CS Capstone Progress Report

February 16, 2018

Smart Home Intercom System

Prepared for

Oregon State EECS

D. Kevin McGrath

Prepared by

Group 25

Team 25

Lazar Sharipoff
Jordan Davis
Glen Anderson

Abstract

This document explains the progress that has been made for the Smart Home Intercom Project as of the midway point of winter term 2018. Specifically, it provides a detailed review of each week including the work that was done as well as a summary of the progress that has been made.

CONTENTS

1 Introduction

The goal of our project is to design a functional home intercom system with additional features such as video streaming, the ability to call a single node, and person detection. This system will use a mesh network to stream audio and video securely between nodes, and log which rooms have people in them so that the appropriate room can be paged.

2 Lazar Sharipoff

2.1 Current Status

We have alpha level functionality for individual portions of the project but still need to integrate them. Currently, the touch screen and IR camera is installed on the Rpi and working correctly which allows a video stream to be transmitted and viewed. Video streaming is working as a proof of concept over UDP streaming with Gstreamer, but it is currently too latent to be considered finished. Audio streaming development has not yet been started, because required hardware has not yet been delivered. Qt 4.8 has been installed on each pi and is ready for GUI development, with a basic model of the home screen having already been developed. Facial recognition for person detection is also working using an installed OpenCV library. In order to get all of this working on the same two systems, we will have to set up a mesh network and create copies of a complete version of the Pi software so that it can be easily copied to other devices.

2.2 What There is Left to Do

2.2.1 Video Streaming

Video streaming is currently in alpha with a proof of concept. A streaming connection is able to be established between 2 Rpi nodes with one node transmitting a video stream to a receiving node. In the beginning stages of development TCP was the protocol being used to stream an H264 encoded video with Gstreamer but proved to be too much for the receiving pi to handle which caused latency and thermal issues. To remedy the thermal issue and hoping would also fix the latency issue we switched to a UDP protocol. The change in protocols reduced the thermal output of the receiving pi and reduced the latency of the stream compared to the TCP streaming. Unfortunately, the latency was not reduced enough to consider video streaming to be finished and will need to be further developed and polished to have low-latency streams.

2.2.2 Audio Streaming

Audio Streaming still needs to be developed but can not be started until the necessary hardware is delivered. With audio streaming working we would be able to stream live audio between two nodes by itself and would also be able to have an audio stream open with a video stream to achieve a live video call between two nodes.

2.2.3 Hardware and Software Compatibility

The audio hats for audio output and mics for audio input will be installed and tested once they are delivered. This will allow audio streaming to begin development and progress the project.

2.3 Problems Impending Progress/Solutions

Currently the video streaming is too latent to be implemented into any further updates of the project. The reason for this is because the Rpi can natively decode H264 video encoding, but not natively encode videos with H264 which can be slowing the transmission from the transmitting node causing thermal issues and latency. A possible solution is to stream raw video from the transmitting node instead of an H264 encoded video, so the transmitting pi can stream the video directly, and the receiving pi will not have to decode anything before the stream is able to be viewed. Another possible solution is to upgrade the transmitting node for about two hundred dollars from an Rpi to an UP board, which is more powerful and would be able to encode and transmit the video stream much quicker.

Thermal issues, as seen with the thermal throttling on the receiving pi, are a cause for concern as the project progresses, because as more processes are run on each node the more heat it produces. This can be resolved installing a heatsink to the Rpi and additionally installing a PWM fan to increase cooling flow.

Audio streaming development is not able to be started yet as the necessary hardware has not been delivered. The needed hardware is a microphone so that audio can be input and transmitted to the receiving node, and an audio hat so that the receiving node can output the audio received from the stream.

2.4 Other Relevant Information

2.4.1 Code Snippets

Install Gstreamer:

```
sudo apt-get install gstreamer1.0-tools
```

UDP Video Streaming Code

```
Transmit: raspivid -vf -t 0 -h 640 -w 480 -fps 25 -hf -b 100000 -o |
        gst-launch-1.0 -v fdsrc ! h264parse !  rtph264pay
        config-interval=1 pt=96 ! gdppay ! udpsink
        host=RECEIVING-PI-IP-ADDRESS port=RECEIVING-PI-PORT
Receive: gst-launch-1.0 -v udpsrc port= RECEIVING-PI-PORT ! gdpdepay !
        rtph264depay ! avdec\_h264 ! videoconvert
        ! autovideosink sync=false
-vf changes whether the video is vertically flipped
-h value changes the video output height
-w value changes the video output width
-fps value changes the framerate of the video
-hf changes whether the video is horizontally flipped
-b value changes the bitrate of the video
```

2.4.2 Why We Are Not Using VLC

There are a couple of reasons why VLC is not being used to stream the audio and video anymore. The first reason is because it is too slow to use as a video streaming tool for our purposes. VLC is also not able to be installed and ran in embedded mode on the raspberry pi, because we are using Qt 4.8 in embedded mode for the GUI it is required that the tool used for streaming is able to run in this mode. We changed to Gstreamer to stream the video and audio

feeds because it is able to transmit and receive feeds much more efficiently, and it is also able to be installed and run in embedded mode.

2.4.3 Why We Are Not Using Salsa20

The reason why we are not using Salsa20 to encrypt our network is because batman-adv, which is the mesh network that we are implementing in this project, can encrypt its own network. Since the video stream is latent with its own encoding, and batman-adv has its own encryption we dont want to risk further reducing the streaming quality and speeds by adding another layer of encryption with Salsa20.

3 JORDAN DAVIS

3.1 Current Status

Due to the breath of our project and the individual components that require work, some portions are in post-alpha and some are in pre-alpha. The five main components of our project are video streaming, mesh networking, audio streaming, graphical user interface and person detection. Video streaming has some issues, but the command for it to operate is down and we will be trying to make sure to have the problems sorted out by the beta phase. It is certainly the selling point of the status of our project. OpenCV has been tested on a Windows machine and has been downloaded to the Raspberry Pi but has not been properly tested with it. The graphical user interface will be drawn up with Qt 4.8. It has a sketch available in Qt Designer but has yet to be ported to Raspbian. The mesh network program, batman-adv has been installed but not tested yet. However, research has been done towards its implementation and it should be a relatively simple portion of the project. The audio streaming is in its infancy. The GPIO microphones our client supplied us with have tutorials online for getting them to work. The streaming will be handled by gStreamer, the same program used to stream video. Ultimately our project is moving along at a comfortable pace albeit behind in some aspects, but the amount of time taken to integrate everything might be being underestimated. Because of this our group meets on Saturdays to verify we are all on the same page and considerate of each others own timeline as well. As a whole, all our hardware that will be required for implementation has been gathered and client verification through to functionality of an outdoor intercom has been covered.

3.2 What is Left

For breaking down what is left, there will be two parts. First is what portions of our project are untested and unimplemented. Both batman-adv and audio streaming are untested and unimplemented. The hardware needs to be added onto the raspberry pi and there are potential case modifications that will be required before we can use the GPIO microphone. Getting the audio should be a relatively simple and assuming gStreamer is friendly to the audio with the video. Batman-adv has been researched but simply remains untested with the devices. The main worrying factor with it will simply be distance and how busy the network is to allow for streaming. It is an important piece of our project and a requirement, but for now we have been testing with a LAN cable between the raspberry pis or on a familiar wireless network. The second and final part is what has been implemented but untested as a whole. The graphical user interface is probably the most behind in this regard. The home screen has a layout and will compile but without knowing what libraries that will be used in full is difficult to implement to standard. The other screens have a skeleton drawn out but have yet to be compiled or drawn in designer. OpenCV is up and running on Windows using

Python but will be tested with C++ and then moved onto the raspberry pi. It should be easy to implement from where it stands as becoming familiar with the library is the hardest portion. Video streaming is ultimately the cornerstone of where we stand. Using gStreamer and raspivid to send and capture video, respectively. Our main goal for the future is the smart home intercom system being able to be shipped. To allow this to happen we have to get batman-adv up and running and fully tested. The end goal is that we can add and remove intercom systems as needed and that each system is aware of the state of the other systems. For this we will be storing the data needed in the XML file Qt creates for program settings. Upon verification that the system will update all other nodes based on that XML file, we can start stress testing with video streaming. Prior to deploying video streaming we need to first get audio up and running. This will happen by installing the GPIO microphones onto the raspberry pis and verifying that they can take in audio and playback audio. Then this will be added onto gStreamer and verified that it can stream between nodes. After the nodes are verified to be transmitting data and verified to be streaming, we can load test batman-adv with the audio and video streams. We will call random nodes. While this is going on, development of the user interfaces will need to happen. An options screen will allow the user to change things in the XML file that will be stored, with some portions private like camera power and some portions public like time. The home screen will be the hub of the intercom system and what the user will primarily view. The status bar will be similar to what one is familiar with on their phone. After completion of this, loading the libraries that are required for everything else will be needed to help move towards an embedded mode. OpenCV will be run when the system is inactive and not in use to track people and update the XML file shared between devices.

3.3 Problems

The main problems we have run into are hardware based. The Raspberry Pi receiving the video stream likes to thermal throttle itself, while the capturing Pi does not have any issues. We are looking into installing a fan and heatsinks to attempt to reduce that issue, especially with the addition of OpenCV. Although documented as working, the transmission between the two devices is extremely laggy. We will be attempting to test if raw video can be streamed instead of H.264, which is likely an overall issue for both overheating and streaming lag. OpenCV is a far larger installation on the Pi than was originally estimated. Qt 4.8 requires a strong backbone from the operating system and as such, implementing Qt on Stretch Lite, or the non-GUI version of Raspbian Stretch, will be difficult due to having to individualize libraries. Overall, the main problems are seemingly from hardware limitations that we have already talked to our client about.

3.4 Interesting Bits

Qt 5.0 does not have an embedded mode in its free version, forcing us to use Qt 4.8. This is actually helpful in multiple ways. Not only does it keep costs low but it also allows us to deploy it without any major concerns for patches both security related and runtime related. It is an all in one package that would theoretically allow us to implement our whole project via its own libraries and interfaces but it would potentially ruin our chances with OpenCV due to the processing of the images, which is easier to perform with raspivid than with any other camera library.

Batman-adv simply ignores security. It is possible to encrypt things using WPA-none, or giving each device a shared key, but ultimately for hard to break security encryption it would need to be done above the network layer. The purpose of this project is to ensure security from node to node so it is likely that we will have to come up with another more secure form of encryption as a whole. It is extremely dependent on how secure we feel our network will be overall.

4 GLEN ANDERSON

4.1 Current Status

4.1.1 Overview

Most of the time spent on the project this half of the term has been trying to get different libraries to work on the Raspberry Pi, and writing some scripts to use them. Facial detection is working, and person detection should be completely done in the next couple of weeks.

4.1.2 Person detection

A critical part of the Smart Home Intercom system is person detection, as this will tell users what room should be paged when they make a call from another node. The person detection in this system needs to exist on every node and be able to tell a person apart from a pet, for example a cat. It also needs to reliably detect someone when they enter a room, even if they are not facing the camera. Currently, facial detection is working using OpenCV and should be trivial to put on each system. The script sets the video input to come from a webcam and it outputs a message when it sees a face. Eventually, this script will need to log the rooms name, the nodes network information, and the rooms current status (occupied if it saw a person within some time period) onto a file that can be shared with other nodes.

4.1.3 User interface

Since this project involves a lot of user interface design, our group is dividing ownership for this work and I am responsible for developing a status bar as well as the user setup interface that will launch the first time a system is booted. So far these have not been implemented, but I have installed QT Developer so that I can begin work on these features in the next couple of weeks. Together, our group planned what each screen should roughly look like and we discussed the user interface extensively with our client, so we have a clear idea of what each piece of the interface has to include.

4.2 What is Left

As a group, we still need to implement BATMAN so that video and audio can be streamed over a mesh network. For testing purposes, we are connecting two nodes together with an ethernet cord to stream video or streaming over a local area network, but have not done stress testing by trying to run facial detection at the same time. Audio I/O has still not been implemented on the hardware side, but we should be able to use the same library to stream the video and audio together (gstreamer even has an option to sync them).

Person detection needs to include infrared sensing so that even if a face is not visible, they can still be identified so that the node can report that the room is occupied. Currently, face detection is the only form of person detection that is implemented, and this will not satisfy the constraint on its own to check if a room is occupied.

While we have completed the design for the user interface, it still needs to be implemented in QT. After this is done, the back-end programs that run video streaming and allow users to set options will need to be linked up with the front-end so that users can place calls and set options by clicking buttons.

Other more minor requirements of the system also need to be implemented, such as the application launching in an embedded mode and allowing the user to set different options like brightness and volume. The user should be able to select from different permission levels on each node, and these levels still need to be implemented and enforced. Any

node should go to sleep after a reasonable amount of time with no use; one way to implement this would be to sync it with person detection.

While our group has alpha level functionality of individual features, we have yet to integrate them all onto two nodes and test full functionality. This will be important because streaming video and audio can be computationally expensive, especially when running programs for facial detection at the same time, and we need to find out as soon as possible if we need to adjust how this works.

4.3 Problems and Solutions

This term our group encountered a variety of problems ranging from hardware difficulties to compatibility issues with software libraries. Early in the term, I had difficulty getting the Raspberry Pi to boot to its operating system. After initially turning it on, the screen would display colored lines that covered the entire screen and it would stay there indefinitely. I eventually discovered that this was due to the power cable I was using, which was a charging cable and could not generate enough voltage, but I spent hours trying to reformat the SD card before finding the solution.

Our group also ran into issues with parts of the project working well together; we initially planned to use VLC to stream audio and video but this could not run in embedded mode, so we had to use gstreamer instead. The cameras we initially used were not infrared capable, but we have some that are and can switch over to those when we begin implementing infrared sensing for person detection.

A potential problem we could run into is a lack of storage and computing power. The Raspberry Pi that was receiving a video stream would begin overheating with our current implementation, and running facial recognition, infrared sensing, and audio streaming at the same time could exacerbate this issue. Furthermore, libraries like OpenCV and QT 4.8 are quite large and with the operating system take up a substantial amount of space. However, we are using 32 GB SD cards so there should be plenty of space for everything we need.

Scheduling continued to remain a challenge for the first half of this term due to commuting and generally busy school and work schedules. However, we were able to work around this by meeting up on weekends when we had to get a lot of work done and finding time during the week to meet up and discuss progress.

4.4 Other Interesting Information

When looking into the OpenCV library, I discovered a lot of additional functionality for object detection and read about interesting applications for the library. In a future version of this project, object recognition could be incorporated. This could be useful for determining likely user actions or seeing what food/supplies they are running out of. These features would be well beyond the scope of the current project, but could be implemented without additional libraries since OpenCV is already installed for facial detection.

Below is the code for detecting faces. Essentially, it tells OpenCV where to get video input then runs an infinite loop that looks for faces. With this current script, it simply prints that a face was detected but more information will need to be logged in future versions. To tell when a face is present, the script checks that the length of the faces object (which holds information about all faces in the current frame) is greater than zero. This script runs on a windows machine with a webcam, so some adjustments will need to be made to set the video input to be the Raspicam and make it generally compatible with Linux. Included in the slideshow portion of the progress report is a picture showing what this looks like when its running.

```
#Script based on this resource:
#https://realpython.com/blog/python/face-detection-in-python-using-a-webcam/
import cv2
import sys


faceCascade = cv2.CascadeClassifier
('C:\Users\Glen\OpenCV\opencv\sources\data\haarcascades\haarcascade_frontalface_default.xml')
#Tell OpenCV where to get video input from
video_capture = cv2.VideoCapture(0)

while True:
    #Capture frame-by-frame
    ret, frame = video_capture.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.1,
        minNeighbors=5,
        minSize=(30, 30),
        flags=cv2.CASCADE_SCALE_IMAGE
    )

    #Outline faces with a rectangle
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
    #If there is at least one face, the room is occupied -
        #eventually this needs to be logged to a file with the node's information
    #Log room name, hostname, status to a file
    if len(faces) >= 1:
        print "Face detected!"
    #Display the resulting frame
    cv2.imshow('Video', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

```
video_capture.release()
cv2.destroyAllWindows()
```

5 Conclusion

This term, our group has continued to work towards a final product and have alpha-level functionality for individual features of the Smart Home Intercom system. These features still need to be merged into the same two nodes so that they can be tested realistically. In the next couple weeks, our group plans to link the user interface with back-end code to have basic paging working for video streaming and implement person detection so that each node knows what room to page. By the end of the term, we hope to have beta-level functionality on at least three Raspberry Pi systems.