

Demo #1 Documentation: Home Security Automation

Group 2

14:332:452:01:00614 Software Engineering

Harmit Badyal

Nikunj Jhaveri

Abhishek Kondila

Kaavya Krishna-Kumar

Kaushal Parikh

Miraj Patel

Nirav Patel

Andrew Russomagno

Sagar Shah

Ashwin Suresh

March 27, 2019

Website: <https://sites.google.com/view/ruhome2019/home>

GitHub: <https://github.com/nikunjghaveri123/HomeSecurityAutomationApp>

All the code for the project in its current state can be broken down into three divisions:

Mobile Application

Arduino

Facial Recognition

This Technical Documentation will be a simplified overview of the software used in each division

Mobile Application

1. AccessCamera
 - a. Allows user to access the Arducam connected wirelessly
2. AdminSettings
 - a. Allows the user to change certain settings within the application when logged in as admin
3. CustomAdapter
 - a. This allows the photos to be viewable on the application
4. DeleteUser
 - a. This allows an admin to delete a user from the application
5. FingerprintHandler
 - a. This manages requests for fingerprint sign-in enabled devices
6. FirstTimeUserSetup
 - a. This registers a user to firebase and sets up an email and password for their account
7. LoginActivity
 - a. This is the page users will interact with when logging into the application
8. MainControlActivity
 - a. This is where the user will be able to interact with any of the devices connected to their system, like lights and alarms
9. NewUserSetup
 - a. This handles requests for new users to the application
10. Photo
 - a. This class lets photos be viewed on the application
11. Pictures
 - a. This is the interface users will use to view the photos
12. RegisterNewFace
 - a. This is the process new users will undergo to add photos of their face to the firebase database
13. User
 - a. This is the class that defines certain characteristics of the current user of the application, such as their email or admin status
14. UserSettings
 - a. This page defines the different settings that a user can change within the application

Arduino

1. AccelerometerAlarm

- a. The functionality of the accelerometer and alarm code is to test these two devices working and how they interact with one another. The first device is the accelerometer. This is the MPU-9250. This requires only 4 port connections - A4 to SCL, A5 to SNL, GND to Ground, and VCC to 5V. The second device is the alarm which can be any standard 2 port alarm or buzzer. The longer pin side of the alarm is connected to pin 8, and the other end is connected to ground. Once hardware is hooked up and code is uploaded, the alarm should start buzzing whenever the accelerometer is moved.

2. Arducam

- a. The functionality of the Arducam is to take a picture when the system detects motion which will then be sent for facial recognition to run on it. The device we use is the Arducam_mini_2mp. There are 7 port connections - SCL to 17, SDA to 16, VCC to 5V, GND to GND, SCK to 13, MISO to 12, MOSI to 11, and CS to 7. In order to run this program, the processor IDE has to be installed. First, the Capture code has to be uploaded into the Arduino. Once that is done, the processor code can be run, and pictures can be taken by command and saved in the same directory. In line 34, if the arduino is on a COM that isnt the default one, then the value '0' has to be updated accordingly. For example, if the output shows COM5 COM6, and the board you are uploading the code to is COM6, then you will have to update the array value to 1.

3. LightOnOFF

- a. The functionality of this code is to show that we are able to manipulate the lights inside the house with signals being sent to it. This code just requires any LED connected to a pin 9, through a resistor, and to ground. Once the code is uploaded to the Arduino, you can input 2 values into the serial monitor but 9600. If you input a zero, the LED will turn on. If you input a 1, the LED will turn off. Any other value will do nothing to the LED.

Facial Recognition

There are four directories in this folder:

1. dataset/ : Contains face images organized into subfolders by name.
2. images/ : Contains test images that will be used to verify the operation of the model.
3. face_detection_model/ : Contains a pre-trained Caffe deep learning model provided by OpenCV to detect faces. This model detects and localizes faces in an image.

4. output/ : Contains output pickle files, a specific type of byte data that is used by the model. The output files include:
 - a. output/embeddings.pickle : A serialized facial embeddings file. Embeddings have been computed for every face in the dataset and are stored in this file.
 - b. output/le.pickle : The label encoder. Contains the name labels for the people that our model can recognize.
 - c. output/recognizer.pickle : The Linear Support Vector Machine (SVM) model. This is a machine learning model rather than a deep learning model and it is responsible for actually recognizing faces.

There are also five other files in the folder:

1. extract_embeddings.py : This file is responsible for using a deep learning feature extractor to generate a 128-D vector describing a face. All faces in our dataset will be passed through the neural network to generate embeddings.
2. openface_nn4.small2.v1.t7 : A Torch deep learning model which produces the 128-D facial embeddings.
3. train_model.py : The Linear SVM model (in recognizer.pickle) will be trained by this script. It'll detect faces, extract embeddings, and fit our SVM model to the embeddings data.
4. recognize.py : This file detects faces, extracts embeddings, and queries the SVM model to determine who is in an image. It'll also draw boxes around faces and annotate each box with a name.
5. recognize_video.py : This file does the same as recognize.py, but instead it is run on frames of a video stream.