Final Presentation

Smart Mirror

ECE 4011/4012 Senior Design Project

Team Name: Myr R
Project Faculty Advisor: X. Ma

Team Members Names, Majors, and Email Addresses

Jessey Sperrey, EE, jsperrey3@gatech.edu
Brennon Farmer, CmpE, bfarmer31@gatech.edu
Daniel Yue, CmpE, dyue9@gatech.edu

Submitted:

Submission Date: 04/30/2020

Table of Contents

1. Introduction

The bathroom mirror is a staple in any public or private bathroom in the world. For an everyday object to be used so frequently, how could it become more useful to the everyday user? The Smart Mirror is an upgrade to the everyday mirror to allow for even more functionality with a focus on entertainment and convenience. The mirror will feature two-way dielectric glass to allow for both common self viewing and simultaneous viewing of a display screen. The screen will allow the user to display his or her day planner/calendar and current weather forecast. An alarm clock and google map application will also be available for the user with the option for user input. Along with the display, the user will be able to play music through the speakers on the mirror using a smartphone. The smartphone will be able to connect to the Raspberry Pi using bluetooth connectivity. The Raspberry Pi is the device that will be the central processor for all of the aforementioned functions along with speakers and LCD display. For a successful final product, the product would be expected to seamlessly output desired information based on user input. If an event is input into Google Calendar, it will seamlessly display onto the LCD screen. The on-screen alarm clock will require the user to input time and it will sound when that time is passed, and user input is also required for the Google Maps application as the LCD will in turn calculate distance and output time of travel based on the destination given. The LCD will also display weather and current time on the main screen. All of the components would ideally be placed within the outer (wooden) casing that holds the dielectric glass with the LCD screen being directly behind that. Overall, the Smart mirror would provide great convenience with ease of use to the user while still maintaining the functionality of an every-day mirror.

2. Project Description and Goals

The original plan for the Smart Mirror Design was to create a convenient and entertaining device that adds onto the uses of an every-day “bathroom” mirror. Using a simple design for a wooden mirror housing, the dielectric, 30% transparent, glass would be on the front-facing side of the housing with the LCD screen placed inside the housing. The speakers would be on either side of the housing, and the components would be on both the under-side and the inside of the housing behind the mirror. The speakers would be connected straight into the Raspberry Pi just as with the LCD screen and the Raspberry Pi would provide all of the code for the applications. The final product and main goal of the product would be to show the following on the LCD screen: an alarm clock that receives a user input and outputs an alarm sound after the time has passed, a weather application that outputs the daily forecast of the current location, a google calendar that shows all upcoming events on ones google calendar as well as new events added on the user's smartphone, and finally a google map application that receives a user input for destination and outputs the time it would take to get to the destination from the user’s current location. The title of the song on the user’s smartphone will also be displayed on the LCD screen as well as audio coming from the speakers. This should be done using bluetooth connectivity to reduce the need for wiring.

Since the semester has progressed, as has the main goals of our project. Because of the Covid-19 outbreak, our design team has moved to finishing the project remotely, thus affecting the final design and goals of the Smart Mirror. The raspberry pi and LCD screen were designed and built separately from the housing and thus were not able to be a part of the final design for time and distance reasons. The final product that was created, i.e. the software that was built and loaded onto the Raspberry Pi, was designed in such a way that it can be “dropped” into a housing in the future. This allowed us to focus mainly on the design of the software and smart mirror interface. The final goals shifted to the team creating an interface that included all of the previously planned applications all located on an LCD screen in such a way that it would emulate the original smart mirror design. The speakers also would be connected to the Pi and music would be able to be played through them from a smartphone. In the end, while the original project design was changed, the smart mirror project was adapted to output a product that still accomplished most of the desired tasks.

3. Technical Specifications

While the plan for the software remained unchanged, the hardware had some minor revisions. The original spec for LCD screen and speakers had to be changed based on availability and cost of previously specified solutions. The Raspberry Pi has four usb ports as well as an hdmi port. It is powered using the micro usb port and also has a camera connector as well as sd card slot that are not in use. We did not change the original choice for the Raspberry Pi. The additions specifications are listed below:

Table 2. Raspberry Pi

Processor	Broadcom BCM2837B0, Cortex®-A53 (Arm®v8) 64-bit SoC @ 1.4 GHz

Communication	2.4 GHz and 5 GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
RAM	1 GB SDRAM
GPU	250MHz VideoCore IV
Memory Storage	10 GB
Input Voltage	5V/2.5A DC

The speakers used were changed to provide a seamless connection to the Raspberry Pi for both audio and power. The speakers are connected from a 3.5mm audio jack port to a usb converter and then into one of the four available usb ports on the Raspberry Pi. This allows for minimal wiring and easy connection. The following specifications on the speakers are below:

Dimensions	W 7.7 x D 5.5 x H 5.7 (cm)
Output Power	3W per speaker
Weight	0.26 KG total

The choice for LCD screen also had to be adjusted because of the remote design approach of our team. A smaller LCD screen was chosen because of price, but the resolution is greater on this choice as well and the capacitive touch capability. It was decided that having a touchscreen would allow for easier user input and reduce the need to use the smartphone for inputting user commands (besides adding calendar events and song selection). The LCD screen chosen still connects using the HDMI port located on the Raspberry Pi. Below are the specific specifications:

Dimensions	85 x 55 (mm)
Resolution	1024 x 600

4. Design Approach and Details

4.1 Software Implementation

Front-End and Back-End systems to retrieve Data

Some applications work by using a backend and front-end system. The back-end system consists of python code which uses various APIs to connect to servers and retrieve data (usually in json objects). The backend system parses the json object and extracts the data that is needed. Afterwards, the backend system sends the formatted data to the front end. The data is sent using the Tornado Python library's WebSocket interface[1]. The web socket interface in Python allows a python script to run as an application that sends data through a specified port that is set by the developer.

The front-end system is responsible for displaying the formatted data in an appropriate style for the end user. The front-end system receives the data through the web socket interface in JavaScript. The web socket listens on a specified port for the back-end system’s sent information. Specifically, the front-end system sends a message (request) to the back-end system and waits for the back-end system to process the request and send a response (parsed and formatted data) to the front end.

The front-end system displays the content to the end users by using HTML5, JavaScript and Cascaded Style Sheets (CSS). HTML5 allows the text, charts, maps, graphics, etc. to be seen by the end user. JavaScript is used to provide a coding language approach to instantiate the web sockets, bind button presses to a functionality such as sending a request to a back-end system, or creating a map application. CSS is responsible for formatting the data such as size of the font, the size of header titles, background color, etc.

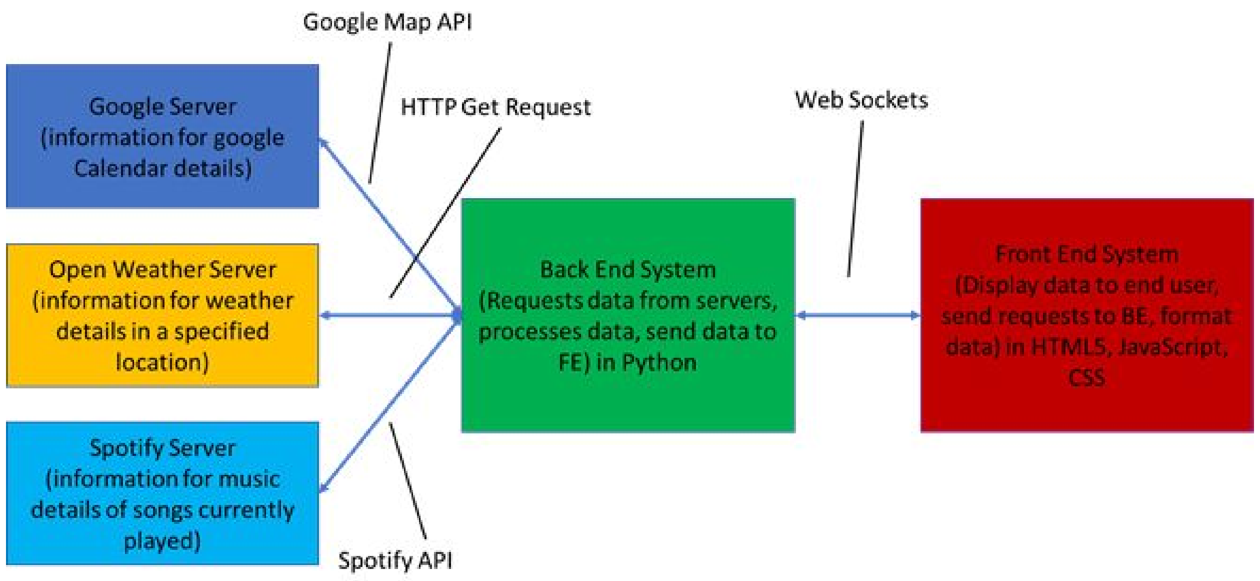

Google Map API
HTTP Get Request
Web Sockets
Google Server (information for google Calendar details)
Open Weather Server (information for weather details in a specified location)
Spotify Server (information for music details of songs currently played)
Back End System (Requests data from servers, processes data, send data to FE) in Python
Front End System (Display data to end user, send requests to BE, format data) in HTML5, JavaScript, CSS
Spotify API

Figure 1. A diagram that shows how data was requested and exchanged for the software portion of the project. The connections between modules represent the exchange of the data. Each connection is labeled to show what software tool was used to exchange information between the modules. FE and BE represent Front End and Back End, respectively.

Calendar Application

This application is integrated with the Google Calendar App to show the user's holidays and first ten events in the user's calendar from Google Calendar. The user can add events to their calendar through the Google Calendar Application. The changes will be reflected in the Smart Mirror's Calendar Application by waiting for it to auto-sync with the Calendar Application in four minutes or by pressing the "Refresh" button to cause instant syncing.

This application uses the front end and back end approach shown in Figure 1. A python script uses the Google Calendar API[2] to request calendar information for a specific user. The information is received as a json object by the back-end system. The information is parsed and processed. The front-end system requests for the information by sending a message through the web sockets to the back-end system. The back-end system receives the message and sends the calendar information to the front-end system.

Weather Application

This application shows the current hour's weather in the end user's location. The location is based on the IP address of the user. This application was created using the back end and front-end approach. The current hour's data is requested from the Open Weather server by using HTTP Get requests that used Open Weather's API [3]. The server responds by sending a JSON object. The back end parses the JSON object and formats the data. Next, the front-end system requests for the information by sending a message through the web sockets to the back-end system. The back-end system receives the message and sends the information about the weather to the front-end system.

Music Player Application

This application allows music to be played through Bluetooth on the Smart Mirror. The application is integrated with Spotify. When a device is connected to the Smart Mirror through Bluetooth and music is being played through Spotify, the song is displayed on the lcd screen.

The application works by using the front-end and back-end systems shown in Figure 1. The back-end system requests and receives data from Spotify servers using Spotify APIs and a python library (spotipy)[4]. The data is parsed and processed by the back-end system as well. The front-end system requests for the back-end's data by sending a message through a web socket that connects both systems. The back end receives the message and sends the data to the front-end to be shown to the end user.

Time Application

The Time Application displays the current time and date to the end user. The time application does not use the back-end system, only the front-end system is used. The application is developed with JavaScript and uses the "Date" object to get the current time based on the user's local time zone.

Alarm Clock Application

The alarm clock module serves the purpose of an everyday alarm clock, but with the convenience of being added to the smart mirror display. The alarm requires user input for selecting the date (MM/DD/YYYY) and the time, including AM or PM. After selecting the time and date for the alarm, the user can select the "Set Alarm" button to actually set the alarm. The user then has the option to "cancel alarm" as well as hit a "5 minute snooze" after the alarm sounds which works as expected. After the time has passed, a pre-selected alarm audio will sound and the "set alarm" button will change to "stop alarm". After hitting that button, the alarm stops and the user has the option to select another time for a future alarm. The code for the alarm clock was written in Javascript and HTML and is a basic use of function and style class calls in javascript to produce the proper display in HTML. The audio file for the alarm sound was a basic mp3 file from an online sound bank.

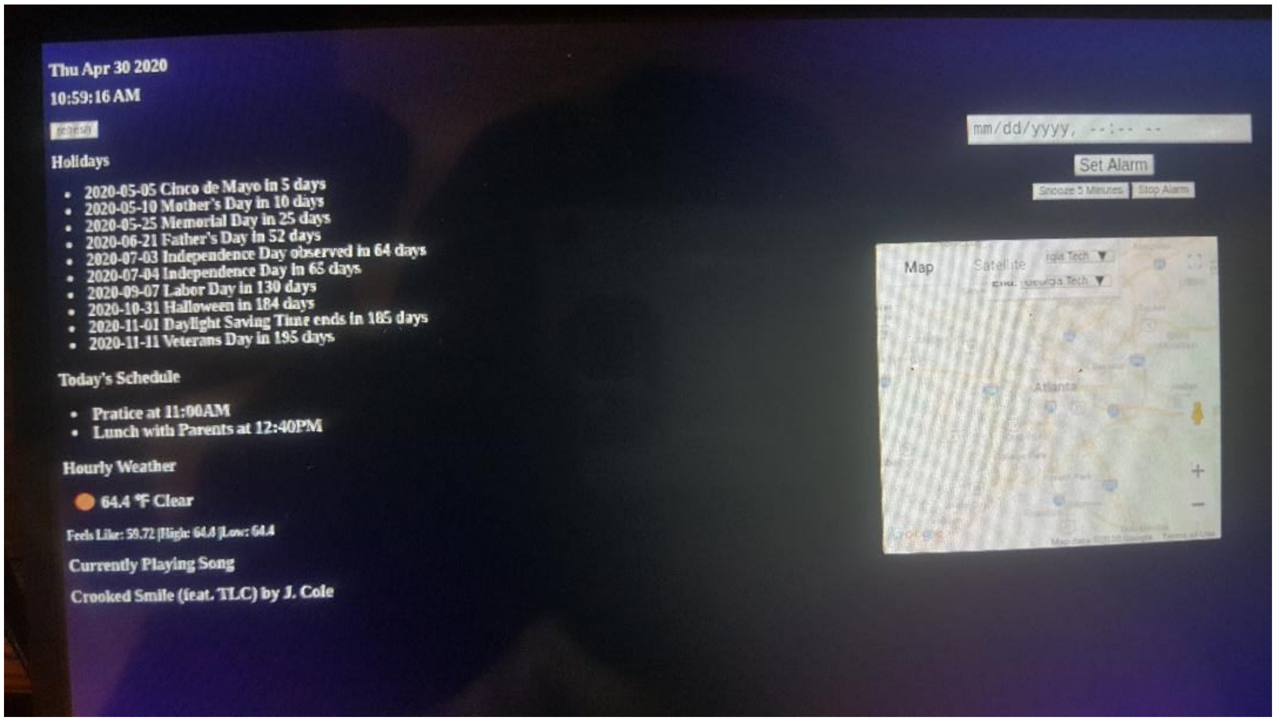
Google Maps Application

This application utilizes the Google Maps API to display a map of Atlanta and driving directions to destinations based on routing information from Google Maps. The user can add a start and end position on a floating panel in the application, and route details will be updated and shown on another panel to the left of the map. These updates are instant, and account for travel times in current circumstances. The routes calculated are always based on an optimal timing of traffic and weather conditions.

This application uses the front end and back end approach. A JavaScript script uses the Google Maps API[5] to request route information for a specific user. The information is received as a response object by the back-end system. The information is parsed and processed according to what is required, in this case, route duration and distance. When an End is chosen, a panel to the left of the map will be created, detailing route instructions. The front-end system displays the appropriate information requested.

Fully implemented software with all applications

The image below shows all the applications displaying on the LCD screen.



Thu Apr 30 2020

10:59:16 AM

Holidays

- 2020-05-05 Cinco de Mayo in 5 days
- 2020-05-10 Mother's Day in 10 days
- 2020-05-25 Memorial Day in 25 days
- 2020-06-21 Father's Day in 52 days
- 2020-07-03 Independence Day observed in 64 days
- 2020-07-04 Independence Day in 65 days
- 2020-09-07 Labor Day in 130 days
- 2020-10-31 Halloween in 184 days
- 2020-11-01 Daylight Saving Time ends in 185 days
- 2020-11-11 Veterans Day in 195 days

Today's Schedule

- Pratice at 11:00AM
- Lunch with Parents at 12:40PM

Hourly Weather

● 64.4 °F Clear

Feels Like: 59.72 |High: 64.4 |Low: 64.4

Currently Playing Song

Crooked Smile (feat. TLC) by J. Cole

Figure 2. The image below shows all the applications for the Smart Mirror on the LCD screen. On the left side of the screen it shows the Time Application, Calendar Application(Holidays), Calendar Application (Today's schedule), Weather Application, and Music Player Application from top to bottom of the screen. On the right side of the screen starting from top to bottom, the alarm clock application and the Google Map application are shown.

4.2 Hardware Implementation

The raspberry pi is connected to a LCD screen through the hdmi port. The LCD screen is also connected to one of the four USB ports for power. USB Speakers are connected to the one of the USB ports for power, and it is connected to the 3.5mm audio port. For debugging purposes, a USB mouse was connected to one of the USB ports. For debugging purposes, a keyboard was connected to one of the USB ports. A 5V adapter was connected to the micro usb port to power the raspberry pi. The image below shows the hardware implementation for the Smart Mirror. It includes components that were used for debugging.

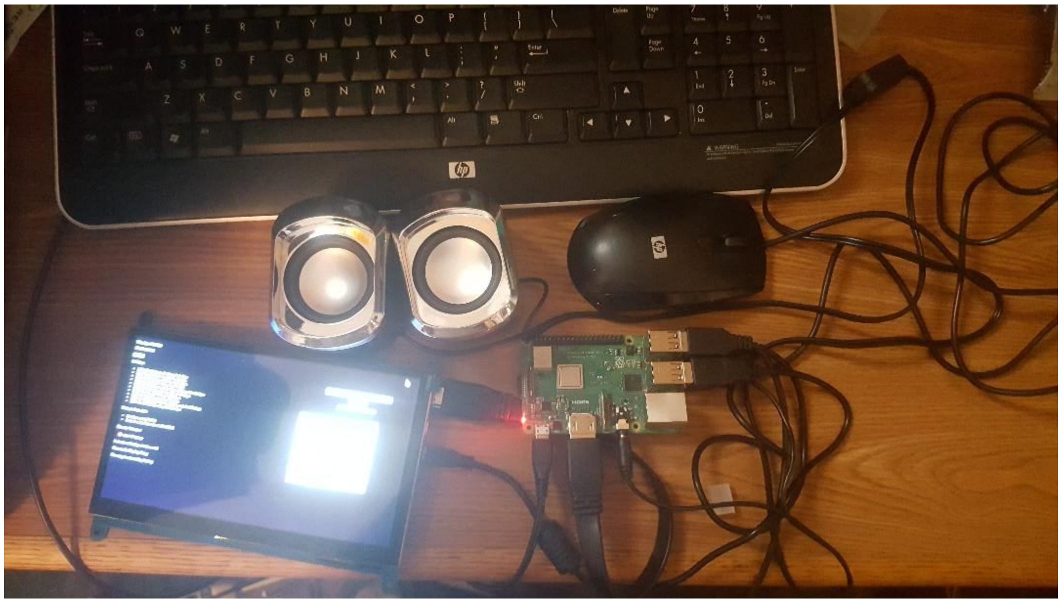


Figure 3. The image shows the wire hook up for the raspberry pi. Components used for debugging are also included.

Basic schematic of the hardware layout is shown below. The debugging components are excluded from the schematic

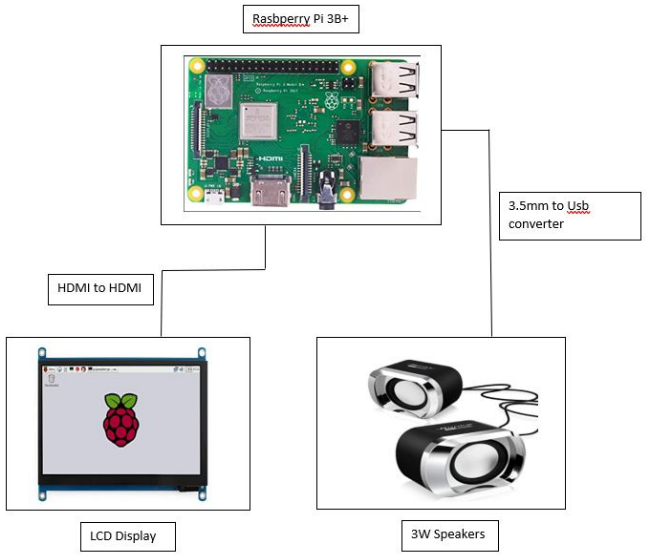


Figure 4. Basic schematic of the Smart Mirror (excluding the debugging components)

5. Project Demonstration

As detailed in the video, we have demonstrated key features of our prototype as shown in the demonstration video. The modules that are used all are able to function in the same window concurrently. Also detailed in the prototype is that the modules should run for a long time, and that was the case in the demonstration video. Multiple components were tested, such as the Google Calendar, Weather, Music Player, Alarm Clock, and Google Maps modules. They were all shown to be functional and fully displayed in their appropriate sections of the LCD screen. A demonstration video is embedded into the senior design website's html coding, for easy access and viewing for advisors, professors, and persons of interest. The demonstration, located at the bottom of the website's page, includes all of the aforementioned components working as intended, with visual demonstrations of both hardware and software present in the device.

Modules were demonstrated for quality and responsiveness, in both the cases of the LCD and speakers. The LCD was shown to be bright enough for viewing from at least 3 feet away, and the speed of software with use in real-time situations was determined adequate for their appropriate situations. Examples of this include displaying accurate calendar and weather information, real-time playing of music from Spotify, and real-time management of the alarm clock and google maps routing information. The sound quality of speakers were decent since the sensitivity level ranged from 70dB - 88dB while playing music. This is shown by using two applications that measure sound(Sound Meter by Tools Dev and Sound Meter by Splend Apps)[6,7] in Figure 4. Each component of every module was demonstrated to be especially accessible to users of either a computer mouse or the LCD's own touch screen, as shown by the demonstration of the alarm clock application and the Google Map application. As a system, the entire hardware package is compact and relatively simple and robust, which makes it especially able to be effectively contained within a black box or cabinet, to reduce exposure to bathroom elements.

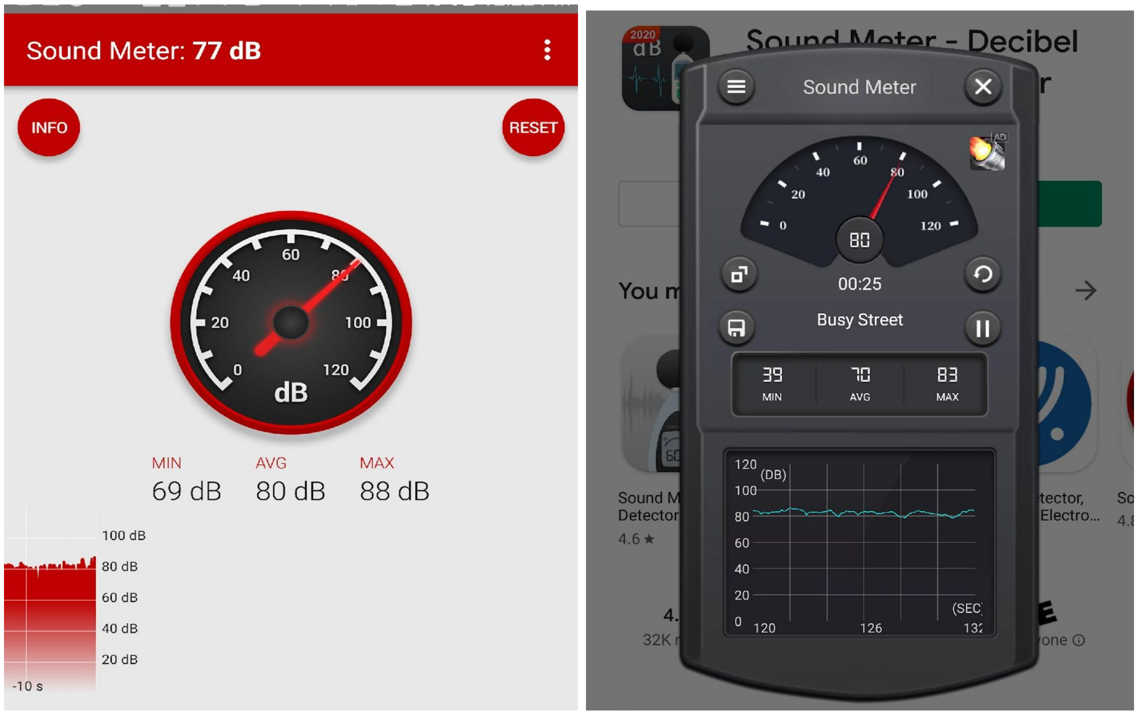


Figure 5. Image showing decibels reading for speakers during a playing song. The left image shows reading from Sound Meter by Splend Apps, the right image is from Sound Meter by Tool Devs.

Because of current events, the mirror was unable to be tested for use with the Raspberry Pi, but the actual mirror was demonstrated to be appropriately reflective and allowing for light from the LCD to pass through. The appropriate hardware was shown to be operational in time with the software as well, with no real issues with consistency in connection or power. A software demo was also included, depicting the use of the Python/JavaScript languages to meet our goals of development. The scripts running the backend of the project were also shown to appropriately render the modules on the LCD, and connected to the internet for the purposes of using Google APIs to generate the Calendar and Maps. The html used for displaying these modules on the LCD was effective in generating all the specified modules. All of the necessary modules are displayed concurrently, with none of the features overlapping with other features on the screen, and all components visibly accessible by either mouse or touch on the LCD.

6. Project Evaluation

Overall, this project was successful in meeting our desired goals, albeit without some components, such as the mirror, and without more advanced testing of our modules and display. The required materials are rather bulky and hard to position in a bathroom environment, although more time and resources would allow for the development of a black box. Otherwise, the vulnerable components (Raspberry Pi, ports) are exposed to bathroom environments. The LCD would be able to be mounted on a wall, displaying the desired module's information accurately and on time, as well as being easily visible.

The LCD allows the user to display their day planner and Google Calendar, as well as the current weather forecast. An alarm clock and Google Maps application will also be available for the user with the option for user input, whether to set an alarm or estimate the route duration to a destination, respectively. Shown with the display, the user is able to play music through the speaker systems on the mirror using a smartphone. The smartphone is able to connect to the Raspberry Pi using bluetooth connectivity, and plays music using Spotify. In the operational, everyday use aspect, the device operates perfectly.

References

1. "tornado.websocket - Bidirectional communication to the browser¶," tornado.websocket - Bidirectional communication to the browser - Tornado 6.0.4 documentation. [Online]. Available: https://www.tornadoweb.org/en/stable/websocket.html. [Accessed: 29-Apr-2020].
2. "Calendars and Events | Calendar API | Google Developers," *Google*. [Online]. Available: https://developers.google.com/calendar/concepts/events-calendars. [Accessed: 29-Apr-2020].
3. OpenWeatherMap.org, "Weather API," *openweather*. [Online]. Available: https://openweathermap.org/api. [Accessed: 29-Apr-2020].
4. "Welcome to Spotipy!¶," *Welcome to Spotipy! - spotipy 2.0 documentation*. [Online]. Available: https://spotipy.readthedocs.io/en/2.12.0/. [Accessed: 29-Apr-2020].
5. "Maps JavaScript API," *Google*. [Online]. Available: https://developers.google.com/maps/documentation/javascript/tutorial. [Accessed: 29-Apr-2020].

6. “ToolsDev,” *ToolsDev*. [Online]. Available: http://toolsdev.applinzi.com/. [Accessed: 29-Apr-2020].
7. “Welcome to Splend Apps!,” *Welcome to Splend Apps!* [Online]. Available: http://splendapps.com/. [Accessed: 29-Apr-2020].