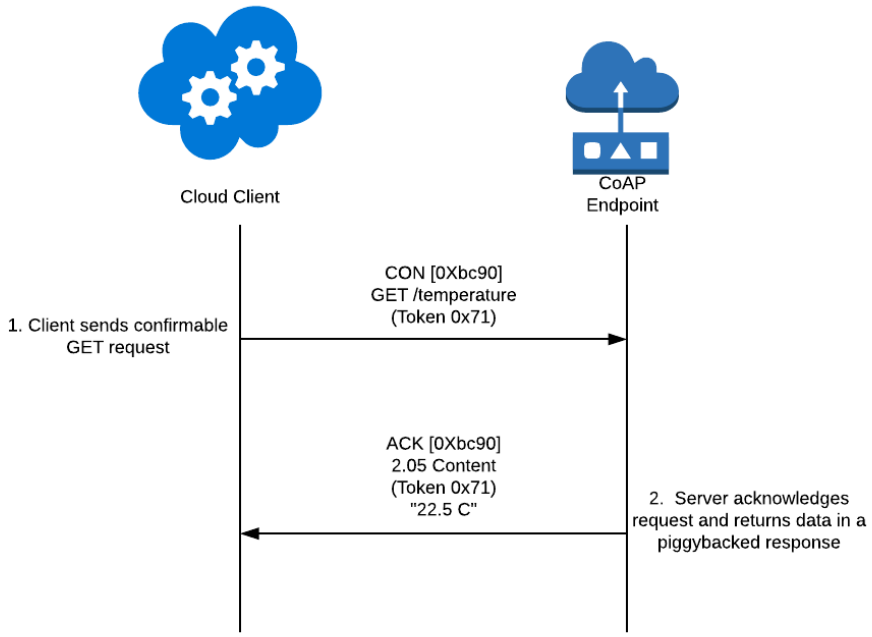

Figure 4: Example CoAP GET request with piggybacked response. (Shelby et al. 2014)

With CoAP endpoints acting as both a client, that sends requests, and a server implementation of CoAP will be needed in each the RPi and the cloud platform. The RPi will then regularly retrieve readings from the sensor and send a POST request at intervals to the cloud CoAP endpoint. This approach will be contrasted with the cloud server observing the REST endpoint on the RPi. The results of both approaches will be evaluated.

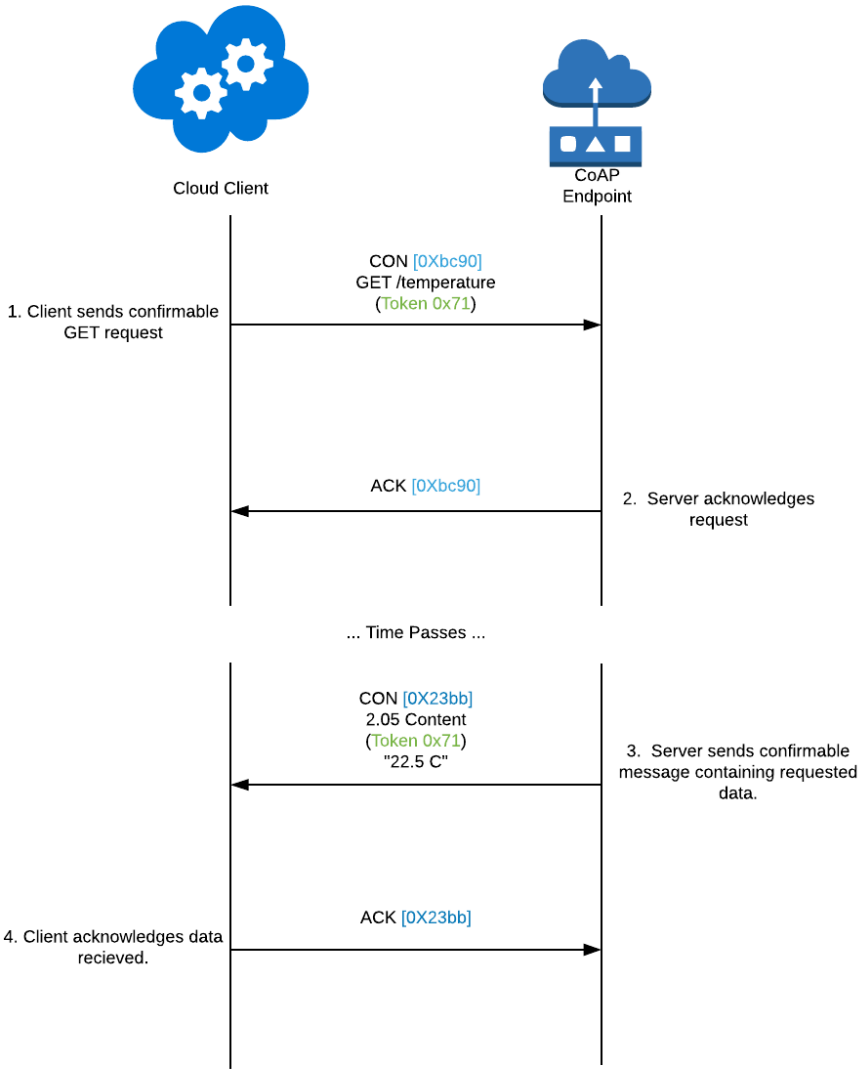

Figure 5: Example CoAP GET request without piggybacked response. (Shelby et al. 2014)

6 Software and Hardware specification

6.1 Hardware

1. Raspberry Pi 3

   Raspberry Pi 3 Model B, includes built in WiFi, GPIO ports and a 1.2GHz Quad-Core processor.

2. Micro SD card

   Used to load operating system onto the RPi.

3. Power cord

   Supplies power to the RPi.

4. DHT22 temperature and humidity sensor

   Connects to the RPi using the RPi's GPIO ports. Will be used to provide data.

6.2 Software

1. Python 3

   The Python programming language will be used to create the scripts and software needed on the RPi. This is due to the languages popularity when creating projects on the RPi and the languages wide selection of networking packages.

2. CoAPthon

   Python implementation of CoAP. Licensed under the MIT license. Github.

3. AdaFruit Python DHT

   Python library to retrieve sensor data from the DHT22. Github.

4. Git Version Control

   Source code version control system to allow for adjustments to the code.

5. Cloud Platform

   Will act as an end point to the RPi where the sensor data will be stored.

7 Conclusion

This research investigates the use of CoAP in transmitting sensor data to the cloud. It aims to explore how CoAP fits into the IoT ecosystem and what advantages, if any, it offers to other protocols. It also shows how a RPi can be used with Python to create an IoT device and connect to the cloud.

Appendices

A Python Code to Get DHT22 Data

```
#!/usr/bin/env python3
"""
Script to get sensor data from DHT22 and provide the data
↪ through a CoAP endpoint
"""

__author__ = "Tom Scott"

import Adafruit_DHT

_SENSOR = Adafruit_DHT.AM2302
_GPIO_PIN = 4

class SensorData(object):
    """
    Data object containing humidity and temperature data.
    """

    def __init__(self, humidity, temperature):
        """
        Set temperature and humidity
        """
        self.humidity = humidity
        self.temperature = temperature

    def has_data(self):
```

```
26          """
27          Returns True if humidity and temperature are not None.
28          """
29          return self.humidity is not None and self.temperature
            ↪  is not None
30
31  def get_sensor_data():
32      """
33      Returns a SensorData object containing the latest data
    ↪   from the DHT22 sensor.
34      The method will 15 times to get data, if there is still no
    ↪   data available
35      it will return an empty SensorData object
36      """
37
38      # Try to grab a sensor reading.  Use the read_retry method
        ↪  which will retry up
39      # to 15 times to get a sensor reading (waiting 2 seconds
        ↪  between each retry).
40      humidity, temperature = Adafruit_DHT.read_retry(_SENSOR,
        ↪  _GPIO_PIN)
41
42      return SensorData(humidity, temperature)
43
44  def main():
45      """ Main entry point of the app """
46
47      # Note that sometimes you won't get a reading and
48      # the results will be null (because Linux can't
49      # guarantee the timing of calls to read the sensor).
50      sensor_data = get_sensor_data()
51
52      if sensor_data.has_data():
53          print('Temp={0:0.1f}*
            ↪  Humidity={1:0.1f}%'.format(sensor_data.temperature,
            ↪  sensor_data.humidity))
54
55
56  if __name__ == "__main__":
57      """ This is executed when run from the command line """
```

```
58      main()
```

References

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

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

Miorandi, D., Sicari, S., De Pellegrini, F. & Chlamtac, I. (2012), 'Internet of things: Vision, applications and research challenges', *Ad Hoc Networks* **10**(7), 1497–1516.
URL: *http://www.sciencedirect.com/science/article/pii/S1570870512000674*

Ofcom (2018), 'The Communications Market 2018: Narrative report'.
URL: *https://www.ofcom.org.uk/research-and-data/multi-sector-research/cmr/cmr-2018/report*

Pi, R. (2018), 'model B', *Raspberrypi. org. Saatavissa: https://www. raspberrypi. org/products/raspberry-pi-3-model-b/. Hakupäivä* **6**, 3.

Shelby, Z., Hartke, K. & Bormann, C. (2014), The Constrained Application Protocol (CoAP), Technical Report RFC7252, RFC Editor.
URL: *https://www.rfc-editor.org/info/rfc7252*