DESIGNING A SYSTEM FOR SMART CONTROLLED USING AWS

[1]CHUNG-YEN WU, [2]KUO-HSUAN HUANG, [3]SHANG-LIN HSIEH

[1,2,3] Department of Computer Science and Engineering, Tatung University, Taipei, Taiwan
E-mail: [1]d10506001@ms.ttu.edu.tw, [2]khhuang@ttu.edu.tw, [3]slhsieh@ttu.edu.tw

Abstract - Recently, IoT technologies is gaining increasing attention with the improvement of networks. However,the network will be unstable in some scenario. In order to ensure the reliability of system for IoT, we designed a system that can control smart and traditional devices offline. In this paper, we presented the design and implementation of a cloud-based IoT infrastructure in AWS. Then, we show that AWS, MQTT and Greengrass are good for unstable network scenario.

Keywords - Greengrass, IoT, Smart scenario, AWS, Offline.

I. INTRODUCTION

With the advancement of technology and the improvement of wireless network technology, the Internet-of-Thing (IoT) devices can be communicated with other devices on the Internet. Most of devices have the ability to connect to the Internet in IoTs. They can provide services that managed, detected and controlled with device, machine and personal, and even the remote control, positioning, anti-theft, etc., for appliances or electronics. Thus, the IoT emphasized that the integration with devices and application inmarket.

In the future, the IoT is expected to have important applications to change human's lifestyle with home-field and business-field. Gartner[1] predicts that will be more than 21 billon devices will be connected to the Internet by 2020. It shown that the IoT development is very rapid. In the development history of IoT, the IoT is evolving from devices with simple network connectivity into a collection of interconnected smart devices. These devices bring endless possibilities for innovative scenarios. As smart devices increase, need a platform that consist of a number of objects connected around the world and provides several services and applications such as integration, security and device management. Current IoT platforms offer for the customers some competitive advantages and features to make our select. According to our survey, Amazon Web Services (AWS) is good choose as our cloud platform. They provide application of consistency, reliability, flexibility, scalability, customization, security and third-party APIs.

The remainder of the paper is organized as it follows. Section II, present an architecture and properties of the system. Section III, we describe the infrastructure's components and present the design. Then, in section IV, presents the evaluation. Finally, thediscussion and the conclusion are presented in section V.

II. PROPOSED ARCHITECTURE

The system overview is shown in Fig. 1. In our system, each smart device is connected over a Controller. Due to smart devices must be able to exchange information with multiple devices. Different smart deviceshave differentcommunication protocols include Bluetooth, Wi-Fi or ZigBee. Moreover, not all devices are not smart devices in smart environment.These devices may use Infrared or Relay to control. Thus, the Controller's hardware feature must be very powerful and include multiple communication interfaces to control any smart devices, likely a gateway. Different from architecture, the user can directly control all devices according to the signal of button or potentiometer on the controller, not use any smart phone. Therefore, we divided into two types in Fig. 2, one is the smart device that controlled byController, and the other is traditional devices that controlledby IR-emitter.

Next, we following the components and target of the system are illustrated:

1) **Sensors and Actuators**
   The system will collect data from user's any operation.Sensors report data when user do something. For instances, the DHT will be reported the current temperature when user turn on air conditioning, the ambient light sensor reported ADC when user change lamp's luminosity, etc.,in some scenarios. Monitor which can display some information, buzzer which warming user has an error and IR-emitter which control traditional device are our system's actuators.
2) **Controller**
   The system operates a Controller that handling multiple inputs as a gateway from real-time operating, stored, processed sensor data, and communicate with the IR-emitter.
3) **Data Storage**
   The data that the current state of the device and user operation has collected by Controller. The data is recorded and stored indefinitely for future analysis - machine learning.
4) **Offline Operation**
   In some scenarios, the quality of wireless is weak, this could be happened that a package loss due to weak signal like in warehouse, construction site, basement, farm, special disaster environment, etc. It could be cause failure to connect to cloud services, loss reliability. Therefore, we need a method to keep service reliability.

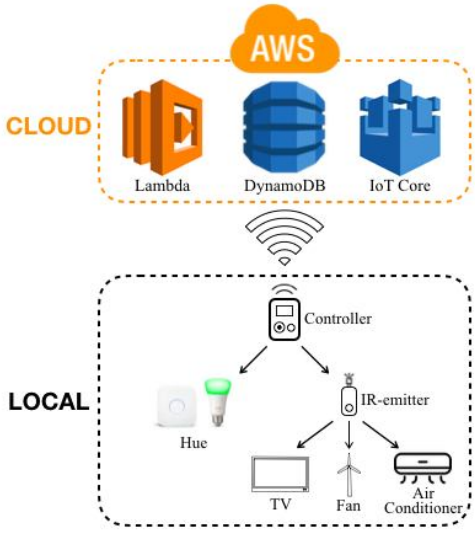


Fig. 1. System Overview

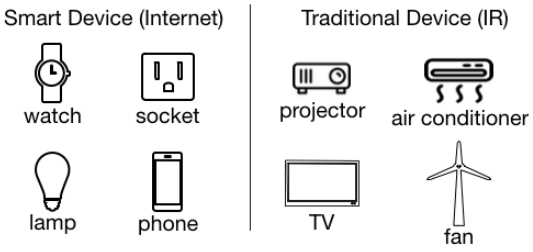


Fig.2. Type of Device

III. IMPLEMENTATION

In this section, we describe our hardware, software and cloud service that used. The cloud service that using AWS to implement the system. AWS was chosen because it is a cloud service provider, the multiple services are already integrated, including IoT Core, Lambda, DynamoDB, Polly, etc. And they provide service of consistency, reliability, flexibility, scalability, customization, security and third-party APIs.

In our architecture, need to complete target system that the offline operation, we install Greengrass[2] which is responsible for providing compute capabilities closer to the devices. Using Greengrass from AWS, the system can in real time respond to local device and keep supply service when cannot access network. The details can reference the Developer Guide[3] of AWS Greengrass.

A. Hardware

1) Raspberry PI 3

The raspberry PI 3 is a single-board just similar a small computer that comes with CPU of 64-bit quad core processor, GPU, USB ports, on board WiFi, Bluetooth and GPIO pins and can connected with external peripherals. Any of the GPIO can be designated as an I/O pin (Fig. 3) and used for a wide range of purposes. Button, buzzer and monitor are connected to PI 3. See the describe given above, the Controller has to powerful like a gateway that deal with multiple smart devices. Thus, PI 3 have all the requirement for our want and low-cost.

2) Zero W

The raspberry PI Zero W is also a single-board computer that similarPI 3. The different from PI 3, Zero W comes in an even smaller form factor. Fortunately, the features that on board WiFi GPIO are not reduce. We connect IR, button and led to GPIO and use WiFi to receive MQTT message. The hardware is enough for our system.

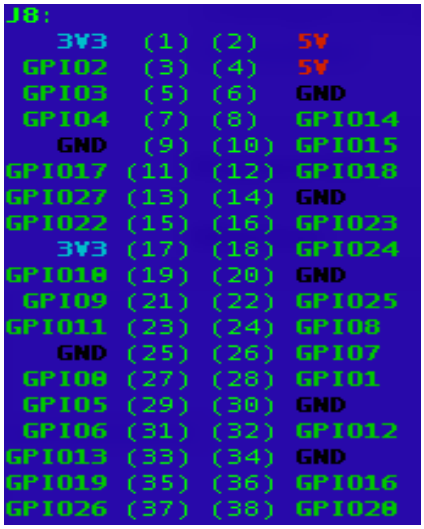


Fig. 3. Pinout of PI

B. AWS Components

1) IoT Core

IoT Core is a platform that enables connected devices to securely connect and manage multiple devices. We use AWS IoT Device SDK that enables devices to connect, authenticate, and exchange message with Message Queue Telemetry Transport (MQTT). For devices that connect using MQTT will maintain long lived, bidirectional connections, enabling these devices to send and receive messages with low latency. And with IoT Core, we create Device Shadow, of each device that includes the device's latest state so that processing logic operation. For instance, user click a button, we have to know what to control and what value to setting. Therefore, we use Device Shadow to store device latest states.

2) Greengrass

AWS Greengrass is cloud service which is responsible run local message, messaging and sync capabilities for register in Greengrass Group. Greengrass authenticates and encrypts data at all device of connection that ensure the security.

The system can in real time respond that using a managed subscription scheme through the MQTT protocol to local events if offline. And Greengrass ensures devices can respond quickly to local events, and Lambda functions can run on Greengrass to interact with local resources.

3) Lambda

AWS Lambda is a serverless compute service that runs code for virtually any type of application or backend service - all with zero administration. Just upload code to Lambda, can automatically run code in responses to multiple events, such as HTTP requests in our system. And Lambda functions can deploy on Greengrass to interact with local resources, operate with intermittent connection. Its easily deploy when update Lambda function. First, update Lambda function code and test, then, deploy on Greengrass. This is an advantage for system that need to manage multiple devices. In this work, Lambda is used for read GPIO and logic operation after read GPIO at local, then, upload user data to DynamoDB, finally, post message by MQTT.

4) DynamoDB

DynamoDB is a fully managed proprietary NoSQL database service by Amazon. It is used for storing data that user operation with time and sensor data in this work.

Next, we will describe architecture in detail.

A. Controller

See the Fig. 1, the Controller that act as a gateway is connected smart devices, user can directly control by button or switch on board. In our system, we connect Philips Hue[4] that can controlled via a website or smart phone application by HTTP method POST and GET to act as smart device. Therefore, in Lambda function, we write program that process read GPIO. Then, do the logicoperationaccording to device shadow. In Fig. 4, the Lambda function that Controller has used is shown.

For instance, when user click button that the Controller is control Hue now, the Lambda function get Controller's shadow to know what is control now after read GPIO of button in readGPIO(), then, do the logic operation to turn on if the state of Hue's shadow is off in logic_operation(), and vice versa. And the other function is uploadthe user related operation to database in upload_to_DynamoDB().

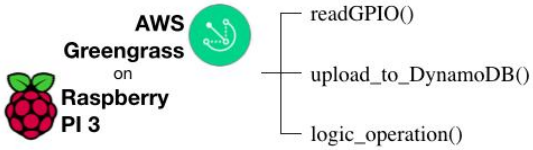


Fig. 4. Lambda Function of Greengrass on PI 3

In Fig. 5, we show the flowchart that Controller's install process. According to flowchart, success to install Greengrass on PI and ensure device get the MQTT message.

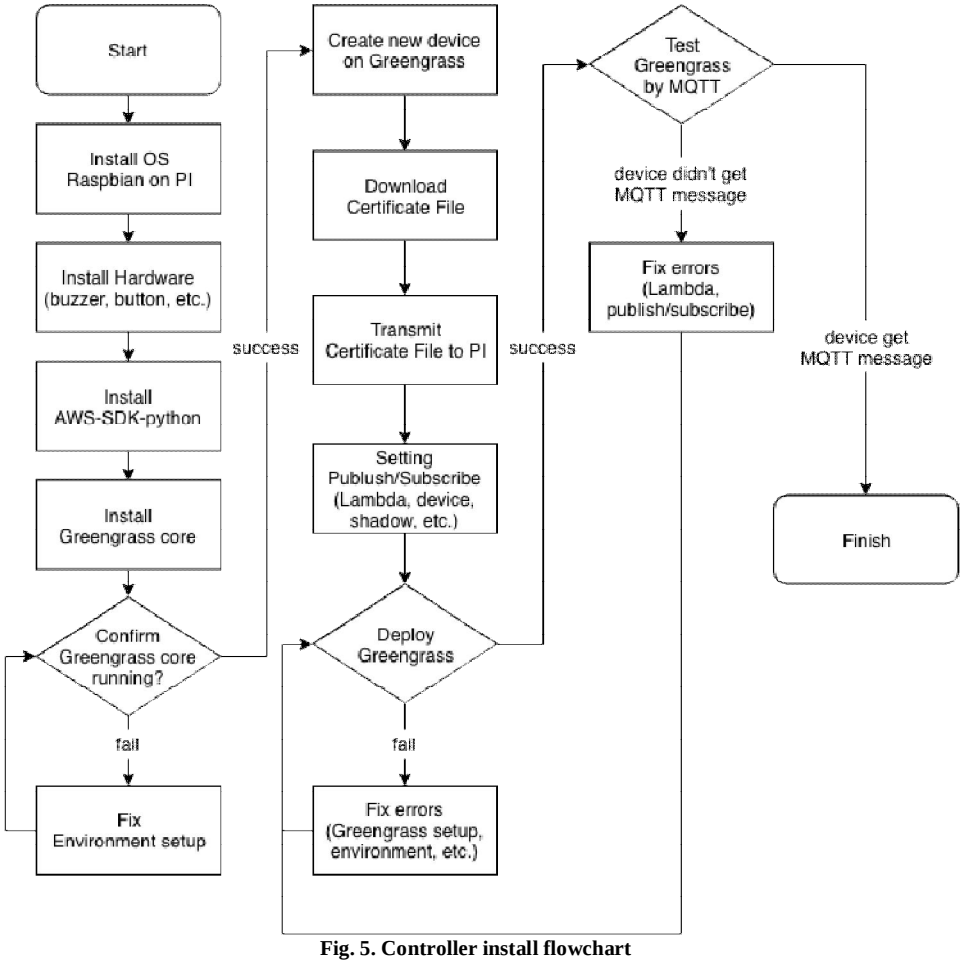


Fig. 5. Controller install flowchart

B. IR-emitter

In Fig. 6, this is a flowchart for IR-emitter that control traditional devices. In our system, we use IR LED that control IR devices such as air conditioning or TV to connect to pins of Zero W. Finishing this process, we ensure all device can get MQTT message by Greengrass.

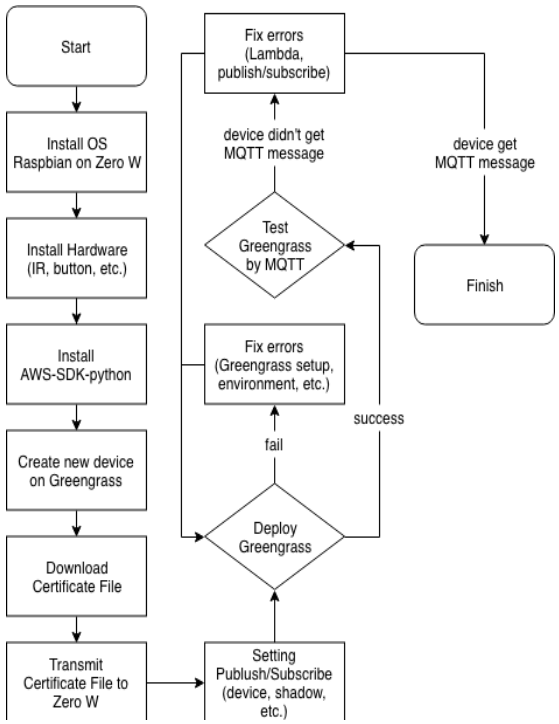


Fig. 6. IR-emitter install flowchart

C. Device shadow

As mentioned above, we describe how to install the basic-software required for Controller and IR-emitter. Next, we show how Greengrass devices can interact with AWS IoT device shadows in a Greengrass group. A shadow is a JSON document that is used to store current state information for a device. In order to sync shadow to AWS Greengrass cloud until internet is reconnect, we set to SHADOW SYNCING TO CLOUD as opposed to LOCAL SHADOW ONLY. The shadow state in the cloud be updated automatically whenever Controller sends an update. This functionality can be used to expose the state of a Greengrass device to the IoT Core cloud. In Fig. 7, we show the shadow format with Controller. It's record what machine is control now, and machine state that Controller can control machine. In Fig. 8, we show the shadow format with IR-emitter. The related information is similar to Controller.

D. Publish/Subscribe

Controller publish MQTT messages on topics. The message will be filtered through topic. On the group configuration page in AWS, choose Subscriptions, and then set up Controller to source, IR-emitter to target. The topic that the broker uses to filter message for each device is setting by yourself. The topic also need to set include shadow. The topic paths must be written exactly as shown in the Table 1.In Table 1, we can see that Lambda function is also setup in Greengrass. Because this is a method to trig a Lambda function in Greengrass.

Table 1. Publish/Subscribe

source	target	topic
Controller	Local Shadow Service	$aws/things/Controller/shadow/update
logic_operation()	Controller	Greengrass/Assistant/hue
Local Shadow Service	Controller	$aws/things/Controller/shadow/update/delta
Controller	Local Shadow Service	$aws/things/Controller/shadow/delete
Local Shadow Service	Controller	$aws/things/Controller/shadow/update/rejected
Local Shadow Service	IR-emitter	$aws/things/IR-emitter/shadow/update/accepted
readGPIO()	Controller	GreenGrass/Assistant/Receiver
readGPIO()	logic_operation()	GreenGrass/call_logic_operation
Controller	Local Shadow Service	$aws/things/Controller/shadow/get
IR-emitter	Local Shadow Service	$aws/things/IR-emitter/shadow/update
logic_operation()	IR-emitter	GreenGrass/Assistant/IR
Local Shadow Service	Controller	$aws/things/Controller/shadow/update/accepted
Local Shadow Service	IR-emitter	$aws/things/IR-emitter/shadow/delete/accepted
logic_operation()	upload_to_DynamoDB()	GeeenGrass/call_update_to_DB
Local Shadow Service	IR-emitter	$aws/things/IR-emitter/shadow/update/rejecter
IR-emitter	Local Shadow Service	$aws/things/IR-emitter/shadow/delete
Local Shadow Service	IR-emitter	$aws/things/IR-emitter/shadow/delete/rejected

IV. EVALUATION

In this section, we designed architecture is evaluated using a scenario based on smart market. Based on this set-up, the result revealed of IR's raw data and the cost and latency analysis are presented.

A. Scenario

To evaluate the designed our system architecture, a scenario was developed in the simulated environment. The scenario is based on smart market. The Inventory Management will pick in warehouse, need to turn on air conditioning and light.In order to reduce work process, using Controller to finish the work, but the wireless is instability.

B. Performance

1) Security

The system's various entities such as Controller and IR-emitter are connected to the IoT Core. The credentials for each device are stored locally on the device. It would represent that Controller and IR-emitter are registered and trusted by the system. Other device could not access IoT Core, and control smart device and traditional device. This satisfies the security and reliability requirements.

2) Scalability

IoT Core provides an interface to add and manage device. The credentials required for devices to communicate with AWS have to be stored locally in each device. In the future, if need to add a relay on a single-board for other methods of operation, the installation step is same as IR-emitter (see section III.). Thus, IoT Core can scale to support thousands of devices.

3) Cost

The cost of deployment is an essential factor in determining the effectiveness of a cloud service provider. Amazon is a company that offers some component of cloud computing - typically infrastructure as a service, software as a service and platform as a service. The large cost is absorbed by Amazon with pay for what use, we only cost local devices such as sensors, single-boards and actuators.

4) Reliability

The device can still work when disconnect. Only DynamoDB unable to store user's operation, and the others feature works normally. DynamoDB enable to restore user's operation once device reconnect, Greengrass sync the data on the device with IoT Core, providing seamless functionality regardless of connectivity.

5) Easily/Forcibly-Updating

In [5]-[8], the device did not update in a timely manner cause issues of information security. In our system, change code in Lambda function to solve security problemif occur and information security incident, then, re-deploy on Greengrass page. The new version willbe forced updating for device that have internet andGreengrass, andreduce the occurrence of the information security.

CONCLUSION

In this paper, we present the architecture that can offline and easily-management and evaluated for scientific experimentation. the hardware architecture and software implementation are discussed. The latency of the system is also presented using AWS. And the global access to AWS is possible and server maintenance difficulties can be eliminated. This system is making well suited for individual or small business IoT service scenarios, includes smart city, smart warehouse, smart traffic and smart site that network unstable. The developed system is a low cost, simple to operate. The data that device state and user operation is collected, then, use machine learning analysis to recommend setting what is users prefer in the future. At the next time, the device will be set according to the setting by machine learning of AWS or cloud service. For instance, user always turn on the air conditioning at 8am. After use machine learning, the air conditioning will automatically turn on at 8am every day. The recommender system based on machine learning may be an effective method of choice.

REFERENCES

[1] N. Eddy, "Gartner: 21 Billion IoT Devices To Invade By 2020 - InformationWeek," Nov. 2015.

[2] S. Singh, "Optimize cloud computations using edge computing," 2017 International Conference on Big Data, IoT and Data Science (BID), 2017, pp. 49-53

[3] Amazon Web Services, "AWS Greengrass Developer Guide," 2018. [Online; Accessed November 2018].

[4] B. Ur, J. Jung, and S. Schecher, "The current state of access control for smart devices in homes," Workshop on Home Usable Privacy and Security (HUPS), 2013.

[5] S. Mohurle, and M. Patil, "A brief study of Wannacry Threat: Ransomware Attack 2017," International Journal of Advanced Research in Computer Science (IJARCS), vol. 8, no. 5, pp. 1938-1940, 2017.

[6] M. Alotaibi, S. Furnell, and N. Clarke, "Information security policies: A review of challenges and influencing factors," 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST), pp. 352-358, 2016.

[7] A. I.Aysan, and S. Sen, "Do you want to install an update of this application? A rigorous analysis of updated Android applications," 2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing, pp. 181-186, 2015

[8] M. Penning, M. Hoffman, J. Nikolai, and Y. Yang, "Mobile malware security challenges and cloud-based detection," 2014 International Conference on Collaboration Technologies and Systems (CTS), pp. 181-188, 2014.

★★★