

# ***Audio Kit for the ESP32-A1S***

*John Talbert - July 2021*



# Table of Contents

<b><i>Audio Kit for the ESP32-A1S .....</i></b>	<b>1</b>
The ESP32	3
ESP32 Help	3
ESP32-A1S	3
Project Goals	5
Board Connections Diagram	7, 8
Slide Pots	9
Switches	9
SD Card	10
MIDI	11
Amp Outs	12
Headphones	13
Key AD	14
Mini Display	15
LEDs	15
Boot/Reset	16
Microphone	17
Board Headers	18
Conclusions	22

# The ESP32

The ESP32 microprocessor is like a supercharged Arduino. It has a 32-bit processor, like the Arduino, and can be programmed with the Arduino IDE app, compatible with over 90% of the Arduino core programming language and many of its libraries.

However, it has expanded capabilities such as wireless WiFi and Bluetooth. Its clock speed is 80MHz/240MHz compared with 48MHz for the MKR. The flash memory for user programs is 4MB/8MB compared to 0.256MB for the MKR. The SRAM memory for user variables is 520KB compared to 32KB for the MKR. The ESP32 processor is dual-core enabling it to run two program threads simultaneously. Its peripherals can include up to 43 GPIOs, 1 full-speed USB OTG interface, SPI, I2S, UART, I2C, LED PWM, LCD interface, camera interface, ADC, DAC, touch sensor.

The programming language for the ESP32 has been expanded to include many useful features that were only accessible in the Arduino by hacking into its internal registers. This includes PWM setup with pulsewidth times and clock speeds, timers with clock setup, mode and interrupt, touch (capacitive) sensor setups, ADC setups with resolution, width and speed.

## ESP32 Help

Given its expanded capabilities, programming the ESP32 can be difficult even given the Arduino's familiar IDE environment. Here are several sources for tutorials and books on the ESP32.

**Espressif**      <https://www.espressif.com/en/products/modules>    ESP32 Developer

**Tech Explorations**    <https://techexplorations.com/pc/esp32/>    Video Tutorials

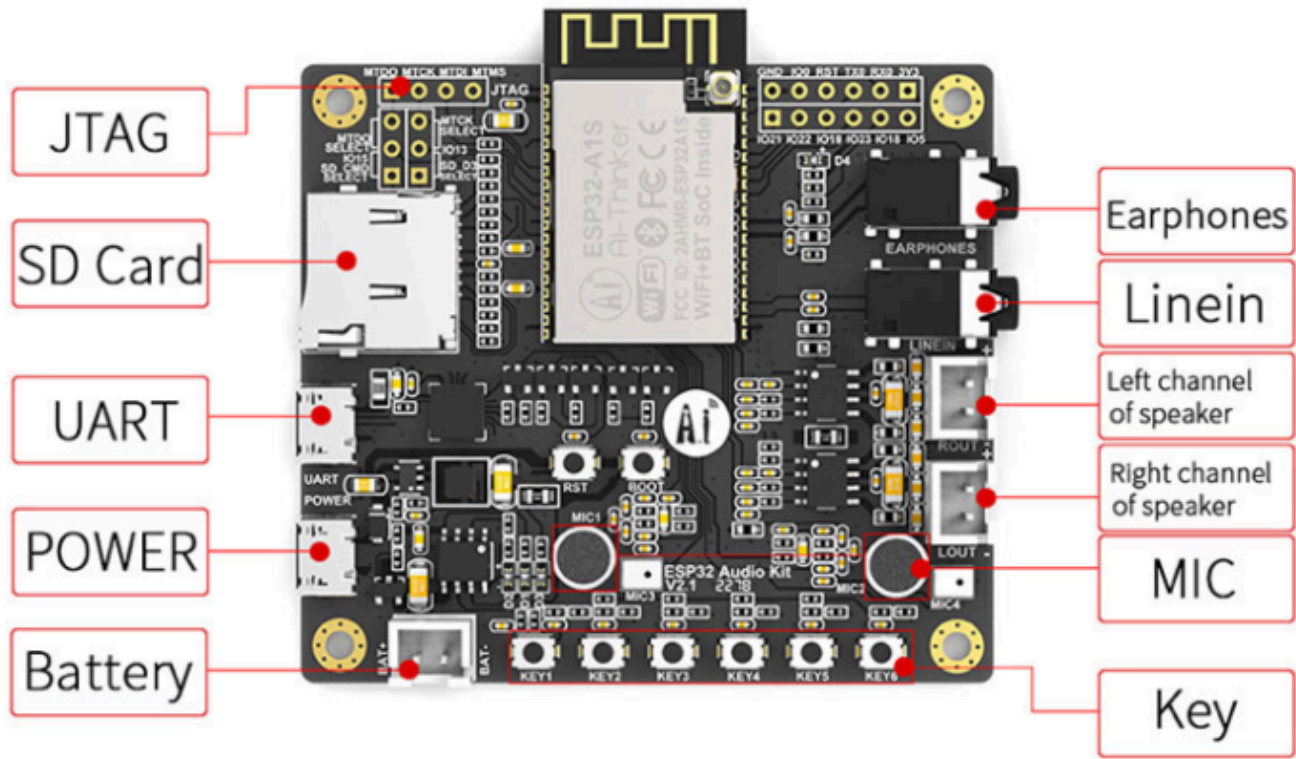
**Random Nerd**      <https://randomnerdtutorials.com/projects-esp32/>    Tutorials, Books

**Books**      <https://bookauthority.org/books/best-esp32-books>

## ESP32-A1S

Since its release in 2016 by Espressif Systems, the ESP32 has experienced an explosion of versions - development boards, boards with cameras, touch screens, antenna, programmable memory, SD cards.

The ESP32 used in this project is special for its high fidelity audio capabilities. The ESP32-A1S ([https://docs.ai-thinker.com/\\_media/esp32-a1s\\_v2.3\\_specification.pdf](https://docs.ai-thinker.com/_media/esp32-a1s_v2.3_specification.pdf)), made by Ai-Thinker, contains an extra AC101 audio codec IC whose IO-pins (line, mic, etc.) are led to the board pins. It comes separately or soldered onto a corresponding audio development board ("ESP32-Audio-Kit" shown below). It comes with 8MB of flash memory for programs and 4MB of SRAM for user variables (especially useful in delay line and reverb processes).

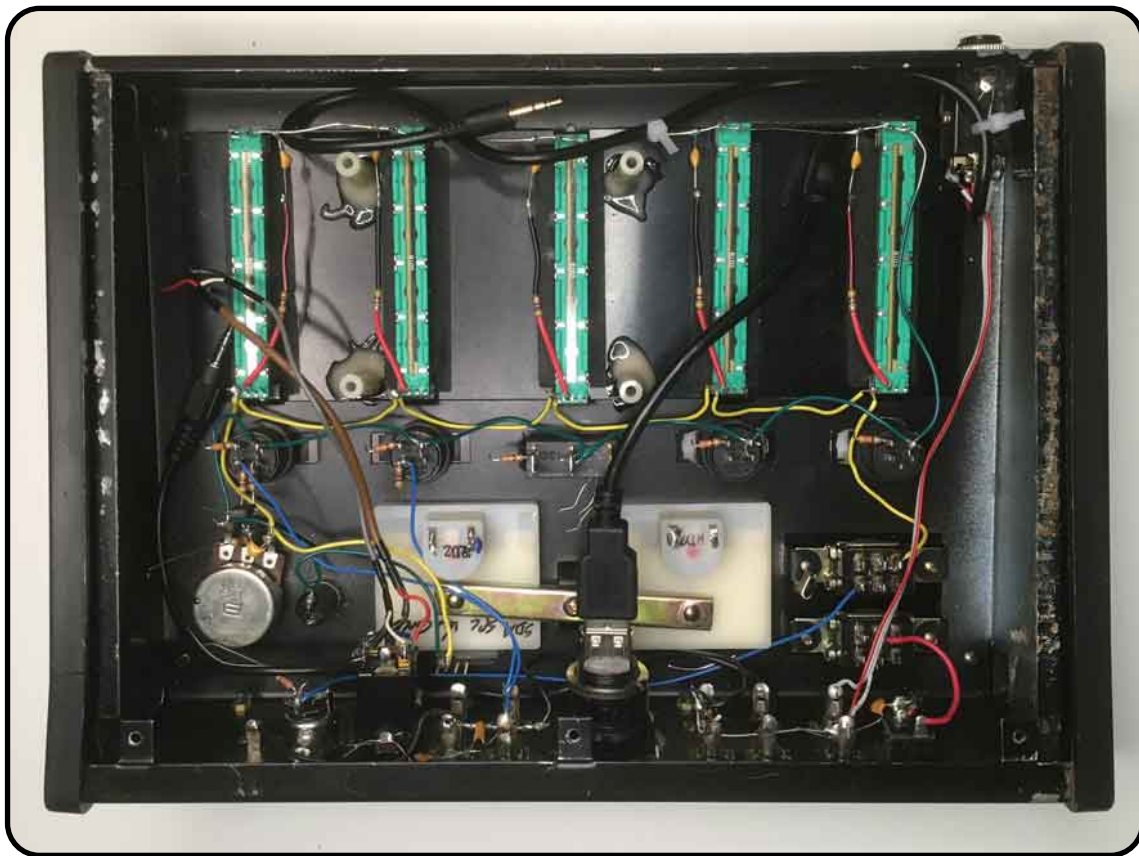


The ESP32-A1S is used by the Deeptronics organization (Free Electronic Circuit Design) for their Blackstomp audio effects processor platform (<https://www.deeptronic.com/blackstomp/>). The circuit boards for their project have been out of production for a while now, but the full software library from the project is readily available with some documentation and full documentation "coming soon".

## Project Goals

While the Deeptronic boards for the Blackstomp audio effects box are unavailable, this project will attempt to adapt the Ai-Thinker ESP32-Audio-Kit board to the Blackstomp design so that the Blackstomp Library can be used to create an audio effects unit. Here are the instructions for installing the Blackstomp Library onto an Arduino IDE platform for use with any ESP32-A1S processor.

*After finished with the board installation for ESP32, we can proceed to the Blackstomp library installation: Download Blackstomp library from <https://github.com/hamuro80/blackstomp> , extract the zip file after download. Make sure you can find the library folder "Blackstomp" inside the "arduino-library" folder. On Arduino IDE, go to menu Sketch > Include Library > Add .ZIP Library, and navigate to the extracted library folder "Blackstomp" [...]*



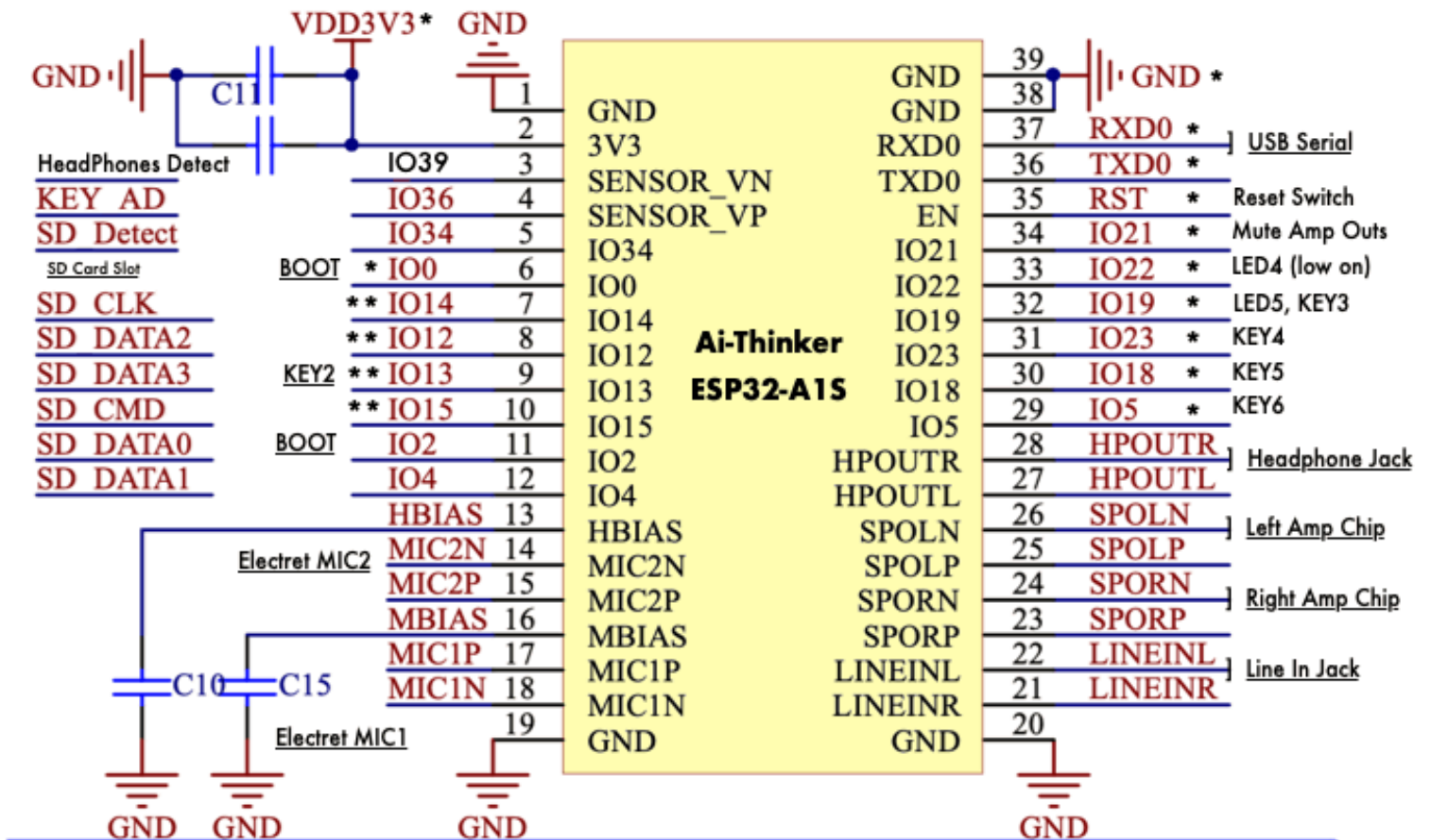
An old Realistic mixer box is used to house the Audio Kit. It has accommodations for 5 slide pots, 5 switches, one rotary pot, and two display/meters, along with a back panel of input and output jacks. Most of these mixer devices will be connected to pins on the ESP32-A1S to be programmed as controllers for the audio effects programs





# **Ai-Thinker** **ESP32-Audio-Kit** **Board Connections**

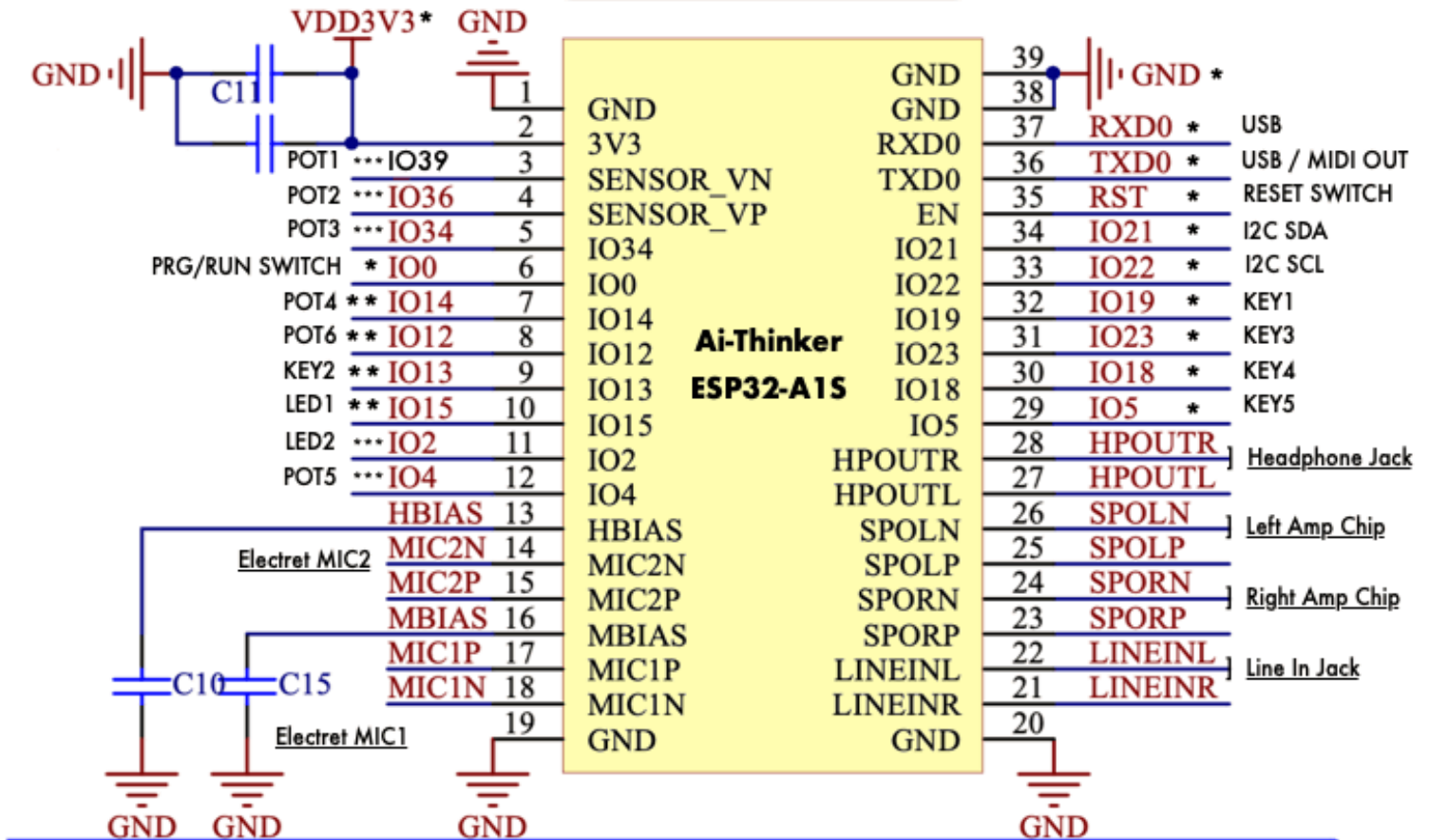
\* 14 pin connector  
 \*\* 4 pin connector



\*\*\* added to 4 pin connector

**Ai-Thinker ESP32-Audio-Kit  
to  
Blackstomp Pedal**

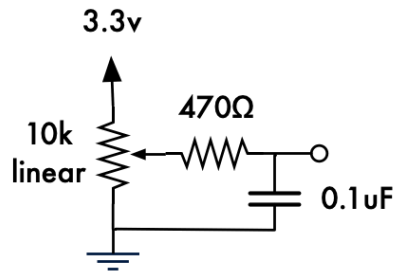
\* 14 pin connector  
\*\* 4 pin connector





## Slide Pots

The Blackstomp Pedal project has 6 potentiometer controllers whereas the ESP32-Audio-Kit board has no provisions for controllers except for a bank of 6 switches. Pins IO39, IO36, IO34, IO14, IO12 and IO4 will each be redirected to a potentiometer circuit as shown below. The only change from the Blackstomp pot/pin assignments is IO12; Blackstomp uses IO13 instead.



The controller pots must have a linear taper. The original slide pots in the Realistic Mixer box were all volume controls with Log tapers and thus needed to be replaced. The 10k resistance value shown above can be replaced with any value as long as it is linear taper. The main problem I had was finding the right length pot to fit this mixer.

The 470Ω resistor is there for safety. If the ESP32 pin is accidentally defined as an output instead of input, its voltage could be shorted out to the ground or 3.3volts connections on the pot. The 470Ω resistor will limit the current to a safe amount if that happens. It should have a negligible effect on the control voltage readings.

Note the ESP32 ADC pot readings will not be fully linear. There is a small range at the top and bottom of the slider movement that will not exhibit any change in value from its maximum or minimum.

## Switches

The ESP32-Audio-Kit has 6 switches mounted on the board. Five of them are connected to pins IO13, IO19, IO23, IO18, and IO5. I have used these five pins to connect to 5 switches on the Realistic Mixer.

Each has a  $330\Omega$  resistor in series with the pin connection. As with the pots it acts a current limiting safety. If a pin is accidentally defined as an output, it prevents the switch from short circuiting the pin to ground.

IO13 was given a 10k pullup resistor to 3.3v to make it "normally" high when the switch is in its normal unpressed position. The other switch pins can use internal pullups by using the "INPUT\_PULLUP" designation in the Arduino software pinMode function.

The board mounted pushbuttons are also ground connecting SPST switches and will not be user accessible. They are fine as is, acting in parallel with the mixer mounted switches.

The Blackstomp Pedal had only one pushbutton switch on IO5. It used an Encoder Switch on IO19, IO23, and IO18 and it used IO13 for a pot, not a switch.

## **SD Card**

Seven pins on the ESP32-Audio-Kit board are dedicated to the SD Card Reader. These pins have been reassigned to pots, switches and LEDs. Actually, these new connections might not prevent the use of the SD as a secondary function though I haven't tested this (I would center the pot positions when trying this).

The SD card slot on the board has no electronics. It is only a carrier that makes connections between an SD card and the 7 ESP32 pins. The SD card has all the data circuitry.

The SD card slot can easily be pried off the board with a screwdriver. Though not necessary, this does uncover some solder pads making it easier to access 3 of the ESP32 pins used for pots. I try not to make solder connections directly on the ESP32 pins to avoid possible heat damage to the ESP32 pins and pads.

One of those pins is SD Detect on IO34. I suspect that this pin has a pullup resistor making it normally high in voltage. When an SD card is inserted, a tab on the slot grounds IO34 making it read zero volts. IO34 has been reassigned to a slide pot.

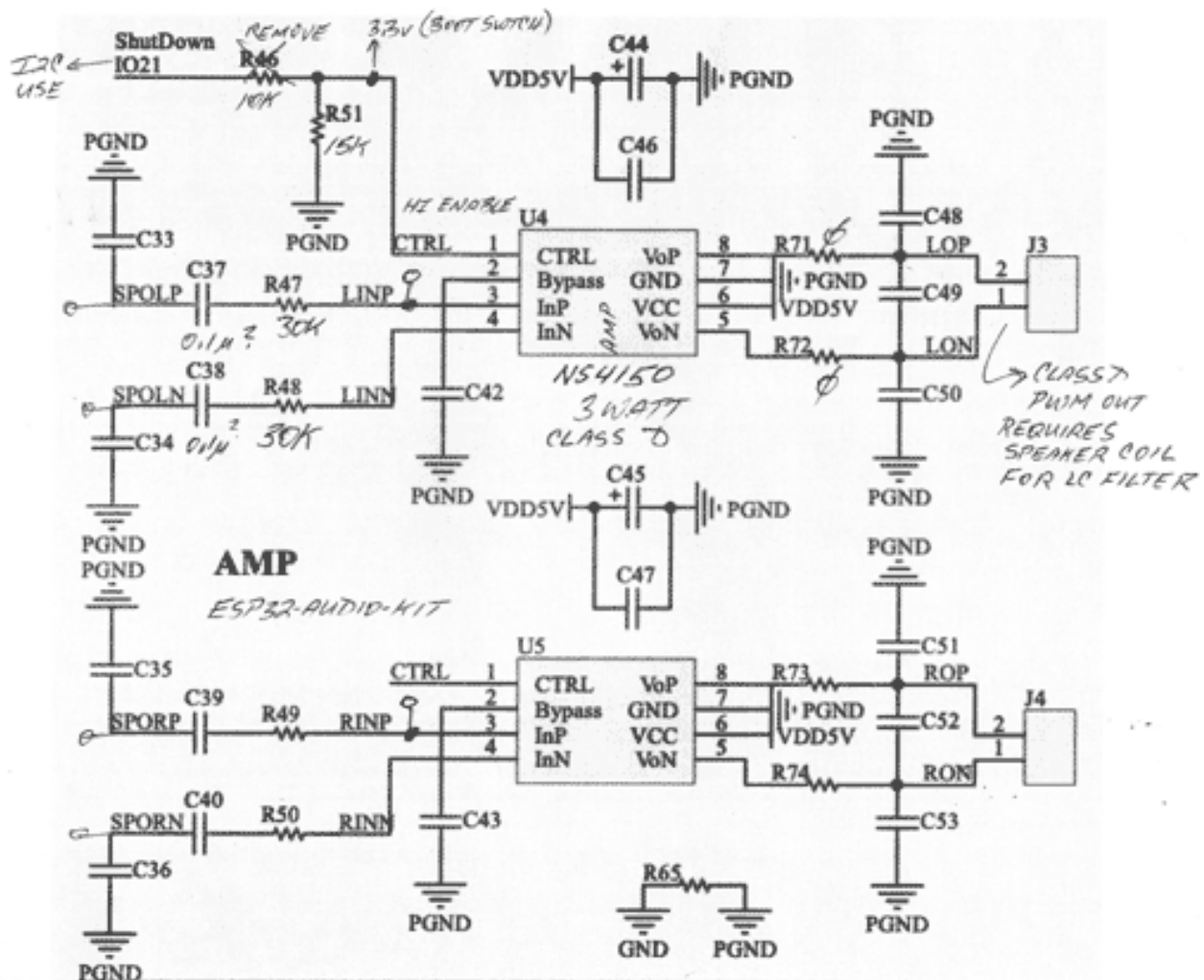
## MIDI

The ESP32 processor has RXD0 and TXD0 pins, serial data receive and Transmit pins. The ESP32-Audio-Kit includes a USB interface that uses those two serial pins. When not used for programming, both RXD0 and TXD0 can easily be used for standard MIDI on 5-pin DIN sockets. Both pins are connected to the DPDT Power On/Off switch on the Realistic Mixer. In that way, any connected MIDI instruments can be disconnected while the board is being programmed (though you could just as easily disconnect the MIDI devices from the MIDI sockets).

MIDI In requires a little extra circuitry in the form of an opto-isolator which was not implemented. MIDI Out is simpler, requiring only two connections, a 220 $\Omega$  resistor to 3.3v connected to the #4 MIDI Out DIN pin and TXD0 connected to #5 MIDI Out DIN pin (through the On/Off switch).

## Amp Outs

The ESP32-Audio-Kit board includes two NS4150 3-Watt chip amplifiers. These are Class D amplifiers with Pulse Width Modulated (PWM) outputs that need to be connected to a speaker coil to form an LC filter to filter out the output pulses. The two onboard white speaker connectors cannot be used as line outputs; they must be connected to speakers if used.

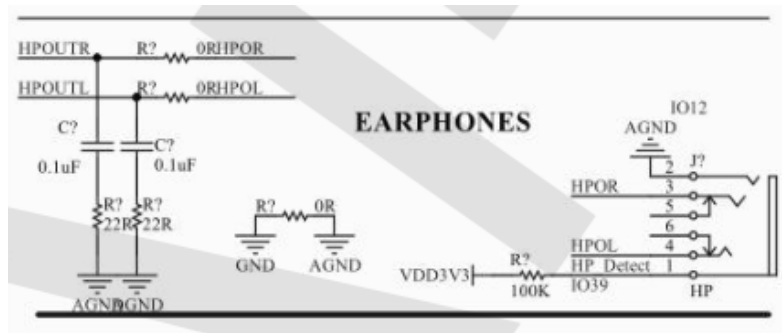


Inputs to the amps come from the ESP32 Audio Output pins SPOLP, SPOLN, SPORP, SPORN. These 4 audio outputs are sent through a 0.1µF blocking capacitor and a 30k series resistor before connecting to the amp chip inputs as LINP, LINN, RINP, RINN. I have connected LINP and RINP to the Phono connectors labeled TAPE OUT on the back of the Realistic Mixer. The series 30K resistor will protect the ESP32 pin outs.

The two NS4150 amp chips have a MUTE control labeled CTRL which is connected to IO21 and labeled ShutDown. The chips outputs are "ShutDown" by a low on IO21. Since IO21 was needed to help drive an I2C Mini Display (described later) it was disconnected from the amp CTRL lines by removing the 10k resistor R46. This is a tiny surface mounted resistor that will easily slide off its solder pads if heated enough. The two CTRL lines (either of the chip's pin 1 at the dot) can then be connected to 3.3v to permanently enable the amp outputs.

## Headphones

The Headphones output on the ESP32-Audio-Kit is an actual stereo mini phone jack mounted on the board. I used a mini phone cable to connect it to the Headphone jack on the front of the Realistic Mixer. This is also connected to the back panel stereo phono jacks labeled OUTPUT.



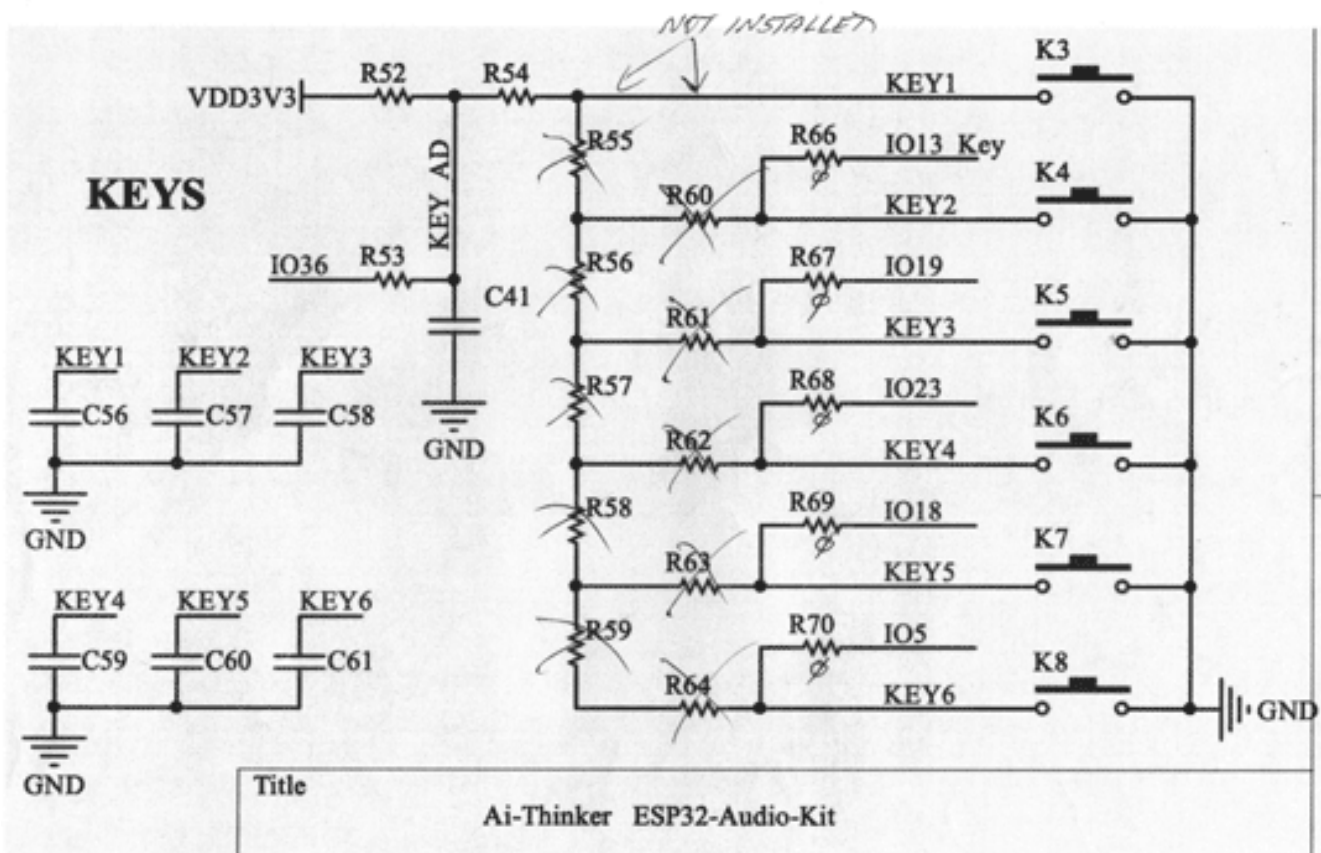
The ESP32-Audio-Kit uses pin IO39 to detect when headphones are connected to the mounted headphones jack. IO39 is connected to the sleeve tab of the headphones jack. It is also connected to a pullup resistor to 3.3v making it normally high at 3.3v. When a headphones plug is inserted in the jack the sleeve tab makes a connection with ground and the IO39 will go low to zero volts.

Pin IO39 is needed for a slide pot controller so its Headphone Detect function must be disabled. This is accomplished easily by simply bending up the Sleeve Tab on the top of the board's headphone jack so that it will no longer make contact with the sleeve of an inserted headphone plug. The Sleeve Tab is also a convenient place to solder a connector wire to go to a slide pot.



## Key AD

The 6 pushbutton switches on the ESP320-Audio-Kit board can be connected to a resistor divider network and read as 6 different voltage levels from one pin, IO36. The circuit is shown below. On my board the resistor divider was not actually installed, though the pushbuttons were there and connected to IO pins.



IO36 is needed for a slider controller. To disable the KEY AD function remove the three middle components (R52, R53, R54) from the 5 components above KEY1. These are tiny surface mounted resistors that will easily slide off their solder pads if heated enough. The top pad of the middle component is a good place to solder a IO36 wire.

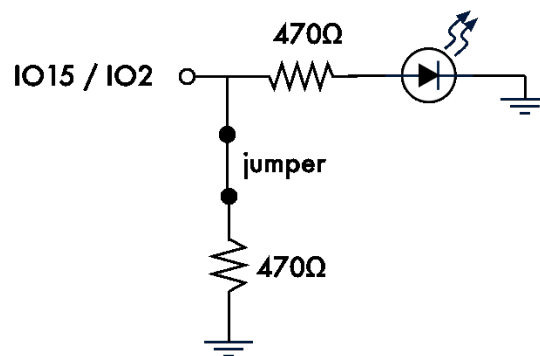
## Mini Display

A HiLetgo 128x64 OLED LCD Display was added to the project (Model SSD1306). This 0.96" display fits perfectly inside one of the two VU Meter casings on the Realistic Mixer. Its four pins, projecting from the bottom of the meter casing, are connected to Ground, 3.3volts, IO21, and IO22. IO21 is the SDA line of the ESP32's I2C serial interface, and IO22 is the SCL line. Both IO21 and IO22 need 10k pullup resistors to 3.3volts (added at pin header).

A "Random Nerd" tutorial (<https://randomnerdtutorials.com/esp32-ssd1306-oled-display-arduino-ide/>) "ESP32 OLED Display with Arduino IDE" is the perfect guide to setting up and programming this display. In this tutorial they use two Adafruit libraries: [Adafruit\\_SSD1306 library](#) and [Adafruit\\_GFX library](#).

## LEDs

Two LEDs were added to the project, using IO15 and IO2, the same LED setup used in the Blackstone Pedal project. The two LEDs were actually part of the pushbutton switches mounted above the first two slider pots. The LED circuits are shown below.



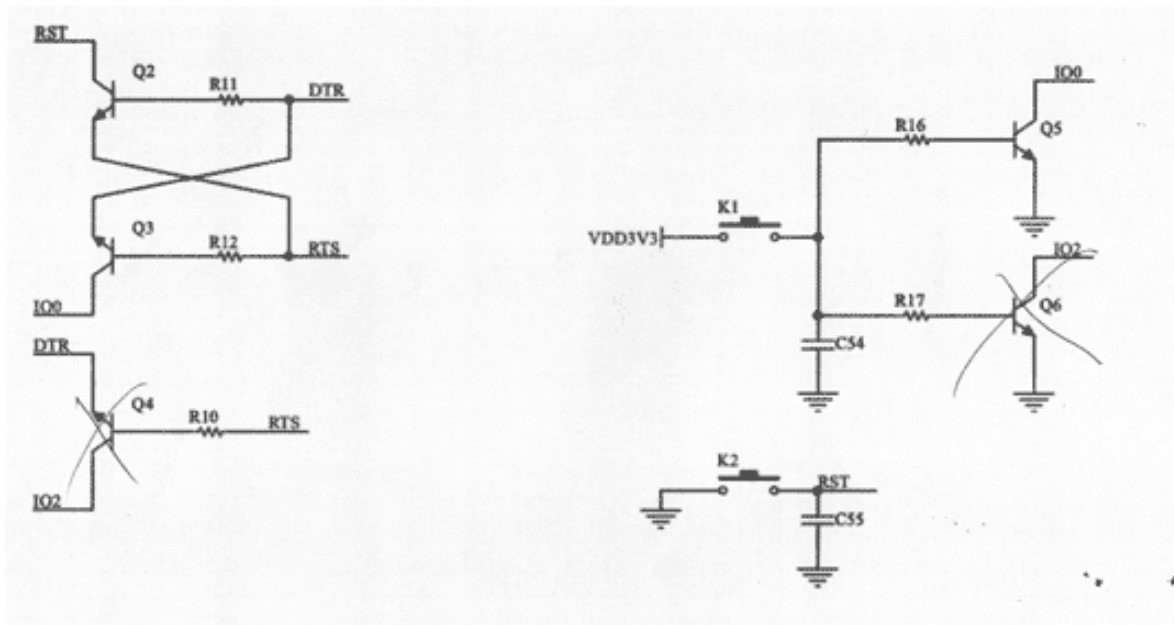
A high on the IO line will light up the LED. The jumpered 470Ω will be discussed in the next section.

IO22 drives the onboard LED4 and IO19 drives the onboard LED5. LED4 and LED5 can be removed since they are not visible inside the Realistic Mixer, and since IO22 and IO19 have other functions now.

## Boot/Reset

Many ESP32 boards have two functional pushbuttons. The RESET switch grounds the RST pin to restart the processor, a similar effect to removing the power and reconnecting. The ESP32-Audio-Kit board has a RESET switch. A second one in parallel with the mounted one was installed to the left of the display on the Realistic Mixer.

The BOOT switch stops the current program and puts the processor in "boot" mode so that it can load a new program. During an Arduino IDE program load, the BOOT switch is pushed after the load has finished compiling the new program and is ready to load it. In effect, it grounds the IO0 pin. The ESP32-Audio-Kit has a pushbutton (K1 in the circuit diagram below) that actually grounds both IO0 and IO2 through common emitter transistors when pushed. When not pressed they are both open collector floating IO connections. A parallel boot switch was installed on the "MODE" switch, top right corner of the Realistic Mixer.



IO2 was assigned the job of driving an LED. Not willing to give up its LED function and also not willing to risk it being shorted to Ground at any point while lighting up its LED, I removed transistors Q6 and Q4. In their place, IO2 is jumpered to Ground

through a 470Ω resistor so that it will always look like it's at zero volts during Reset or Boot.

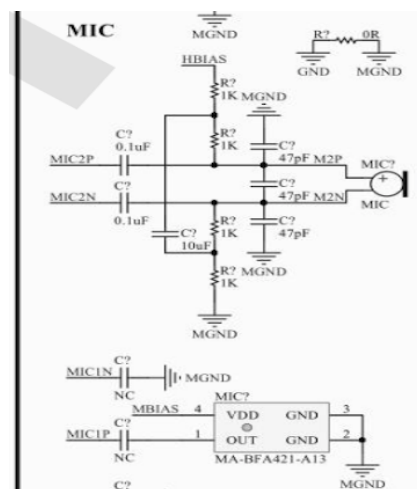
During Reset and Boot the ESP32 turns a number of pins into Inputs and then examines their levels for decisions I don't quite understand. IO0, IO2, IO15 and IO4 are some of the main pins used in this process. After getting rid of Q6 and Q4 affecting IO2 I discovered the following settings that seem to work for both Boot and Reset.

1. Ground IO2 through a jumpered  $470\Omega$  resistor
2. Don't use the grounding jumper on IO15
3. IO0 is left alone. The BOOT switch is not needed
4. Make sure Pot6 on IO4 is at its zero setting.

With these settings the ESP32 Resets without problem and Loads new programs from the Arduino IDE without any manipulation of the BOOT switch.

# Microphone

The ESP32-Audio-Kit has two Electret Microphones mounted on the board with their biasing circuitry. Mic1 was removed in order to add an external Mic attached to the Mic3 pad located next to Mic1.



The Mic3 pad does not appear to have the electret mic biasing and thus might work with an external dynamic mic using pins 1 and 2 (OUT and GND).

## Board Headers

Removing components on the ESP32-Audio-Kit is fairly easy. They are all tiny surface mounted components that will easily slide off their solder pads if heated enough. However, soldering anything to the board pads can be very difficult. I used 30 awg gauge, single strand, wire wrapping wire to solder to the board and then hot glued the wire to the board to prevent any movement at the solder joint.

Luckily, the board includes two pin headers (2.54mm pitch) that facilitate most of the off board connections.

The 14 pin header (two rows of 7 pins) was easily accessed with a 14-wire female ribbon connector. Here is a list of the 14 ribbon connections made between the ESP32 board and the enclosure devices.

1. 3.3 volts	8. IO23 -- KEY3
2. GND	9. RST -- Reset Switch
3. 3.3 volts	10. IO19 -- KEY1
4. IO5 -- KEY5	11. IO0
5. RX0 -- (MIDI In)	12. IO22 -- Display I2C SCL
6. IO18 -- KEY4	13. GND
7. TX0 -- MIDI Out	14. IO21 -- Display I2C SDA

A 4 pin header on the board is connected to 5 small chip switches. When switches 1, 4, and 5 are set to "ON" the 4 pin header carries IO15, IO13, IO12, and IO14.

Six additional connections to the board are soldered using wire wrapping wire. One side is soldered to a board connection and the other side is wire wrapped/soldered to a 7-pin male to male header. All six wires are hot glued to the board surface.

IO39 is soldered from a tab on the Headphones jack.

IO36 is soldered from a pad above KEY1

IO34 is soldered from the top pad found under the SD card carrier

IO4 is soldered from a second pad found under the SD card carrier

IO2 is soldered from a third pad found under the SD card carrier

Boot Switch wire is soldered from the top right of the board's boot switch











A second 14 wire ribbon cable connects to the board's 4-pin header. The 7-pin male to male header with the 6 wired board connections is inserted into the other side of the ribbon cable connector (as shown above). In effect, the 4-pin header is expanded with this 7-pin male to male header to a 14 pin header. The ribbon connectors are shown below.

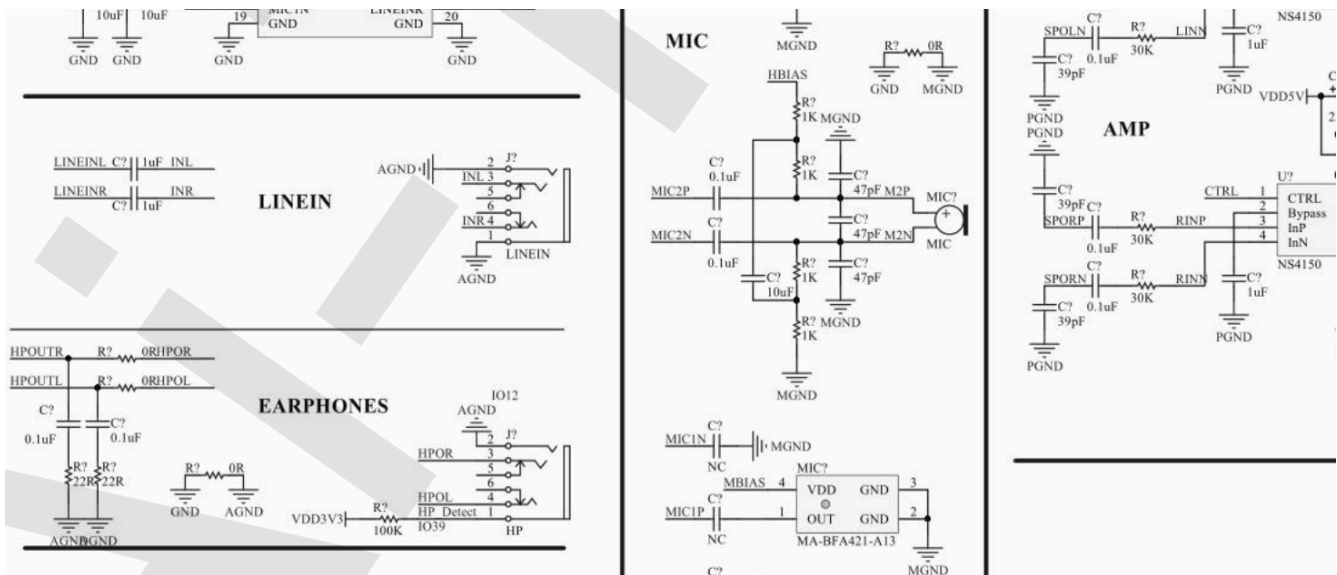
1. *IO15 -- LED1	8. IO4 -- POT5
2. IO39 -- POT1	9. --
3. *IO13 -- KEY2	10. IO2 -- LED2
4. IO36 -- POT2	11. --
5. *IO12 -- POT6	12. Boot Switch
6. IO34 -- POT3	13. --
7. *IO14 -- POT4	14. --

# Conclusions

As it turns out, most of the extra features on the ESP32-Audio-Kit are not being used for this audio signal processor project -- the SD Card Slot, the two Audio Amps, the on-board Electret Mics, the on-board switches, the on-board HeadPhone Jack and Input Jack, the battery circuit.

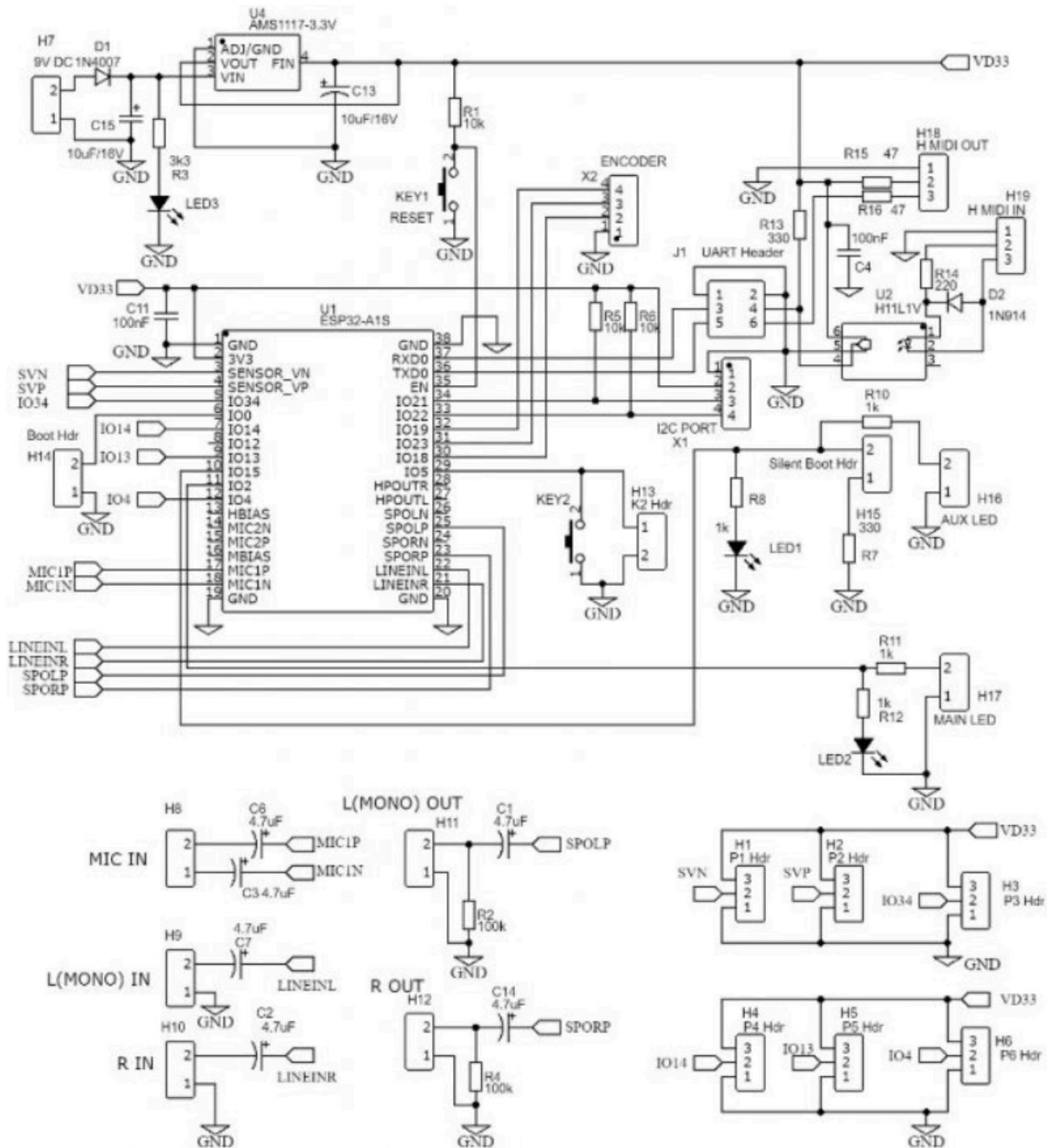
The only devices that were absolutely essential were the USB circuitry for programming the board and the power supply circuit. The Blackstomp Pedal project uses a USB-to-TTL cable or converter which simply connects to the RX, TX, and GND pins of the ESP32.

This entire project could be built from just the ESP32-A1S mounted to a circuit board for easy access to its pins. A few extra capacitors and resistors may be recommended for connection to the ESP32-A1S audio pins. For this check out either the ESP32-A1S minimum system diagrams, or the Blackstomp Pedal circuit diagram, both shown below.



**ESP32-Audio-Kit Circuit**





**Blackstomp Pedal Circuit**

