Final Project Report

Jiaming Wu, Lekai Song, Anqi Guo

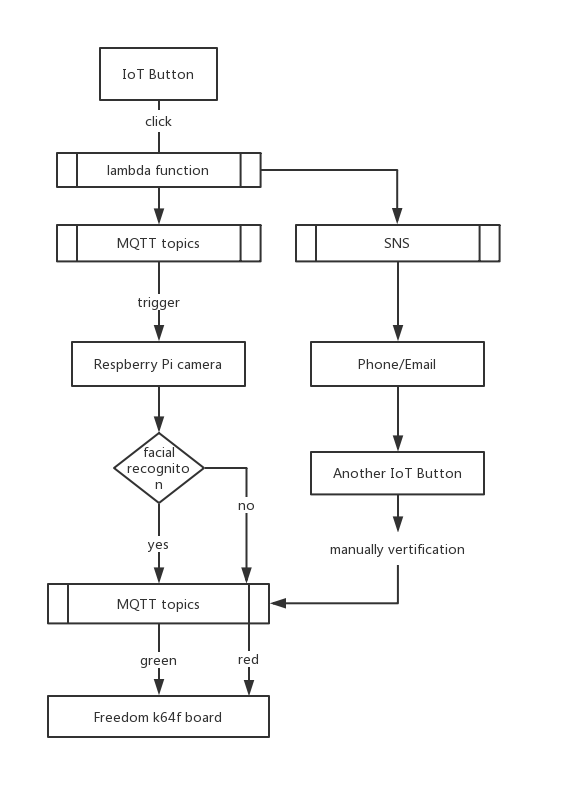
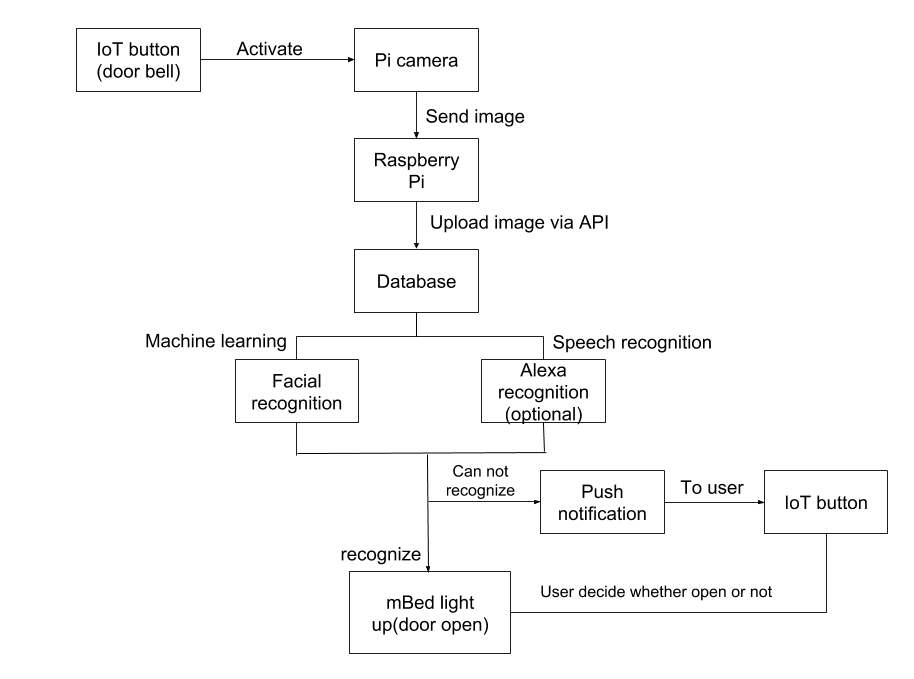
Introduction

The goal of our project is to build a secure smart doorbell. There are several similar products like Dropcam, Nest Cam Indoor which are able to help you look after your home and family even when you are not at home. However, these products are expensive and they may not have some functions that we need. What if we want a smart doorbell which can recognize the person who is pressing the doorbell, and if this person has access in our database, the door will open automatically. The user will receive a notification message and an email at the same time. We are still working on realizing the function that the Pi will send the image to the user’s phone and then user can send a signal to open the door when recognition fails.

In this report, we will introduce the flow chart, implementation, failures and successes we have met as follow.

Flow chart

User presses button (knocking at the door) -> IoT button sends signal to AWS topic -> Pi receives signal from the topic -> trigger OpenCV facial recognition to detect and check whether this person has access -> If recognition succeeds, then sends signal to AWS topic -> freedom board gets signal from the topic -> turn on green light (open the door).

****

Current flow chart (left) and original flow chart (right)

Implementation

IoT button

We are using AWS MQTT channel to intercommunicate with Pi. When the button is pressed, it will send string “RUN” to our topic (freertos/demos/echo) since we set up a lambda function to publish a message to the topic. For using IoT button, you need to download keys and certificate from AWS and configure them into your button. And In lambda function, you need to define your endpoint, the topic and the payload(message) you want to publish. Once you set up things above, you are able to send a message to the AWS topic.

Raspberry Pi

Pi is expected to receive a message from the topic, thus you need to add new thing Pi in your AWS console. We are using AWSIoTPythonSDK to realize the function of receiving a signal and sending a signal from/to the topic that you subscribe and a Pi camera to capture the images. Also, download the key and certificates, then set up your endpoint, certificate path, key paths, and the topic. The customCallback() function is to receive the message from the MQTT topic. If the message we get from the topic is “RUN” sent by the button, it will trigger OpenCV and recognize the person by the model we trained. If the visitor matches one of the people in our database, it will send a signal to myAWSIoTMQTTClient.publish() which will publish a message to MQTT topic. “SINGLE” is the keyword in our demo.

Freedom K64F board

For freedom board, we are using C SDK to receive a message from MQTT topic, which is the same topic channel as before(freertos/demos/echo). If the message freedom board gets equals to “SINGLE” same as Pi published, it will turn green light on, which means the door opens.

Discussion of failures:

The Alexa speech recognition module is not implemented since time is limited and speech recognition could be considered as a secondary recognition mechanism without which the system is still fully functional.

For the aspects implemented but not in the original proposal, we tried the AWS C SDK to subscribe and publish messages to a topic with Raspberry Pi. The problem is the subscribe function cannot be implemented since we did not find something like a callback function. Though messages can be received and output when we make all files, we cannot find the exact variable to set up conditional expressions, thus cannot trigger the facial recognition module. Having struggled for some time, we refer to the aws-iot-device-sdk-python. Python SDK makes the whole system more connective since the facial recognition module is also implemented with OpenCV-python and it is more consistently to import more modules together. Unlike C SDK which has several files and has to make all files together every time compiling them, python SDK can be compiled faster and all functions are implemented when we install the library, which saved us a lot of time and effort.

Another aspect that doesn’t run as we expected is the facial recognition module run on Raspberry Pi. Due to the computation ability of Raspberry Pi, the facial recognition dataset preparation takes a long time when capturing photos for each person with Pi camera. The possible solution is to try importing the frequent visitors’ photos from their social media instead, which may affect the accuracy of confidence since the images come from different sources, there exists a contrast of image resolution among Pi camera and other devices. Moreover, the facial recognition process takes a long time to get a result of a visitor’s confidence. To make the whole system working in fluent progress, we make a compromise to simplify the structure of the training module and thus the prediction is more smooth now; the drawback is the accuracy falls down.

Detailed stacks:

When we deliver messages through AWS MQTT service, several stacks include MQTT, crypto, authentication, and serverless computing. When connecting the IoT Button and Raspberry Pi, a lambda function between them is required, where serverless computing allows us to build and run our functions without thinking about the server. This part is designed based on Lab1 where we use a lambda function to send an email when the IoT button is clicked.

Another important stack is the AWS MQTT service. The whole system is connected through MQTT service. When the button is clicked, a specific message is published to MQTT channel which is already subscribed by the Raspberry Pi. After the Raspberry Pi receives the message, it runs the facial recognition module and if succeeds, publishes the result to the MQTT channel which Freedom k64f board subscribed. Subsequently, the board receives a signal to turn on the green light, which means the door opens. This part is related to Lab4 where we use MQTT to get Raspberry Pi connected.

Technical report of success

Following the proposal, we successfully developed a system with 4 parts as follow and connected them together.

1. IoT Button controlling module:

When the button is clicked, a message would be posted to an AWS topic to trigger the Raspberry Pi. The IoT button module is implemented to realize the function that when a strange visitor has accessed the doorbell and pressed the button, a notification will be sent to the user who can decide whether to activate the door by clicking another IoT button. To implementing this approach, the Raspberry Pi and IoT button must be connected to Amazon AWS IoT core service. On the AWS IoT Core, we can set up our own Lambda function to generate a certain signal when the IoT Button is clicked. This signal will be sent to Raspberry Pi via MQTT, then further analyzed by facial recognition and then sent to the Mbed board to simulate the door’s open by lighting up.

2. Facial recognition module:

The most significant section is the facial recognition module. To realize facial recognition functions, OpenCV libraries have strong computing abilities for picture recognition and is very suitable for human facial classification. To build up a complete facial recognition module, three steps are fundamental: Human faces detection and data gathering; training the recognizer; facial recognition.

The primitive task for facial recognition is facial detection. We need to first detect faces before comparing and recognizing them. The easiest approach to detect faces is using pre-trained OpenCV classifier to detect human eyes, smiles, and other features. We can build a human face detector with these pre-trained modules and functions. The detector we applied is based on Cascade classifier.

The next step is to train the recognizer. Before that, we need to collect enough pictures and create a dataset to store them. Each person is given a typical name ID. This process is called data gathering. Next, we need to extract pictures from the dataset and feed the face recognizer with pictures and ids of each face so that the recognizer could start to learn. This process could be executed by OpenCV functions and a model(.yml) will be generated after this process. In the last phase, we will capture a new face through a Pi camera. If this face is similar to those trained before, the recognizer will return a name ID and confidence of this result. When the confidence is higher than an exact number(like 60%) we set, that means this person pass the recognition, and thus a typical signal will be published to the topic. Otherwise, there is no way to open the door unless the user gets the notification and approves the access by clicking another button. The algorithms we use is LBPH(local binary patterns histogram) and all script are run on Raspberry Pi.

3. Freedom k64f board:

The Mbed board is to simulate a real doorbell system. There will be a light indicating whether a door is open or not. The Mbed should receive signals from Raspberry Pi and make a certain response with our programming. We install freertos operating system in the freedom board and successfully program the board to subscribe to the messages from AWS topics. When a recognized signal is received, the green light will be on indicating the door is open, otherwise, the red light will remain on indicating that the door keeps closed.

4. Notification module:

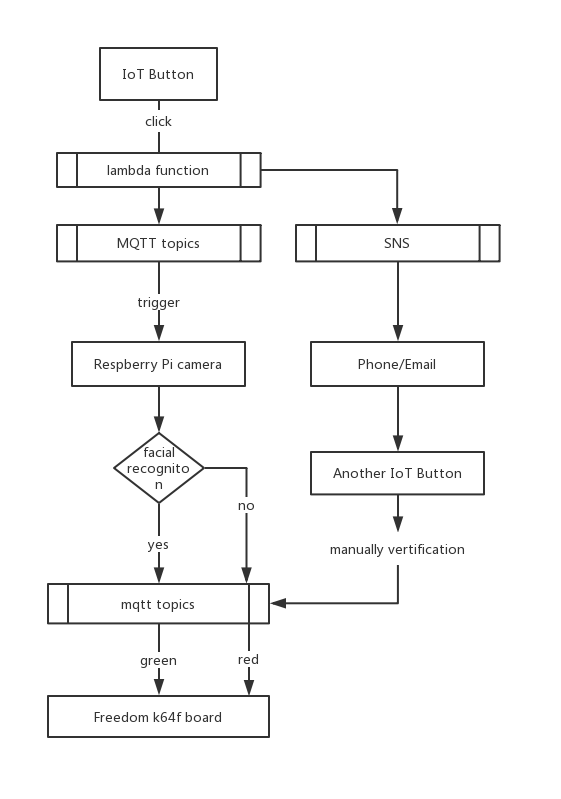
We successfully achieved the function that when a person is accessing the smart doorbell system, both the owner’s phone and email will receive a message. We implemented these based on lambda functions.

Open resources

OpenCV, AWS IoT Device Python SDK, AWS Lambda function, Google email API documentation.

Software architecture:

The system contains three modules: IoT Button, Raspberry Pi, and Freedom k64f board. Each module connects with others through AWS MQTT service. In the end, the freedom board will give out a signal of light, where green light refers to open door, while red light refers to keep the door closed.



Things learned from the project

From this project, we learned how to connect things through AWS MQTT service; How to set up lambda functions to make IoT button functioning; how to use Raspberry Pi as a microcomputer in an embedded system. Although there are some functions we cannot implement at all currently, thus sometimes it looks like a simple project, we come to understand the core of ‘internet of things’, putting what we’ve acquired into practice.