Development of an IoT Device With MQTT Cloud Communications

David C

CMP408: IoT and Cloud Secure Development

2024/2025

Note that Information contained in this document is for educational purposes.

1 CONTENTS

2 Introduction 2
2.1 Background 2
2.2 Aim 2
3 Procedure 3
3.1 Overview of Procedure 3
3.2 Procedure 3
3.2.1 sendMqtt.py 3
3.2.2 getSysInfo.py 4
3.2.3 buttonListener.c 4
3.2.4 userspaceListener.c 5
4 Discussion 7
4.1 General Discussion 7
4.2 Conclusion 7
5 Bibliography 8
Appendices 9
5.1 Appendix A – Flow Diagram 9

2 INTRODUCTION

2.1 BACKGROUND

Internet of Things (IoT) and cloud computing uses and collects data across IoT devices and utilises cloud infrastructure for storage, processing and analysis (Hewlett Packard Enterprise, 2024). As we enter a new age of computing where artificial intelligence is integrating into everything that we use, the need for businesses to take advantage of emerging technology such as smart devices and cloud computing is sky rocketing. Over the past 10 years the share of business data stored in the cloud has doubled from 30% of businesses storing data in the cloud to 60% indicated in recent research (Edge Delta, 2024).

The ability to storage data in the cloud is a cheaper and more efficient option for businesses as they are not having to maintain the hardware themselves, nor the costs for the upkeep of servers and their security but instead shifting these costs to the organisations offering their services. Unfortunately, whilst businesses are utilising cloud services there are still legacy components in most organisations that won't be moved to the cloud due to the layout of these organisations systems, this is where smart devices bridge the gap. Allowing for systems that could not be moved to the cloud for logistical reasons to connect through this middleman. Although it should be considered, given the increase in systems now responsible for what was originally controlled by one device, how does security fit into this?

The developer aims that throughout this report to highlight their success in creating a solution that highlights the importance of transferring data securely between devices whilst allowing for key functionalities that organisations require on a daily basis to operate as intended. The report will document the developers aims and procedure, highlighting the importance of each component used throughout the project and discuss the security measures in place.

2.2 AIM

The developer aimed throughout this project to create a project that would allow for cloud and IoT components to communicate efficiently whilst also maintaining a secure channel to ensure security throughout the entire process. The project aims to communicate JSON data across the MQTT protocol to be uploaded to the cloud upon the press of a button directly connected to the Raspberry Pi used throughout this project. To allow to the developer to measure their success when evaluating this project, the project has been split into three main aims:

- Create scripts to store system information and upload it to the cloud
- Create system architecture to allow for a physical component to be altered because of the projects intended functionality, or responsible for the running of the project's functionality
- Develop a user space application that acts as the middleman between the system architecture and scripts.

3 PROCEDURE

3.1 OVERVIEW OF PROCEDURE

Whilst the developer has not followed a specific methodology throughout this project, they feel that it is worth mentioning the general structure of this project and therefore the necessary steps to ensure that each section is able to work in the most efficient way possible. The project consists of a Linux Kernel Module (LKM) which acts as the back-end listener for this project, maintaining control of what should happen depending on the user's interaction with the physical components present. The LKM speaks directly to a user space application referred to as "usListener". usListener is triggered when the individual involved in the project presses the button, triggering a signal from the LKM that is passed to the application which then controls the additional scripts in the project, sendMqtt.py and getSysInfo.py. getSysInfo.py is responsible for conducting the checks against the system and storing the necessary data into a JSON format which is then stored into a file on the system under "usage.json" whilst sendMqtt.py is responsible for connecting to the cloud and uploading the data stored in "usage.json" to the cloud. Best practices are taken throughout the programming languages used throughout this project to ensure that there are no credentials leaked, chances of memory leak or overreach from any program to affect other areas of the filesystem and integrity of the Raspberry Pi. A flow diagram is available at Appendix A – Flow Diagram.

3.2 PROCEDURE

Our procedure starts with the two Python scripts that are interacted with through usListener. The developer decided to begin their procedure with these Python scripts so that once the functionality of the cloud and software component was working, they could begin building the IoT and hardware components afterwards.

3.2.1 sendMqtt.py

This scripts pure functionality is to connect to the AWS MQTT topic and upload the data provided and then close its connection. The script contains little complexity as it calls certificates to verify itself through AWS' security systems before posting the data present to the topic defined by the developer and then closing the connection.

The developer has configured their AWS MQTT topic to work based on "Things". Things are representations of IoT devices and effectively show AWS' systems that they are communicating with a specific device (Amazon Web Services, 2024). When creating their thing for this project, the developer also created a policy for the certificates they were downloading. This policy would allow any devices

operating under the generated certificates to have full access to any resources with full permissions over them. The full policy can be seen in Figure 1.

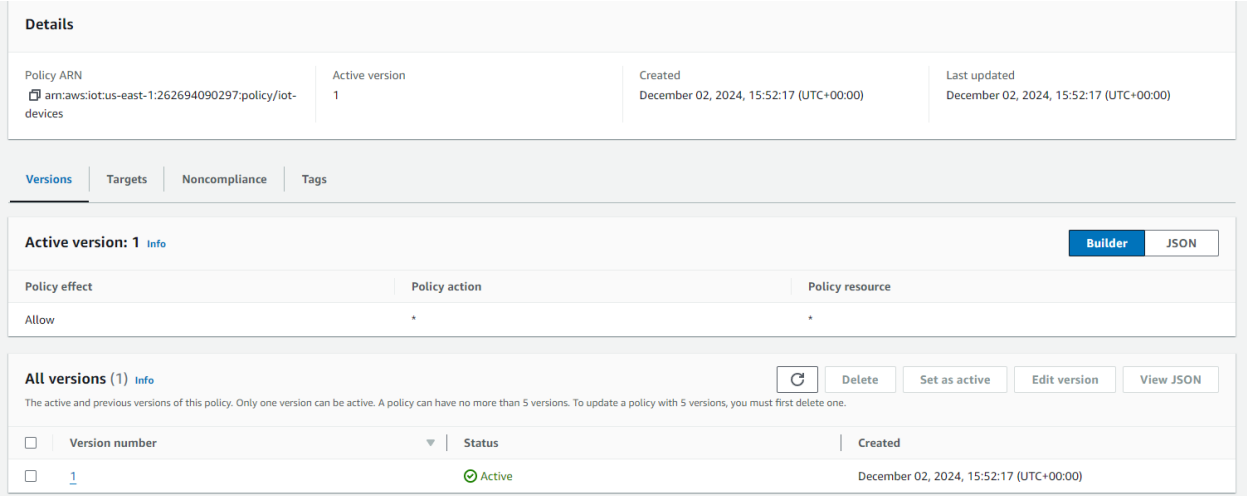

Figure 1 - AWS IoT Policy

The developer was then able to head to the "MQTT Test Client" whereby they were able to receive their endpoint and then began coding their solution, which connected using the information mentioned before and sent the JSON data retrieved from getSysInfo.py to the cloud.

3.2.2 getSysInfo.py

This script is responsible for collecting all of the information that is sent to the cloud and presented on the website. A majority of this is done through the "psutil" library which is responsible for retrieving information on running processes and system utilisation (Rodola & Rodola, 2024). The script collects the necessary information, converting the necessary data into readable formats before storing it in a JSON format and utilising the "json" library to format it in such a way, outputting to "usage.json".

3.2.3 buttonListener.c

Following the creation of the Python scripts, the developer shifted their focus to the LKM. The program starts with its initialisation by which checks are completed versus a defined pin to ensure that the pin is valid and can be altered. Further checks and setup functions are ran to provide the button and LKM with the necessary functionality to operate with the necessary functions. An interrupt request handler is created and serves its purpose of when the LKM is interrupted, due to the press of the button, indicating that the user is wanting to execute the user space listener and handling the interrupt appropriately.

Following the successful initialisation, the LKM waits for a signal and once triggered, activates the "signalHandler" function which, assuming there has not been another button press within quick succession, passes the signal on to the application running in the user space which begins the process

highlighted in the following section. An example of this happening can be seen in Figure 2 - LKM and User space application communicating Figure 2.

```
[  +0.012625] Loading buttonListener module
[  +0.000683] Module initalised.
[  +0.000016] Pin is mapped to IRQ number160
[ +23.370925] Button pressed! Sending signal to userspace
[Dec15 09:08] Buttong pressed.
[  +0.000048] Userspace app with PID 1388 is registered
[Dec15 09:09] Button pressed! Sending signal to userspace

pi@raspberrypi:~/proj $ sudo ./listener
Running under Process ID: 1388
Setup complete, waiting for signals.
Signal received, running scripts.
/home/pi/proj/sendMqtt.py:9: DeprecationWarning: Callback API version 1 is deprecated, update to latest ve
rsion
  client = mqtt.Client()
Connected to: 0
Scripts complete, check for output.
```

Figure 2 - LKM and User space application communicating

3.2.4 userspaceListener.c

Finally, this program is responsible for waiting for the signal passed from the LKM to activate. When the signal from the LKM is passed to the interrupt request handler (IRQ) this program activates, alerting the user that a signal has been received and then running both Python scripts previously mentioned. This is effectively updating the JSON file and therefore the website whilst simultaneously uploading it to the cloud. This can be seen happening in Figure 3 & Figure 4.

System Information

Timetstamp:	2024-12-15 09:11:41.054852
Hostname:	raspberrypi
Architecture:	Linux-5.4.51+-armv6l-with-debian-10.4
Total Memory:	432MB
Memory Available:	287MB
Percentage of Memory Being Used:	33.6%
Total Disk Space:	14356MB
Percentage of Disk Being Used:	43.3%
Bytes Sent:	0MB
Bytes Received:	0MB

Figure 3 - Data being updated on the website simultaneously

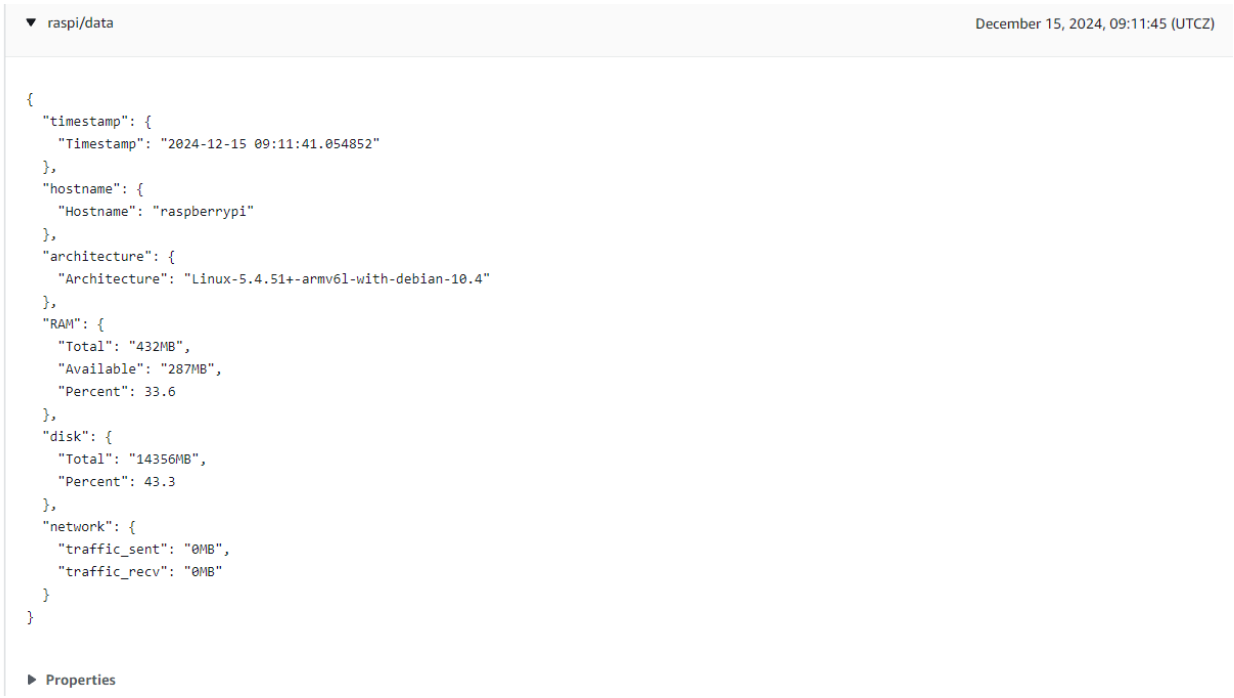

Figure 4 – Data being updated on the cloud simultaneously

4 DISCUSSION

4.1 GENERAL DISCUSSION

Upon reviewing their project, the developer believes that they achieved the aims that they had set for themselves with the creation of two Python scripts that were able of retrieving system information and uploading them to the cloud. At a bare minimum this project would be able to operate with these two scripts being ran by the user.

Secondly, the developer believes that they were able to successfully develop a piece of system architecture in the form of an LKM that would either alter a physical component or run the aforementioned scripts due to the interaction with a physical component, which the latter being the option decided by the developer with the implementation of a button that would run the scripts.

The final aim set by the developer was to develop a user space application that would act as a middleman between their other aims, which they believe was also successful meaning that, assuming the LKM and user space application were previously running, any user would be able to interact and see the results from this project.

4.2 CONCLUSION

Overall, the developer believes that they were able to successfully create a project that combines Cloud security and IoT security to develop a solution to communicate data efficiently through the use of the MQTT protocol. An LKM is used to control a button which, when pressed, activates the functionality of this project and is comparable to the real-world functionality of smart devices therefore highlighting this projects real-world similarity.

5 BIBLIOGRAPHY

Amazon Web Services, 2024. *Thing types.* [Online]
Available at: https://docs.aws.amazon.com/iot/latest/developerguide/thing-types.html
[Accessed 14 December 2024].

b. & Rodola, G., 2024. *psutil 6.1.0.* [Online]
Available at: https://pypi.org/project/psutil/
[Accessed 15 Decembe 2024].

Edge Delta, 2024. *How Many Companies Use Cloud Computing in 2024? [10 Statistics and Insights].* [Online]
Available at: https://edgedelta.com/company/blog/how-many-companies-use-cloud-computing-in-2024#:~:text=94%25%20of%20major%20companies%20worldwide,data%20stored%20in%20the%20cloud.
[Accessed 15 December 2024].

Hewlett Packard Enterprise, 2024. *IoT Cloud Computing.* [Online]
Available at: https://www.hpe.com/in/en/what-is/iot-cloud-computing.html
[Accessed 15 December 2024].

(2022). Thing Ontologies for the Semantic Web of Things. Available from: 10.1109/iisa56318.2022.9904401

Harshali, Rohit, Kadaskar., Vaibhavi, Ramesh, Kamthe. (2024). An overview of AWS. International Journal of Scientific Research in Modern Science and Technology, 3(7), 22-30. Available from: 10.59828/ijsrmst.v3i7.223

Oroos, Arshi., Aryan, Chaudhary. (2024). Fundamental Concepts of IoT. 42-69. Available from: 10.1201/9781032656694-2

APPENDICES

5.1 APPENDIX A – FLOW DIAGRAM