Smart Mirror

Project Report

Sponsor: MakerKid

Student Members:

Quang Trung Trinh

Tuan Minh Nguyen

Haider Ibrahim

Status

/1 Hardware present?

/1 Title Page

/1 Declaration of Joint Authorship

/1 Proposal (500 words)

/1 Executive Summary

Declaration of Joint Authorship

We are: Tuan Minh Nguyen, Haider Ibrahim, and Quang Trung Trinh, confirm that this project is contributed work between the group and expressed by our own words. Any materials used in any form (table, research, etc...) in our report will be properly referenced and acknowledged at the end of the report. The work breakdown is as follows: Each of us provided functioning, documented hardware for a sensor or effector. Tuan Minh Nguyen provided functioning and documented hardware for the Ranging Sensor Circuit. Haider Ibrahim provided functioning and documented hardware for the Microphone Circuit. Quang Trung Trinh provided documented hardware for the Speaker Circuit. Specification of our works contribution including: Tuan Minh Nguyen is the lead for peripherals connectivity to microcontroller system; Haider Ibrahim is the lead for establishing the mirror interface and internal software; and Quang Trung Trinh is the product manager, lead for further development of our mobile application, connecting the system to database, budget managing and report generating.

Proposal

We have developed a mobile application (android), interactable with databases, completed a software engineering course and produced three small embedded system with a custom PCB as well as an enclosure laser cut. Each and everyone of us finished our Internet of Things (IoT) capstone project and had the following materials ready for contribution: a working computing model of a smart phone application, an internet accessible database, an enterprise wireless (capable of storing certificates) connected embedded system prototype with a custom PCB and an enclosure laser cut, and are documented via this technical report targeting OACETT certification guidelines.

Intended project key component descriptions and part numbers

Development platform:

Sensor/Effector 1: Ranging Sensor Circuit - Arduino IDE and Raspberry Pi 3B+

Sensor/Effector 2: Microphone Circuit - Raspberry Pi 3B+

Sensor/Effector 3: Speaker Circuit - Raspberry Pi 3B+

We will continue to develop skills to configure operating systems, networks, and embedded systems using these key components to create a finished fully functioning smart mirror that interactable, accessible via the internet and modificable via mobile application

Our project description/specifications will be reviewed by, Jenifer, ideally an employer in a position to potentially hire once we graduate. They will also ideally attend the ICT Capstone Expo to see the outcome and be eligible to apply for NSERC funded extension projects. This typically means that they are from a Canadian company that has been revenue generating for a minimum of two years and have a minimum of two full time employees.

The small physical prototypes that we build are to be small and safe enough to be brought to class every week as well as be worked on at home. In alignment with the space below the tray in the Humber North Campus Electronics Parts kit the overall project maximum dimensions are 12 13/16" x 6" x 2 7/8" = 32.5cm x 15.25cm x 7.25cm.

Keeping safety and Z462 in mind, the highest AC voltage that will be used is 16Vrms from a wall adapter from which +/- 15V or as high as 45 VDC can be obtained. Maximum power consumption will not exceed 20 Watts. We are working with prototypes and that prototypes are not to be left powered unattended despite the connectivity that we develop.

Executive Summary

In this Smart Mirror project, we have made a mirror not only able to reflect physical object but also able to be hooked up with a computer system and display information. This opens the wide horizon of potential uses of a furniture that take up a lot of space but highly functional once it is interactable via the internet. Our Smart Mirror right now can display the weather, time, news header, calendar. It is also interactable with the user via our microphone and speaker system which is backed up by a google internal software. The mirror display, sound and LED system can be controlled and customized via our android application. With this application, you can control the volume of the speaker, customize the mirror display with simple drag and drop procedure, change the LED color, set a time for a voicemail to be played and connect your phone to the mirror speaker to play any audio file that you want.

Although the smart mirror has been implemented advanced functionality to stand out any regular mirror in the market, we still see a lot of space for development and improvement. Smart mirror can be customized to be a communicational device for video streaming, video call, it can be customized as a perfect fitness device with its features of both reflecting and displaying decently, or it can be a multifunctional smart device like a smart TV but with reflection functionality added on. The horizon is wide and potential of this device is highly scalable. In this project, we have demonstrated a wide range of technical skills such as: electrical system designs and production, mechanical production, programming in multiple platforms and languages, computer system

troubleshooting, etc… We hope we can further advance and able to use our knowledge in developing the product with a position of IoT developer.

Contents

List of Figures

1.0 Introduction

Smart Mirror is a device that not only able to reflect physical object but also able to be hooked up with a computer system and display information. This opens the wide horizon of potential uses of a furniture that take up a lot of space but highly functional once it is interconnected to the internet. It allows users to acquire current weather, time, news header, date and also helps children to get familiar and to learn more about technology as the device is visually cool and the application is easy to use. It also interactive with the user via our microphone and speaker system which is driven by a google service.

1.1 Scope and Requirements

Our Smart Mirror project is an Internet of Things (IoT) capstone project that uses a distributed computing model of a smartphone application (android application), a database accessible via internet, an enterprise wireless (capable of storing certificates) connected embedded system prototype with a custom PCB as well as an enclosure laser cut, and is documented via an OACETT certification acceptable technical report.

Our hardware included a Raspberry Pi 3B+ as a central microprocessor that control the mirror's parts and connected to the online database. We use Raspbian OS as our operating system and development platform for the internal software of the mirror. The mirror peripherals including the following parts: Monitor (Output Device), Speaker (Output Device), Microphone (Input Device),

LED (Output Device), and Ranging Sensor (Input and Output Device). The internal software of the mirror has the functions of:

+, Read and send signals from and to the mirror's peripherals.

+, Read and send data from a to our online database.

+, Processing input signal from the mirror's peripherals.

+, Able to run media player.

We will also use a 3D printed frames for our smart mirror to hold all the electrical parts together as a finished product.

For database, we use Firebase real-time database to store mirror's configurations and user identifications in JSON node model. A user-end software was also developed using Android Studio development platform. The program was written in Java and Xml. The name of the software is Smart Mirror and is downloadable via Google Play Store. The application is capable of:

+, Signing In / Signing Up users

+, Connecting to a Mirror's database with serial number

+, Controlling the Speakers output and play time

+, Changing the LED color, turn on/off LED and Ranging Sensor

+, Monitoring the display of the mirror

+, Read and Send data to Firebase database

The Mirror system and the software will be connected via Firebase real-time database and not directly connected to each other.

Report

/1 Hardware present?

/1 Introduction (500 words)

/1 Scope and Requirements

/1 Background (500 words)

/1 References

2.0 Background

Firstly, we would like to thank our mentor Jennifer Turliuk from MakerKids for supporting our project. “MakerKids runs summer camps, after-school programs and parties for kids from the age of 5 to 13 to learn about Robotics, Coding and Minecraft” (MakerKids). Their main purpose is to help the kids to develop their skills as well as their confidence. The iDTech website wrote:” Kids have big imaginations; too big to be contained. Where in the past they only had art supplies like crayons and colored markers at their disposal to get those ideas out and into a conveyable form, they now have computers, tablets, and so much more to help them turn such thoughts into reality.” (Ryan, 2018). As a result, the kids will have qualified level of knowledge to become innovators, inventors and entrepreneurs in the future by making their imaginations become visible.

Secondly, we would like to thank our professor Kristian Medri for helping us a lot throughout this project. He gave us good advices on how to meet the requirements of MakerKids.

Our task is to create an object that can help the kids to learn about technology by having characteristics such as: visually cool so the kids will be interested in; hackable by the kids so the code must not be too hard; easy and simple interaction. This is not only to attract the kids but also to develop their skills interacting with Technology to solve visible problems or change visible peripherals. Our project and MakerKids want to help children

understand that Technology is not just about writing some boring code, it is about solving life problems and make life easier.

3.0 Methodology

3.1 Required Resources

Report

/1 Parts/components/materials (500 words)

/1 PCB, case (500 words)

/1 Tools, facilities (500 words)

/1 Shipping, duty, taxes (250 words)

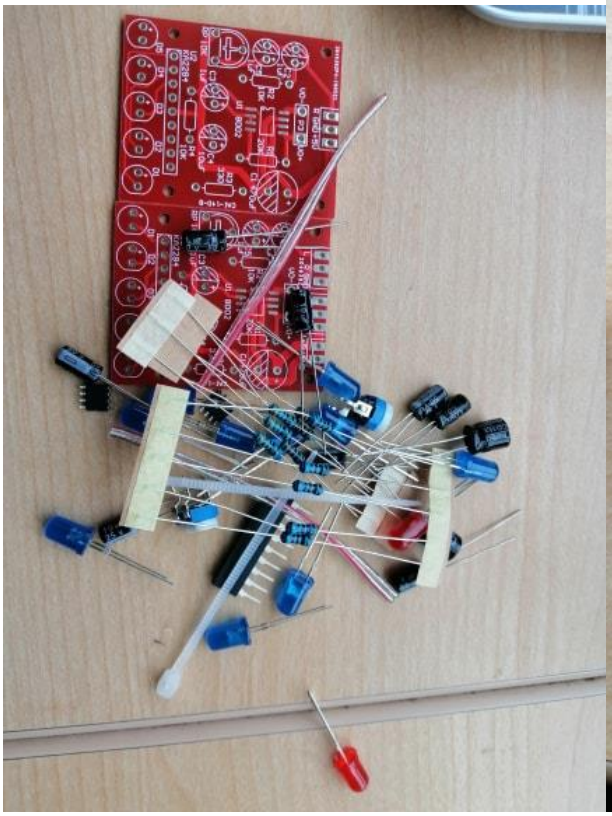
/1 Working time versus lead time (250 words)

3.1.1 Parts, Components, Materials

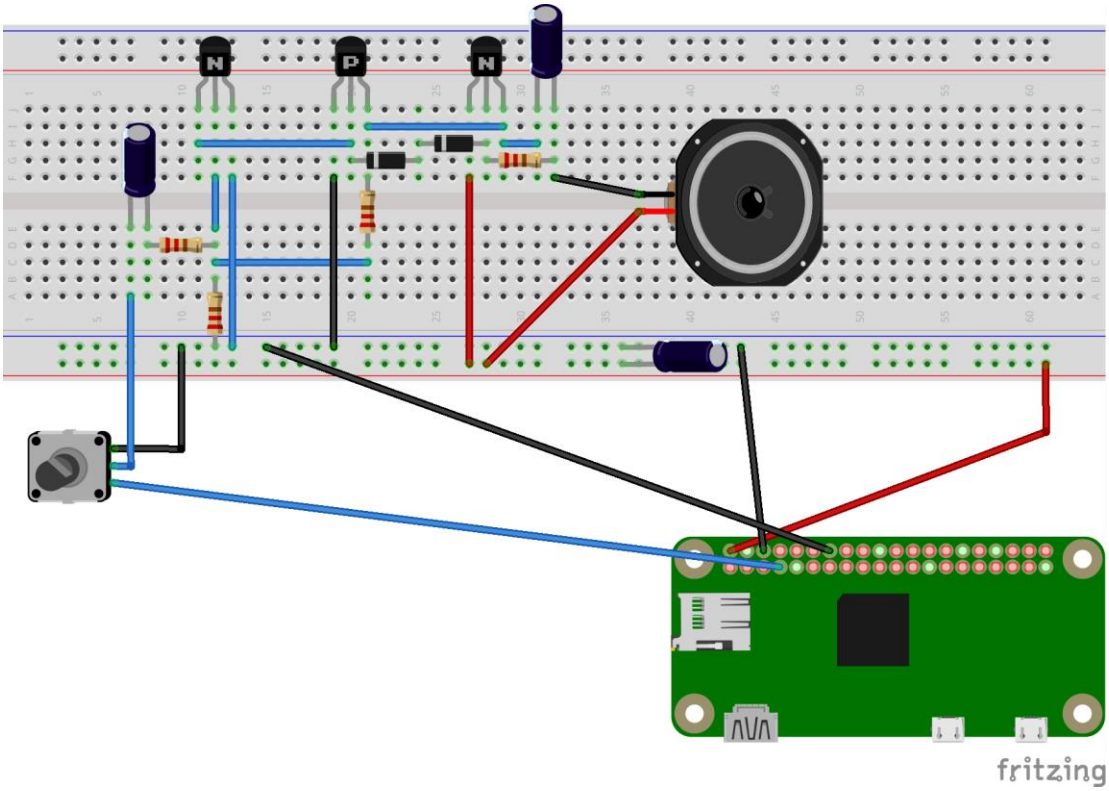
On the hardware aspect of our project, we are using a Raspberry Pi 3B+ as our microcontroller board. There is a newest version of Raspberry Pi which is Raspberry Pi 4, however, we chose to work with Raspberry Pi 3B+ since we have more experience working with Raspberry Pi 3B+ from our previous semesters and courses. Beside Raspberry Pi as our microcontroller, we also have our peripherals that will be implemented in our smart mirror system and will be controlled by the Raspberry Pi 3B+. Here is the listing and description of our parts:

A finished speaker implemented with PCB board with amplifier circuit and an enclosed laser cut case. Our team member, Quang Trung Trinh, has designed and solder the circuit in last semester in CENG 317 class with the help of professor Kristian Medri. The speaker PCB is consisted of multiple transistors, resistors, capacitor that combined to make an amplifier circuit. It also consists of some LEDs lights that will

indicate the audio output level. A computerized chip is also implemented in the speaker to monitor the light on the PCB and also help amplifying the audio output. The PCB board consists of some connector to the speaker hat, an USB and 3.5mm audio hack cable as audio input. The speaker is held with a plastic laser cut design. The design and specification of the PCB designs as well as the final product will be shown in the image down below.



\


N
P
N
fritzing

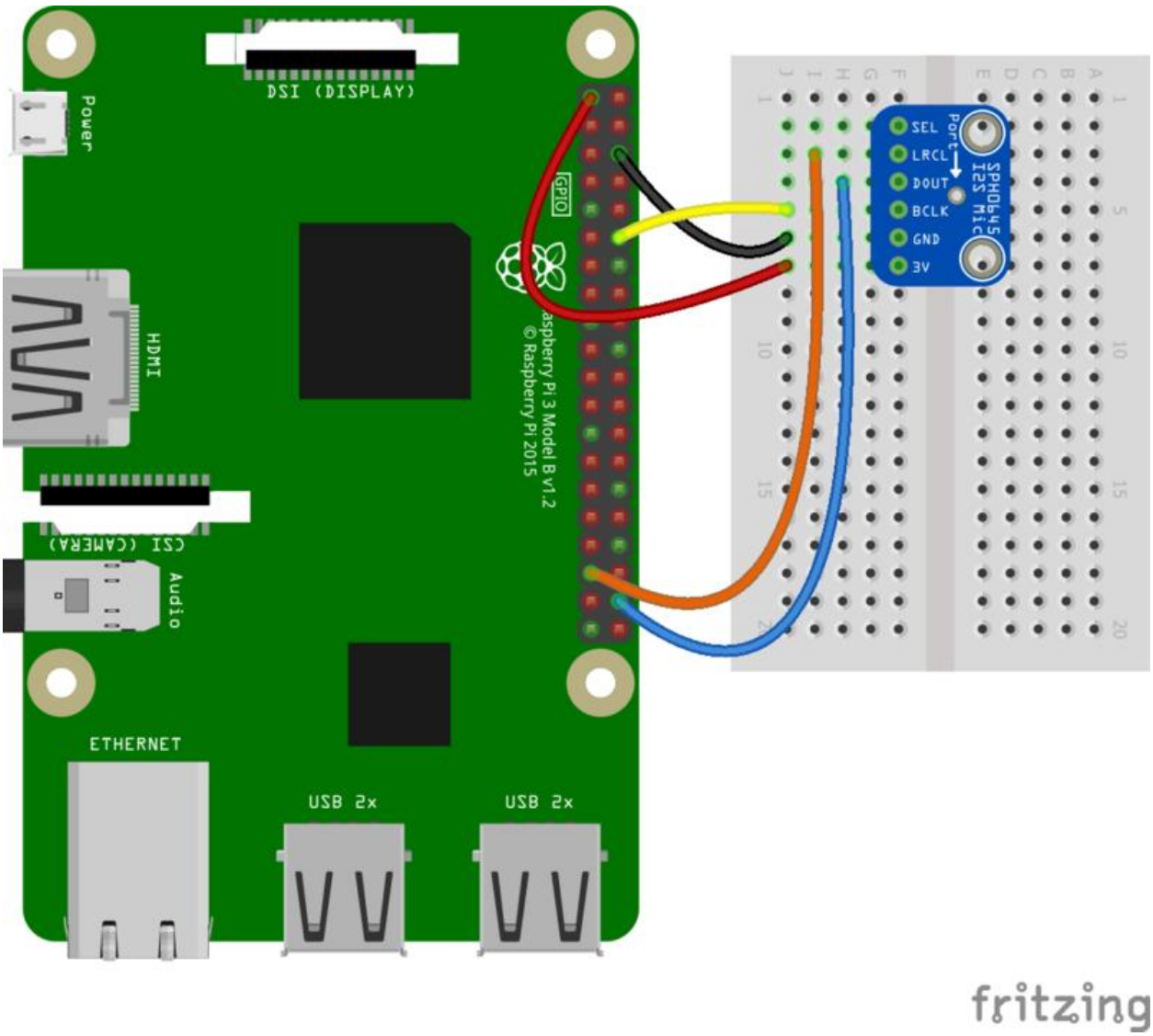
We also having our microphone module implemented in our system. The microphone module is made intractable with our microcontroller system by our team member, Haider Ibrahim. For many microcontrollers, adding audio input is easy with one of our analog microphone breakouts. But as we do our research of bigger and better microcontrollers and microcomputers, we find that we don't always have an analog input, or maybe we want to avoid the noise that can seep in with an analog mic system. Once getting past 8-bit micros, we often find an I2S peripheral that can take digital audio data in.

Instead of an analog output, there are three digital pins: Clock, Data and Word-Select. When connected to your microcontroller/computer, the 'I2S Master' will drive the clock and word-select pins at a high frequency and read out the data from the microphone. No analog conversion required!

The microphone is a single mono element. We can select whether we want it to be on the Left or Right channel by connecting the Select pin to power or ground. This I2S MEMS microphone is bottom ported, we make sure we have the hole in the bottom facing out towards the sounds the system want to read. It's a 1.6-3.3V device only, so not for use with 5V logic. Many microcontroller boards *don't* have I2S, so we made sure its a supported interface before you try to wire it up. This microphone is best used with Cortex M-series chips like the Arduino Zero, Feather M0, or single-board computers like the Raspberry Pi – our microcontroller. Here is some indication of our microphone system:

- Mic 3V - Pi 3.3v
- Mic Gnd - Pi Gnd

- Mic SEL - Pi Gnd (this is used for channel selection. Connect to 3.3 or GND)
- Mic BCLK - BCM 18 (pin 12)
- Mic LRCL - BCM 19 (pin 35)
- Mic DOUT - BCM 20 (pin 38)



Those are two main modules that we really wanted to mention in details. We also have our Monitor for the display of our smart mirror and act as one of our component in our development platform. The ranging module, keyboard,

Bluetooth mouse, power supply cable, connection cable, designed PCB boards with minor electrical components (resistors, capacitors, etc…) and designed plastic frame are also parts of our material list.

3.1.2 Manufacturing:

Printed Circuit Boards (PCBs) form the backbone of our project. These miraculous inventions pop up in nearly all computational electronics, including simpler devices like digital clocks, calculators etc. For the uninitiated, a PCB routes electrical signals through electronics, which satisfies the devices electrical and mechanical circuit requirements. In short, PCBs tell the electricity where to go, bringing our electronics to life.

PCBs direct current around their surface through a network of copper pathways. The complex system of copper routes determines the unique role of each piece of printed circuit board.

Before PCB design, circuit designers (us) are recommended to get a tour of a PC board shop and communicate with fabricators face to face over their PCB manufacturing demands. It helps prevent designers making any unnecessary errors from getting transmitted during the design stage. However, as more companies outsource their PCB manufacturing inquiries to overseas suppliers, this becomes unpractical. With our experience of designing PCB circuit board, here are some steps taken that we recommended to follow

Circuit boards should be rigorously compatible with, a PCB layout created by the designer using PCB design software, in our case it was Fritzing.

Once the PCB design is approved for production, we export the design into format their manufacturers support which is the prototype lab in our case.

After a thorough examination, designers forward PCB file to the prototype lab. To ensure the design fulfills requirements for the minimum tolerances during manufacturing process, almost all PCB Fab Houses run Design for Manufacture (DFM) check before circuit boards fabrication. Our files sometimes got sent back due to unqualified electrical designs, so we would have to redesign and re-start the cycle. After the design is approved, the PCB board will be made and we can solder and implement our electronics to make a completed circuit.

3.1.3 Tools and Facilities:

For tools and facilities, multiple electronics, hardware and development platform has been utilized. From our tool kit, we used bread board, electronic parts (resistors, capacitors, LED, jumper wires etc…) and Arduino Uno. These components are used to electrically and programmatically test our speaker amplifier and ranging sensor modules. We also use the Arduino integrated development environment (IDE) as our development platform. We were able to find multiple sources of code that works with our

hardware in google search for testing purposes. The facility in Humber that we primarily used is the prototype lab, and the CENG 355 (Computer System Project) lab. In these rooms, we had access to the solder station, mechanical and electrical parts and tools, electric meters, power source and computer station. We also received help from our Professor and staffs in these room for some procedure such as: PCB design, laser cut printing, 3d printing and technical advices for the making of our capstone project. We also borrowed computer peripherals (keyboard, mouse, connectors) from the crib in the process of booting up our Raspberry Pi. These three facilities that were listed are all located on the 2nd floor, J wing of Humber North Campus.

On the software side, we used Arduino IDE to program our ranging sensor module to get the reading. Raspbian OS is the operating system that we used to run our Raspberry Pi, on top of this OS, we installed google assistant API and Magic Mirror application to process voice recognition and take care of the displaying of our mirror. The terminal on this OS is also used as our development platform for the microphone module. Another tool that we use is the Android Studio, an Integrated Development Environment that we used to develop our android application named Smart Mirror. This IDE provided the framework for us to design the front-end of the application, making the front end interact able with users, validating information, creating application flow and interacting with our back end database. We used the Firebase Real-time Database to store and pull information. It acts as the middle man between the android application – Smart Mirror and the mirror, enable users to control the physical mirror's peripherals. We also use Hostinger web service and phpMyAdmin as our backup database

3.1.4 Shipping, duty, taxes:

Shipping, duty and taxes came about 315$ in total. This total come from buying of monitor, Raspberry Pi 3B+, Raspberry Pi connectors, sd card, monitor, Speaker package, microphone package, ranging sensor package Acrylic mirror and LED strip lights. We don't have a lot of part that we already have so there are a lot to buy for this project. The Raspberry Pi, Sd card and Monitor alone cost 200$ in total. Trung and Haider actually already had four Pi (2 Raspberry Pi 3 and 2 Raspberry Pi Zero). Trung Raspberry Pi Zero couldn't boot up after Raspbian is installed on the Sd card, Trung Raspberry Pi 3B+ is booted up, working on Raspbian for 4 times and never come up working again. Haider Pi Zero and Pi 3 follow the same story except Haider's Raspberry Pi Zero actually booted up and working but we decided we will need a microcontroller with more computing power for our project. That's why we had to purchase a brand new Raspberry Pi 3B+. The new Pi, Monitor, acrylic mirror and LED strip are bought on this semester by Trung and Haider. Speaker package was bought from Trung from the last semester, got soldered, connected and fully working. Microphone package was bought from Haider last semester and the ranging sensor package and Arduino Uno was bought from Minh last semester. We ordered all of our components from Amazon. Amazon is really the best choice for us, they're very reliable in term of giving the arrival date of the bought items. Trung and Minh have Prime membership so most of the parts came within one day of purchase and the shipping for most of the materials are free.

3.1.5 Time expenditure:

For this project, the time expenditure was the length of two semesters, which is 26 weeks (including the break time in the middle). The actual combined time for the whole project would be around 5 days. In the first semester, we focused on three separate parts of the project: getting our custom designed hardware modules ready for connection and use, develop an android application to control the mirror peripherals and set up a database as the middle man of the application and the mirror hardware module. There are several issues which delays our process of making the smart mirror, however, we managed to get the smart mirror done on time. In the process of making hardware, Minh's and Haider's PCB design were sent back twice due to copper overlapped. This event delays us a week of re-design and remake PCB board. Trung had to redesign the circuit and reorder the new speaker amplifier circuit due to the termination of parts shipping, this event delays Trung 2 weeks in the process of making speaker amplifier. Trung had to change the decision of database, this delays the team 2 weeks (including the valuable reading week) in the process of finishing the android application. We chose to go with AWS web service as our cloud database at the beginning. However, later on, we found out that the service is quite expensive (for our budget) and hard to use. We finally come to the decision of working with Firebase, a cheap price, pay as you go real-time database. This database work well with Android Studio applications and both of the tools come from one company which is google. Trung also had a web service with Hostinger that use the LAMP stack as a backup database, time of configuring this would not be counted in the time budget of making this project. However, if we need to change the database one more time, the time of

configuring Hostinger database would be counted as the part of project time budget. We have this database in case of inability to connect both android studio and the mirror system to the firebase database. The process of making android studio application was smooth and on time with the help and supervision of professor Haki in Software Project class. To summarize, there were some delay in the process of making project but we manage to finish the project on lead time.

Working time versus lead time.

3.2 Development Platform

3.2.1 Mobile Application:

Status

/1 Hardware present?

/1 Memo by student A + How did you make your Mobile Application? (500 words)

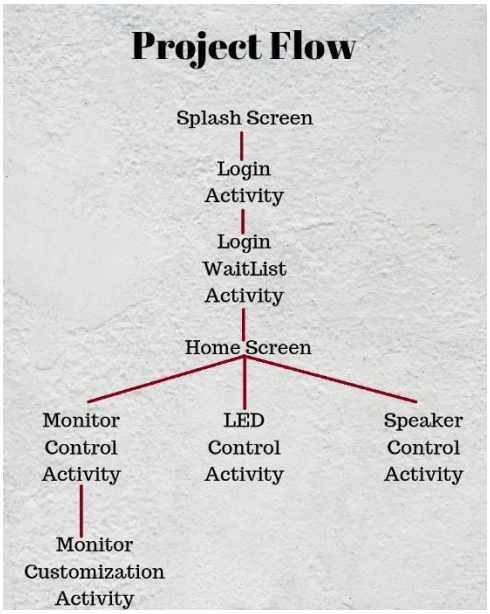
/1 Login activity

/1 Data visualization activity

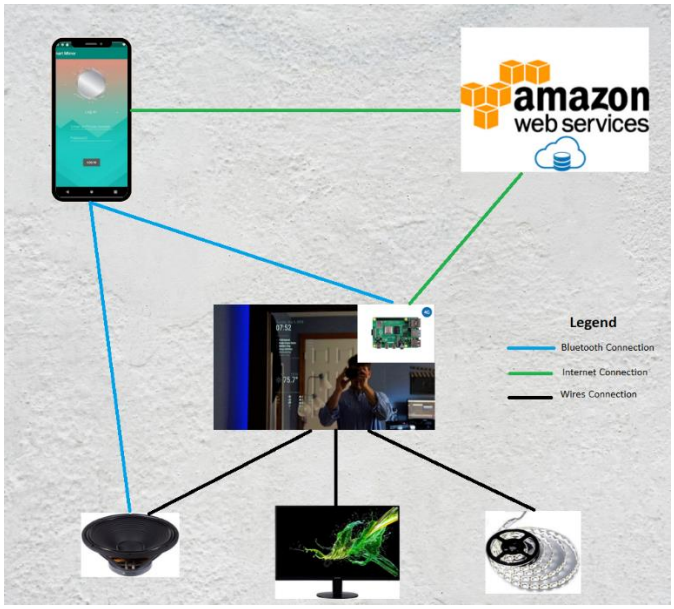
/1 Action control activity

The Mobile Application was finished in the first semester with the combination work of the whole team. We started off by brainstorming the idea of the application. What would the purpose of the application would be? How would it help user in using the smart mirror? We decided to make an application that is able to control the mirror peripherals like: speaker, display, LED lights. We think the ability to control the smart mirror would really excited users with curiosity since they have more control of the system and this would bring more sense of control and ownership of the user to their

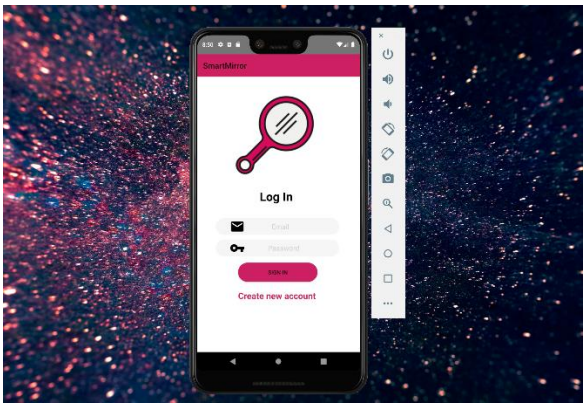
purchased smart device. The team move on to create the application flow and functions of different activity of the application. This was necessary so that we can plan forward our milestone and deadline, make it easier to determine the different pages (activity) and function of each page of the application. We also make the information flow architecture design to make clear which is connected to what and a mockup of application. Then a Gantt chart was fully completed to determine the milestone of each stage of the application.



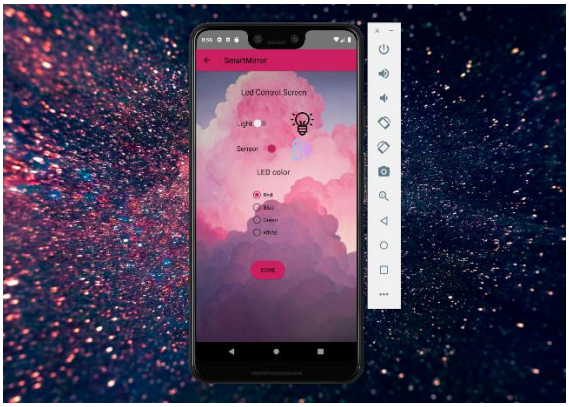
Project Flow



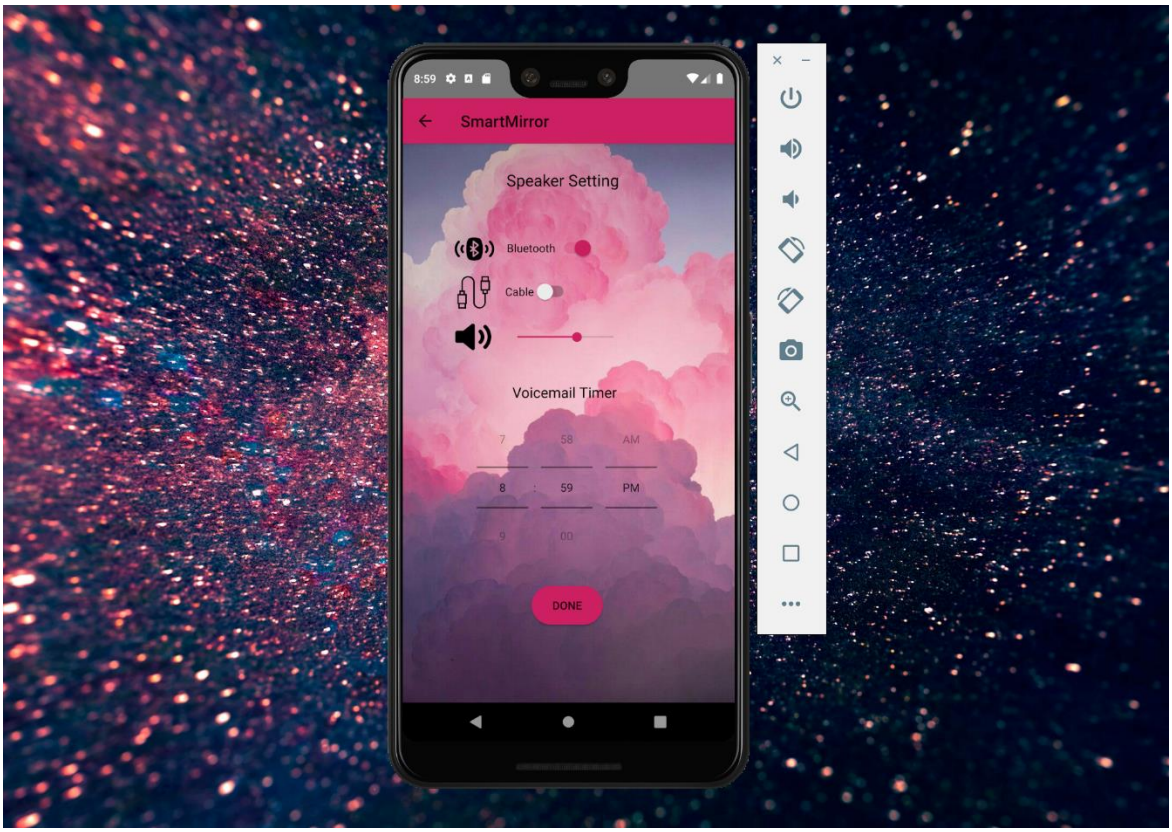
Architecture Design



Login Screen



Data Visual Screen



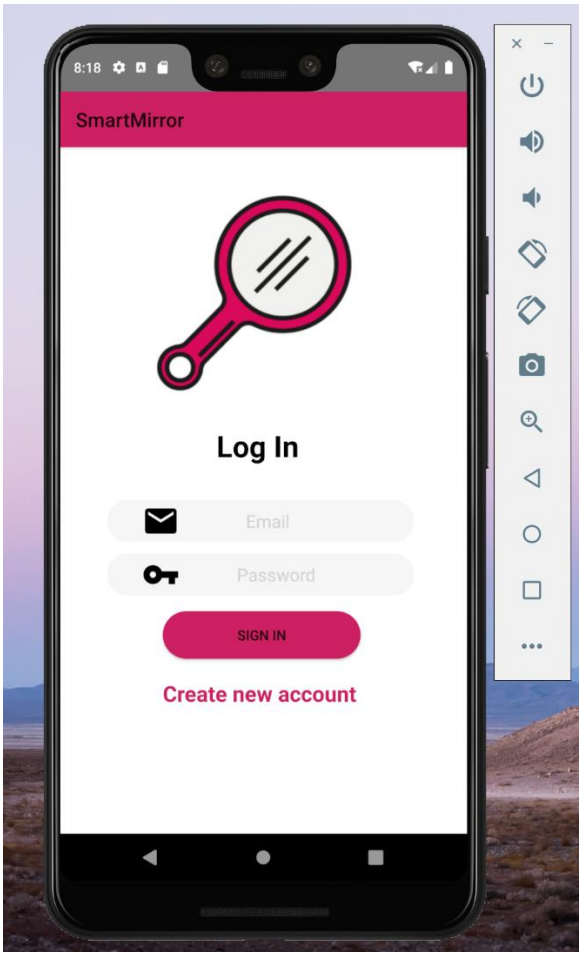
Data Control Screen

After the completion of the Gantt chart, we start working on our actual app. We decided each individual will take care of different part of the application. Minh created the front end of the application, finish the front end of all the activities in the application

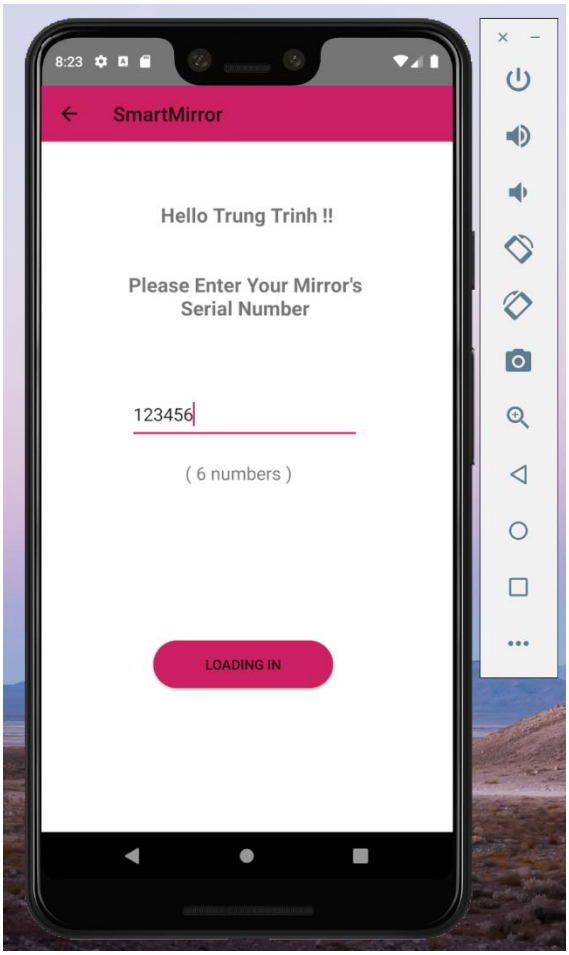
(except for monitor control activity). Haider take care of the monitor control activity, this is the hardest part of the application in term of designing and data input/output. Trung takes care of the database part, make sure the application fully functioning. Firstly, Minh and Haider design the whole application front end without implementing the backend. Meanwhile, Trung was researching and implementing different backend technologies (AWS, phpMyAdmin and Firebase) on separate android application to determine the best technology for the project. Both sides have had some issues, made big changes and finally combined to have the application working. The front-end needed to change the background, adding the side navigation for the sake of user experience, added one more activity, added the splash screen. For the backend, an AWS database was set up and ready for using but it was not sustainable in term of cost and hard to send and receive data with android studio application. A major shift of database was made from AWS to Firebase. Finished all those parts, now we have the application with the full application flow as planned. However, it is not fully functioning yet, users unable to actually register and login to the system to interact with the database, none of the data is sent and receive from application to database. Trung finish the application by implementing the Firebase backend to make the application fully interactable with Firebase. User now can sign up for account with their email, user name and password, login with proper validation, choose the mirror that they want to monitor by a serial number, send a receive data to and from Firebase.

We used Android Studio to make our application. It provides the framework in which we can use the drag and drop procedure to design our front end. Android Studio also generate the .xml file that determine the items positions and page design. We can

also change the design by changing the .xml file. Java programming language was use to capture application events, handling information validation, establish connection with Firebase and interacting with the Firebase database. Down below is our login activity, data visualization activity and data control activity.

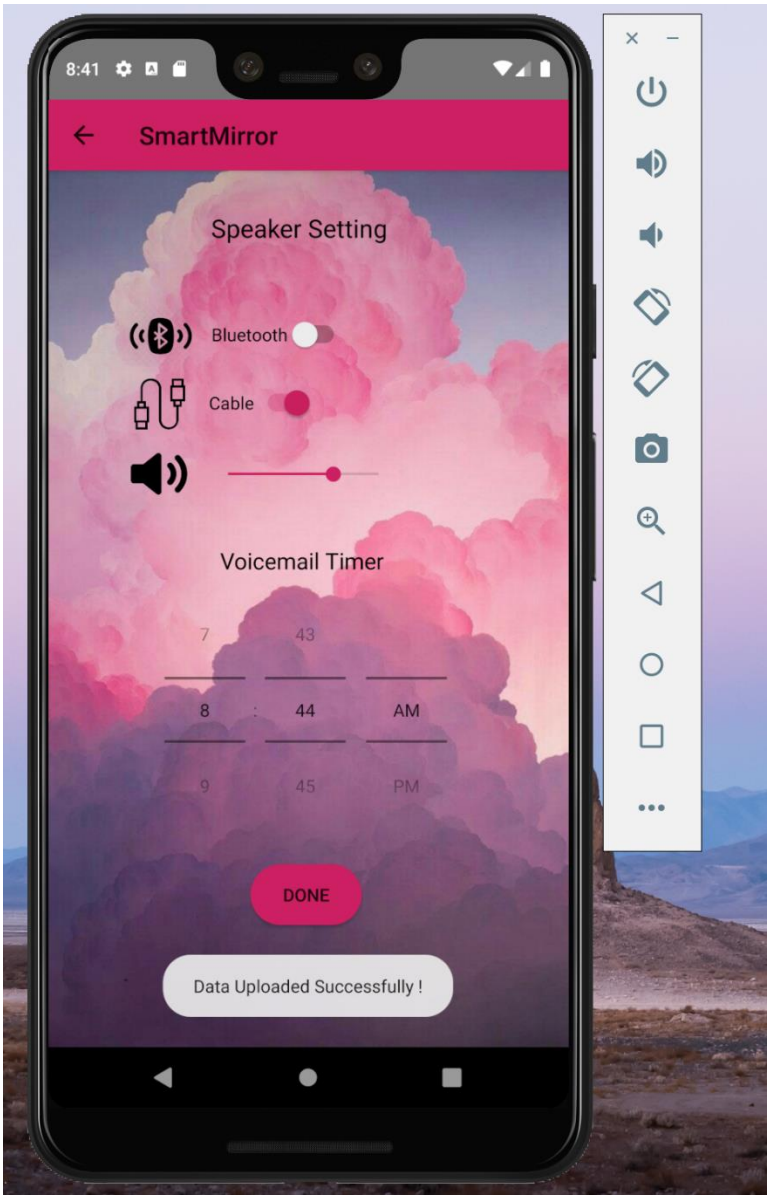


Login Activity



Data Visualization Activity

(After logging in, there is a text View that has the name of the user, in this case we have: “Hello Trung Trinh”, the User name Trung Trinh is taken from the database)



Data Control Activity

(The data of this activity

got sent over to the database)

3.2.2 Image/firmware

Status

/1 Hardware present?

/1 Memo by student B + How did you make your Image/firmware? (500 words)

/1 Code can be run via serial or remote desktop

/1 Wireless connectivity

/1 Sensor/effector code on repository

Our project is an interactive smart mirror, and by most crucial hardware components is a monitor screen, which will display information in the form of widgets such as news, calendar, weather, etc. Displaying information as widgets and retrieving the data from API is not tricky the part. The tricky part is creating a software backend system that can take all of these widgets and spit out a nice front-end. At the same time, the system widgets have to be modular and loosely coupled with actual software backend. The user should be able to swap any widgets/modules without breaking the code. After a lot of research, I found software in GitHub called "MagicMirror2". It's an open-source smart mirror platform that is based on a modular plugin system, and it's built using the Electron framework, which is a hybrid of desktop and web framework that gives the best two worlds. Electron is a desktop app that runs a minimal browser by using Chromium and display web pages built using web technologies such as JavaScript, React, etc. At the same time, it can interact with the computer operating system and has access to the files of the computer by using Node.js, so it has the best of two worlds. Another feature of Not magic mirror software itself, it has a decent community of developers and

contributors. There is an extensive database of modules that were developed by them with a variety of applications, such as voice control, health, transport, etc. Therefore, I can utilize or modify the already existing module without the hassle of creating one from complete scratch. Now I have MagicMirror2 software running with default modules. For our smart mirror, we need to implement two modules, one module for google assistant that uses i2s microphone and another module to turn on/off the mirror based on the ultrasonic distance sensor. I currently have all the code for google assistant and trying to get google assistant working, but unfortunately, without much luck because of python errors, I am still trying to understand. My Idea is to test google assistant by sending wav file as input than if it's a success I can create a script for recording and sending wav file to google assistant. In parallel to this, I tried to look for an already existing module in a database that can be used for putting a smart mirror to sleep when the user is not present. There was dozen of the modules that used the ultrasonic sensor for gesture detection, but none of the modules used the ultrasonic sensor for putting the smart mirror to sleep. Initially, my first step to resolve this problem was to take the gesture detection module and modify it in such a way that when it senses motion, it put the smart mirror to sleep. I tried for a week, but I had issues such as unable to put a smart mirror to sleep and instead got a black screen when I start my MagicMirror2. To resolve this issue, I found a module called "MMM-PIR-Sensor" that meets the project requirement, but it forced me to switch from the ultrasonic sensor to the passive infrared sensor. I installed the module, and everything works perfectly. If the user is present, the mirror will be on; otherwise, the MMM-PIR-Sensor module turn off the screen.

3.2.3 Breadboard/Independent PCBs

Status

/1 Hardware present?

/1 Memo by student C + How did you make your hardware? (500 words)

/1 Sensor/effector 1 functional

/1 Sensor/effector 2 functional

/1 Sensor/effector 3 functional

As we have said before, Printed Circuit Board (PCB) is one of the most important objects of our project. It provides us mechanical support and electrical connection between our sensors, computer and electrical source. Our team built smart mirror's hardware which includes all 3 sensors: ranging sensor, microphone and speaker connected to Raspberry pi 3 based on 3 steps: design, test and print/solder. For this part, we will focus on explaining how we do the design step for the hardware.

Before designing, our team had to decide what components to be used. We must decide what we must change in our old components to be fit with others. After a week of discussion and exploring new components, we decided to use a passive infrared sensor (PIR sensor) instead of our old hc-sr04 ultrasonic sensor to be easy to build on the same Raspberry pi 3. Because our old ranging sensor was hard to be coded and connected to Raspberry pi 3. That was the only change in our components and then it was pretty easy for us to do the connection between those sensors on the same PCB as they are all able to be connected to Raspberry pi 3.

The designing part started right after we had decided required components for our project. For design, we used Fritzing as our application because it was always be recommended by our professor, we had already done many designs on Fritzing and had earned experiences using this app so it was easier for us to use.

At first, we design the Schematic version for the hardware, the components' connections in schematic view was just all about right connection between Raspberry pi's pins and three sensors, therefore, it wasn't very hard to design. There was a little problem while we design the schematic view that Raspberry pi 3 and Passive Infrared Sensor cannot be found in the parts of Fritzing, so we had to download then import it from the Internet. The next step was to look up for pin layout of Raspberry pi 3 on Google and connected all those sensors to it. After connecting all wires, I recommend you to put colors to the wires so we can identify connections between power and ground or connections between sensors. This is our hardware's schematic view:

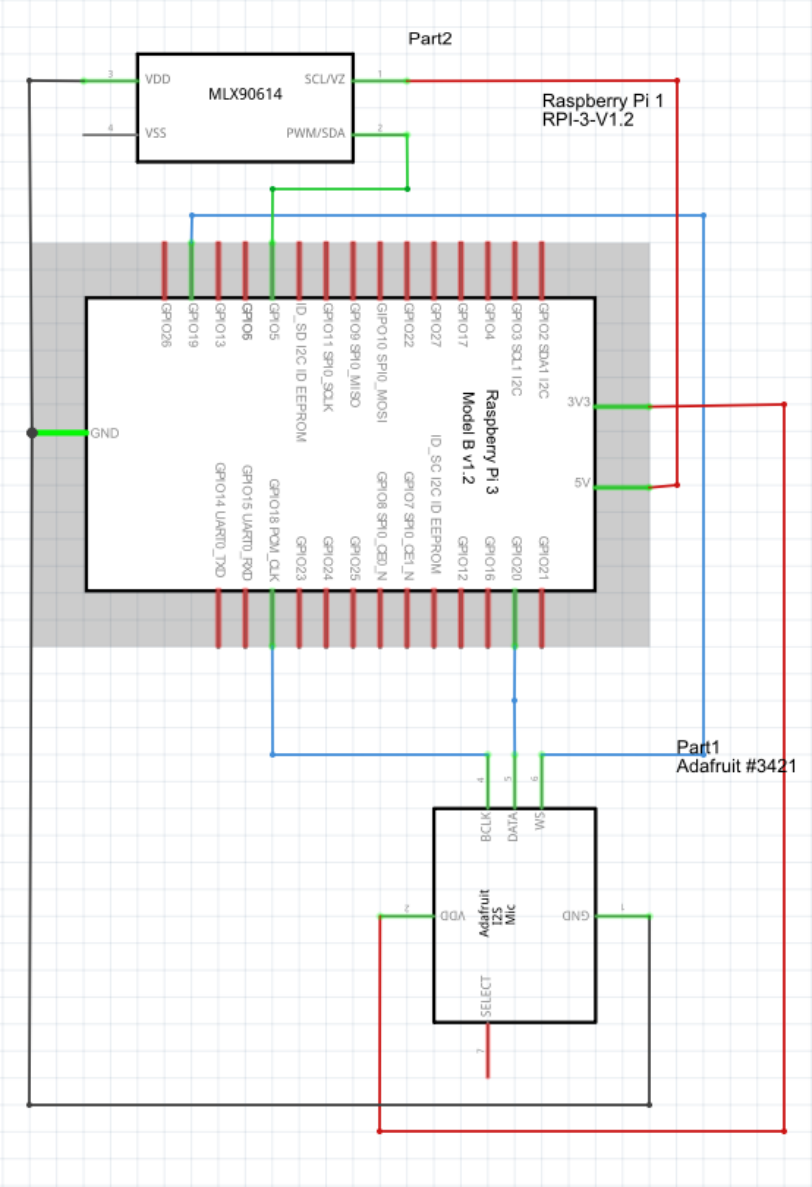


Figure 3

(Power connection: red wire; ground connection: black wire; others: blue wire for adafruit sensor and green for PIR sensor)

Secondly, based on the schematic view (Figure 3), we design the Breadboard version which ensures that all the connections are right (the same as schematic version's connections) and all the wires' colors are the same with schematic view so

that the viewers won’t get confused when looking at our design. Here is our Breadboard version:

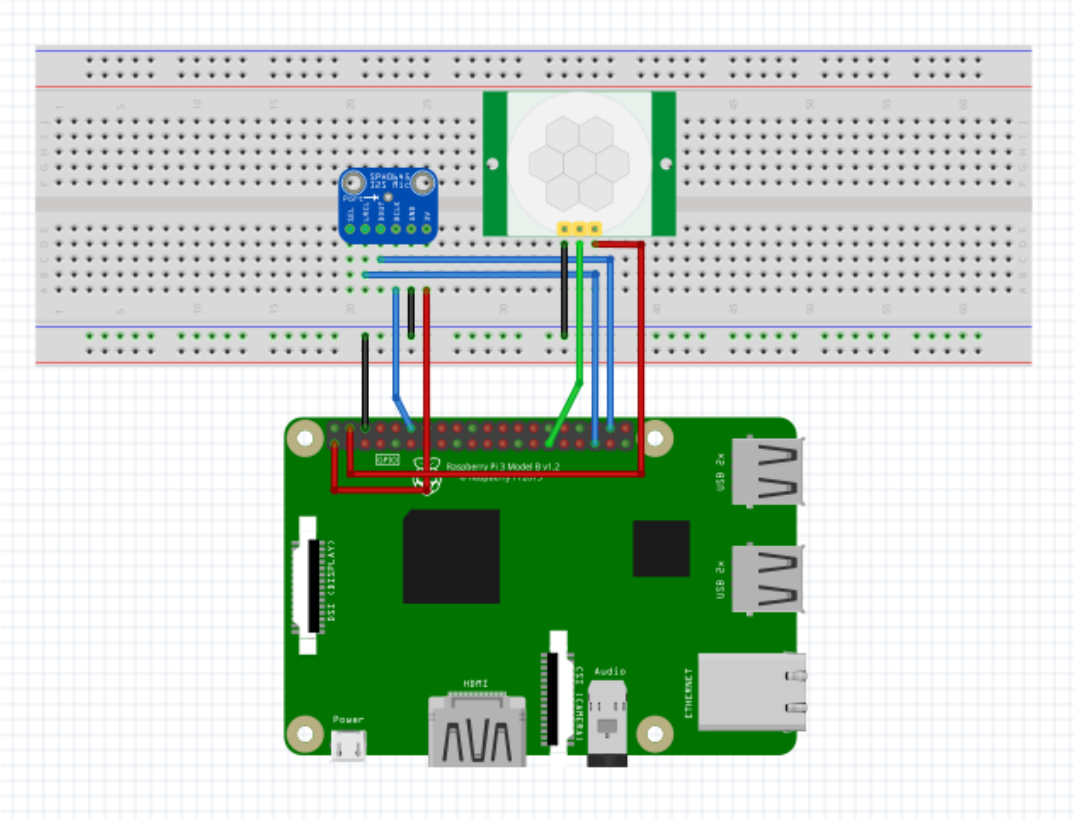


Figure 4

The prototype was absolutely the same with the Breadboard design version in Fritzing. There were no difficulties doing it. Moreover, we can directly connect all sensors to Raspberry pi 3 without the help of Breadboard.

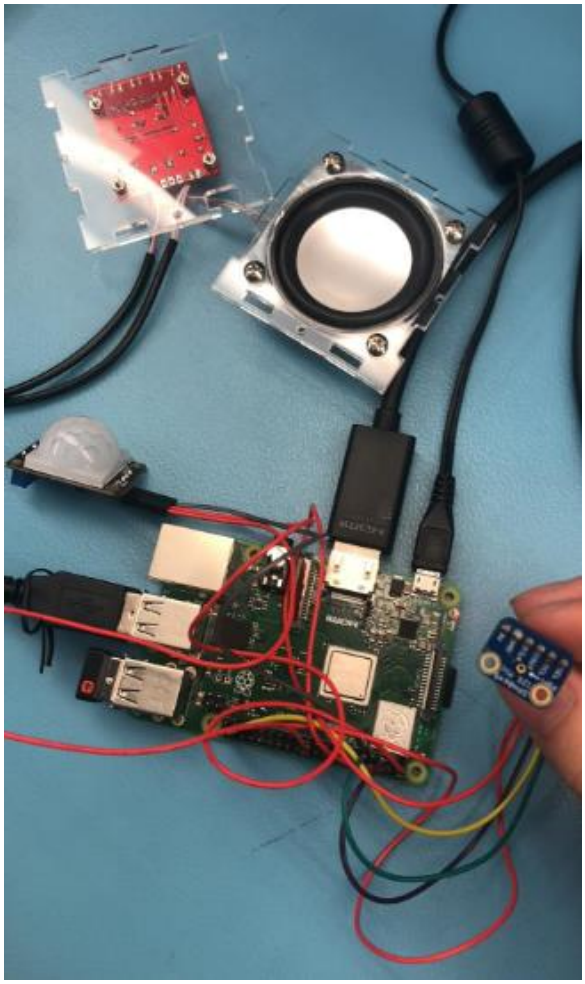
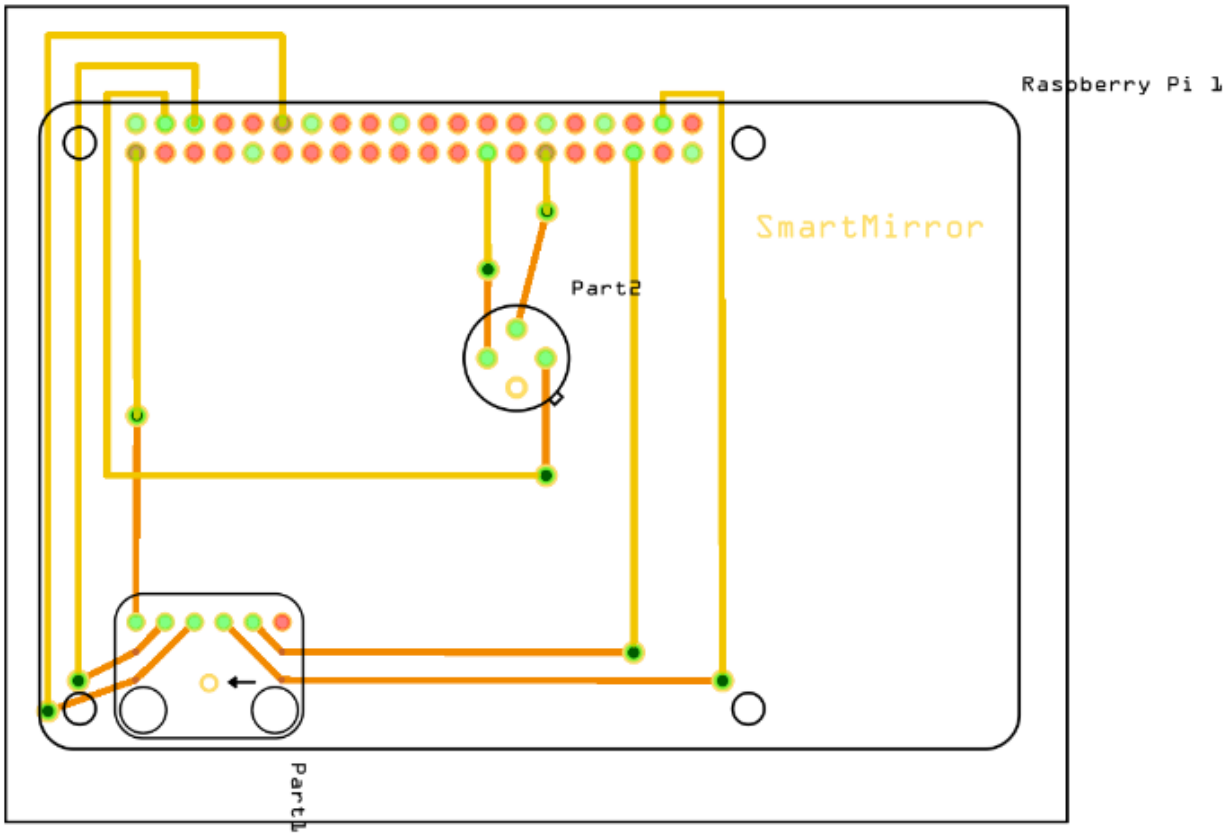


Figure 5

Last but not least, as we already done the schematic (Figure 3) and breadboard (Figure 4) versions, we did the PCB version with the same connection. We had to make sure all the wires do not cross each other so the PCB can work perfectly. Moreover, we had to create via holes (with recommended size by our professor in order to solder easier) to transfer connection from one side of PCB to another. Otherwise, we learned from our experiences and made sure all connections are on the right side of the PCB (connections from via holes to the raspberry pi will be on top side, connections from via holes to sensors will be on bottom side). Here is our PCB design:



3.2.4 Printed Circuit Board

Demo

/1 Hardware present?

/1 PCB Complete and correct

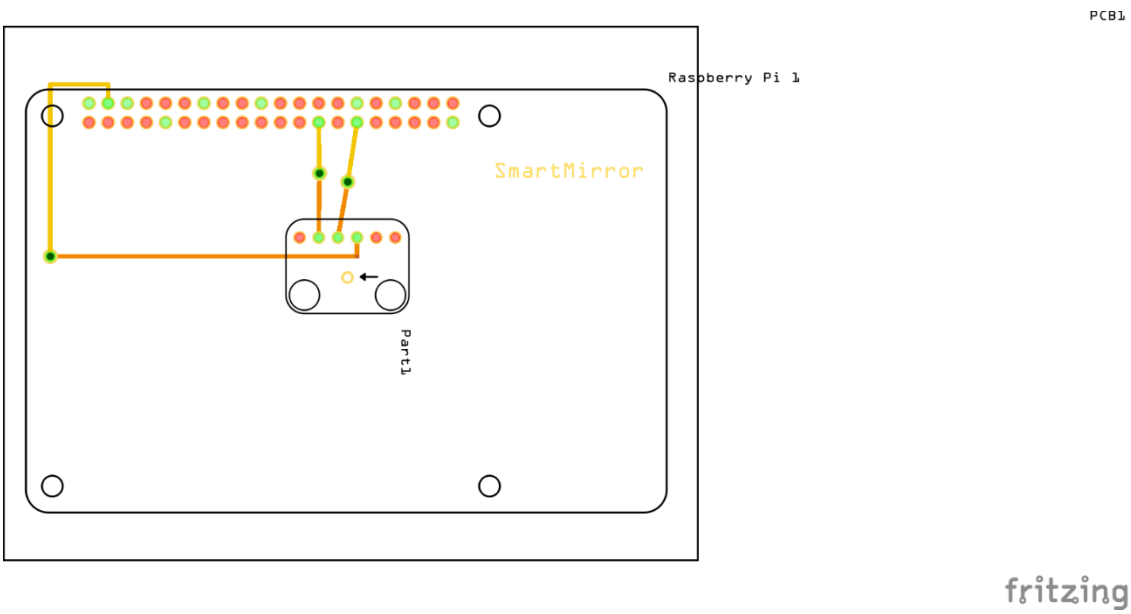
/1 PCB Soldered wire visible but trim, no holes or vacancies

/1 PCB Tested with multimeter

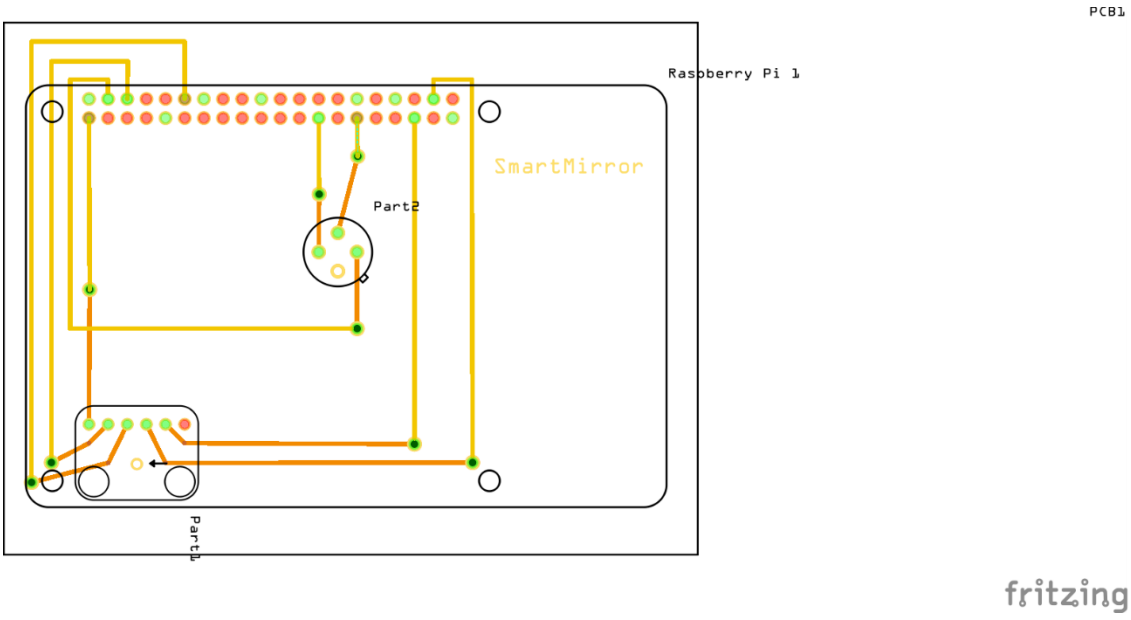
/1 PCB Powered up

How did you build your Prototype: PCB?

After having the design for PCB in Fritzing from the previous part (3.2.3), we had to check for mistakes again before laser cutting the PCB. First, we had to make sure every connection was on the correct side of the PCB (wires from Raspberry Pi to via holes are on top layer, wires from via holes to sensors are on bottom layer). Secondly, after checking several times, we had found out that all the PIR modes from the Internet don't have 3 legs alignment in PCB design so we need to alter it with another sensor which has 3 legs aligned so we would be able to connect the PIR into those 3 legs after laser cutting. Here is our alternate 3 legs alignment design for PIR:



And here is our first design:



3.2.5 Enclosure

Demo

/1 Hardware present?

/1 Case encloses development platform and custom PCB.

/1 Appropriate parts securely attached.

/1 Appropriate parts accessible.

/1 Design file in repository, photo in report.

How did you build your Prototype: Case?

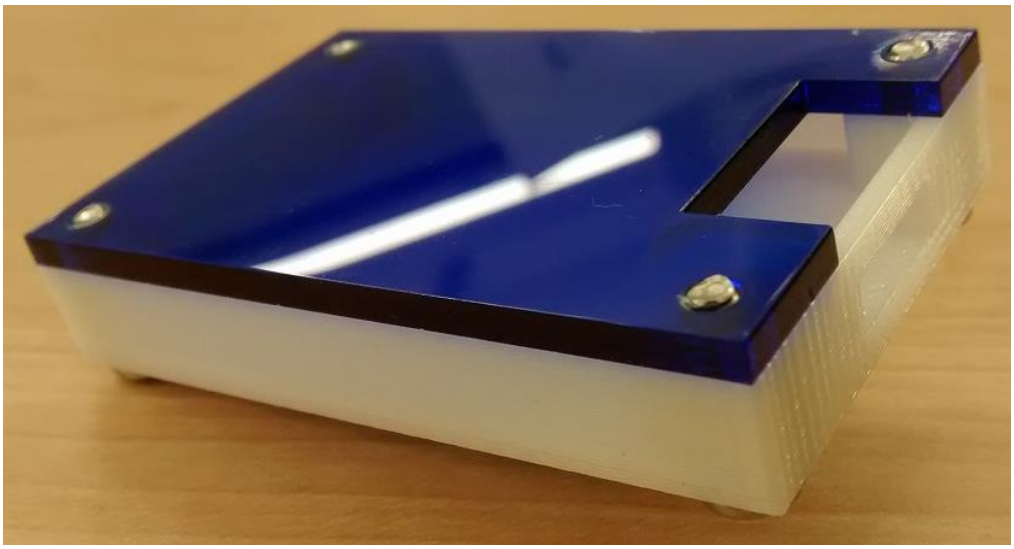


Figure 1. Example enclosure.

3.3 Integration

Demo

/1 Hardware present?

/1 Data sent by hardware

/1 Data retrieved by mobile application

/1 Action initiated by mobile application

/1 Action recieved by hardware

Report

/1 Enterprise wireless connectivity (250)

/1 Database configuration (250 words)

/1 Security considerations (500 words)

/1 Unit testing (900 words)

/1 Production testing (100 words)

3.3.1 Enterprise Wireless Connectivity

How did you make a Database accessible by both your Prototype and Mobile Application?

For every IoT project, it's crucial to have a wireless connectivity – a connection between the hardware, software and databases so that user can control the hardware using application on PCs, mobiles or tablets. In our project – Smart Mirror, we want to control the peripherals such as weather, news, etc. shown on the mirror using SmartMirror Android Application. In order to implement, we need a connection between the Hardware and our Application – a link that both of them can access and deliver information. Therefore, we create the Serial Numbers in our Firebase to play a role as the link between those. Each mirror will contain its own Mirror Serial Number, and the Python Code for the Hardware makes all the peripherals' information deliver to that Serial Number section and that specific mirror will only accept peripherals' changes that come from its own Serial Number section on Firebase. And to make changes to the peripherals, our Application will ask users to fill in their Mirror's Serial Number. After having the Serial Number, the Application will enter that specific Serial Number section on Firebase and take information from last peripherals' changes to display on the screen as well as users can also adjust the peripherals' information such as turn on/off sensor or turn on/off speaker or even set auto timer for the mirror; then the information will be sent back to Firebase only in their Serial Number Section and the hardware peripherals of that specific Mirror will be changed according to the new information.

3.3.2 Database Configuration

Our Database consists of two main sections: Accounts and Mirror_Serial_Numbers. The “Accounts” section is where we keep our users’ signup information: email address and user name. The user name will be passed to the next activity as “Welcome *Username*” to inform users that they have successfully logged in. The users’ signup information (email address, time created, encrypted password) are also stored in Authentication section in Firebase so that we can authenticate users logging in to SmartMirror application in order to inform whether they are already a member or not. The second section of our database is Mirror_Serial_Numbers – a link between Hardware and the application. It contains Mirrors’ identified numbers so that users can control their own mirrors by giving the mirrors’ serial numbers. As they have provided the numbers, the peripherals’ information of their mirrors will be identified and shown in Setting section of the application. Therefore, users can change the information as they want and the information will immediately deliver to be stored in their serial number section of Mirror_Serial_Numbers on Firebase. As the result, the smart mirrors’ peripherals will change as their information had been changed by users. For now, we only list three Mirrors’ Serial Numbers on our Firebase and as we are doing only one mirror, only one serial number is linked to both the mirror and the application which is number “102030”. Other two serial numbers can also store information on Firebase as if users enter their identified Serial Numbers, but there are no connections between those two numbers to any hardware because there are no other smart mirrors.

3.3.3 Security

The security level implementation in our Smart Mirror project is regular, and not sophisticated because the smart Mirror doesn't contain any critical information such as the address of the user, never the less, all information is protected by some security measure because certain data can act as a gateway for discovering more data and personal information about the user. For example, the email of the user is a very important piece of information because it contains a confidential message about his accounts from various websites such as Amazon. Email is included in our database, the hacker can send a phishing email, it's going to direct the user to a fake website such as Facebook, Amazon, looks perfectly identical, When the user logs in to this fake site, the hacker receives his password, and from there, the hacker can get his credit card, home address, history, and other information. Therefore it's important to protect all data at any cost, and even if it's minor, any piece of data is like a treasure hunt to find more.

The two security measures that are implemented, the user needs to enter the correct login. The second is to enter 6 digits security serial number to synchronize with the mirror.

When the user creates an account in the android app by entering the username, email, and password, this information gets pushed into the database. All information can be viewed in the table, but the password section is hashed by the default algorithm in the Firebase, to provide some security to the customer even against the administer or the hacker. Moreover, we can change the hashing algorithm for encryption, however,

hashing is preferred because of its one-way function that scrambles user password and produces a unique random string of letters and numbers. It's very hard to reverse this function to get user password back, and the default hashing algorithm in Firebase it's good to prevent such reversal. Another security light security measure is when the user enters the wrong password, a warning message will be displayed.

The other layer of security is the serial number that authenticates with the mirror. This feature hasn't been implemented yet, due to the level of difficulty of integrating such thing to already existing complex node.js codebase of MagicMirror2. However, the idea is a smart mirror will have a unique 6 digit serial number. When the user enters the serial number, a notification will be sent to the mirror via the internet or data will push into the database to indicate " the authentication starting". The smart mirror will process this notification or data, and send back a message to the user "Login is complete ". From this point on, the user has the setting and control of the smart mirror input/output devices such as speaker and monitor.

In conclusion, the smart mirror prototype implements two security measure that is good enough for a prototype. If this project attracts investments and customers, further security measure has to be taken to protect all users of the smart mirror.

3.3.4 Testing

Unit testing is a very crucial part of the system development phase because it ensures components in system functionality, in case of our smart mirror prototype, this was, in fact, more important, and time-consuming than building the actual smart mirror

prototype. The reason being is the smart mirror runs several software multiple software components concurrently. The first one is MagicMirror2 software that is responsible to display, update, pull necessary data for all widgets such as a clock, bind everything together, and weather. It's writing in the node.js, Python and use Electron as a wrapper. The second components are mostly are service application. Google assistant service is responsible to authenticate with google and facilitate the communication between the user of the mirror and google via executing python script. Another service application is the speaker service that is responsible read in Volume data from the database and use to update the volume of raspberry-pi via the jack. Lastly, there is Monitor state service, similar, is also responsible to read the data from the database and determine whether to turn on or off the mirror based on the data. Another important note to mention is, why did we decide to run services currently with MagicMirror instead of integrating everything into one software? the simple answer is because we founded very difficult to understand the large node.js code base and we didn't have any prior experience with node.js, especially node.js syntax and structure. Anyhow, the unit testing was important in our case because it taught and helped us along the way. Unite testing allowed us to ensure that all software components are producing their correct output, each unites test taught us a new lesson that allowed us to fix new errors from other software components. More importantly, taught how to be a better debugger and narrow down the problem quickly.

Unit testing for the MagicMirror2 easy because MagicMirror software had various "detection and checking modules " such as MM-PM2, MM-error, and MM-out. For

example, the MM-PM2 module contains more than 30 unite test script to check various defects and expected results. Also, they log any errors to console as users start the application, for example, each module like weather or clock, if the modules are working correctly, they would display a log such as "clock is activated and working..". Otherwise, they would display detailed logs about the issue. Also, all of these logs get stored in a text file for future reference and they come from the server-side of the application. This helped in resolving issues when install, or updating modules, in the case of client-side of the application, we used chrome developer tools test to the behavior of widgets. However, we didn't have much problem with the client-side because most of these widgets were built by an experienced developer.

Before getting into unit testing for other software components. This unit test was a hybrid between a unit test and a "developer test", that is firing up everything, setting some breakpoints and getting the correct output for each portion. Although we acknowledge this mistake, we understand how important is unit testing is especially for big and complex software systems, but we didn't have enough experience to write a professional unit and tried to write unite test.

Unit testing for google assistant service consisted of multiple tests. The first test was testing google's assistant by itself. I pretty much wrote a python script that will try to activate google assistant using parameters such as sampling rate and audio file, so I can catch any early defect. The second test was to ensure that the service knew the location of the google assistant script.

Unit tests for speaker service also consists of two smaller tests. The first one to ensure that it can read data from the database without a problem. In the second one for service, I checked for various defects related to volume performance with different volume levels.

The monitor service unit test consists of a test for python script and service executing and running Linux command to turn off the monitor screen. In the case of a python script, I wrote tests to ensure it can read from data. In the case of service, I wrote a script to ensure service is enabled when raspberry-pi is booted up and it can execute the "vcgencmd display_power " command to turn off and on the monitor via HDMI.

4.0 Results and Discussions

The prototype our group developed for a smart mirror project it's not perfect, but it contains %70 of all the features we initially conceptualize at the beginning of this project. The reason being that we had a range of issues from "what is achievable" to project management skills to not always being consistent, but more importantly, we learned a whole lot of skills from each aspect of the project.

Without a doubt, we learned and improved our techniques skills from software programming to hardware troubleshooting. For example, we learned how to work with 3rd party software such as Google assistant and work with open-source software from GitHub. In more detail, we learned how to configure and install the necessary software packages to enable google assistants and MagicMirror2. During the configuration process, we faced tremendous issues related to incompatible packages, software error, incompatible hardware, most of these issues were resolved by consulting with documentation of google assistant and MagicMirror2. Reading the documentation and configuring 3rd software packages, whether it's open-source or proprietary software is very useful skills for software engineers. Another technical skill we learned how to come up with alternative software solutions based on our current skill level because some solutions we conceptualize we couldn't skill-fully implement due lack of experience with languages and understanding code-base of open-source software. For example, we wanted to add a feature to Magic-mirror software such as creating alarm widgets to alert the user, however, the task was too very difficult because we had to first understand the Magic-mirror complex Node.js, learn Node.js syntax and functions, importantly finish the

task before the deadline. These software difficulties pushed me to find an alternate solution, which is Linux service, which is an application that can be run back around to do something. This concurrent behaviour can be used with Magic-mirror software. In another word, we implemented each feature as a Linux service to run concurrently with MagicMirror2. This helped us to learn more about Linux services, Linux commands, and where else it can be used. In terms of the hardware and embedded system, we learned how to integrated various input and out sensors into our raspberry pi. In more detail, we learned each pin in raspberry-pi had a different purpose, how to read and write to pin. We learned how to test the PCB board for connection between the traces by using a multi-meter, better soldering techniques. In equal value to technical skills, we learned in the hardware and software part of the project, we learned a lot of software skills. For example, in the case of project management, we tried to follow an agile project management method because of its most common project method, and it makes sense than waterfall. We implemented scrum for each feature we wanted to prototype to have at a minimum, and we created a list of the backlog that contain all features. Each sprint was one week, and during that sprint, we picked the highest priority feature from the backlog and worked on it. When finished each feature, we review it and check if it meets all requirement or if want to change modify if we didn't like it. There are a lot more skills such as being consistent, debugging, writing emails, technical writing, and a lot more. Needless to say, the skills we learned will be in big use in our careers or even in other areas of life.

5.0 Conclusions

Creating a large number of smart mirrors for the rest of the Canadian population that can bring a lot of interesting challenges that would haven't risen if only we produced a dozen of them. When it comes to hardware there is not much of a challenge, it's just plug in a microphone, infrared sensor, and build a nice case that fits everything. The true challenge comes from the software side, install all necessary software, and creating necessary software infrastructure to connect all smart mirror. It was easy to install open-source and proprietary software in one raspberry-pi with all necessary support packages but it would be time-consuming to install all necessary software to thousands mirror. There are two solutions to this time-consuming problem. The first to create an agreement with owners of the raspberry pi to pre-install of this software just after manufacturing for sale percentage. The second solution its to create our hardware similar to raspberry pi minus all stuff that is not needed. Most likely we will continually update, improve the user experience, connect all users, help all user issues without coming to there door, only if we connect all our smart mirrors into a single cloud platform. This platform will gather data from all smart mirrors and enable our team to resolve the issue before the customer give us a call.

Report

/1 Hardware present?

/1 Checklist truthful

/1 Valid Comments

/1 Results and Discussion (500 words)

/1 Conclusion

6.0 References

OACETT. (2017, March). *I need to Complete a Technology Report*. Retrieved from The Ontario Association of Certified Engineering Technicians and Technologists: https://www.oacett.org/Membership/Technology-Report-and-Seminar

MakerKids. (2020, January 17). Empowering Kids to be Creators, Not Just Consumers. Retrieved January 20, 2020, from https://makerkids.com/

Ryan. (2018, May 10). Positive Effects & Benefits of Technology for Children: Kids' Development. Retrieved January 20, 2020, from https://www.idtech.com/blog/benefits-of-technology-for-children

7.0 Appendix

7.1 Firmware code

Demo

/1 Hardware present?

/3 Code runs concurrently for all sensors/effectors

/1 Project repository contains integrated code

Status

/1 Memo including updates

/1 Financial update

/1 Progress update

/1 Modified Code Files in Appendix

/1 Link to Complete Code in Repository

7.2 Application code

Demo

/1 Hardware present?

/1 Memo by student A

/1 Login activity

/1 Data visualization activity

/1 Action control activity

Report

/1 Login activity

/1 Data visualization activity

/1 Action control activity

/1 Modified Code Files in Appendix

/1 Link to Complete Code in Repository

```
package
smart.mirror;

                import android.content.ContentValues;
                import android.content.Context;
                import android.database.sqlite.SQLiteDatabase;
                import android.database.sqlite.SQLiteOpenHelper;

                import androidx.annotation.Nullable;

                import java.sql.SQLData;
                //Smart Device (Haider Ibrahim, Minh Nguyen , Trung Trinh)
                public class DataBaseHelper extends SQLiteOpenHelper {

                    public static final String DATABASENAME = "waitinglist.db";
                   public static final String TABLENAME = "waitingListStack";
                   public static final String USERNAME = "name";

                   public DataBaseHelper(@Nullable Context context) {
                      super(context, DATABASENAME, null, 1);

                   }

                   @Override
                   public void onCreate(SQLiteDatabase db) {

                      String action = "CREATE TABLE ";

                     db.execSQL(action + TABLENAME + " ("+ USERNAME + "TEXT PRIMARY KEY"+ ")");
```

```
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

        String action = " EXIST ";
     db.execSQL("DROP TABLE IF"+ action  + TABLENAME);
    }

    public boolean insertData(String name)
    {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentvalues = new ContentValues();
        contentvalues.put(USERNAME,name);

        long result =db.insert(TABLENAME,null,contentvalues);

        if(result==-1){
            return false;

        }else{
            return true;
        }
    }
}
```