EDUCATIONAL STICK SHIFT ASSISTANT

By

Aadhar Patel (alpatel2)

Ian Kidder (ikidder2)

Maulin Patel (mpate222)

Final Report for ECE 445, Senior Design, Spring 2021

TA: Alex Sirakides

5 May 2021

Project No. 4

Abstract

For our project, we designed an educational device that teaches users how to drive manual transmission cars. The Educational Stick-Shift Assistant plugs into any compatible vehicle's OBD-II port, from which it draws power and communicates with the car's engine control unit. The module reads information regarding the car's speed, engine RPMs, and throttle position to output real-time instructions for the user to follow. Through a series of instructional pre-programmed lessons, the user can master manual transmission vehicles in an isolated environment at slow speeds. We implemented the full functionality of the Educational Stick Shift Assistant, except that it is not compatible with all of the cars we initially designed it to work with.

Contents

1. Introduction

Knowing how to drive a manual transmission vehicle is a valuable skill for anyone traveling abroad, buying a new car, or looking to save some money. However, it can be hard to learn how to drive a manual without in-person instruction. The Educational Stick Shift Assistant provides instructions to anyone who wants to learn to drive a manual car independently through interactive pre-programmed lessons. Our solution is a system that reads the car's speed and engine RPMs to output audio instructions for the driver to follow. Our goal is to have the driver become familiar with shifting gears, giving them the knowledge and confidence they need to drive a manual car.

COVID-19 has caused mass layoffs, closed down businesses, and halted the economy. As a result, the global production of cars has decreased [1]. In addition, due to social distancing and safety guidelines, many people want to avoid using public transportation. One safe and economical alternative is to buy used cars. However, the price of used cars has tremendously increased [2] due to this pandemic. An affordable option is cars with manual transmissions, which are on average $1000 cheaper [3] compared to their automatic counterparts. However, manual transmission vehicles are inaccessible to people who do not know how to drive them. Knowing how to drive a manual car also equips the person with a vital travel skill as nearly 80% of cars on the road in Europe have manual transmissions as of 2020 [4]. In conclusion, knowing how to drive a manual car is a beneficial skill for almost anyone.

2 Design

2.1 Block Diagram

Figure 1 represents the block diagram of the Educational Stick Shift Assistant and provides a visual representation of the components used to achieve the final product. The Engine Control Unit (ECU) Interface Module is responsible for retrieving vehicle data such as RPMs, vehicle speed, and throttle position when requested by the Control Unit. The Control Unit is responsible for analyzing data from the ECU Interface Module and input from the user through the Button Breakout Board and sending commands to the Audio Module. The Audio Module outputs the desired audio cue stored on the SD Card through the AUX Connector for playback by Vehicle Speakers. The Power Module is responsible for regulating and providing power to all the submodules/components in the design. The Program & Debug Module is not user accessible and enables easy programming and monitoring of the microcontroller during prototyping and final build stages.

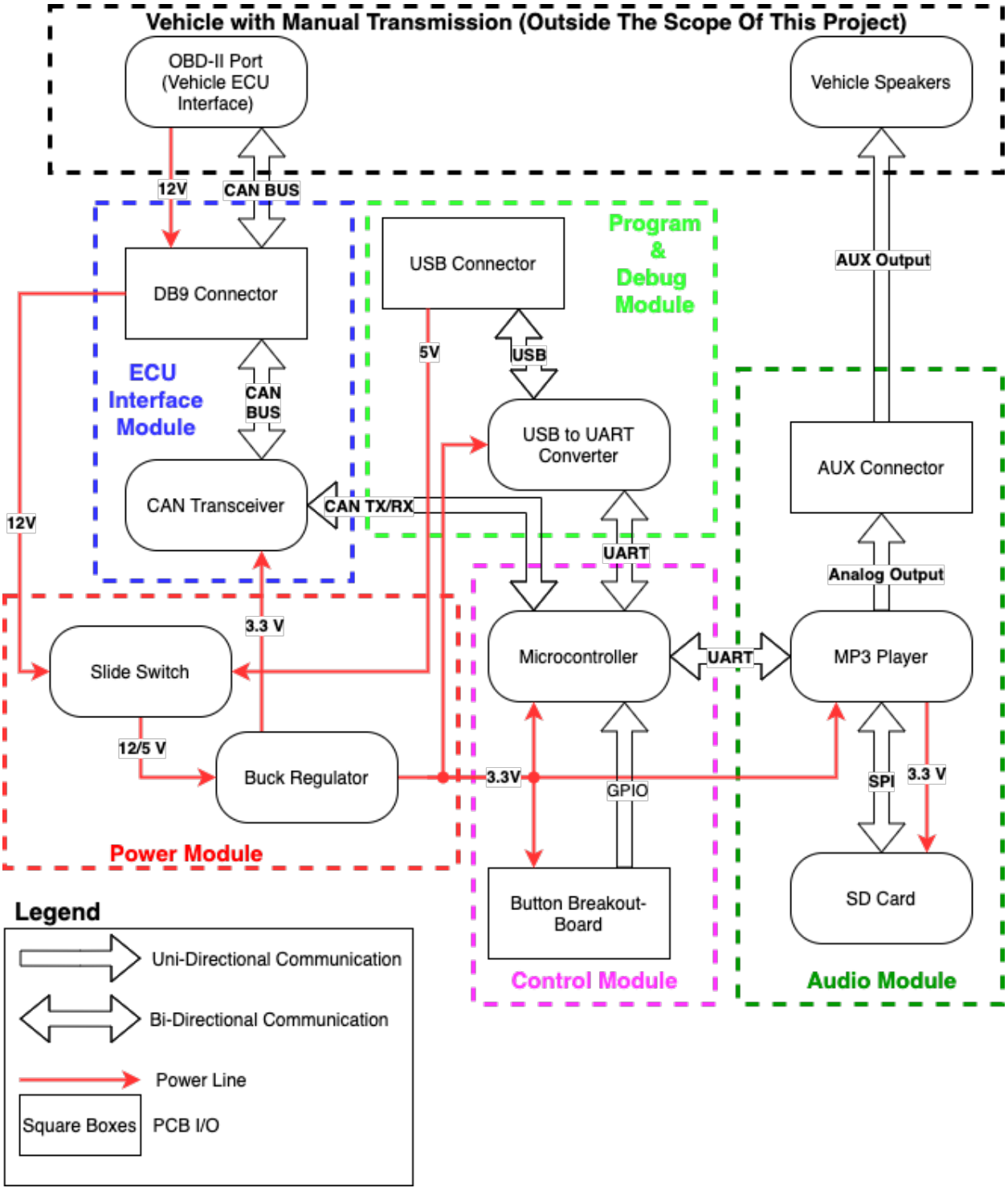

Figure 1: Stick Shift Assistant System Block Diagram

2.2 Power Module

The Power Module is responsible for regulating and providing power to the rest of the components in the design via a 3.3 V rail. We designed the Power Module to source power from two sources, the USB connector in the Program & Debug Module and the DB9 Connector. The DB9 connector provides access

to the car's 12 V battery which connects to the input of the slide switch. The other input to the slide switch is the 5 V supply provided by USB. Regardless of the input, the output is stepped down to a constant 3.3 V by a buck regulator. Figure 2 shows the buck regulator connected to the slide switch and both input voltages.

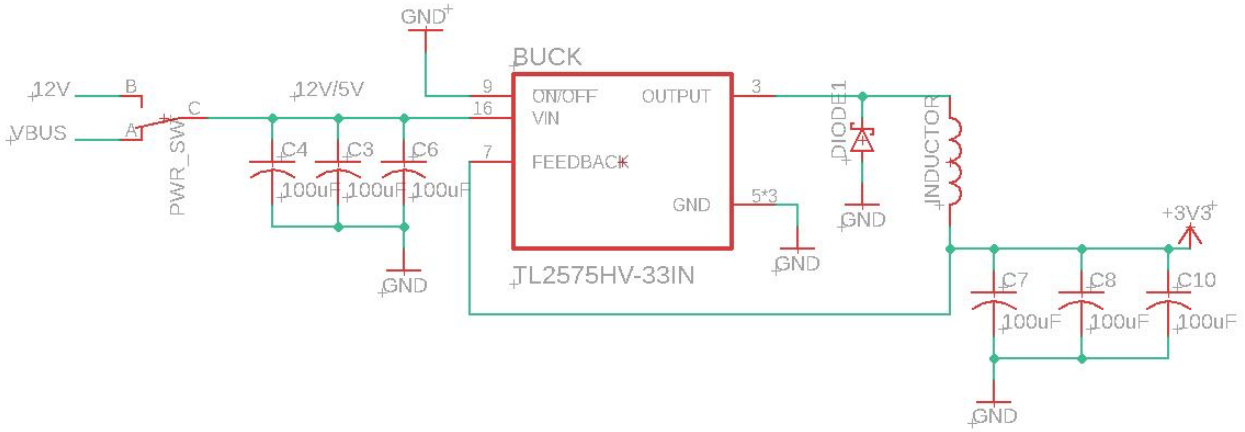

Figure 2: Power Module Schematic

We used a CAN transceiver and USB-UART converter in our initial design that each required 5 V inputs. To power them, we were planning on using a slide switch configuration similar to figure 1, except that the inputs to the slide switch would be the output from a fixed 5 V buck regulator and the 5 V USB output. To power the 3.3 V components on the board, we planned on placing a 3.3 V low-dropout linear regulator (LDO) at the output of the slide switch.

We moved away from this design because it would require more power conversions and lower overall energy efficiency while adding unnecessary complexity. We were able to eliminate the LDO from our design by switching to 3.3 V tolerant versions of the CAN transceiver and USB-UART converter, therefore removing the need for a 5 V source anywhere on our board. To accommodate this change, we switched to a fixed output 3.3 V buck regulator for the final design.

2.2.1 Buck Regulator

The buck regulator steps down both the 5 V USB input and the vehicle's 12 V battery to 3.3 V to power the rest of our board. In order to utilize this component, we had to reference the datasheet [5], which allowed us to create a typical application circuit. Some of the passive components that were needed consisted of capacitors, an inductor, and a diode.

We encountered one issue while implementing the Power Module on our PCB because the first 3.3 V buck converter we ordered was only available in a 3 mm x 3 mm package with 0.25 mm wide pads, which were far too small for us to solder by hand. The small pad size forced us to switch to a buck regulator in a larger package, so we selected a through-hole buck regulator that we placed on pin headers.

We then had an issue where the buck regulator was outputting a constant 4.2 V across all input voltages when attached to the pin headers on the PCB. This was surprising, especially since we had successfully tested the Power Module on a breadboard beforehand, so we were confident that the buck regulator

was functioning correctly. After extensive testing, we suspected a loose connection between the buck regulator and the PCB caused by the pin headers, so we desoldered the pin headers and soldered the buck regulator directly to the PCB. This solved the issue, but unfortunately, the 4.2 V burnt out our ESP32 by exceeding its maximum input voltage of 3.6 V [9].

For the input capacitor, we chose a ceramic capacitor of 100 uF with a 16 V rating based on the recommendation made in the datasheet. Similarly, for the output capacitor, the datasheet offered a range from 100 uF to 470 uF. The output capacitor that we chose was 100 uF with a 16 V rating again. For both the input and output, the voltage rating for the capacitor had to be greater than or equal to 4.95 V, as shown below.

$$Vout = 3.3\ \text{V} \quad [1]$$

$$Voltage\ Rating\ for\ Capacitor\ \geq 1.5\ * Vout\ =\ 4.95\ V \quad [2]$$

We performed a series of calculations for the inductor to figure out the minimum current rating it must handle. The value of the inductor given by the datasheet was 330 uH for a fixed 3.3 V output. To find the inductor's minimum required current rating, we performed the following calculations.

$$Nominal\ Switching\ Frequency, fs\ =\ 52\ kHz \quad [3]$$

$$L = 330\ uH \quad [4]$$

$$Vin\ (max)\ =\ 15\ V \qquad Vin\ (min)\ =\ 5\ V \quad [5]$$

$$Max\ Duty\ Cycle, D\ = \frac{Vout}{Vin(min)}\ =\ 0.66 \quad [6]$$

$$Inductor\ Ripple\ Current, \Delta I_L\ = \frac{(Vin(max)\ -\ Vout)\ *\ D}{fs\ *\ L}\ =\ 0.45 \quad [7]$$

$$Minimum\ Current\ Rating, IL\ (min)\ =\ I\ max\ +\ \frac{\Delta Il}{2}\ =\ 1.225\ A \quad [8]$$

From the calculations up above and from equation 8, we can see that the minimum current rating for the 330 uH inductor has to be 1.225 A. We chose an inductor rated at 1.6 A, which meets the specifications required for the buck regulator.

To test and verify we could meet our requirements for the buck regulator, we swept the input voltage to the buck between 4.7 V and 5.3 V to simulate the worst-case scenario from the USB cable. The input voltage from the USB cable should never drop below 4.7 V nor exceed 5.3 V. The output voltage from a 12 V car battery can vary between 10.5 V, for an unhealthy battery, up to 14.5 V, for a battery being charged by an alternator. Therefore we swept the input voltage between 10.5 V and 14.5 V to ensure that we achieved a stable 3.3 V output across all possible input voltages. We also tested these voltage conditions at the minimum and maximum current load scenarios. We found the minimum current load to be 84 mA when the Stick-Shift Assistant is idling and the maximum current load to be 110mA during peak operation. As shown in figure 3, the buck's output voltage has a maximum deviation of 37 mV (1.12%) from 3.3 V across all possible input voltages and load conditions, well within our specifications for all components.

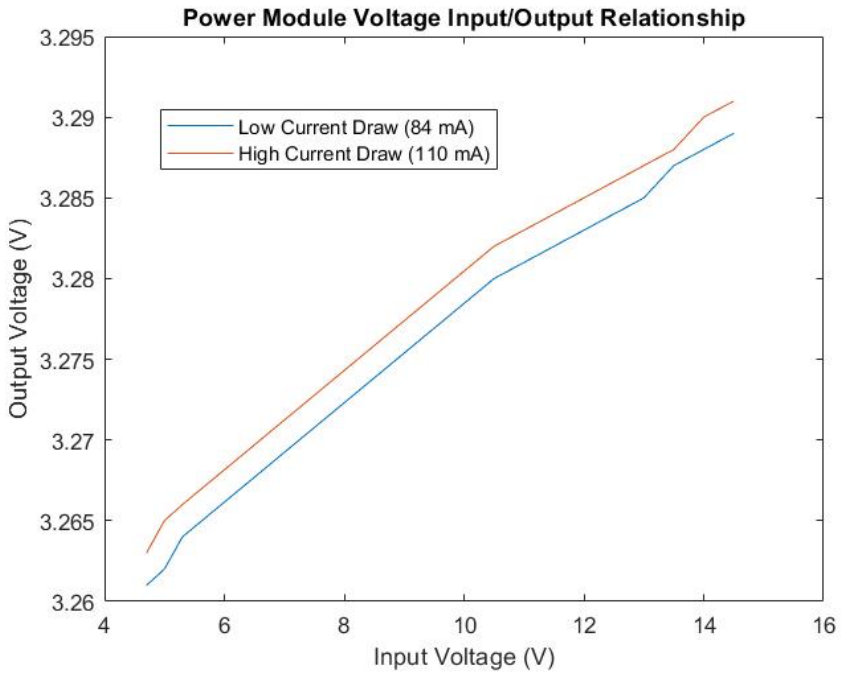

Figure 3: Voltage Input vs. Voltage Output Relationship

2.2.2 Slide Switch

The slide switch takes in the 5 V output from the USB cable and approximately 12 V output from the DB9 connector and allows us to choose which source we want to power the board with. This ensures that we can power the board when programming the ESP32 while retaining the ability to draw power from the 12 V car battery once finished. This switch is enclosed within the product box, making it inaccessible to the end user. Figure 2 shows how the slide switch connects to the buck regulator and the USB connector.

2.3 ECU (Engine Control Unit) Interface Module

The ECU Interface Module begins with an OBD-II to DB9 cable, which provides access to the vehicle's ECU and 12V battery for us to use as a power source. The DB9 cable's data pins connect to the CAN Transceiver, allowing the Control Module to request and interpret ECU signals from the vehicle. Figure 4 shows an example circuit diagram for the TL2575HV-33IN CAN Transceiver. In this figure, CANH and CANL are the high and low CAN physical bus pins, respectively, and connect to the vehicle through the DB9 connector. The D and R pins are transmitting and receiving pins, respectively, connected to the microcontroller via GPIO pins. The IC is then powered by the 3.3 V power line and grounded to the signal ground provided by the DB9 connector.

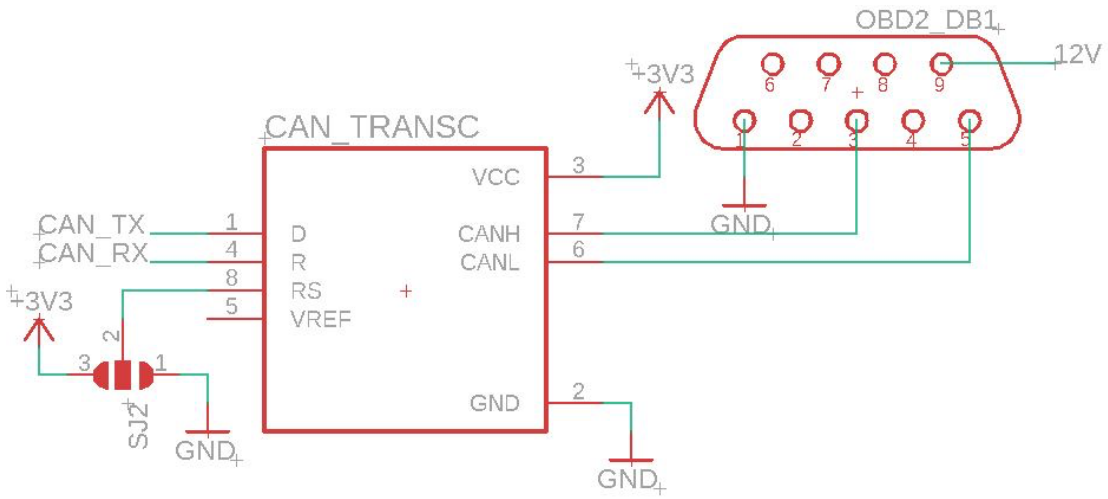

Figure 4: ECU Interface Module Schematic

2.3.1 CAN Transceiver

The CAN transceiver is the interface between the CAN physical bus and the CAN protocol controller (microcontroller) [6]. This sub-module is used to help convert the digital signals from the CAN controller to Analog signals on the CAN physical bus when transmitting commands and vice-versa when receiving data. The CAN transceiver enables the communication between the CAN protocol controller and the ECU. This enables us to request vehicle data such as vehicle RPMs, vehicle speed, and throttle position for further analysis by the microcontroller.

During the testing phase of the CAN transceiver, we used a CAN simulator which we utilized with a RedBoard and CAN bus Shield. We used this simulator to send out data that was interpreted by the CAN transceiver. The baud rate at which the CAN transceiver successfully communicated with the CAN bus and received data was 500 Kbps. We chose this baud rate based on the fact that it is common in most vehicles. Figure 5 shows the control module receiving data at 500 Kbps.

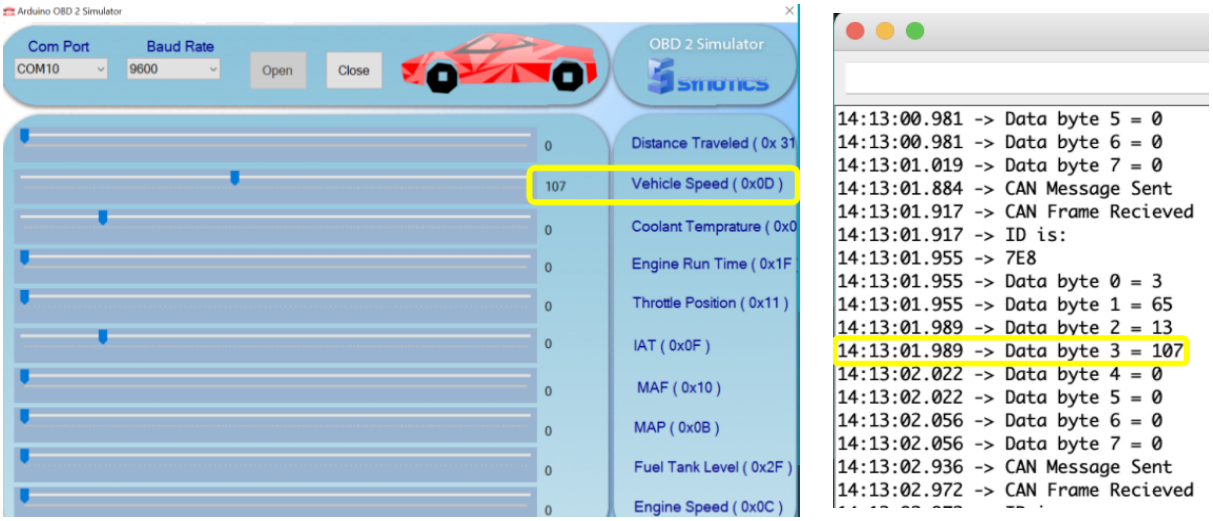

Figure 5: CAN Transceiver Verification

2.3.2 DB9 Connector

The DB9 connector allows communication with the vehicle ECU by connecting to the OBD-II port via a DB9 to OBD-II adapter cable. We chose the DB9 connector for its smaller physical footprint and part availability compared to the OBD-II port. An adapter cable connects the DB9 connector to the OBD-II port in the vehicle. See figure 4 for a diagram depicting DB9 pinouts and which physical pins we are using for this project.

During testing, we identified an issue where signals from the OBD-2 port to the DB9 connector did not have continuity. We eventually determined that the SnapEDA footprint for the DB9 connector was incorrectly labeled, resulting in PCB traces connecting to the wrong pinouts on the DB9 connector. This was solved by desoldering the PCB mount DB9 connector and utilizing a DB9 connector soldered to the PCB via extension wires.

2.4 Control Module

The Control Module consists of the microcontroller and input buttons. It is in charge of handling, processing, analyzing, and transmitting data. The microcontroller processes data received from the ECU interface module and gave commands to the audio module. The buttons allow the user to choose what kind of feedback they want from the assistant. Figure 6 shows the circuit schematic for the Control Module.

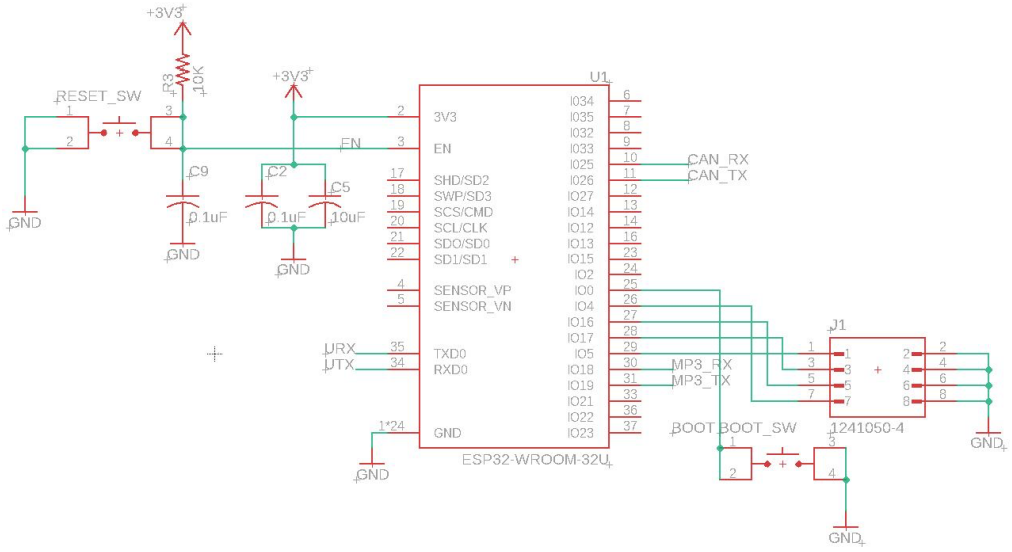

Figure 6: Control Module Schematic

2.4.1 Microcontroller

The microcontroller is the central processing unit and has three main functions. The microcontroller retrieves live vehicle data such as RPMs, vehicle speed, and throttle position from the vehicle ECU, enables the user to input which lesson to learn via the Button Breakout Board, and controls the MP3 player to playback audio cues through the Audio Module.

We chose the ESP32 microcontroller from ESPRESSIF Systems to achieve these functions for its extensive user-friendly and open-source C libraries to achieve these functions. One significant advantage of the

ESP32 WeCAN controller library specifically for the ESP32, which we used in our design. To program and test the ESP32 microcontroller, we used the ESP32 Core available in the Arduino IDE libraries. We encountered multiple difficulties programming the ESP32, starting with the boot button on GPIO0. On an ESP32 development board, a prefabricated auto-program circuit handles the flash sequence, so we overlooked the need to hold GPIO0 to ground to flash our code to the standalone ESP32. We initially solved this by soldering a jumper wire between GPIO0 and ground when flashing the chip. In the next iteration of our PCB, we added a boot button connected between GPIO0 and ground to solve this problem robustly.

As mentioned in section 2.2.1, we accidentally burnt out our first ESP32 during testing by supplying 4.2 V to the input pin. We figured out that the ESP32 burned after it repeatedly failed to flash code from our laptops, giving the error: “Timed out waiting for packet header.” After figuring out that we had burned the chip, we abandoned this PCB and ESP32, moved to our improved PCB design, and used a spare ESP32-WROOM-32-D that a TA had in the lab. The functionality of the ESP32-WROOM-32-U and ESP32-WROOM-32-D are effectively the same for our purposes, the only difference being that the U model does not have the built-in antenna present on the D model.

2.4.2 Button Breakout Board

The button breakout board allows drivers to interface with the software. The driver can choose which lesson to learn and can repeat a lesson if desired. The microcontroller communicates with the input buttons via GPIO and monitors them to make various decisions in software. The buttons mount to the outside of the housing and connect to the microcontroller via jumper wires. As shown in figure 6, the buttons are active low with the 3.3 V power rail as their source. The buttons are all debounced via software and inputs are ignored when the vehicle is in motion to discourage distracted driving.

2.5 Program and Debug Module

The Program and Debug module includes the USB connector and the USB to UART converter. This module is strictly for programming and debugging the microcontroller and monitoring the data received from the ECU interface module during the prototyping and build phases. This module is not accessible to the user and serves no purpose in the finished product. However, note that this module can both program/communicate with the microcontroller and also power the board independently, eliminating the need to plug the board into a car’s OBD-2 port every time. The USB connector connects the USB to UART converter, which allows communication to the microcontroller as the standalone ESP32 microcontroller does not have USB D+/D- pinouts. Figure 7 shows a schematic view of the Program and Debug module.

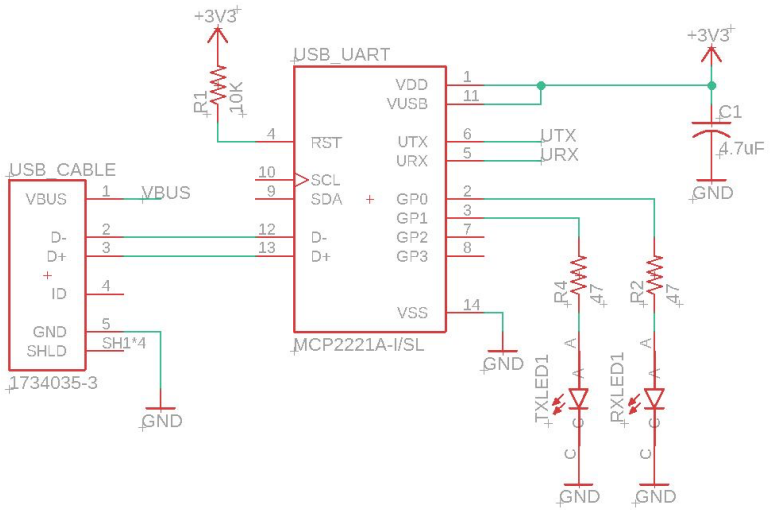

Figure 7: Program and Debug Module Schematic

2.5.1 USB to UART Converter

The USB to UART converter receives data from a laptop over USB and converts the data to serial UART, which interfaces directly with the microcontroller. This device serves as a bridge between the USB connector and the microcontroller.

2.6 Audio Module

The Audio Module consists of the AUX connector and a breakout MP3-player board with a built-in SD card slot. This module provides audio instructions to the driver through the vehicle's speakers. The audio instructions are pre-recorded MP3 audio files. The AUX connector provides DAC_R and DAC_L signals used for audio playback by the vehicle speakers. Figure 8 shows a schematic view of the Audio Module.

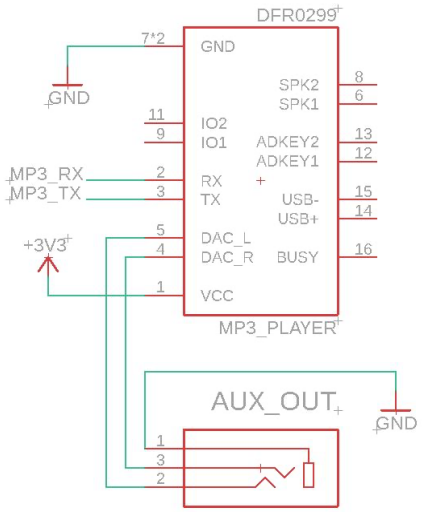

Figure 8: Audio Module Schematic

2.6.1 Mini MP3 Player

The MP3 player receives playback commands from the microcontroller and outputs the appropriate audio file from the attached SD card reader. The command set is specified in a provided library. The MP3 player sends its left and right channel audio output via an AUX cable. This AUX cable connects to the vehicle's AUX input for playback through vehicle speakers.

3. Software Design

3.1 Flowchart

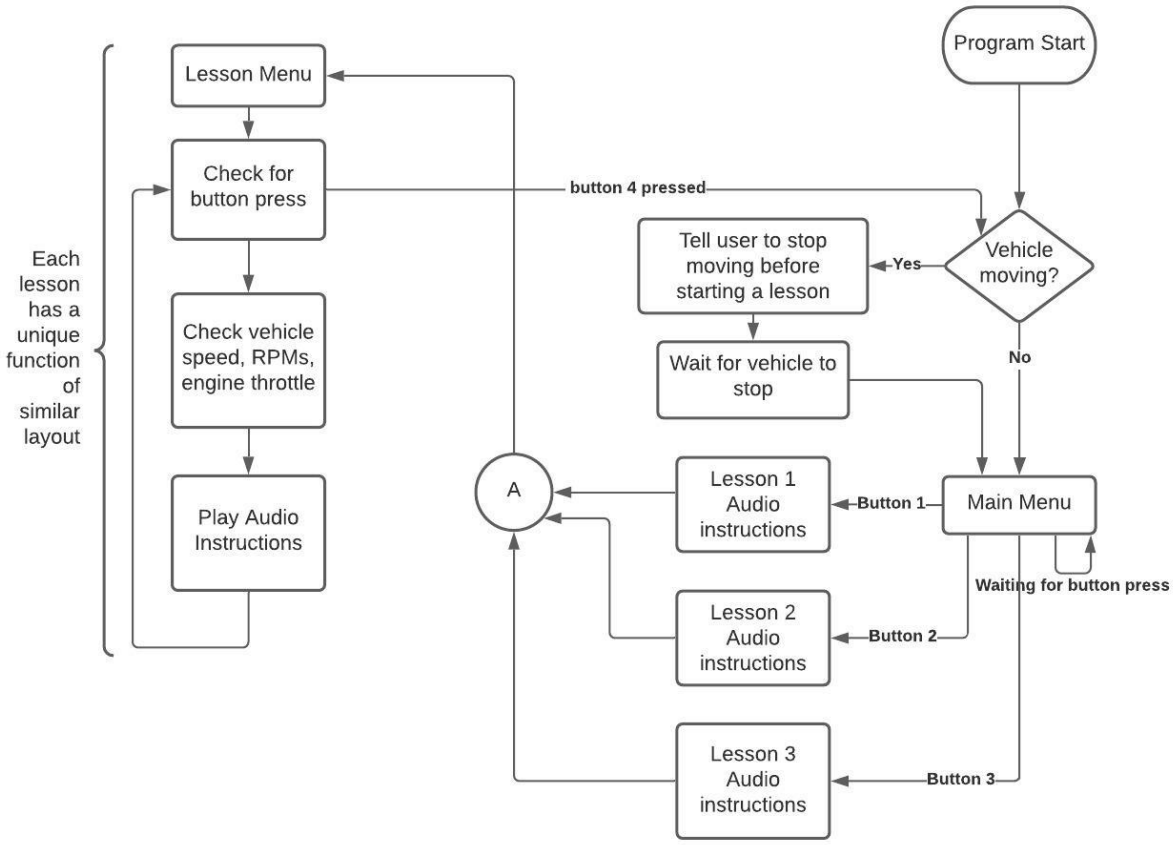

Figure 9: Software Flowchart

Figure 9 shows an abstract layout of the Educational Stick Shift Assistant's software implementation. The software implements many features that affect user experience. First, the software ensures that the input buttons, intended for the user to interact with the software, are disabled when the vehicle is in motion. From the Main Menu, the user can select between the various lessons. From a Lesson Menu, the user can elect to repeat the current lesson, return to the Main Menu, or continue with the current lesson (only available for some lessons). The flowchart depicts when the module is actively reading ECU data and outputting audio instructions. An abstract structure of individual lesson functions is shown on the left side of the flowchart. Note that all lessons differ somewhat and that this is just a general breakdown of significant actions the device performs to achieve full functionality.

3.2 Driving Lessons

This module delivers driving instructions in audible format to teach users how to drive stick-shift. To achieve this, the user can select from three driving lessons, each with varying difficulty levels. The module delivers two key lessons that a driver may practice multiple times to help them learn to drive stick-shift at low speeds in a safe environment away from traffic and pedestrians.

In addition to the standard accelerator pedal, brake pedal, and steering wheel, manual transmission vehicles also have a clutch pedal and a stick shift. The first lesson introduces the clutch and stick shift to get the driver comfortable with their operation while the car is stopped. This lesson also teaches the user how to start moving forward and stopping the car without stalling. In contrast, the second lesson introduces making the transition between first and second gears. Once these concepts are understood, all other gear shifts are essentially the same. Finally, the third lesson is the continuous monitoring mode which only gives shifting guidelines without further explanation.

The driver can replay any lessons at any time to encourage repeated practice and promote mastery of each essential skill. As this is meant to be an instructional device and used in an isolated area by the driver away from traffic or pedestrians, the device only focuses on low-speed driving in the first two gears. We expect the driver to master the basics first and then apply the same skills to learn at higher speeds and in live traffic by taking lessons with a licensed instructor or on their own at their own risk.

4. Cost and Schedule

4.1 Cost of Parts

Table 1: Part Cost Breakdown

Quantity	Part Name	Part Number	Cost
1	Audio Jack	SJ1-3525N	$0.76
1	CAN Transceiver	TL2575HV-33IN	$1.20
1	MP3 Mini Player Board	DFR0299	$8.90
1	USB to UART Converter	MCP2221A	$2.50
1	Microcontroller	ESP32-WROOM32	$8.95
1	DB9 Connector Pin	PRT-00429	$1.75
1	Mini USB Socket	1734035-3	$1.24
1	Buck Regulator	TL2575HV-33	$1.94
1	Slide Switch	CS12ANW03	$2.25
1	Button Breakout Board	KONA-01	$3.98
1	Header for Input Buttons	FTSH-104-01-F-DV	$2.02
2	Push Button	TL3305AF260QG	$0.18
2	Resistor (47 Ohm)	RCG080547R0JNEA	$0.50
1	Resistor (10K Ohm)	CR0805-FX-1002ELF	$0.10
2	Capacitor (4.7 uF)	CC0805MKX5R6BB475	$0.25
4	Capacitor (0.1 uF)	C0805C104K5RACTU	$0.15
1	Capacitor (10 uF)	CC0805ZKY5V6BB106	$0.22
1	Capacitor (0.01 uF)	C0805C103K5RACTU	$1.21
1	Capacitor (0.47 uF)	CC0805ZRY5V8BB474	$0.20
1	Schottky Diode (20 V, 1 A)	SB120	$0.43
1	Inductor (330 uH, 1.6 A)	5900-331-RC	$2.21
3	Ceramic Capacitor (100 uF, 16 V)	EMK325ABJ107MM-P	$1.87
		Total Cost	$47.93

4.2 Labor Cost Breakdown

Table 2: Labor Cost Breakdown

Team Member	Hourly Wage	Weekly Hours	Number of Weeks	Multiplier	Cost Per Member
Aadhar	$38.00	20	12	2.5	$22,800
Ian	$38.00	20	12	2.5	$22,800
Maulin	$38.00	20	12	2.5	$22,800
				Total Labor Cost	$68,400

Our hourly wage is calculated based on the average salary for a newly graduate Electrical Engineer. The cost per member is calculated using the following formula:

$$Hourly\ Wage * Weekly\ Hours * Number\ of\ Weeks * Multiplier = Cost\ per\ Member$$

4.3 Schedule

Table 3: Schedule

	Aadhar	Ian	Maulin
2/22	Block Diagram, Design Document	Schematics, Design Document	R&V Tables, Design Document
3/1	Finalize Lesson Plans, start ordering Parts	Finalize Circuit Schematics, start ordering Parts	Finalize R&V Tables, start ordering Parts
3/8	Setup coding environment for ESP32 - using CAN API (ECU Module)	Create PCB Layout	Create PCB Layout
3/15	Begin writing software for ESP32 - interfacing with Audio Module	Finalize PCB Layout after passing Audit	Finalize PCB Layout after passing Audit
3/22	Begin writing software for ESP32 - Lesson Plans	Begin soldering components on the board	Begin soldering components on the board
3/29	Verify Audio playback through vehicle speakers	Verify Power distribution	Verify Power distribution
4/5	Debug and Test	Debug and Test	Debug and Test
4/12	Debug and Test	Debug and Test	Debug and Test
4/19	Prepare for Mock Demo, start writing final paper/teamwork evaluation	Prepare for Mock Demo, start writing final paper/teamwork evaluation	Prepare for Mock Demo, start writing final paper/teamwork evaluation
4/26	Demonstration and Presentation, continue working on final paper/teamwork evaluation	Demonstration and Presentation, continue working on final paper/teamwork evaluation	Demonstration and Presentation, continue working on final paper/teamwork evaluation
5/3	Lab Checkout, finalize Lab Notebook and Teamwork Evaluation	Lab Checkout, finalize Lab Notebook and Teamwork Evaluation	Lab Checkout, finalize Lab Notebook and Teamwork Evaluation

5. Conclusion

5.1 Accomplishments

Our project successfully achieved the high-level requirements that we had set in place before starting the project. We could access the CAN bus in a vehicle and request data such as vehicle speed, engine speed, and throttle position. Utilizing the microcontroller and the USB to UART converter, we successfully programmed our microcontroller. We processed the data that was being received from the CAN bus to implement our driving lessons. We used the DFPlayer Mini MP3 player to play audio which was vital in providing instructions successfully. All in all, we are able to incorporate all design elements and develop a working device that met all our requirements described in further detail in Appendix A.

5.2 Ethical Considerations and Safety Hazards

When creating an educational tool, it is imperative to instruct the learner in a safe manner. Teaching someone how to drive prompts many factors that can harm the public. The IEEE Code of Ethics states "to disclose promptly factors that might endanger the public or the environment" [7]. In response to this, we included a safety warning system that discloses all the pertinent information before even instructing the driver. Furthermore, the Illinois 2020 Rules of the Road "prohibits the use of handheld cell phones, texting or using other electronic communications while operating a motor vehicle" [8]. This rule is something we expect anyone who uses this assistant to be aware of. We anticipate that anyone who uses this tool will have a license and has some information about the rules and regulations of driving on the road. In addition, to deter distracted driving, we have prevented the use of input buttons while the vehicle is moving. These button presses allow the user to choose a lesson and are only operational when the vehicle is stopped, ensuring the driver's safety.

5.3 Further Work

Future work has great potential regarding this project in the following areas: functionality, upgrades, and compatibility. An important feature to add to this project is the feedback loop to verify that the driver has shifted into the correct gear positions and pressed down on the clutch. This gives the driver the verification that the correct steps have been taken to drive a manual car. Another improvement to the project is upgrading the lesson plans to go more in-depth about certain aspects such as stalling. This project can also enhance its compatibility to include support for more vehicle models by implementing various other ECU communication protocols. All these improvements and modifications will result in a more well-rounded product.

6. References

[1] T. Hofstätter, M. Krawina, B. Mühlreiter, S. Pöhler, and A. Tschiesner, “Reimagining the auto industry's future: It's now or never,” *McKinsey & Company*, 06-Nov-2020. [Online]. Available: https://www.mckinsey.com/industries/automotive-and- assembly/our-insights/reimagining-the-auto-industry's-future-it's-now-or-never. [Accessed: 20-Feb-2021].

[2] J. M. Vincent, “What You Need to Know About Coronavirus and Cars,” *U.S. News & World Report*, 14-Dec-2020. [Online]. Available: https://cars.usnews.com/cars-trucks/coronavirus-and-cars. [Accessed: 10-Feb-2021].

[3] Autolist Editorial, “Manual vs Automatic - Pros and Cons,” *Autolist*, 29-Jul-2019. [Online]. Available: https://www.autolist.com/guides/manual-vs-automatic. [Accessed: 10-Feb-2021].

[4] S. Gautam, R. Pansare, A. Chaudhary, and K. Gupta, “Why Does Europe Prefer Manual Cars Over Automatic Ones?” *Get My Parking Blog*, 17-Feb-2020. [Online]. Available: https://blog.getmyparking.com/2020/01/20/why-does-europe-prefer-manual-cars-over-automatic-ones/. [Accessed: 10-Feb-2021].

[5] Texas Instruments, “TL2575, TL2575HV 1-A Simple Step-Down Switching Voltage Regulators,” TL2575HV-33IN datasheet, (Revised Nov. 2004). [Accessed 30-Mar-2021]

[6] “CAN Transceivers,” *Maxim Integrated*. [Online]. Available: https://www.maximintegrated.com/en/products/interface/transceivers/controller-area-network-transceivers.html#: ~:text=A controller area network (CAN, loading down the system microcontroller. [Accessed: 25-Feb-2021].

[7] “IEEE Code of Ethics,” *IEEE*. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 12-Feb-2021].

[8] Publications/Forms. [Online]. Available: \https://www.cyberdriveillinois.com/publications/. [Accessed: 20-Feb-2021].

[9] “ESP32-WROOM-32,” *Espressif Systems*. [Online[. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf (Revised 2021). [Accessed 30-Mar-2021]

Appendix A - Requirements and Verification Tables

1.1 Power Module

Table 1: Requirements and Verification for Buck Regulator

Requirement	Verification
1. Accept an input voltage (VIN) in the range: 4.7 < VIN ≤ 14.5 V. 2. Must operate in stable conditions in temperatures between -15 °F and 100 °F. 3. Must output a maximum of 1.0 A to avoid burning out the Buck Regulator.	1. Using a Digital Multimeter, we probed the VIN pin of the buck regulator to ensure that the voltage is in the range of 4.7 V to 14.5 V when the car is on. 2. We did not have access to an IR thermometer, so we did touch tests and verified that the module never got noticeably warm. 3. Using a Digital Multimeter and electronic load machine, we probed the VOUT pin to ensure a maximum of 1.0 A output was possible.

1.2 ECU (Engine Control Unit) Interface Module

Table 2: Requirements and Verification for CAN Transceiver

Requirement	Verification
1. Must communicate with an ECU at a baud rate of 500Kbit/s 2. Must have an IO voltage level of 3.3 V +/- 5%.	1. Verified via simulator and on an actual car, we could send and receive meaningful messages to and from the ECU. 2. Probed all output pins with a DMM and verified voltages were within specification.

1.3 Control Module

Table 3: Requirements and Verification for Microcontroller

Requirement	Verification
1. Must have an input voltage (VDD) of 3.3 V +/- 5%. 2. Must be operational in temperatures between -15 °F and 100 °F. 3. Must be able to process ECU data and output a command to the MP3 player under 5 ms.	1. We used a DMM to verify that the microcontroller was receiving within 5% of 3.3 V on VIN. 2. We did not have access to an IR thermometer, so we did touch tests and verified that the module never got noticeably warm. 3. Used timestamps to verify that data processing took well under 5ms.

Table 4: Requirements and Verification for Input Buttons

Requirement	Verification
1. Input Buttons must be debounced to prevent processing multiple presses. 2. While vehicle speed is greater than 0 MPH, changes in the value of input buttons will be ignored.	1. When a button value changes from digital low (0) to digital high (1) we started a short, predefined timer. Once the timer ended, we sampled the button value again. If the button's value has returned to digital low, we recorded a button press. Otherwise, it was ignored, successfully debouncing the button. 2. While the vehicle was moving, a passenger tried changing the current lesson via the buttons but could not do so.

1.4 Program and Debug Module

Table 5: Requirements and Verification for USB to UART Converter

Requirement	Verification
1. LEDs must light up when data is being transmitted/received. 2. Data must be transmitted through a USB cable to the MCU.	1. Verified by acknowledging the LEDs light up on the board during programming. 2. Verified by connecting an LED and resistor between the GPIO port and ground. We then sent the GPIO_SET_LEVEL command from the ESP32's GPIO.h library to toggle the LED.

1.5 Audio Module

Table 6: Requirements and Verification for Mini MP3 Player

Requirement	Verification
1. Must have an input voltage (VDD) of 3.3 V +/- 5%. 2. Must be operational in temperatures between -15 °F and 100 °F. 3. Must have a stereo audio output for the auxiliary cable. Driver must be able to listen to audio cues at a reasonable volume.	1. Used a DMM to verify that the MP3 player was receiving within 5% of 3.3 V. 2. We did not have access to an IR thermometer, so we did touch tests and verified that the module never got noticeably warm. 3. We played audio instructions during live testing in a car and were able to hear all instructions clearly.

Appendix B - Miscellaneous Diagrams and Pictures

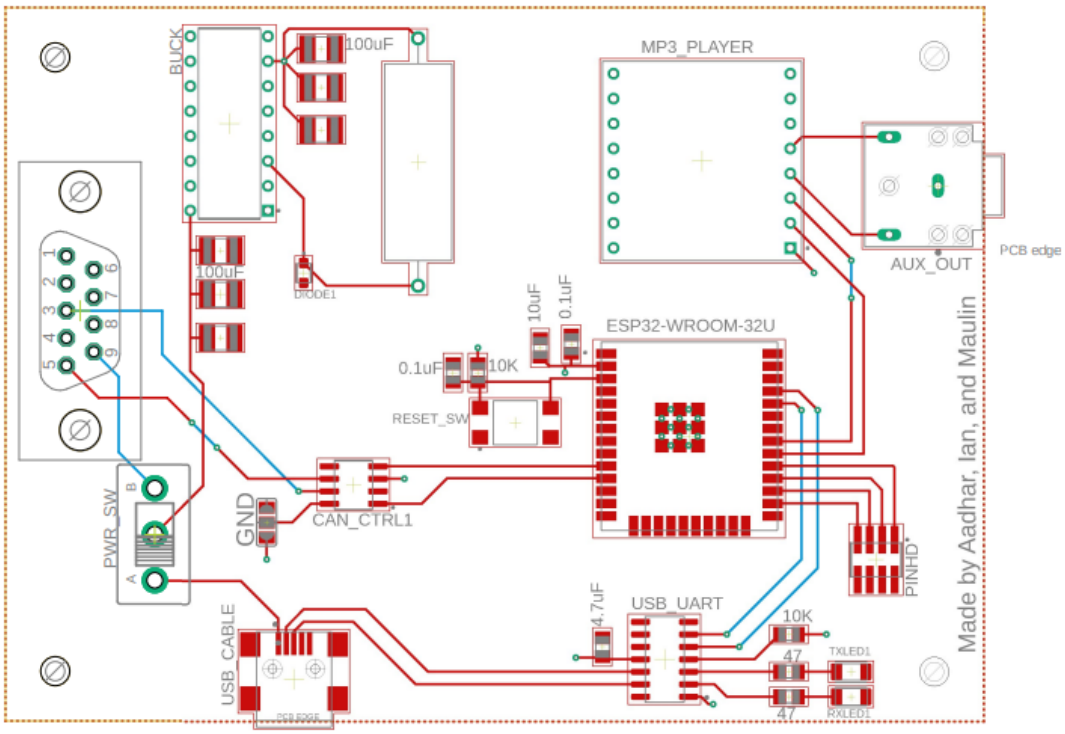

Figure 1: PCB Layout

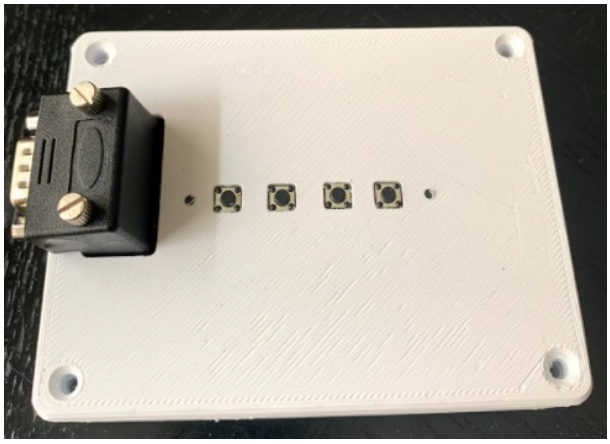

Figure 2: Final Design