Phone Audio FM Transmitter

ECE 445 Final Report

Team 18
James Wozniak (jamesaw)
Madigan Carroll (mac18)
Dan Piper (depiper2)

TA:
Abhisheka Mathur Sekar

Senior Design Spring 2023
3 May 2023

ABSTRACT

Our project is a device that receives audio from a user's device over a Bluetooth connection and transmits that audio with a frequency modulation (FM) transmitter. It automatically scans the FM spectrum to find the channel with the lowest power, for the best audio quality, and sets the transmitting frequency accordingly upon user input. The device works as intended, and additional features were added on top of the basic requirements. We encountered problems with noise in the device, but we solved this by replacing the switching voltage regulator with a linear regulator. Further work is needed to reduce the size of the device and take advantage of all the functionality the microcontroller offers.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 PROBLEM

In cars with older stereo systems, there are no easy ways to play music from your phone as the car lacks Bluetooth or other audio connections. There exist small FM radio transmitters that circumvent this problem by connecting to a user's phone via Bluetooth and broadcasting the phone audio on some given FM wavelength. A significant safety flaw with these devices arises when the user enters an area in which a radio station is transmitting on the same frequency they are using. While operating the vehicle, the user must manually scan through the channels, identify an open frequency, and set the transmitter to that frequency, taking their attention off driving.

1.2 SOLUTION

Our solution builds upon these existing devices, adding the functionality of automatically searching for an open channel and switching the transmitting frequency. This creates a safer and more enjoyable user experience. This objective requires several components: a Bluetooth connection to send audio signals from the phone to the device, an FM receiver and processing unit to find the best wavelength to transmit on, and an FM transmitter to send the audio signals to be received by the car stereo. This took the basic form shown in Fig. 1.

1.3 HIGH LEVEL REQUIREMENTS

1. Connects and receives audio data via Bluetooth.
2. Transmits audio via FM.
3. Scans FM stations and automatically selects one with minimal interference.

1.4 SUBSYSTEM OVERVIEW

Our design is broken up into six total subsystems, shown in Fig. 1. The power subsystem interfaces with the auxiliary power port of the vehicle and steps the 12 V input down to 3.3 V for use in the rest of the subsystems.

The Bluetooth subsystem is completely encapsulated by the control subsystem, as it is included as a part of the integrated circuit (IC).

The control subsystem manages the communication buses used, inter-IC (I^2C), inter-IC sound (I^2S), and general-purpose inputs and outputs (GPIO), and performs all the computational tasks required of the system.

The user interface subsystem takes input and provides feedback. It consists of a display controlled via the I^2C bus and buttons connected directly to the GPIO of the microcontroller.

The receiver subsystem consists of an IC that handles FM reception and an antenna. The IC communicates with the control subsystem via I^2C. The receiver measures the strength of signals transmitted on FM stations and relays this information back to the control subsystem.

The transmitter subsystem consists of an IC, which handles FM modulation, and an antenna. The transmitter IC receives digital audio from the control subsystem via I^2S and receives and sends control data via I^2C. It outputs an FM signal that can be received by the user.

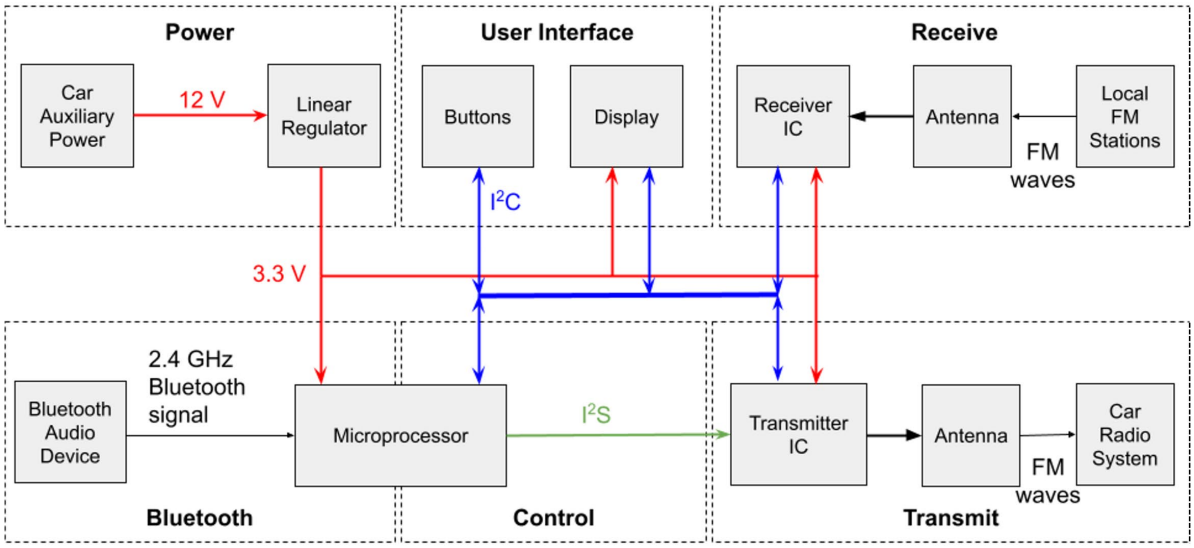


Figure 1: Block Diagram

2 DESIGN

2.1 PHYSICAL DESIGN

The physical design of our project is similar to current designs already on the market and other car accessories making use of the auxiliary power port of the vehicle. We elected to make the design three pieces, a front plate, a mounting bracket, and a back plate, all connected to one another by small screws. This simplified assembly of the device and provided ample room for maneuvering and routing the wires for the power connector and antennas. The front faceplate features holes to provide access to the display and pushbuttons and on the back plate to insert the end of an auxiliary power connector. The inside of the case includes small standoffs to give enough room for the components on the printed circuit board (PCB) and encourage airflow for the cooling of the voltage regulator. Figure 2 shows the computer aided design (CAD) diagram of the enclosure design. The front and back of the PCB are shown in Fig. 3 and Fig. 4 respectively.

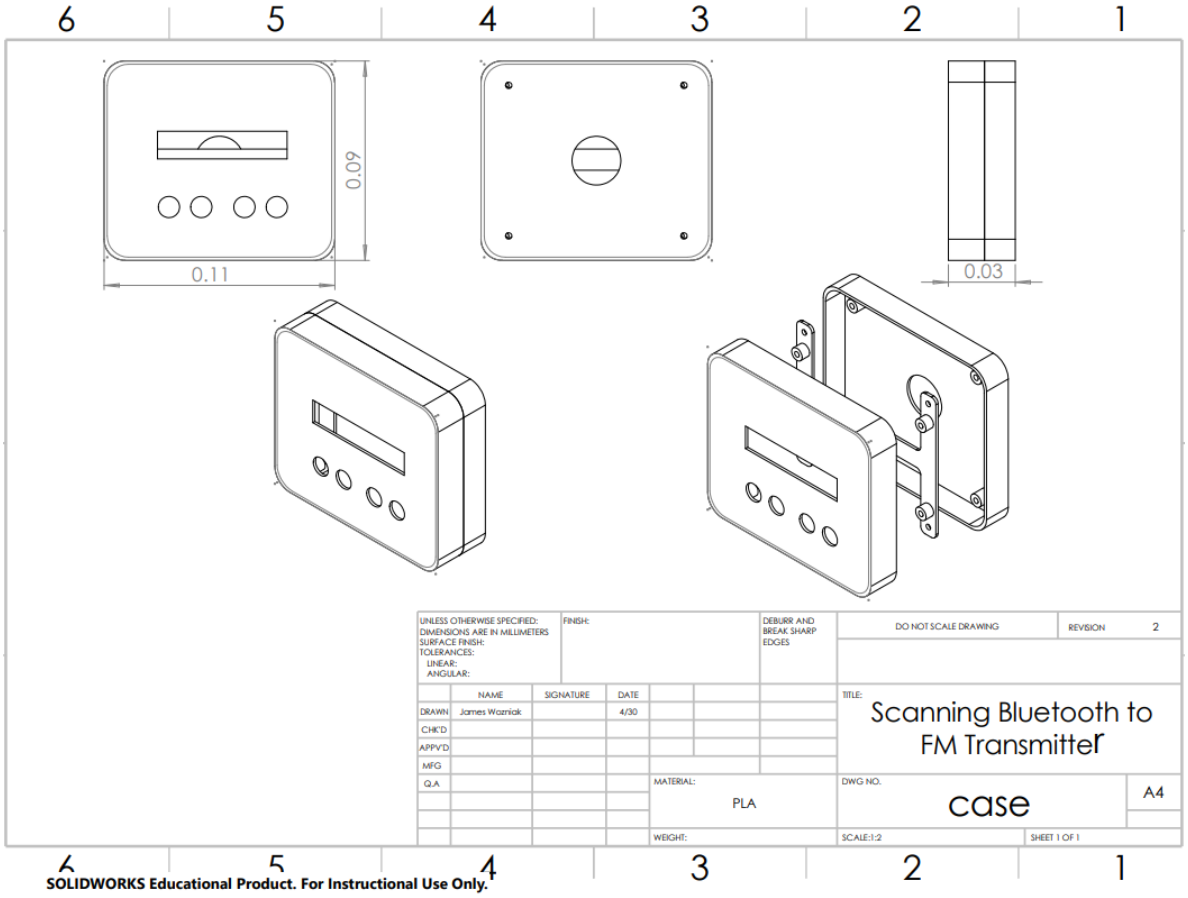


Figure 2: CAD Drawing of Current Design

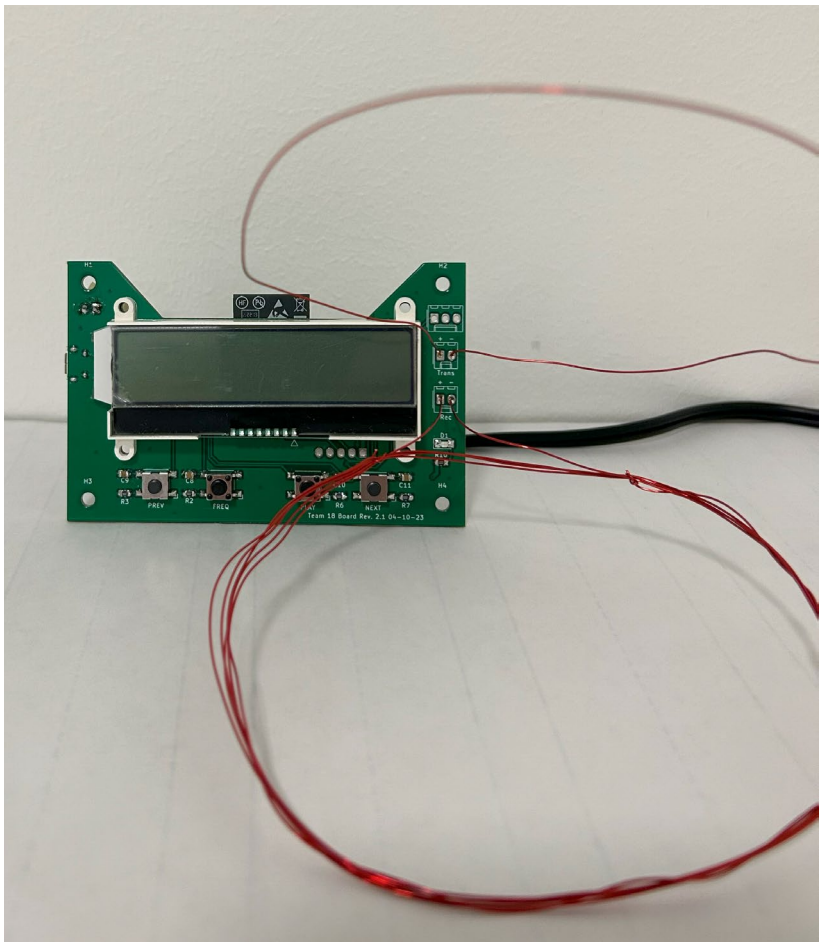


Figure 3: Front of PCB, showing transmitting (top) and receiving (bottom) antennae

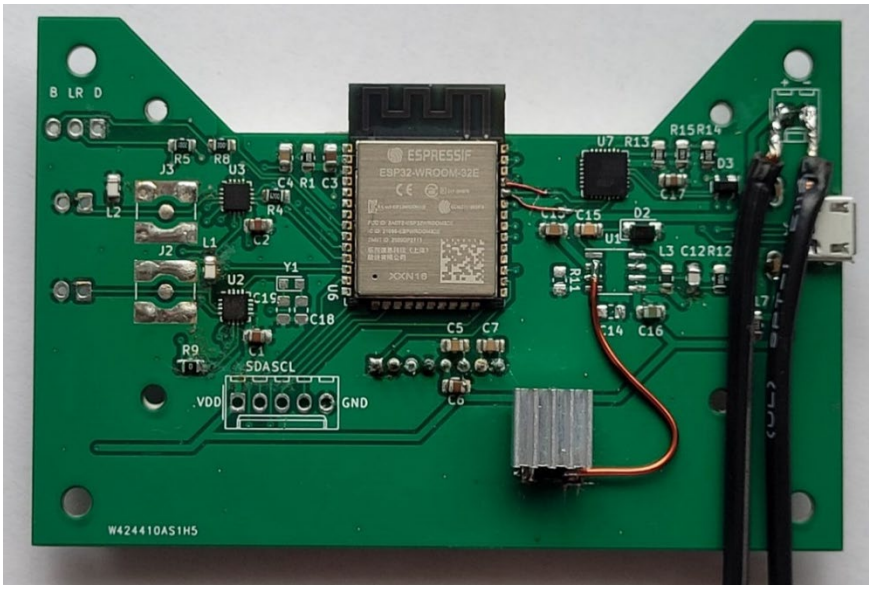


Figure 4: Back of PCB

2.2 CONTROL

Design Procedure

Our device requires multiple communication protocols to handle Bluetooth, IC control, and audio streaming. We could have used multiple devices to implement this but chose to look for a microcontroller that had all this functionality built in to simplify our device and lower our part count.

Design Details

We built our device around the ESP32 line of microcontrollers because they offer all the functionality we desired in a single package. Initially, we chose the S3 variant of the ESP32. This device offers many GPIO pins, Bluetooth, and the I^2C and I^2S communication protocols. Additionally, programming is straight forward because it has direct inputs for universal serial bus (USB) data lines and does not require any additional communications hardware beyond a USB port to interface with a PC. We discovered after our initial PCB order that the S3 variant only has Bluetooth Low Energy, not Bluetooth Classic, and is unable to receive audio data. While a setback, this was remedied by changing to the standard ESP32 device as shown in Fig. 5 that does offer Bluetooth Classic. This device cannot be programmed directly and requires the use of a USB to universal asynchronous receiver-transmitter (UART) bridge and related circuitry as shown in Fig. 6.

The EN pin must be low until the voltage becomes stable on the ESP32, so we included a delay circuit to ensure this. The ESP32 features strapping pins that define its startup state. These pins include pull-up and pull-down resistors for a default configuration that met our requirements, so we left them disconnected.

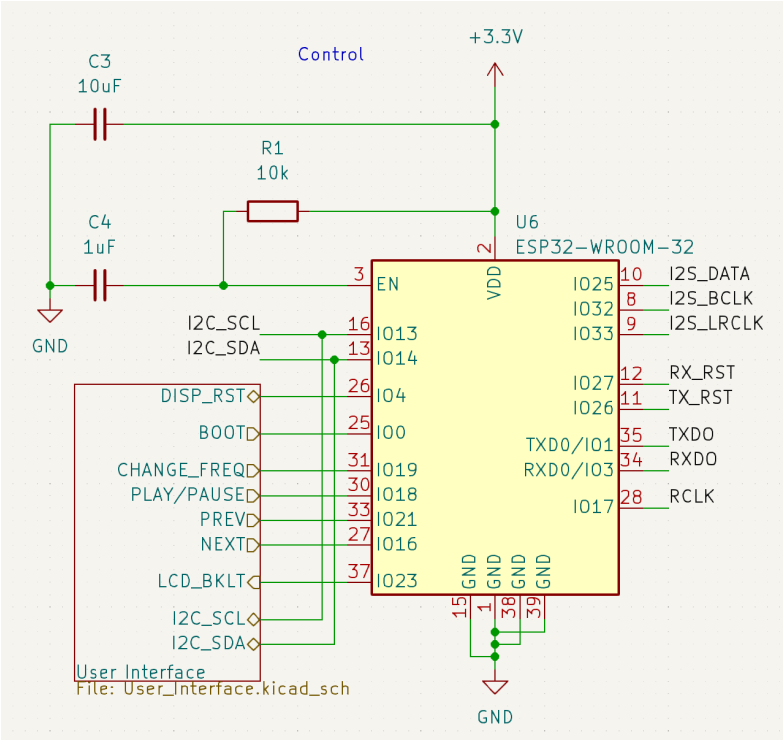


Figure 5: Control schematic

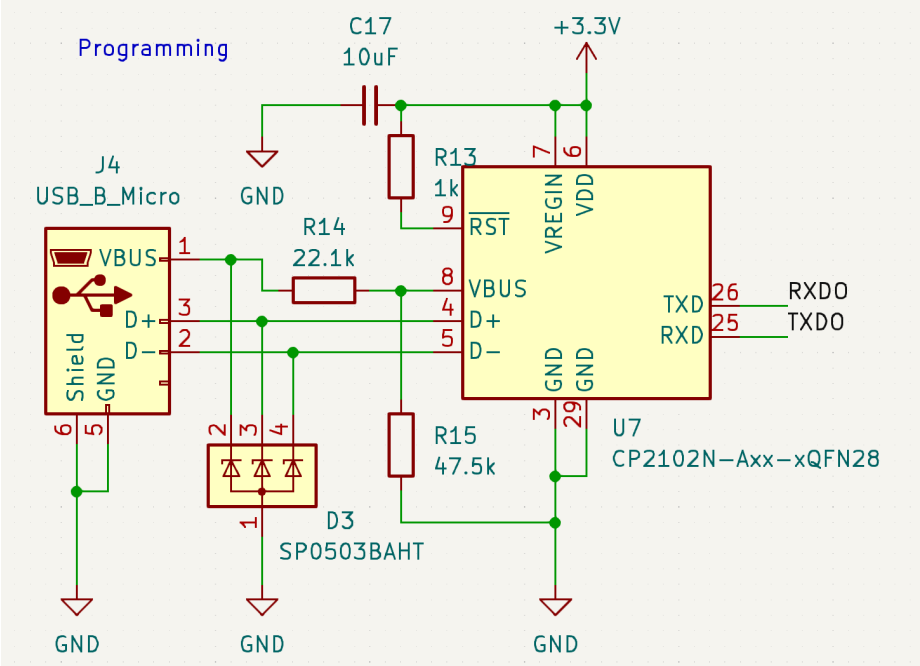


Figure 6: Programming schematic

Verification

The ESP32 manufacturer, Espressif Systems, offers their Internet of Things Development Framework (IDF). The IDF provides useful tools and functions for ESP32 devices. When flashing the device, it provides feedback that it is properly communicating with the ESP32. Initially, we received an error that the IDF could not communicate with the ESP32. After referencing the USB-UART bridge datasheet [1] we realized the TX and RX pins were not properly connected to the ESP32. We corrected this by cutting the trace and adding wires to properly route the signals. After this, we flashed the device and received confirmation from the IDF that it was successful, verifying the functionality of the programming circuit.

We verified the operation of the I^2C and I^2S buses. Using a logic analyzer attached to test headers, we wrote and read messages to the buses from the ESP32 and confirmed they were correct.

To verify GPIO functionality, we flashed a simple program to write the state of the GPIO to a serial terminal and saw that as we connected the GPIO pins to ground or 3.3 V that the state was correct.

2.3 BLUETOOTH

Design Procedure

As mentioned above, the initial microcontroller used does not support the Bluetooth Classic stack, only Bluetooth Low Energy. The Bluetooth Advanced Audio Distribution Profile (A2DP), the profile that facilitates the streaming of audio from one device to another, is only available in the Classic stack, so a different version of the chip supporting A2DP was required.

Design Details

We created the system with the functionality provided by Espressif Systems' Audio Development Framework (ADF), another set of tools specifically designed for audio applications on ESP32 devices. The "A2DP Sink Stream" project, included in the examples of the ADF, served as the base for the Bluetooth code. Modifications included changing the pins for I^2S, removing references and functions relating to ESP32 development boards, and adding interrupts. These interrupts provide the ability to control audio playback and view audio metadata via the Bluetooth Audio/Video Remote Control Profile (AVRCP), a feature not planned for in the original design but included for further ease of use.

Verification

The IDF contains several functions for logging the output of programs to a serial monitor, including the state of the Bluetooth subsystem. Using these functions, we monitored the terminal as a smart phone connected to our device and confirmed Bluetooth functionality. We verified A2DP and AVRCP functionality by logging song metadata and playback status to the monitor after a successful connection.

2.4 FM TRANSMITTER

Design Procedure

Our design requires the transmitter module to take digital audio as input and output an FM radio signal. We chose the Si4711, from Skyworks Solutions, because it contains almost all the necessary hardware in a single IC. The chip performs the modulation digitally and therefore is compatible with the I^2S protocol, eliminating the need for any external digital to analog converters. It is capable of multiple serial communication protocols. All our other devices use I^2C, so this chip allows us to use the same communication bus.

There were options for transceiver chips. Having a single chip for receive and transmit functionality would reduce the complexity of our communications, and our PCB, but would not allow us to scan while transmitting. The Si4711 gave us all the functionality required in a single package and gave us the flexibility to use a separate receiver.

Design Details

We used the schematics shown in the Si4711 datasheet [2] as a guide for our design. For the antenna, we used guidelines [3] provided by the manufacturer.

We connected the relevant pins on the IC for I^2C and I^2S data to their respective buses. We left the analog audio inputs disconnected as they are unused. The I^2C address of the chip is selected by connecting the SEN pin to either the low or high voltage level. We connected it to the high voltage level.

Based on our reading of the datasheet, we initially thought that the RCLK pin could be grounded and was not necessary to be driven. However, after initial testing, we realized that this pin has to be driven by an external clock signal. In our second board revision, we connected this pin to the GPIO 17 pin on our microcontroller and drove a pulse width modulation (PWM) waveform at the frequency specified by the datasheet [2]. Our final schematic is shown in Fig 7.

We placed a subminiature version A (SMA) connector across the output of the transmitter to allow us to view the output on a spectrum analyzer. We used a loop antenna around the edge of our board for transmitting, all per the antenna design guidelines [3]. The guidelines specify that a single loop should be eat least 13 cm in circumference and have 5 mm of clearance from the ground plane [3]. We exceed these requirements as our board is over 20 cm in circumference.

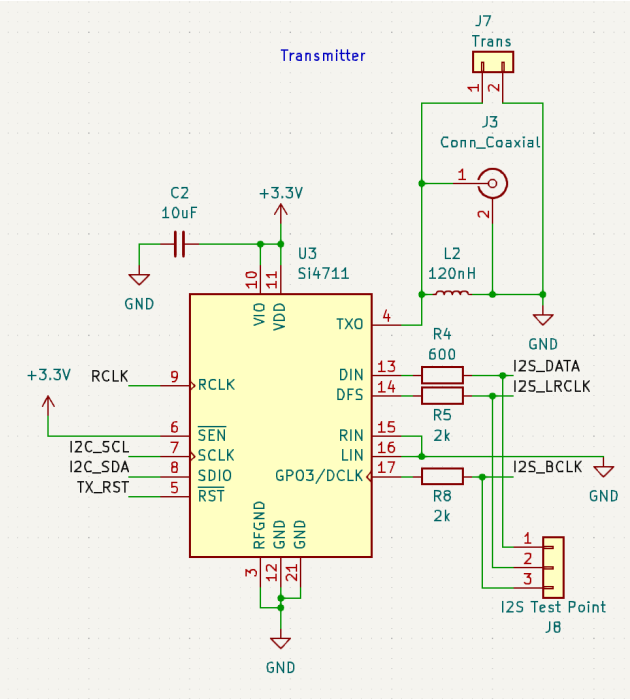


Figure 7: Transmitter schematic

Verification

As with all peripheral devices connected to the I^2C bus, we first verified that the device was communicating with our microcontroller and able to respond in the expected ways. We then proceeded to verify the ability to transmit. We connected the SMA connector on the transmitter to a spectrum analyzer and set the transmitting frequency to specific channels and inspected the output as per our verification tables from our design document (see Appendix B). While transmitting, we observed a large peak at the specified frequency on the spectrum analyzer, shown in Fig. 8. Once Bluetooth was connected and sending audio to the transmitter chip, we further verified transmission by listening to it on an FM radio. We found that we can clearly hear the audio being transmitted from the phone we are connected to.

Federal Communications Commission (FCC) regulations specify that for an unregulated transmitter the emitted electric field strength should not exceed 250 µV/m past 3 meters from the transmitter. However, we did not have access to the equipment to measure this precisely. To do our best to comply with these regulations, we tuned the output power settings such that beyond 3 meters, the reception on our portable radio degraded rapidly.

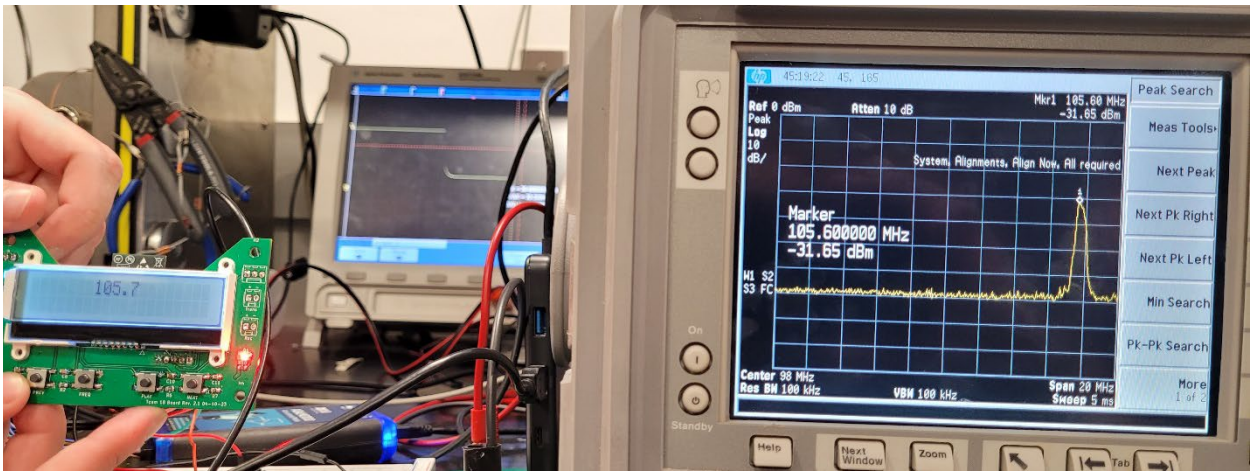


Figure 8: Transmitter test setup

2.5 FM RECEIVER

Design Procedure

Our design requires the receiver module to measure the strength of signals being transmitted on specific channels in the area and communicate that data to the microcontroller.

We chose the Si4706, from the same manufacturer, because it also contains all the necessary hardware onboard the chip. Critically, the chip measures the strength of the signal at the receiver's input after tuning to a particular FM channel. We use this feature to determine whether a station is broadcasting at a given frequency.

Design Details

We used the schematics shown in the Si4706 datasheet [4] as a guide for our design. Our final schematic is shown in Fig. 10. For the antenna, we used the same design as the transmitter because, according to the design guidelines [3], both are compatible with the wire loop antenna.

We again connected the relevant pins for I^2C to the bus. We left the analog and digital audio output pins unconnected because they are not used. The receiver uses the same address selection (setting the SEN pin high or low) as the transmitter, so we set it low. This ensures that the transmitter and receiver have different addresses on the I^2C bus.

The Si4706 also requires the same external reference clock signal as the Si4711, so once we realized this, and made the change to the transmitter, we made the same change to the receiver.

After initially using a loop antenna of the same dimensions as the transmitter, we found that increasing the number of turns increased the magnitude of the received signal. Four turns is optimal for our receiving loop antenna.

Once we were able to verify the signal strength measurement, we implemented our channel selection algorithm. Our system constantly sweeps the entire FM spectrum and updates a rolling average of the last five signal strength values at each channel. It then finds local minimum values below a threshold and sorts those minimum values from least to greatest signal strength. Then starting with the lowest value, it

checks if the two neighboring stations on either side of the channel are below the threshold. If they are, it selects this channel, thus ensuring a clear region in the FM band surrounding the transmitting frequency. The visual aid in Fig. 9 demonstrates this process with an arbitrary spectrum.

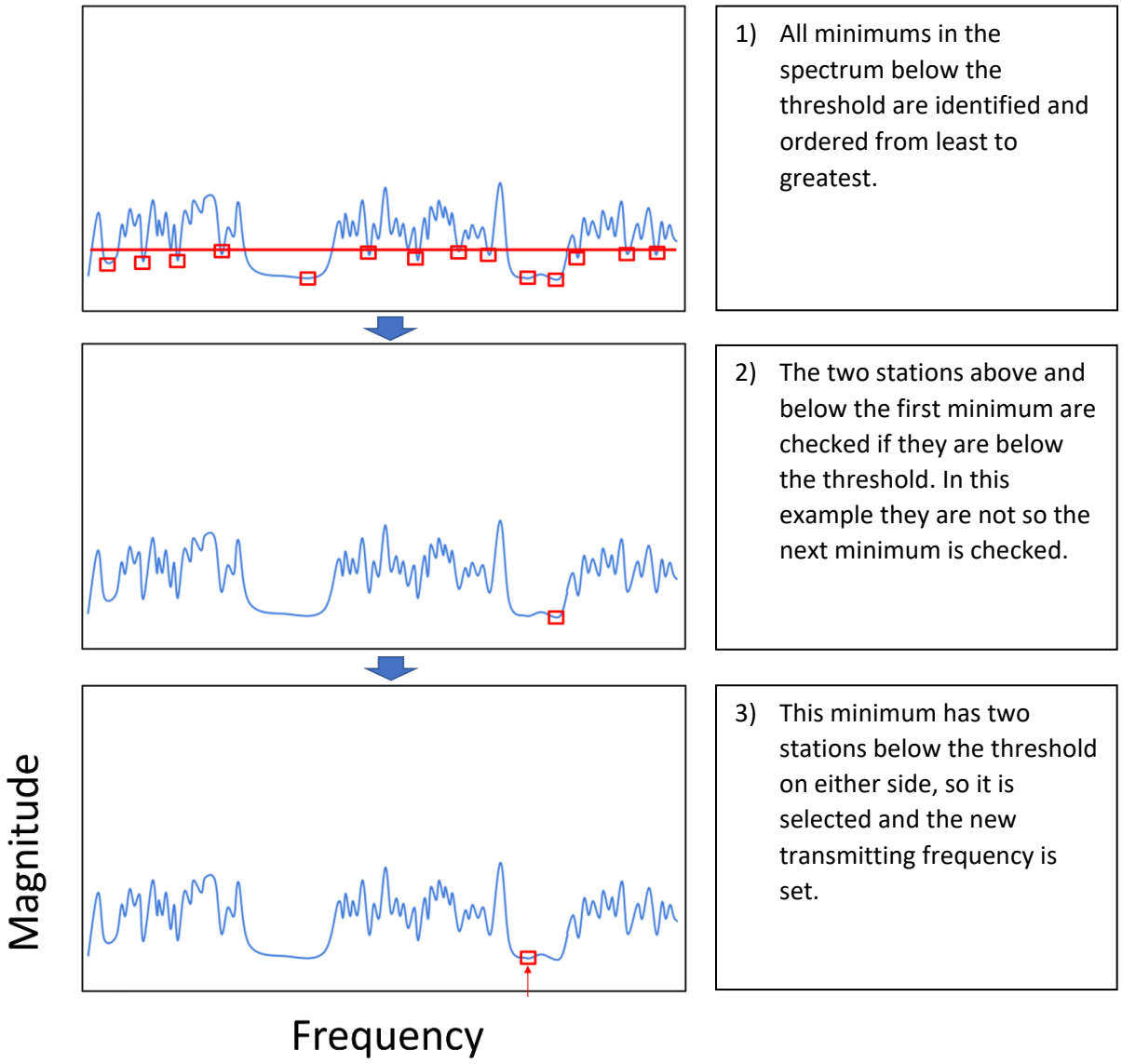


Figure 9: Algorithm visual aid, shows process of selecting from an arbitrary spectrum

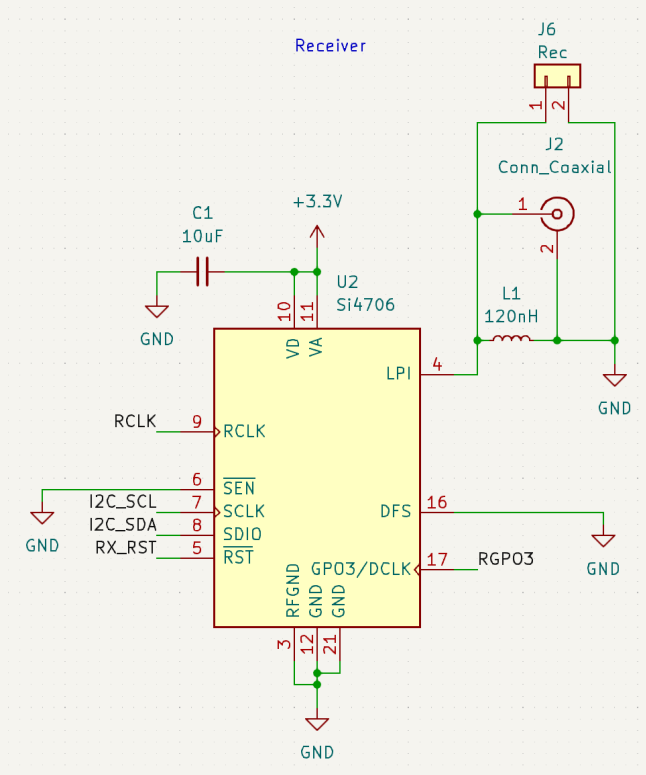


Figure 10: Receiver schematic

Verification

As with the transmitter, we first verified I^2C connection and proper communication with the microcontroller. We then verified the ability to measure the received signal strength using a signal generator connected to the SMA port on the receiver. We injected a 96.5 MHz tone directly into the receiver input and then scanned all the channels in the FM spectrum. We read back the received signal strength values, and Fig. 11 shows these results. The peak at 96.5 MHz represents the tone from the signal generator. This verified we properly measure received signal strength.

Even with no antennas connected, we observe strong coupling from the transmitter to the receiver. During the testing shown in Fig. 11, the transmitter was set to 104.5 MHz. The peak at this point in the figure demonstrates this coupling with our receiver.

We performed the above tests powering the 3.3 V system directly from a bench power supply. Once we powered the entire board from our regulator, we noticed that the noise floor of the receiver was elevated, obscuring the peaks of local channels. After further diagnostics, we determined the switching regulator was producing broad-band noise strong enough to drown out the reception of local stations. To solve this, we switched to a linear regulator and are now able to properly receive.

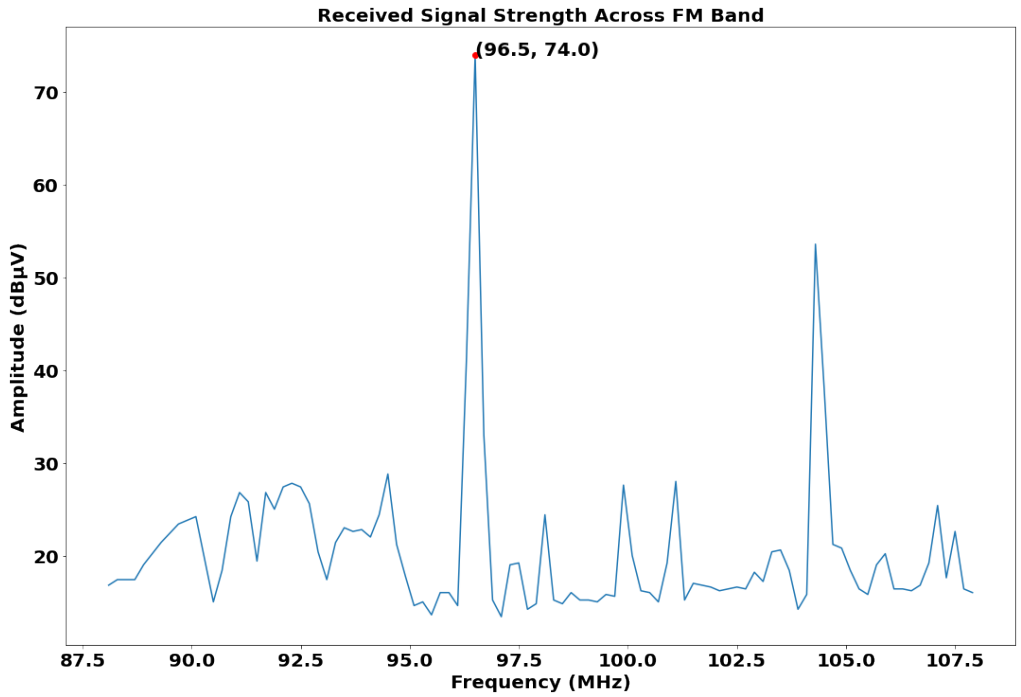


Figure 11: Signal strength measurements across FM spectrum from receiver testing

2.6 POWER

Design Procedure

An automotive auxiliary power port provides 12 – 15 V. Therefore, we require a power supply that can take a variable input voltage across that range and output a stable 3.3 V. We initially chose to use a linear regulator for its simplicity and low noise characteristics. After further consideration, we decided to use a switching regulator for its efficiency. As described in the receiver section, this turned out to be a mistake and our final device uses a linear regulator as shown in Fig. 12.

Design Details

For the switching power supply we designed, we chose to use the TPS5403 regulator. This regulator met our requirements as it has an input range of 4.5 – 28 V and can output up to 1.7 A continuously at 3.3 V. Additionally, it varies its switching frequency during operation. We wanted this feature to minimize issues related to switching noise. We designed the circuit, shown in Fig. 13, based on the example provided in the TPS5403 datasheet [5]. To further minimize noise, we included bypass capacitors to ground as close as possible to the 3.3 V inputs of every IC in our device and on the 12 V input. When we decided to switch to the linear regulator the circuit became much simpler, only the bypass capacitors remain.

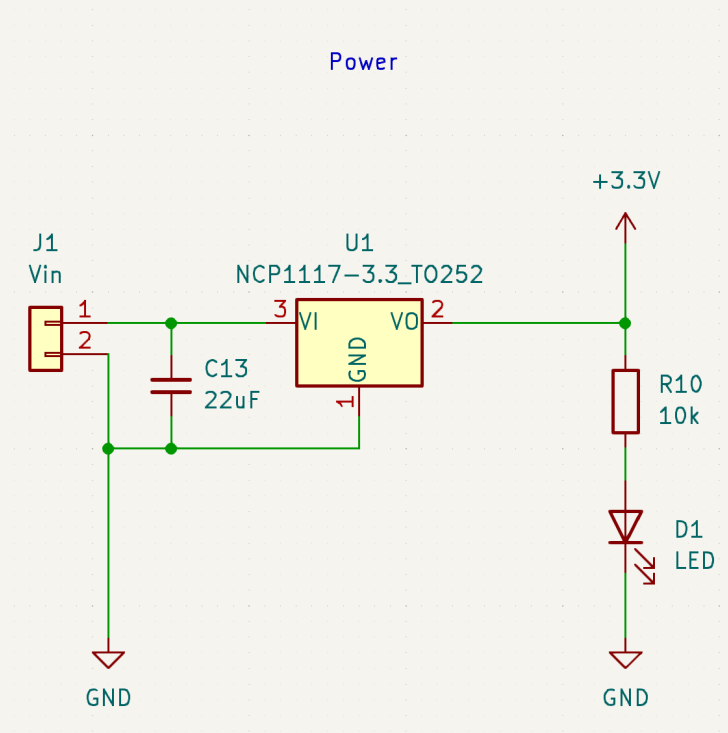


Figure 12: Linear power supply schematic

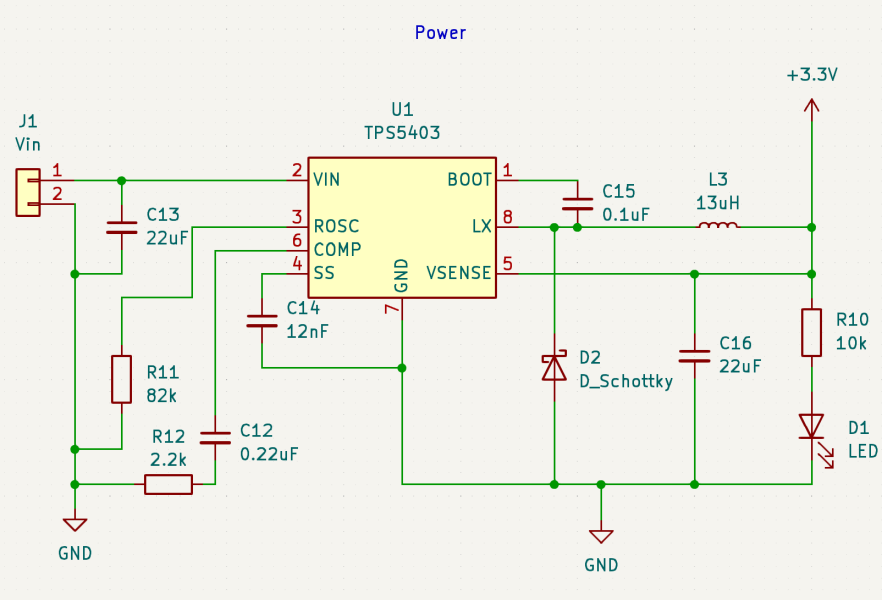


Figure 13: Switching power supply schematic

Verification

When we attempted to verify the switching power supply, we found it could only output 160 mA before the voltage dropped below our required threshold of 3.0 V. Although this circuit should have been able to function properly according to the datasheet, we found we needed to increase the switching frequency from 300 kHz to 750 kHz by changing the resistor labeled R11 in Fig. 13. After this change, the power supply met our desired threshold of outputting 600 mA at 3.3 ±0.3 V.

We tested the linear power supply under a 600 mA load and found that after 58 seconds the output voltage drops due to the regulator overheating. While we initially set our verification load as 600 mA, the max current all our devices could technically draw simultaneously, we were not using many of the features built in to the ESP32 and would therefore never see that load in our usage. In our testing we see a max current draw of 250 mA, and only for a brief instant on startup. The linear regulator is able to handle a 300 mA load indefinitely and therefore is verified for our device.

2.7 User Interface

Design Procedure

The device requires some way for users to provide input for control and a way to give feedback to the users on the current operation of the device. Pushbuttons and a liquid crystal display (LCD) were chosen for these purposes respectively. There are four buttons present on the final device, one to change the frequency and the remaining three to control media playback. The LCD shows the frequency of the channel currently being transmitted on. Another screen or display could have been used, especially if trying to minimize the footprint of the device. A common choice in existing devices on the market is a seven-segment display, but we opted for the LCD to provide additional options for displaying other, non-numerical information.

Design Details

The schematics for display and button subcircuits are shown in Fig. 14 and Fig. 15 respectively. The ESP32 allowed for configuring the GPIO pins with either pull-up or pull-down internal resistors. We used the pull-up resistors to reduce the number of components needed. We connected the pushbuttons to a resistor-capacitor (RC) circuit for switch debouncing purposes. The time constant of a RC circuit can be determined by

$$\tau = \mathrm{RC} \tag{2.1}$$

where *R* is the resistance of the circuit and *C* is the capacitance. The buttons have a maximum bounce time of 5 ms, so using Equation (2.1), the resistance was chosen to be 10 kΩ and the capacitance 1 µF for a time constant of 10 ms. This ensures that on any given press of the button, only one falling edge will be seen at the output. We set the buttons up as peripherals using functionality from the ADF. This allows them to act as interrupts from the main program, meaning that we do not need to constantly poll them, speeding up and simplifying our code.

The LCD used, the Newhaven Display C0220BiZ, communicates via I^2C which enabled simple integration with the other ICs and microcontroller. Additionally, the display includes a built-in font table, so characters can be sent directly to it for immediate viewing, without a need for additional components

for decoding. The LCD has pins to control the backlight, a useful feature for nighttime driving, but we set it to be always on in software since the device lacks any sensor or control to automatically adjust the screen brightness.

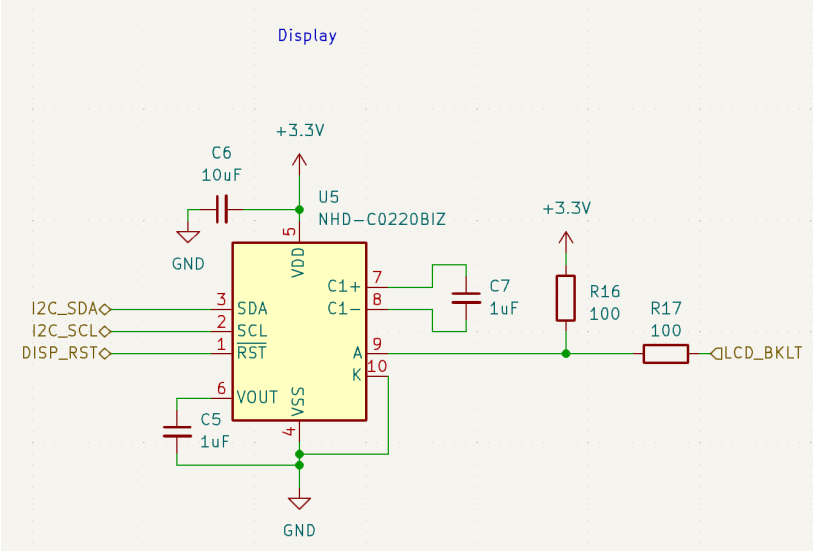


Figure 14: Display schematic

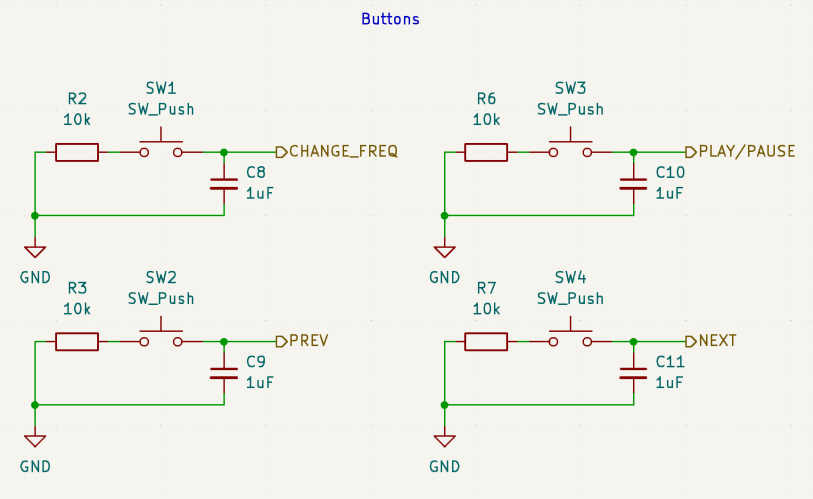


Figure 15: Button schematic

Verification

We verified button functionality using a serial monitor and oscilloscope. We created a simple test program which upon pressing a button, would print to the monitor. This provided a quick method to see that the switch is properly debounced, but we also used the oscilloscope to verify this is the case.

We verified the LCD qualitatively by sending it various characters and control sequences over the I^2C bus. We found that it was able to properly read and display any sequence of characters we sent. We checked that the backlight is able to be toggled, so although this functionality is not used by our device, it can easily be enabled and used in further iterations.

3 COST AND SCHEDULE

3.1 COST ANALYSIS

Table 1 Bulk production cost breakdown

Part No.	Manufacturer	Unit Price	Quantity	Price
ESP32-WROOM-32E-N16	Espressif Systems	$3.6000	1	$3.6000
SI4711-B30-GM	Skyworks Solutions, Inc.	$3.5150	1	$3.5150
SI4706-D50-GM	Skyworks Solutions, Inc.	$3.7500	1	$3.7500
NHD-C0220BIZ-FSW-FBW-3V3M	New Haven Display Int.	$10.7160	1	$10.7160
NCP1117DT33G	onsemi	$0.2723	1	$0.2723
CP2102N-A02-GQFN28R	Silicon Labs	$2.3836	1	$2.3836
SP0503BAHTG	Littelfuse Inc.	$0.2937	1	$0.2937
B3FS-1010P	Omron	$0.5319	4	$2.1278
100 Ω Resistor	Generic	$0.0058	2	$0.0115
600 Ω Resistor	Generic	$0.0051	1	$0.0051
1 kΩ Resistor	Generic	$0.0096	1	$0.0096
2 kΩ Resistor	Generic	$0.0046	2	$0.0091
10 kΩ Resistor	Generic	$0.0050	6	$0.0300
22 kΩ Resistor	Generic	$0.0046	1	$0.0046
47 kΩ Resistor	Generic	$0.0047	1	$0.0047
22 µF Capacitor	Generic	$0.0237	1	$0.0237
10 µF Capacitor	Generic	$0.0168	5	$0.0841
1 µF Capacitor	Generic	$0.0149	7	$0.1040
120 nH Inductor	Generic	$0.0415	2	$0.0829
24 AWG Wire	Generic	$0.3486	1	$0.3486
Case	Custom Manufactured	$0.2500	1	$0.2500
Red LED	Generic	$0.0685	1	$0.0685
10118194-0001LF	Amphenol ICC	$0.4600	1	$0.4600
PCB	Custom Manufactured	$0.1000	1	$0.1000
			Total	$28.2549

The projected unit cost for our device when produced in bulk is shown in Tab. 1 as $28.2549. This cost is higher than we would prefer it to be but would be brought to a more reasonable value if the display was changed to something smaller. This would not have a negative impact on our device as we have extra screen space that we do not use.

Based off our 252 hours of labor and a rate of $40 per hour, we come to a total design cost of $25,200 for this device.

3.2 SCHEDULE

Table 2: Development schedule

Week Ending On	Item	Individual
26-Feb	Complete Schematic	Dan Piper
	Part Order #1	James Wozniak
5-Mar	Complete and Order PCB Rev. 1	Dan Piper
12-Mar	Complete CAD Model and Print Enclosure	James Wozniak
19-Mar	Test Bluetooth Subsystem Functionality	James Wozniak
	Assemble PCB Rev. 1	All Team
26-Mar	Complete and Order PCB Rev. 2	Dan Piper
	Part Order #2	James Wozniak
	Test User Interface Subsystem Functionality	Dan Piper
2-Apr	Complete and Order PCB Rev. 3	Dan Piper
	Test Receiver Subsystem Functionality	Madigan Carroll
	Part Order #3	James Wozniak
9-Apr	Test Transmitter Subsystem Functionality	Madigan Carroll
	Test Power Subsystem Functionality	All Team
16-Apr	Test Control Subsystem Functionality	All Team
	Complete Final Product	All Team
23-Apr	Perform Verification	All Team
30-Apr	Complete Final Report	All Team

Table 2 shows the completion dates for different parts of the project, as well as the general work distribution for each task.

4 CONCLUSION

4.1 ACCOMPLISHMENTS

The device scans each channel of the FM spectrum and selects the channel with the lowest received power to transmit on. It also pairs to a Bluetooth enabled device and streams audio directly to the transmitter, which can then be heard when a radio is tuned to the specified frequency. Audio playback can be controlled by the buttons on the front panel of the enclosure, and song metadata can be seen on the receiving radio.

4.2 CHALLENGES AND LESSONS

As mentioned in the discussion of the receiver, there is significant coupling between the transmitter and receiver ICs. We know that this is largely due to their proximity to one another, so although it does not affect our final product itself, it makes for a good learning opportunity. In future designs, we will avoid placing the transmitter and receiver close to one another to limit the effect the coupling has on the circuit.

We knew that a switching regulator would produce more noise when compared to the original linear regulator. We assumed that this noise would be negligible against the outputs of the transmitter and receiver. While we considered many ideas for limiting the noise produced, due to time constraints, we reverted to using the linear regulator. We learned to be more cognizant of noise, especially in the context of RF circuits.

4.3 FURTHER WORK

We would like the device to be closer to existing products on the market in terms of its footprint. As it stands, our current design is several times larger than the others. A reduction in size could be achieved by removing most of the PCB features included for test purposes, like the SMA ports and pin headers. In addition, the LCD takes up most of the space on the PCB and since its text displaying capabilities are not being utilized, a more compact display could be used.

We also feel that the device could serve as a base or accessory for building an automobile-based Internet of Things (IoT) network. The ESP32 microcontrollers commonly see use in other IoT projects, and our selected microcontroller already provides built-in functionality for Wi-Fi and hosting a local network. We feel that future work could utilize these capabilities in ways that we did not initially consider.

4.4 ETHICS AND SAFETY

In creating a device related to an activity as inherently risky as driving, it was imperative that throughout our design and production process we acted ethically, using the Institute of Electrical and Electronics Engineers (IEEE) [6] and Association of Computing Machinery (ACM) [7] Codes of Ethics as our guiding principles. First and foremost, we met the requirements of IEEE I.1 regarding safety and public welfare, which is also succinctly represented in ACM 1.2 to “Avoid harm” in protecting driver safety.

Our device allows drivers to operate their vehicles more safely by automating the frequency selection process of the FM transmitter. No additional distracting features were added, and the only difference between our solution and existing products is the use of an LCD. While an LCD can increase the risk of distracted driving if filled with text, our solution only displays the broadcasting channel, making it functionally identical to simpler displays.

Additionally, the electromagnetic spectrum is subject to the regulations and oversight of the FCC here in the United States, so in order to meet the standard of ACM 2.3 to know and respect the rules relating to our work, we followed the FCC requirements to the best of our ability.

4.4.1 FCC Regulations

Under the FCC rules, no license is required for this project given it is not sold commercially and that we follow the parameters for our transmitter specified by Part 15 [8]. These rules state we are allowed a maximum bandwidth of 200 kHz within the 88-108 MHz frequency range. Additionally, the maximum emitted field strength is 250 μV/m at 3 meters away. The testing environment required for verification of these requirements was not available to us, however, we attempted to adhere to them to the best of our ability. The testing of the signal reception at 3 meters away was meant both as an ethical consideration to avoid interrupting other radio listeners and as a model for the FCC rules.

REFERENCES

[1] Silicon Labs, "CP2102N Datasheet," November 2020. [Online]. Available: https://www.silabs.com/documents/public/data-sheets/cp2102n-datasheet.pdf. [Accessed 26 March 2023].

[2] Skyworks Solutions, Inc., "Si4710/11 Datasheet," 13 September 2021. [Online]. Available: https://www.skyworksinc.com/-/media/Skyworks/SL/documents/public/data-sheets/Si4710-11-B30.pdf. [Accessed 2 February 2023].

[3] Skyworks Solutions Inc., "Si47xx Antenna, Schematic, and Layout Design Guidelines," 8 September 2021. [Online]. Available: https://www.skyworksinc.com/-/media/Skyworks/SL/documents/public/application-notes/AN383.pdf. [Accessed 23 February 2023].

[4] Skyworks Solutions, Inc., "Si4706-D50 Datasheet," 2 November 2021. [Online]. Available: https://www.skyworksinc.com/-/media/Skyworks/SL/documents/public/data-sheets/Si4706-D50.pdf. [Accessed 2 February 2023].

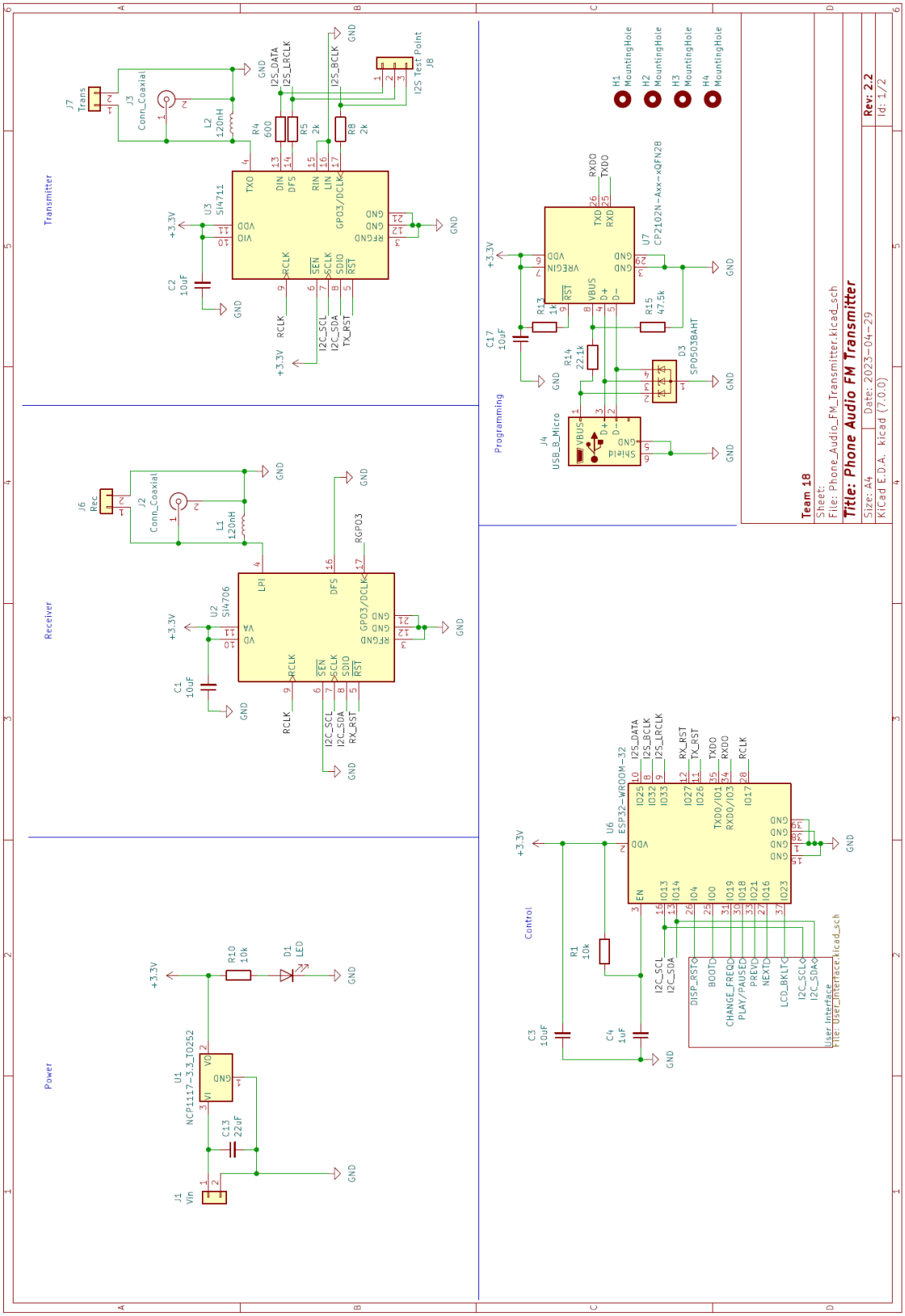
[5] Texas Instruments, "TPS5403 Datasheet," September 2012. [Online]. [Accessed 2023].

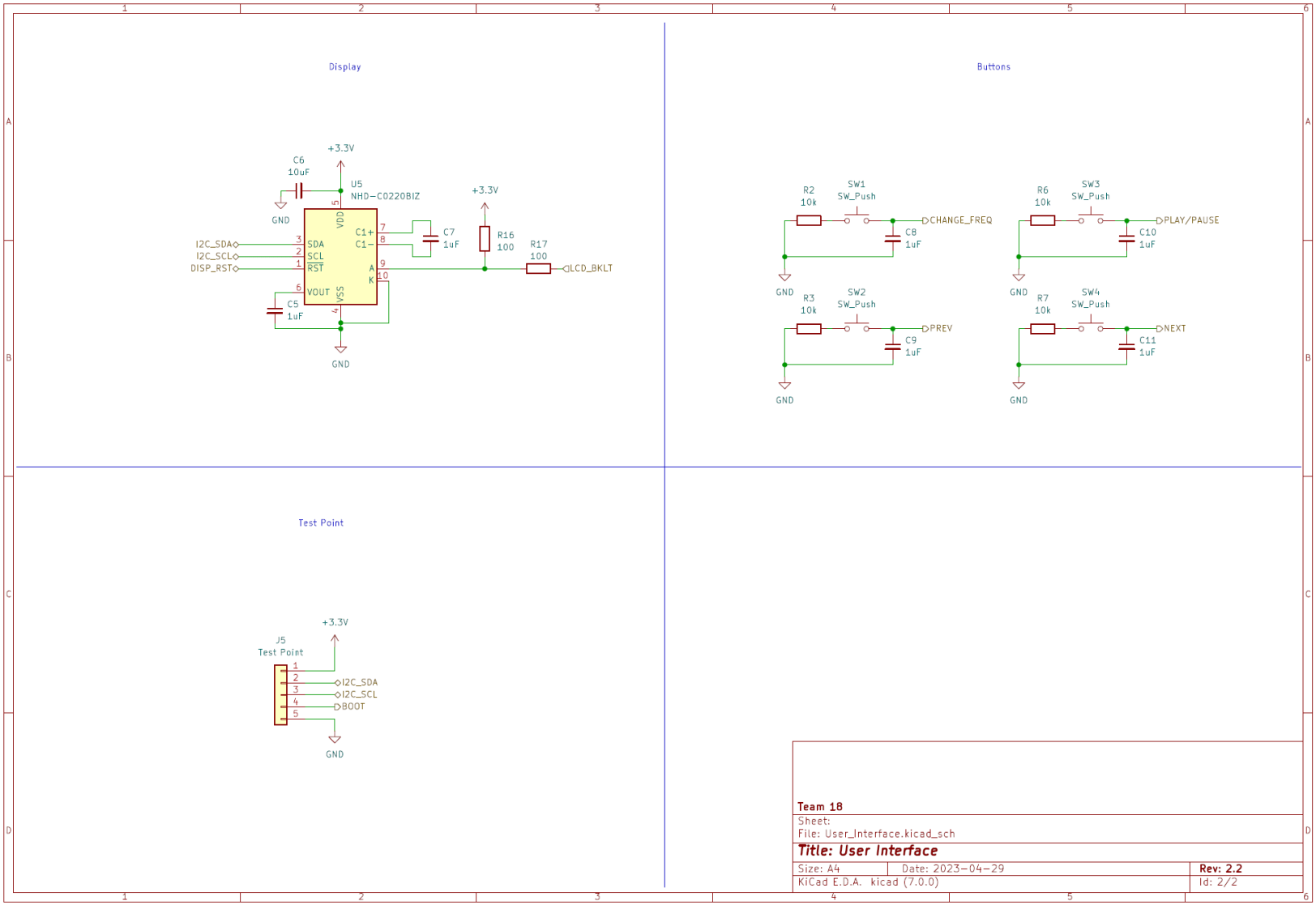
[6] IEEE, "IEEE Code of Ethics," June 2020. [Online]. Available: https://www.ieee.org/content/dam/ieee-org/ieee/web/org/about/corporate/ieee-code-of-ethics.pdf. [Accessed 2 February 2023].

[7] ACM, "ACM Code of Ethics and Professional Conduct," 22 June 2018. [Online]. Available: https://www.acm.org/code-of-ethics. [Accessed 2 February 2023].

[8] FCC, "Code of Federal Regulations," [Online]. Available: https://www.ecfr.gov/current/title-47/chapter-I/subchapter-A/part-15. [Accessed 7 February 2023].

APPENDIX A – DEVICE SCHEMATICS




Display
+3.3V
C6
10uF
GND
U5
NHD-C0220BIZ
VDD
SDA
SCL
RST
VOUT
VSS
C1+
C1-
A
K
I2C_SDA
I2C_SCL
DISP_RST
C5
1uF
C7
1uF
+3.3V
R16
100
R17
100
LCD_BKLT
GND
Buttons
R2
10k
SW1
SW_Push
CHANGE_FREQ
C8
1uF
GND
R6
10k
SW3
SW_Push
PLAY/PAUSE
C10
1uF
GND
R3
10k
SW2
SW_Push
PREV
C9
1uF
GND
R7
10k
SW4
SW_Push
NEXT
C11
1uF
GND
Test Point
+3.3V
J5
Test Point
I2C_SDA
I2C_SCL
BOOT
GND
Team 18
Sheet:
File: User_Interface.kicad_sch
Title: User Interface
Size: A4
Date: 2023-04-29
Rev: 2.2
KiCad E.D.A. kicad (7.0.0)
Id: 2/2

APPENDIX B – REQUIREMENTS AND VERIFICATION TABLES

Requirements	Verification
Microcontroller	
Can write and read data from the I^2C bus	Writing to bus:
	1. Flash a program to the board which writes a singular byte to the bus repeatedly.
	2. Read the data on the bus using a digital signal analyzer.
	Reading from bus:
	1. Flash program to board that reads data from the bus and outputs it to the serial connection.
	2. Using a function generator, send SDA and SCL signals as inputs to the bus pins.
	3. Using a serial client, check the data output by the microcontroller against the input.
Can write audio data to the I^2S bus	1. Flash a program to the board which writes a 2kHz sine wave to the bus and outputs each sample to the serial connection.
	2. Read the data on the bus using a digital signal analyzer.
	3. Using a serial client, check the data output to the bus.
Takes input from the user interface	1. Flash program to board that reads data from the user interface and outputs it to the serial connection.
	2. Provide inputs to the user interface.
	3. Using a serial client, check the data output by the controller against the input.

Bluetooth	
Device is discoverable and able to pair to a separate Bluetooth audio device.	1. Flash a program to the board that configures the Bluetooth system with a unique name and writes Bluetooth data to the serial connection.
	2. Using a smartphone, find and connect to the appropriately named Bluetooth device.
	3. Check on the smartphone and a serial client for a successful connection.
Bluetooth audio signal from connected device is properly received at the microcontroller.	After Bluetooth connection is established, via serial connection to the microcontroller:
	1. Flash program to board which uses functions within the ESP library to extract AVRCP data, specifically song metadata and writes it to the serial connection.
	2. Connect to the microcontroller with PuTTY or other serial client and check output data for correctness.
Transmitter	
The transmitter IC is able to transmit a signal at a frequency specified by the control unit. The	With the antenna not connected to the PCB:
	1. Connect the output SMA port to a spectrum analyzer.
	2. Flash the board with a program to change the transmission frequency at specified intervals.
	3. Program sends command to set frequency.
	4. Program sends frequency argument.
	5. Measure the output frequency at the spectrum analyzer.
When the control unit is transmitting digital audio via the I2S data channel, the transmitter IC is able to receive that data, modulate it, and transmit it at the chosen frequency.	1. Connect the output SMA port to a spectrum analyzer.
	2. Flash the board with a program to send various single-frequency audio tones to the transmitter at a predetermined transmitting frequency.
	3. Program selects transmitting frequency.
	4. Program sends tone frequency via I^2S.
	5. Observe the output on the spectrum analyzer.

Receiver	
The receiver IC must be able to measure the signal strength of an FM channel specified by the control unit, and communicate that back to the control unit	With the antenna not connected to the PCB:
	1. Connect the input SMA port to a spectrum analyzer
	2. Set up a signal generator to output a low frequency tone ~ kHz, modulated to a specified FM channel
	3. Flash the board with a program to measure the signal strength at that frequency, and at another frequency with much lower power
	· Control unit sends command to set receive frequency.
	· After waiting a minimum of 300µs control unit sends command to query the signal quality indicators
	· Control unit sends signal strength measurements to serial port, to be read on the PC.
	4. Compare received power indicated at the frequency output by the signal generator against the power indicated at the other frequencies.
Power Supply	
Output 600mA continuously at 3.3V ±0.3V	1. Solder the power supply to PCB before all other components.
	2. Connect an oscilloscope in parallel with a 5Ω resistor across the output voltage rail and ground. This will draw a minimum of 600mA across the entire acceptable voltage range.
	3. Measure the output voltage for at least 5 minutes and ensure it remains within the acceptable range.
Interface properly with an automotive auxiliary power port	1. Plug the device into the port.
	2. Ensure the power LED illuminates.

User Interface	
LCD displays alphanumeric characters received on the I^2C bus.	1. Flash a program to the board that writes a character or sequence of characters to the I^2C bus (i.e., “Test”).
	2. View the LCD display for correct characters.
Buttons provide a clear signal when pressed/depressed. Once the signal falls below 30%±5% of the maximum, it does not increase until the button is released.	1. Measure the outputs of each button on an oscilloscope.
	2. Press each button and check the voltage curve for adherence to the mentioned requirements.