CS CAPSTONE TECHNOLOGY REVIEW

NOVEMBER 21, 2017

SMART HOME INTERCOM SYSTEM

PREPARED FOR

OREGON STATE EECS

D. KEVIN MCGRATH

PREPARED BY

GROUP 25

GLEN ANDERSON

Abstract

The Smart Home Intercom System project will require implementing features that rely on various technologies. This document aids in understanding which technologies will be used to implement this project by discussing three topics: audio I/O hardware, data management systems, and person detection. After reviewing and analyzing options for each, this document provides a recommendation on which technologies to use.

CONTENTS

1 AUDIO I/O HARDWARE

1.1 Overview

Each node in the Smart Home Intercom System will need to be able to get audio input from a user, stream it to another device, and output the audio. Since the Raspberry Pi 3 does not have a built-in audio I/O, additional hardware will be required to satisfy this requirement.

1.2 Criteria

Selecting which audio I/O hardware to use will depend on several criteria, including quality, cost, and ease of configuration.

1.3 Potential Choices

1.3.1 PI-DAC+ with PI-AMP+

This configuration would support high quality audio I/O with the Raspberry Pi 3 and require minimal setup [1]. It is also built to protect the hardware it runs on, so even if this component failed the rest of the system should remain intact. Furthermore, it is better documented for the Raspberry Pi 3 than the other listed options, making setup and troubleshooting easier if related problems are encountered. However, it is significantly more expensive than the other options.

1.3.2 Cirrus Logic Audio Card

Like the PI-DAC+ with PI-AMP+, this option would cover audio I/O with high quality. However, the compatibility between this card and the Raspberry Pi 3 is questionable since it was designed for the Raspberry Pi A+ and B+ [2]. However, it can be configured to work with the Raspberry Pi 3 [3]. This option is less expensive but would still cost substantially more than using the onboard audio card, and could require more configuration than either of the other options depending on compatibility issues. Still, this configuration is documented and this project would not be solving the problem for the first time.

1.3.3 Onboard audio card

The onboard audio card on the Raspberry Pi 3 would support audio I/O, but with very low quality [4]. For a final product, this quality of audio would not be acceptable but it may serve the purposes of a prototype if cost becomes a major factor. This would also require no additional configuration, and the built in USB ports could be used for a microphone and speakers.

1.4 Discussion

The best quality option for the least cost reviewed in this document is the Cirrus Logic Audio Card. Since audio I/O is a core piece of this project and the software libraries needed for this project are free, this is a reasonable expenditure. However, the setup could prove difficult if compatibility issues arise. The PI-DAC+ with the PI-AMP+ would offer similar quality and functionality but at a higher cost, but are well documented and fully compatible with the Raspberry Pi 3 (without additional configuration). While it would make setup minimal and contribute no additional costs beyond a microphone and speaker, the built-in audio card for the Raspberry Pi 3 has poor quality compared to the other options.

1.5 Conclusion

Due to its higher quality, cost-effectiveness, and reasonable ease of configuration, the Cirrus Logic Audio card is the best option for audio I/O for this project. It will allow high quality audio I/O which is integral to the Smart Home Intercom System.

2 Data management/storage

2.1 Overview

For this project, the last known location of a person (by room) needs to be tracked. In order to do this, some data will have to be stored to associated a person to a room so that it can be fetched later. Video and audio should never be stored, so these do not have to be considered for this part of the project.

2.2 Criteria

The criteria for data management for this project are overhead, data integrity, and speed (although speed does not need to be emphasized). Cost is not a criteria because there sufficient free options.

2.3 Potential Choices

2.3.1 mySQL-Python

mySQL-Python is a Python interface to mySQL which is compatible with Linux, which would make it a viable option for this project [5]. Of the options discussed, this has the most overhead in terms of storage space and memory usage. However, it could be situationally faster than the other methods and would provide a convenient way to interact with a database for this project since Python will be used for much of it. Since this would use mySQL, data integrity could be easily enforced.

2.3.2 SQLite

SQLite provides a low-overhead method of interacting with an SQL database while providing data faster than typical file I/O [6]. While this library is still in-progress (has potential for unforeseen bugs), it is well reviewed, tested, and typically reliable. This would have higher overhead than simply using a flat file format, but could potentially speed it up. Like above, an SQL database allows data integrity to be enforced.

2.3.3 Flat files

While using flat files for data storage and system calls to read from them is typically slower, it is still fast on small files. Furthermore, the information that each node needs to keep track of would be minimal, so managing this with flat files and parsing them with a script would be simple. Data integrity would have to be enforced by the program reading and writing to the file; inputs and outputs would need to be checked.

2.4 Discussion

This project does not involve storing massive amounts of complex data, but rather just associations for people and what room they are in. This should take a minimal amount of space and would be able to be queried quickly with any of the methods discussed above. Furthermore, the speed of a query wouldnt be a difficult consideration since they will be

infrequent. Due to the simplicity and small size of the data that needs to be stored, there is no need to setup a complex database. The difference between the speed of queries using flat files and system calls versus an SQL database would be negligible on this scale. Therefore, the greatest considerations are the overhead and ability to enforce data integrity of each. Since there is a small amount of data to consider, the overhead of reading from the file itself would be insignificant. However, an SQL database would be able to enforce data integrity whereas a flat file system would rely on a program to do so.

2.5 Conclusion

For the small amount of data that needs to be stored and queried for this project, directly reading and writing to flat files will be sufficient and should not contribute substantially to overhead.

3 PERSON DETECTION

3.1 Overview

In order to keep track of what room was last occupied, each node must have a method of identifying a person when they enter a room. In accordance with the project requirements, the system needs to be able to tell a person apart from a pet or other non-human stimulus.

3.2 Criteria

The criteria for this are cost of associated hardware, software overhead, and effectiveness.

3.3 Potential Choices

3.3.1 IR-capable camera

An infrared capable camera could be used to detect when a person walks in front of it by monitoring changes in infrared light. When a significant enough change is detected, the camera would trigger a software interrupt that would update the last known location of a person in a room. In addition to these capabilities, this type of camera would encompass the functionality of a standard camera so other methods of person detection could also be implemented.

3.3.2 Standard camera

With a standard camera, various forms of image analysis and motion detection could be used to detect a person in a room. One method of doing this is facial recognition, for which their are open source libraries such as OpenCV (discussed below).

3.3.3 Dedicated motion sensor

A dedicated motion sensor (no camera capabilities) would provide additional privacy to users, as there would not be a camera constantly running to detect a person. However, it would add additional costs and bulk to the system since the system will already require a camera. It would also likely be difficult to configure this in a way that it wouldnt be triggered by a pet or other movement.

3.4 Discussion

An IR-capable camera (which the client has offered to provide) could determine a persons presence with little software overhead. It would also be a convenient solution since this project will require the use of a camera anyway. A standard camera would also be capable of detecting and identifying a person using facial recognition or other image analysis, but it would require libraries that would add overhead. While a dedicated motion-sensor couldnt detect a persons face, it could be configured to a certain height or sensitivity to only detect people. The drawback to this approach is that it would require additional hardware, adding bulkiness to the system and contributing to the cost of the system. Furthermore, it would be redundant as it is simple to detect motion with a camera. If facial recognition was a desired feature of the system, it could still be implemented with an IR-capable camera using a library such as OpenCV, which is open source face recognition software [7]. In addition to the ability to detect a generic human face, it can be trained to recognize a specific person.

3.5 Conclusion

An IR-capable camera would be the best solution for person detection, as it would allow multiple methods of identifying a person. It also encompasses all of the functionality of a standard camera and a motion sensor which will allow for more flexibility in implementation.

REFERENCES

[1] IQaudIO. [Online]. Available: http://www.iqaudio.com/downloads/IQaudIO.pdf

[2] Cirrus Logic Audio Card for Raspberry Pi A plus and B plus. [Online]. Available: https://www.element14.com/community/docs/DOC-71261/l/cirrus-logic-audio-card-for-raspberry-pi-a-plus-and-b-plus#documents

[3] Cirrus Logic Audio Card working and streaming 24/7. [Online]. Available: https://www.youtube.com/watch?v=iuHWzpzS_hU

[4] Raspberry Pi and realtime, low-latency audio. [Online]. Available: https://wiki.linuxaudio.org/wiki/raspberrypi

[5] MySQL-python 1.2.5. [Online]. Available: https://pypi.python.org/pypi/MySQL-python/1.2.5

[6] About SQLite. [Online]. Available: https://www.sqlite.org/about.html

[7] Face Recognition with Videos in OpenCV. [Online]. Available: https://docs.opencv.org/2.4/modules/contrib/doc/facerec/tutorial/facerec_video_recognition.html