Smart Music Interface and Display (S.M.I.D.)

Major Design Project Proposal

Illinois Institute of Technology

ECE 441

PREPARED BY

Cameron Haley - *B.S. Computer Engineering*

Anthony Banuelos - *B.S. Computer and Cybersecurity Engineering*

Colin Prochnow - *B.S. Computer and Cybersecurity Engineering*

Kamil Kaczmarczyk - *B.S. Computer Engineering*

EXECUTIVE SUMMARY

A part of our everyday lives as students is music. It is something that helps us get through the day while we are heading to work, working out at the gym, or even grinding out homework problems. Our idea is to come up with a smart speaker system that allows you to display your current music playing along with other useful stats such as temperature, humidity, etc. This will be displayed on a mirror with a display that shows through a glass covered with two-way film which will give you a two-way mirror. It will allow the display to portray onto the mirror so that you don't have just another screen lying around your room, it can blend in looking like a mirror. Along with that, there will be voice recognition to help control your music using the Alexa API and while your music is playing there will even be LEDs synchronized with the beats of the song hanging around the frame of your mirror. We want to make something that someone can use every day and style up in your room. The smart display will also include a Bluetooth function and all of this will be powered on a Raspberry Pi connected to a display.

1. Project Overview

This project intends to design a Smart Music Display that allows a user to connect wirelessly via Bluetooth to stream music. The device will provide a clean UI that displays all relevant information, including the typical “Now Playing” screen, including things such as the title, artist, and album art. Additionally, data from temperature, humidity, and pressure sensors will be displayed. There will also be a strip of LEDs surrounding the enclosure that will light up in sync to the music, providing a more personalized listening experience. Finally, the device will have Alexa built-in so that it may recognize voice commands, control music playback hands-free, and control the various peripherals.

2. Materials & Budget

Name / Link	Description / Specifications	Price	Quantity	Amount
Audio				**$81.94**
Dayton Audio RS100-8 Full-range Driver	30W RMS Full-range speaker (can handle the full frequency spectrum). 8Ω impedance to match the amplifier. One for each channel. 4” so it fits in the enclosure.	$29.98	2	**$59.96**
TDA7492 Digital Audio Amplifier	Max 32W per channel @1% THD with 25V power supply. Our power supply is 48W (24V@2A), so this will be closer	$17.98	1	**$17.98**
Port Tube	TBD	$4	1	**$4**
Poly-fil	Sound dampening. Will buy only if deemed necessary.	$8.99	1	**$8.99**
Smart Mirror				**$99.07**
Display (Monitor)	A monitor a member happened to have. Price listed is for comparable MSRP.	$79.99	1	**$79.99**
Glass panel	Sits in front of the monitor. Two way film is applied to it to turn it into a two-way mirror. This is cheaper than buying a two-way mirror.	$8.08	1	**$8.09**
Two-way Film	Only allows light in one direction. Light from the monitor will be allowed out, but light from the room will be reflected, creating a mirror-like effect.	$10.99	1	**$10.99**
Electronics				**$85.45**
Raspberry Pi 4B	Controls and interfaces with the entire system. Bluetooth 5.0 for streaming music to the device.	$29.99	1	**$29.99**
SD Card	Data storage for Raspberry Pi	$4	1	**$4**
WS2812B Addressable RGB LED Strip	16.4 ft, so we can create two loops if necessary. Each LED is individually addressable, so “animation” effects can be made.	$18.69	1	**$18.69**

BME280 Digital Temperature/Humidity/Pressure Sensor	Supports I2C and SPI protocols. Has fairly good accuracy that will work perfectly for our application.	$8.99	1	**$8.99**
24V 3A DC Power Supply	Used to power the whole system. 24V because the amplifier supports up to 26V.	$12.99	1	**$12.99**
DC Step Down Converter 24V -> 5V	Used to step down the 24V power supply to 5V (3A) for the Raspberry Pi	$9.99	1	**$9.99**
3.5MM Male to Screw Terminal	Need to put on Amplifier to plug into Raspberry Pi's 3.5mm jack	$1.30	1	**$1.30**
TOTAL COST				**$266.96**
COST TO US				**$186.97**

3. Hardware

The hardware will be connected as follows:

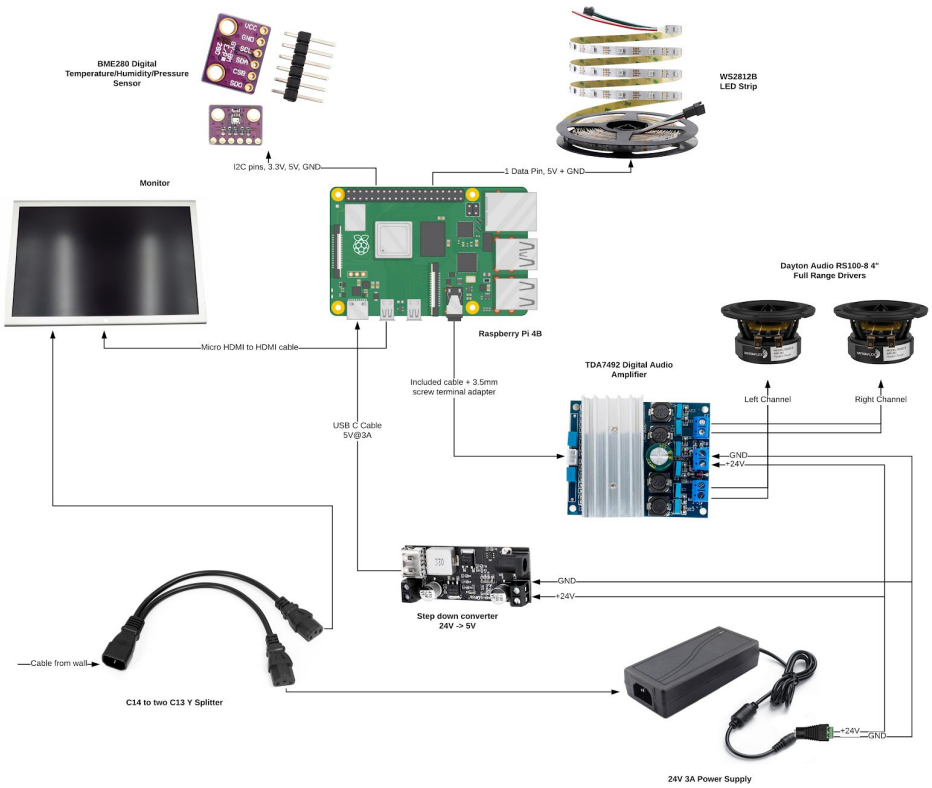

Figure 1: Components of the Smart Music Display

Power/Electronics

Colin Prochnow and Kamil Kaczmarczyk

The main driver of the system will be a Raspberry Pi 4B. This device will consume 5V@3A, which means it will require 15W of power. Additionally, it will provide GPIO pins for interfacing with peripherals, Bluetooth audio for allowing devices to connect, and analog audio out for supplying audio to the amplifier.

According to the documentation, the TDA7492 amplifier is rated at 32W RMS per channel for 1% Total Harmonic Distortion (THD). As previously mentioned, the Raspberry Pi will require an additional 15W of power. Therefore, at most this project will require a power supply of 79W.

Because of this, a 72W power supply was chosen (24V@3A). The amplifier supports an input of up to 26V, so this can be connected directly to the amplifier. However, the Raspberry Pi 4B does not support such a high voltage, so a step-down voltage regulator was chosen that is capable of converting the 24V input into a 5V output with at least 3A of current.

The monitor may be powered by using a Y-cable adapter, since both the monitor and power supply take a C13 cable.

Audio Equipment

Cameron Haley

The speakers chosen are Dayton Audio RS100-8 full-range speakers. These were chosen due to their capability of producing the full audio spectrum, which negates the need for multiple speakers and a crossover circuit, which would be difficult to fit due to size constraints. They also were generally well-reviewed in terms of their audio and build quality.

Based on the size of the monitor used, we think that a reasonable size for the audio enclosure would be 8”x4”x12”. This would give a volume of 0.22 ft^3, which is perfect for our speakers. The speakers’ specifications [1] recommend a 0.1 ft^3 (vented) volume per speaker, which is part of the reason the speakers were chosen.

Frame/Glass

Anthony Banuelos, Kamil Kaczmarczyk, and Colin Prochnow

In order to construct the frame, we can make use of ½ inch plywood. We will use a CNC machine to cut out the wood to form the cabinet to house the audio equipment along with cutting a notch in which the glass will sit. A two-way film will be applied to the glass to achieve a two-way mirror at a cheaper cost of buying a two-way mirror pre-constructed.

4. Software

User Interface

Cameron Haley

In order to build a user interface, Electron [2] will be used. Electron essentially allows a GUI to be constructed with familiar web technologies, such as HTML, CSS, and JavaScript, by using Node.js and the Chromium engine. This was chosen due to familiarity with building websites that should translate over to a quicker, more efficient design process.

On top of Electron, the following technologies will likely be used:

- **TypeScript** - Static typing to lessen errors and provide type safety
- **Svelte (or perhaps React/Vue)** - A lightweight framework to help manage data, state, and reactivity
- **TailwindCSS** - Provides utility classes to easily and consistently style with CSS

However, because most members of the team are not familiar with Node.js, the backend Node.js server will spawn and communicate with a Python process, most likely using python-shell [3], that will handle the bulk of our operations (such as interfacing with the LEDs, fetching Bluetooth information, etc.).

The Electron server will simply receive messages from this Python process (“The temperature is now 70 degrees”) and then update the web application as necessary.

Bluetooth Audio

Anthony Banuelos and Colin Prochnow

We want the user to be able to connect to the mirror via Bluetooth on their phone or other devices and from there, be able to stream music to the speakers. We also want the music info to appear on the display, including accurate album art, the artist, the track, and what time they are in the song. Now we have a couple different of options for our Bluetooth library:

- BlueAlsa [4]
- ALSA
- PulseAudio [5]

The most up-to-date library appeared to be BlueAlsa which is what we will first attempt to implement. In order to do this we have to go within the Bluetooth library to see how we can properly do so. The first thing we are going to want to do is configure Bluetooth within our pi and have it discoverable as an A2DP sink. This way any user on their phone may be able to see this and connect. Now this does have its pros and cons. Clearly anyone is able to connect if they are close enough, although it is useful if someone new wants to connect and play their own music. After doing this we must run BlueAlsa on reboot.

For security purposes, ee may add a fixed pin which allows a password in order to pair to the display and in this it gets past the event where someone random may walk past a dorm room and connect, for example.

Although now we have the difficult part of trying to access the songs metadata. For this we will have to use PyBluez [6] which can help us get the additional info such as album title, song name, etc.

Voice Commands

Cameron Haley

Due to the mirror being hung on a wall, it will be inaccessible and inconvenient for a user to use any kind of button controls. It is also impractical to have buttons on the display itself, as it is a mirror and would collect fingerprints very easily. For these reasons, we would like to implement voice commands to not only control the music, but also to control the LEDs, report the readings from the sensors, and any other useful features that may reveal themselves during the build and testing phases.

In order to have a reliable, extensible framework to build these commands on, we decided to use Amazon's Alexa Voice Services (AVS), which allows us to have Alexa built-in to the device. This should be easy to integrate with on the Raspberry Pi, as they have guides on their site [7] that steps through the process.

By default, this should allow voice commands for controlling music, although this will have to be tested. However, for controlling other things, a bit more work is involved. Our research revealed a helpful YouTube video [8] that explains how AVS provides a functionality called Smart Home for AVS [9] that allows for controlling connected devices (such as the LEDs in our case) without the need for creating custom Skills or hosting a web service. We could use this to control things such as:

- Turn on/off LED lights (Alexa.PowerController)
- Change polling interval for sensors (Alexa.RangeController)
- Enable/disable audio-only mode (Alexa.ToggleController)
- Set color for LED treble frequency (Alexa.ModeController)

This is most easily accomplished by using the C++ AVS SDK [10] and handling the response appropriately. We will most likely simply invoke Python scripts that will carry out the response, instead of duplicating peripheral control logic in C++ again.

Controlling Peripherals

Colin Prochnow and Kamil Kaczmarczyk

LEDs

For our design we selected to use LED strip model number WS2812B which is individually addressable. What "individually addressable" means is that each of the individual LEDs brightness and color can be controlled. This can be contrasted to the commonly purchased LED strips that have become popular as of late where the entire strip of LEDs can only be one brightness level and one color. This offers us greater control over visual effects, such as changing the LEDs to the beat of the music. Two parts that do not come with the WS2812B LED strip and are found in commonly available single color LED strips are a controller and a power supply. This is not a problem as we can use the Raspberry Pi, which is central to our design, to act as the controller for the LED strip.

As shown by the schematic below, we will only need 1 DATA line that can be supplied by a GPIO port from the Raspberry Pi, along with a GND and 5V connection which are also available on the Raspberry Pi in the form of GPIO pins.

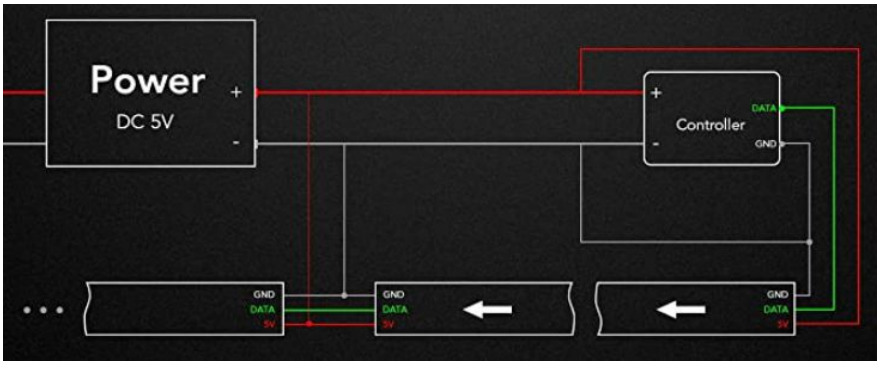

Figure 2: Schematic of the circuit for the WS2812B [11]

Using the image below as reference we could, for example, make use of pins 4, 6, and 8 in order to control the WS2812B strip.

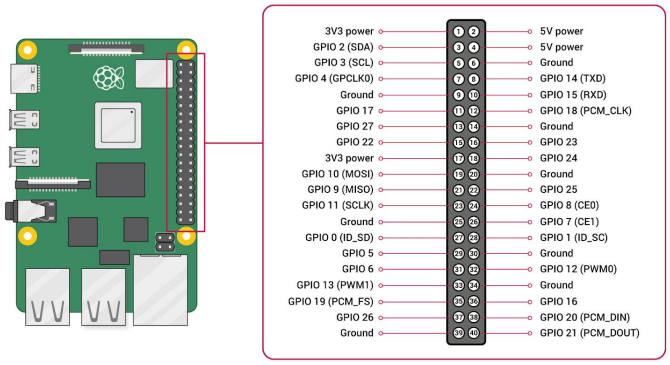

Figure 3: Layout of the GPIO Pins for the Raspberry Pi 4B [12]

In order to actually sync the music to the LEDs, we will have to intercept the Bluetooth audio in Python using the PyAudio [13] library, perform DSP using FFT from numpy [14], and then convert this processed data into some kind of visualization, which we still need to research more. We found a great library [15] that does 1-dimensional visualizations in a very similar manner that we will use as a reference when building our LED system.

Temperature/Humidity/Pressure Sensor

We will be using the BME280 atmospheric pressure, temperature and humidity sensor with either the Raspberry Pi. Another problem is the sensor being too close to the Raspberry Pi and reading the heat coming off it giving us a wrong temperature. This can be solved by extending it on 5 male-to-female jumpers and placing the sensor at a more optimal place like at the top of our design. The schematic of the sensor is the following.

All the necessary ports are supported by the Raspberry Pi. With this we will be using a Python Library called Adafruit_CircuitPython_BME280. [16]

Additionally, we can add a function to the program to record all data to a spreadsheet and we can possibly run algorithms and use trends to either predict the weather or just simply display prior days. We can easily access this information remotely from a SSH client on our desktop by enabling SSH on our Raspberry Pi and running the python file remotely.

We could also use the ESP32 chip. [17] This allows us to add an additional sensor outside to measure outside temperature. Of course doing that would need a sustainable power supply, potentially small 5v solar panels.

5. Technical Challenges

In this section, we will explore the three biggest challenges we expect to face in implementing all the aforementioned components of our project. Additionally, we will provide a brief explanation of how we plan to overcome these challenges.

Bluetooth Connection

Within this section, it is going to be very difficult in order to pull the track information properly as it is uncharted territory for most of us and we will have to learn how to interface with Bluetooth. We've looked into various libraries that should allow us to extract the information we need, such as PyBluez [18], which should allow us to interface with the connected Bluetooth device. A lot more research is definitely needed here, as this is one of the most crucial aspects of our project.

Additionally, having this Bluetooth connection and Alexa commands integrating together well is another struggle we will encounter, as we have to ensure that everything is able to operate together well.

Voice Commands

In using Alexa Voice Service [19] (AVS), we anticipate to encounter many challenging obstacles. From setting up the basic service to setting up integrations to control the LEDs, sensors, and display, there are many components that need to work together properly to provide a seamless, truly smart experience.

In our research thus far, we have a general understanding of how AVS works and what parts of it we need to make use of in order to get the functionality we desire. For example, in order to control the peripheral devices, we can make use of Smart Home for AVS [20] to not only allow voice control of these devices, but also to allow changing the values in the Alexa app on any smart device. This library works by sending a custom directive to the device, which we then implement handlers for in C++ using the AVS SDK.

Of course, the documentation is very expansive and we won't truly know the little details and hurdles we'll encounter until we start integrating with the API, but our research has shown that this problem is surmountable and is feasible for our project.

LED Synchronization

One of the biggest challenges will be intercepting the Bluetooth audio on the Raspberry Pi and converting it in real time to some kind of nice, aesthetically pleasing animation. Extensive research has already been done, and as previously mentioned, an incredibly helpful library [21] was found that will be used as a reference.

However, even with this excellent reference, there's still a few pieces we will have to figure out. First, it's very likely that Bluetooth and the Alexa Voice Service may complicate things and may make it harder to "intercept" the audio with PyAudio for processing in real time. Next, this library performs single dimension animations, but we want to perform two-dimensional animations (around the enclosure).

Additionally, for this synchronization to be effective, it has to be low-latency - out of sync lights won't be effective. This limits the amount of processing we can do, so some trade-offs will have to be made in order to get a desirable effect.

For these reasons, we anticipate this component to be a major challenge and will continue to do research to get the best result.

6. Milestones and Reporting

Milestone	Tasks	Assigned To	Date
1 - Development			
1.1	Order all parts	Everyone	20 Feb
1.2	Test out all the parts, ensure they work	Everyone	27 Feb
1.3	Circuit designed	Cameron Haley Colin Prochnow	2 Mar
1.4	Basic LED control	Colin Prochnow	6 Mar
1.5	Design enclosure	Anthony Banuelos Kamil Kaczmarczyk Colin Prochnow	16 Mar
1.6	Bluetooth connectivity and information working	Anthony Banuelos Colin Prochnow	13 Mar
1.7	Speakers and Amplifier working and playing audio from the RPi	Anthony Banuelos Cameron Haley Colin Prochnow	20 Mar
1.8	Enclosure assembled	Everyone	27 Mar
1.9	Basic AVS integration	Cameron Haley	3 Apr
1.10	Basic GUI built	Cameron Haley	10 Apr
1.11	**Integration Test**	**Everyone**	**17 Apr**

7. Conclusion

The Smart Music Display is an entertainment device that outlines information to the user. It will use a user interface to display all information regarding the music playing such as title, artist, and album art. The user will also be able to see the temperature, humidity, and atmospheric pressure which will be gathered through a sensor attached to a Raspberry Pi. The design will also include LED lights that will light up in sync with the music giving the user a better, customized experience. Additionally, the user will be able to use voice

recognition to control multiple peripherals and say certain commands to control the music. Finally, it is important to note that we have challenges which were previously addressed in the report. These challenges consist within the bluetooth connection, voice commands, and LED synchronization. It is extremely important that these challenges are highlighted early within the project so that throughout the creation of the product we are able to focus heavily appropriately approaching these challenges.

8. References

1. https://www.parts-express.com/Dayton-Audio-RS100-8-4-Reference-Full-Range-Driver-295-352
2. https://www.electronjs.org/
3. https://www.npmjs.com/package/python-shell
4. https://github.com/Arkq/bluez-alsa
5. https://www.raspberrypi.org/forums/viewtopic.php?t=235519
6. https://github.com/pybluez/pybluez
7. https://developer.amazon.com/en-US/docs/alexa/avs-device-sdk/raspberry-pi.html
8. https://www.youtube.com/watch?v=WkgoMm9l0ec
9. https://developer.amazon.com/en-US/docs/alexa/alexa-voice-service/smart-home-for-avs.html
10. https://developer.amazon.com/en-US/alexa/devices/alexa-built-in/development-resources/sdk
11. https://www.amazon.com/dp/B0882KJCG7
12. https://www.raspberrypi.org/documentation/usage/gpio/
13. http://people.csail.mit.edu/hubert/pyaudio/
14. https://numpy.org/
15. https://github.com/scottlawsonbc/audio-reactive-led-strip
16. https://github.com/adafruit/Adafruit_CircuitPython_BME280
17. https://www.amazon.com/dp/B07Q576VWZ
18. https://github.com/pybluez/pybluez
19. https://developer.amazon.com/en-US/alexa/devices/alexa-built-in
20. https://developer.amazon.com/en-US/docs/alexa/alexa-voice-service/smart-home-for-avs.html
21. https://github.com/scottlawsonbc/audio-reactive-led-strip