EC 544 Project

Smart Doorbell

Github: https://github.com/ziyu39076/Smart_DoorBell

Web app: http://ec2-3-86-230-21.compute-1.amazonaws.com

Team Member	Email Address
Qianhao Liulin	linqianh@bu.edu
Ziyu Zhao	zepher@bu.edu

1. Introduction

In the context of the Internet of everything, the conception “smart home” has penetrated into all aspects of people's life. For example, in the field of Intelligent Security closely related to consumers, intelligent security devices represented by smart cameras have become an important support in family security activities. As a very important part of family security, intelligent doorbell undoubtedly plays the role of "frontier sentry", adding a security guarantee to the family door.

We want to build a smart doorbell for modern family which is capable of automatically open the door for friends and family members and send alerts to the user if their home is visited by strangers. Our idea is that use an embedded system to capture and preprocess the photo of visitor, then send the photo to a web server which will classify whether this visitor is permitted to enter, if so, activate the door, if not, record this visit and send alert to the user, and the user could proceed with further operations such as call the police or simply keep the door closed.

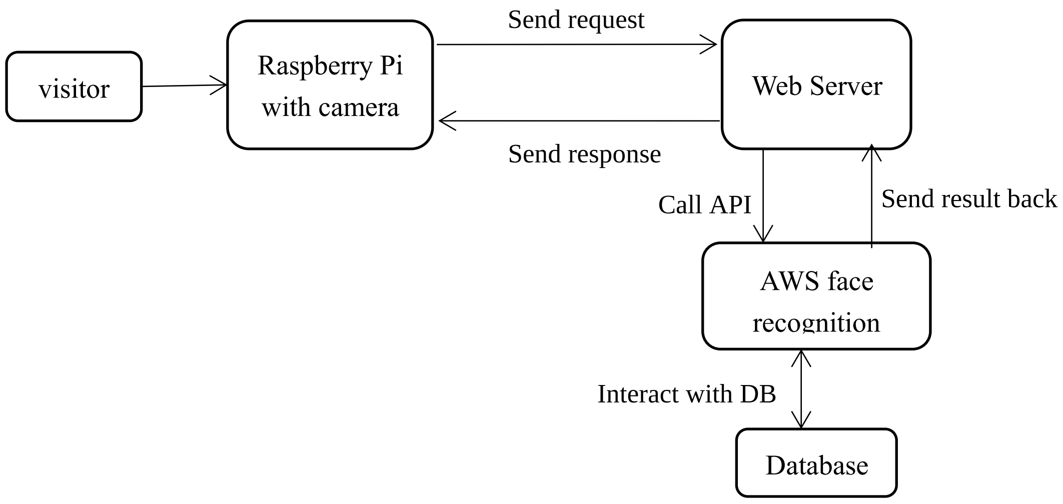

Figure 1: project flowchart

2. Resources

1) Hardware
    a) Raspberry Pi 2 Model B
    b) Raspberry Pi Camera Video Module
2) Software
    a) Linux Ubuntu LTS 18.04

b) Python 3.6.9
c) Nginx web server
d) Gunicorn
e) Supervisor (a software in Linux enviornment)
f) Git
g) Flask Framework
h) Bootstrap
i) Javascript
j) CSS
k) HTML
l) Database (SQLite)
m) VSCode
n) PyCharm

3) Web Server
a) AWS

3. Organization and Structure

1) Hardware part

a) Discussion of failure

i. M2M between Piboard And Webserver

Although the Piboard can be connected to the server, M2M is not a simple data transmission between machines, more importantly, it is an intelligent and interactive communication between machines. It can't automatically upload the captured photos to the server, and can't automatically receive any return from the server. The reason why I didn't finish this is largely due to limited time. I don't have enough time to learn enough programming skills to deal with this problem. My teammates and I have been struggling with no suitable environment to jointly develop such a connection so that when people don't send real time signals, the machine will also actively communicate with each other according to the established program, and intelligently make choices according to the data obtained. The failure of this task also means that we cannot optimize on this, whether using hash function to verify the integrity of the source, or using encryption technology to prevent the third party from intervening.

ii. From Web Server Sending Orders to the Door

In addition to the above problem, it is also important that we do not have any connection between the FRDM board which can replace the door function and the server. As a result, FRDM board cannot accept any instructions

b) Report of Success

i. Successfully assembled the Piboard and Picamera, downloaded and burned the respbian system, configured the SSH and VNC environments, redistributed the video memory, and installed all necessary environments including python2 and python3. In terms of power supply, any USB device charger can be competent. Because raspberry PI 2 model B does not have its own Wi Fi adapter, I have to establish network sharing so that the PC network can be shared to Piboard through the Ethernet cable. After testing, the hardware part including camera has no problems at all. After adapting the Piboard with VCNviewer, you can control it directly and remotely without the monitor

ii. Picture Capturing program

Write and configure a picture capture Python program that can automatically turn on or off through pixel changes. That means if pixels change are detected, it will start capturing the picture itself. The purpose is to start the program to generate pictures only when someone passes by. And you can write python pyfile_Path in /etc/ rc.local to set this program automatically run when you turn on the Piboard system. The program can easily adjust the accuracy, path of store and other parameters. The only regret is that it can not automatically interact with the server.

iii.

2) Software part (Flask web app)

a) Discussion of failure

i. Fail to add reset password link for user in case a user forgets his/her password

ii. Fail to add delete permitted visitor functionality

iii. Fail to send alert email to user if a stranger visited the user's home

(these unfinished functionalities could be finished with more time and efforts)

b) Technical report of success

i. Developed a fully functional web app using Python with Flask framework for user to register, login, add permitted visitor and review visit records

ii. Successfully achieved interaction between web app and database using Flask-SQLAlchemy

iii. Implemented face recognition using AWS face-recognition API

iv. Managed project version and updated project iteratively with Git version control system

v. Configured Linux server for hosting web app in AWS EC2 instance

vi. Deployed flask app on AWS EC2 instance with Nginx and Gunicorn

vii. Achieved web app auto-start and auto-restart with Supervisor

3) Web app details

a) Web app development

i. In Flask app, each page is navigated by function decorated with routes, and each page is a rendered template with style defined in Bootstrap and local CSS file

ii. Date submission is handles with build-in Class called WTForm in Flask-WTF, forms could be passed into flask templates to be displayed in pages

iii. Date storage and manipulation is handled with built-in class called SQLAlchemy, each table is represented by a defined class and we do not need to write SQL commands when developing web app

iv. User session management is achieved with LoginManager imported from flask_login

v. In client side, the raspberry pi could simulate as a user with request lib in python then login and post images for web app to identify

b) Server configuration

        i. Nginx web server is deployed to handle incoming requests
        ii. Gunicorn is used to call python scripts to run our application
        iii. Supervisor is utilized to achieving running our app in the background with auto-start and auto-restart feature

4) Learned skills
    a) Web app development with Python and Flask framework
    b) Database interaction and basic SQL commands
    c) Client server communication and basic URL and HTML parsing
    d) Basic HTML and CSS knowledge
    e) Calling external APIs and setting up credentials
    f) Version control with Git
    g) Linux web server configuration

5) References
    a) Flask tutorial videos: https://www.youtube.com/playlist?list=PLosiE80TeTs4UjLw5MM6OjgkjFeUxCYH
    b) Git official doc: https://git-scm.com/book/en/v2
    c) https://projects.raspberrypi.org/en/projects/raspberry-pi-getting-started
    d) https://github.com/kevinam99/capturing-images-from-webcam-using-opencv-python
    e) https://help.realvnc.com/hc/en-us/articles/360002253198-Installing-and-Removing-VNC-Connect#removing-vnc-connect-0-17