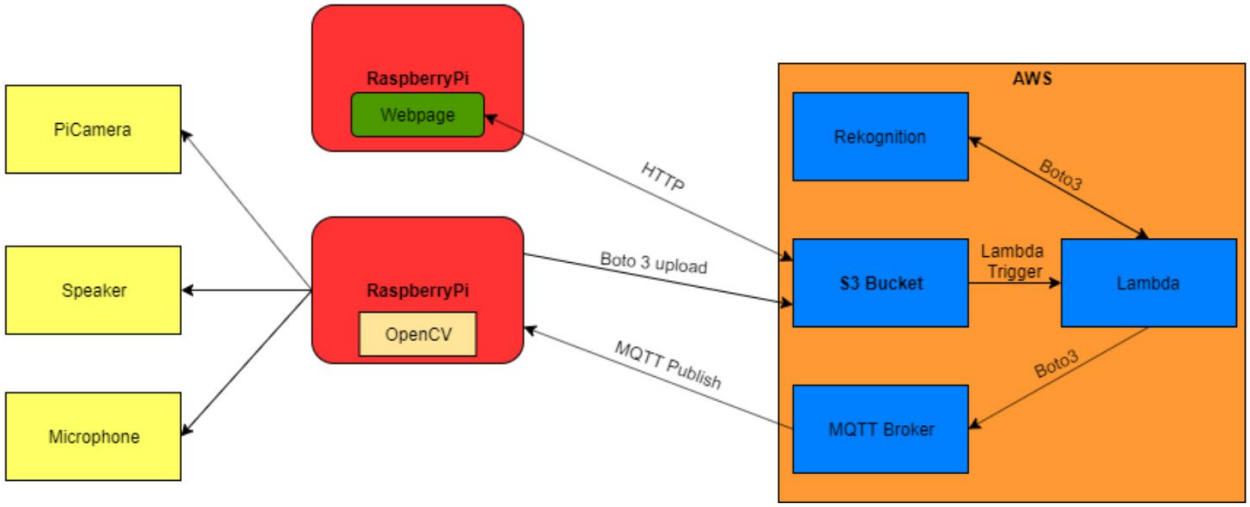
EID Project 6

Facial Recognition Security System

By Ayden Blotnick & Ajitesh Batra

Final System/Architecture Diagram and Statement



Once the application is started, a videostream will be created and continuously sent to OpenCV. Once OpenCV detects a face within the videostream, it uploads the image taken to an S3 bucket. A lambda trigger was created whenever an upload to the S3 bucket occured. The Lambda function uses AWS rekognition in order to determine if the newly uploaded image matches the approved images in another S3 bucket. The images of the newly recognized faces and ones already approved are displayed on a web page hosted on another raspberry pi. The speaker and microphone interactions are handled by espeak and SpeechRecognition python libraries respectively.

Project Deviation Statement

The project deviated in the tools/libraries we decided to utilize. Instead of using Polly for text to speech, we decided to use a less cumbersome library provided by python called *espeak*. Additionally, we diverted away from doing speech to text using AWS transcribe, and instead went with a python library known as *SpeechRecognition*.

Third-Party Code Usage Statement

- Used this tutorial in order to download OpenCV onto the raspberryPi: https://www.pyimagesearch.com/2018/09/26/install-opencv-4-on-your-raspberry-pi/
- This github repo was used as a basis to create a videostream with a pi camera and use openCV to detect images: https://github.com/just4give/home-surveillance
- This tutorial provided by AWS was used to pull images from an s3 bucket and display them on a webpage: https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/s3-example-photos-view.html
- Used this tutorial for setting up SpeechRecognition python library: https://pythonprogramminglanguage.com/speech-recognition/
- Various tutorials/posts from stackoverflow.com and w3schools.com were also used for minor issues found along the way.

Project Observations Statement

- Initially, I assumed that OpenCV would be a simple pip3/apt-get install from the command line and in a few lines I would be on my way to spy level facial recognition ...oh how wrong I was. Due to the native ARM architecture of a raspberryPi, OpenCV needed to be downloaded and built from the source code. This provides the benefit of being able to install the “latest and greatest” version of the code, but the build is quite heavy and arduous for the Pi to perform and took quite a lot of time to complete.
- The more I used AWS, the more I was able to perform quick and useful connections between the tools. Originally, I thought I would have to use a series of MQTT messages to trigger Lambda functions and upload to s3. I found that Lambda functions provide triggers that allow you to execute a Lambda on various actions. The one I chose was an S3 Upload. Whenever a new image was added to the S3 bucket, a Lambda function is triggered and utilizes AWS rekognition.
- As we learned that AWS transcribe takes a while for the Speech Recognition process, we looked for other options. We tried installing Amazon Alexa Voice Services on the Pi, but it always got stuck during installation at around 60%. In the end, we decided to use the SpeechRecognition python library which was very easy to work with. It has its issues but works pretty good for the purpose of the project.