IoT Project: A Physical Authentication System For Your Home

We will build a physical authentication system in this project. We use
a Raspberry Pi and a few sensors to authenticate a person and allow an
admin user to view the person being authenticated and supervise the
authentication process.

We will provide each of you with a Raspberry Pi, a proximity sensor,
and a webcam. We expect you to do this project in Python.

Milestone 1 due: 10/31/2017
Milestone 2 due: 11/23/2017

Milestone 1:

1. Connect a distance sensor to Raspberry Pi. Learn how to read data
   from the sensor. You can read more about the sensor here:
   https://www.adafruit.com/product/3317 (local copy: here). You can
   use the Python library available at
   https://github.com/johnbryanmoore/VL53L0X_rasp_python (local copy:
   here) to use the sensor.

2. Connect a webcam to Raspberry Pi. Learn how to collect image frames
   or videos from the webcam. You can use the Python library available
   at https://github.com/gebart/python-v4l2capture (local copy: here)
   to use the webcam.

3. Learn how to use face recognition API and run it on the image
   collected by the webcam.

4. Learn how to upload the images to a server in the cloud using your
   Python program and run that program on Raspberry Pi.

5. Integrate these functionalities into a single or a set of Python
   programs to collect image, some summary of distance

measurement,
face identity and upload to the server.

6. Create an Android app that downloads the image from the cloud and
displays the image to the user on the app. The user specifies the
rate at which the image should be refreshed. Run this Android app
on the Android Emulator.

7. Now download other information (face identity, distance) along with
the image from the server. When the image is displayed, display
information about the user by merging content from a simple
database and face recognition API.

8. The Android app should ask the user to approve authentication using
a simple GUI control. Authentication should require the correct face
and corresponding unique distance for that person.

9. The authentication status is posted to a file which is periodically
and frequently checked by Raspberry Pi to know the authentication
status or the user. This is called "polling" approach.

10.Use the speaker connected to the audio port of Raspberry Pi to give
result of authentication to the user. For example: "Welcome home,
Bob" where the name corresponds to the person the system
recognized. You can use synthetic voice API for this part. If
authentication fails, the system could sound: "System does not
recognize you, try again". You can use Amazon or Watson TTS service
for this part.

Milestone 2:

1. Measure the number of bytes that are sent from Raspberry Pi to the
cloud; from cloud to the Android app and in the reverse

direction
   corresponding to one image frame or one authorization command. What
   factors determine the network overhead? Design experiments to
   understand the network overhead and run those experiments. Your
   experiments for this and subsequent questions may need to support
   non-interactive, i.e., scripted use of your system.

2. Study the processing and network tradeoff when you run face
   recognition software on Raspberry Pi vs using an API over the
   Internet.

3. What is the fastest image frame rate you were able to support? How
   did you achieve that? What are the bottlenecks?

4. Implement non-polling authentication status push from the server to
   Raspberry Pi. This is non-trivial because Raspberry Pi has private
   IP address behind a NAT. So you will need to use an appropriate and
   easiest technique to make this possible.

5. Discuss the positive and negative aspects of polling vs non-polling
   approach with data from actual measurements.

6. Now connect the Raspberry Pi to a WiFi AP that has dynamic data
   rates. Your system should adjust the image quality depending on the
   data rate available so it can continue to meet the frame rate
   specified by the user. This also means you need to have a data rate
   measurement system in place so that the application can make
   decisions on the image resolution and/or format to use.

7. Design the simplest possible way to "program" a new user into the
   system. A user has a unique face and sensor proximity distance.

8. Design and implement at least one new feature on your own.

9. Write a detailed project report in a DIY tutorial format on how
   others can build such a system. This report should also have a
   section called "System Performance" where you put all the
   performance related issues and insights and a section called "What
   you will learn" section where you describe what the person reading
   your tutorial and following through will learn. Your code should be
   in appropriate folders in your git repository. There should be a
   README that has a short version of the tutorial and a reference to
   the file with the full report. Thus the repository should look
   exactly like how other open source repository look: documentataion
   and code.