Jazz Hands: Hybrid Saxophone

by AndrewChi

Jazz Hands is a DIY project to turn your saxophone into a general purpose MIDI controller without losing the playability of your original sax.

The idea is to put sensors on the keyholes and convert it to MIDI on a microcontroller.

Features:

Open source PCB's

Sparkfun "ESP32 Thing" based (Arduino compatible)

MIDI OUT over **Bluetooth** (BLE)

MIDI OUT 5pin connector **(DIN)**

expression pedal input

2 capacitive touch sensors: extend the midi octave range up to 6 octaves

Designed to be hacked and easily extendable

This project is presented as a proof of concept. It does work well and it's a lot of fun to play with.

It's not too hard to install the system, but prepare for handling small cables, soldering and manual work.

All files are included, so you can modify things as needed.

This guide will be updated with new features and firmware updates.

The system has been designed to be reversed/uninstalled at any time, without leaving a single trace unto the saxophone. Not that you can put everything on before a gig... but if you want to remove the system at one point, you definitely can without damaging your precious horn.

Let's get started!

DISCLAIMER:

This project is a part of my doctoral research in the Arts at ARIA.

This project was kindly supported by IDLab.

I'm a musician, not an engineer. I'm very open to suggestions by experts on the subject matter.

Big shoutout to Gordon Good, who's code has been a great source of inspiration.

Supplies:

SensorBoard PCB fabrication files available at Github

SensorsPCB fabrication files available at Github

JazzHands maker code from Github

3x 6m 38AWG stranded insulated copper hookup wire in different colours

3x 50cm 28AWG solid core hookup wire in different colours or 3 x 25 Dupont style connectors (female)

20 hall effect switches

20 3mm neodymium magnets

2 metal discs of app.1.5cm diameter

2 x 24g of Sugru adhesive paste

3 x 40cm heat shrink tubing small in different colours

Electric breadboard

3 male jumper wires to alligator clips

soldering iron

lead free solder

wire stripper

computer running Arduino IDE

microUSB cable of 2m or larger

expression pedal (type Roland EV-5 or similar)

standard 5-pin MIDI DIN cable length 2m or larger

Sparkfun ESP32 Thing microcontroller

DIN female connector

TRS 6.35mm (1/4 inch) female connector

Pin headers 2.54mm (40 pins)

24 resistors of 100kOhm or 16DIP package resistor array chips

2 resistors 10kOhm

3 16DIP package SN74HC165 shift register input IC chips

3 or 6 16DIP sockets (6 if you use the DIP resistor arrays)

electric multimeter

pair of pliers

non drying sticky putty such as "Pritt buddies"

some clear adhesive tape, some adhesive sticky putty and a soldering 'helping hand' may prove very handy!!

'hook & loop' tape (velcro) app. 3 x 8cm

piece of flexible styrofoam

sheet of arts & crafts rubber foam

Step 1: Get the PCB's

The first step towards your Hybrid Saxophone would be to fabricate the PCB's.

You could do this yourself, but it's pretty difficult indeed. It's far easier to get the work done by a PCB fabrication service. You can go the cheap way and use a service like PCBWay or JLCPCB in China. Or you can do what's right and get the work done by a European manufacturer like Eurocircuits or Multi-Circuit-Boards. They are a lot more pricey, but hey, they also live up to the standard in terms of environmental impact and fair wages. Agreed, you'll probably pay up to 10 times the price. But your karma will make it up in the long run I believe. Friends in the USA can opt for a fabrication service in the States here.

I paid a little under EUR 60 for 2 Sensorboards (including shipping) and some EUR 25 (shipping included) for the SensorBoard.

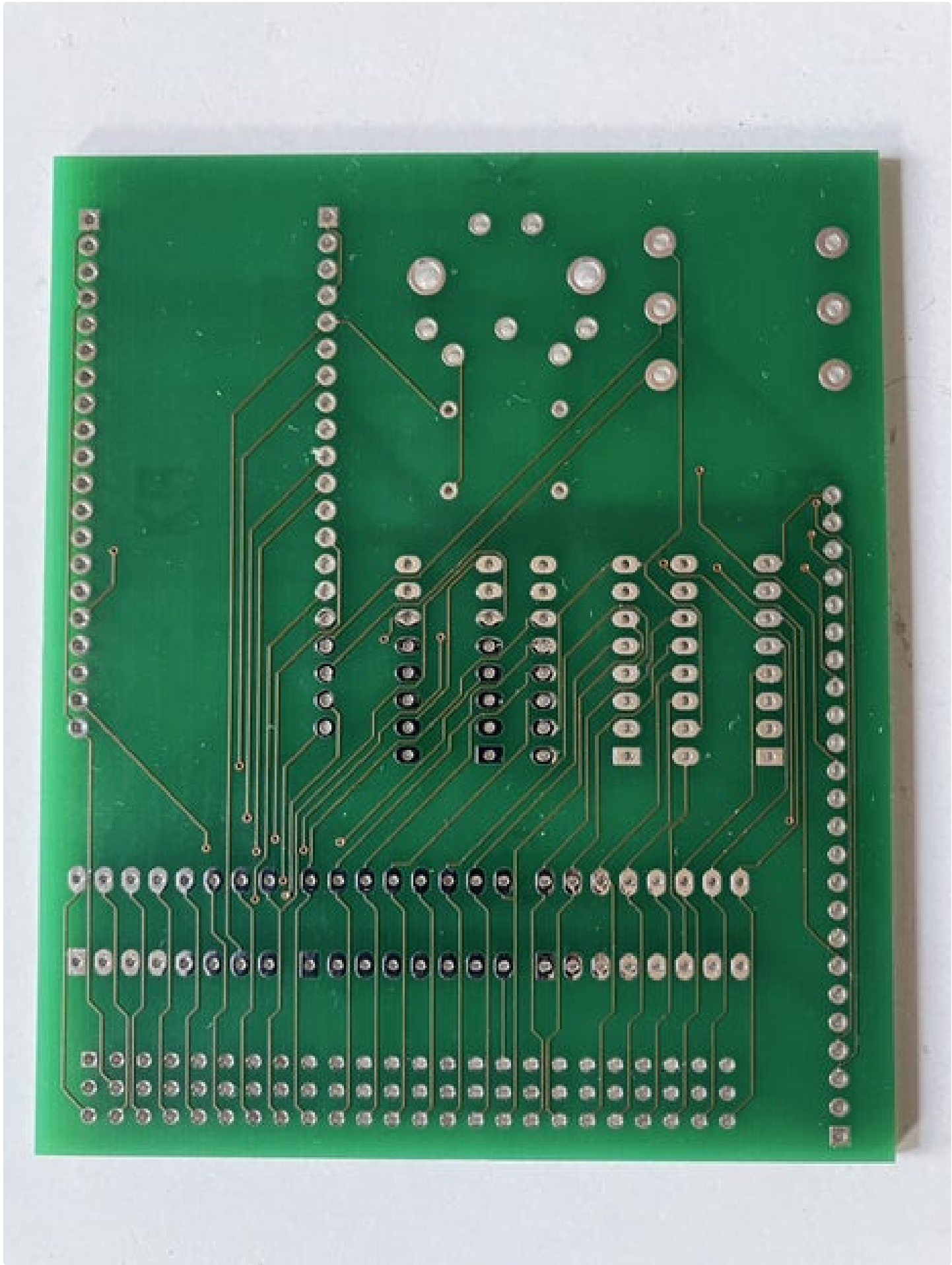
Download the PCB maker files here on Github:

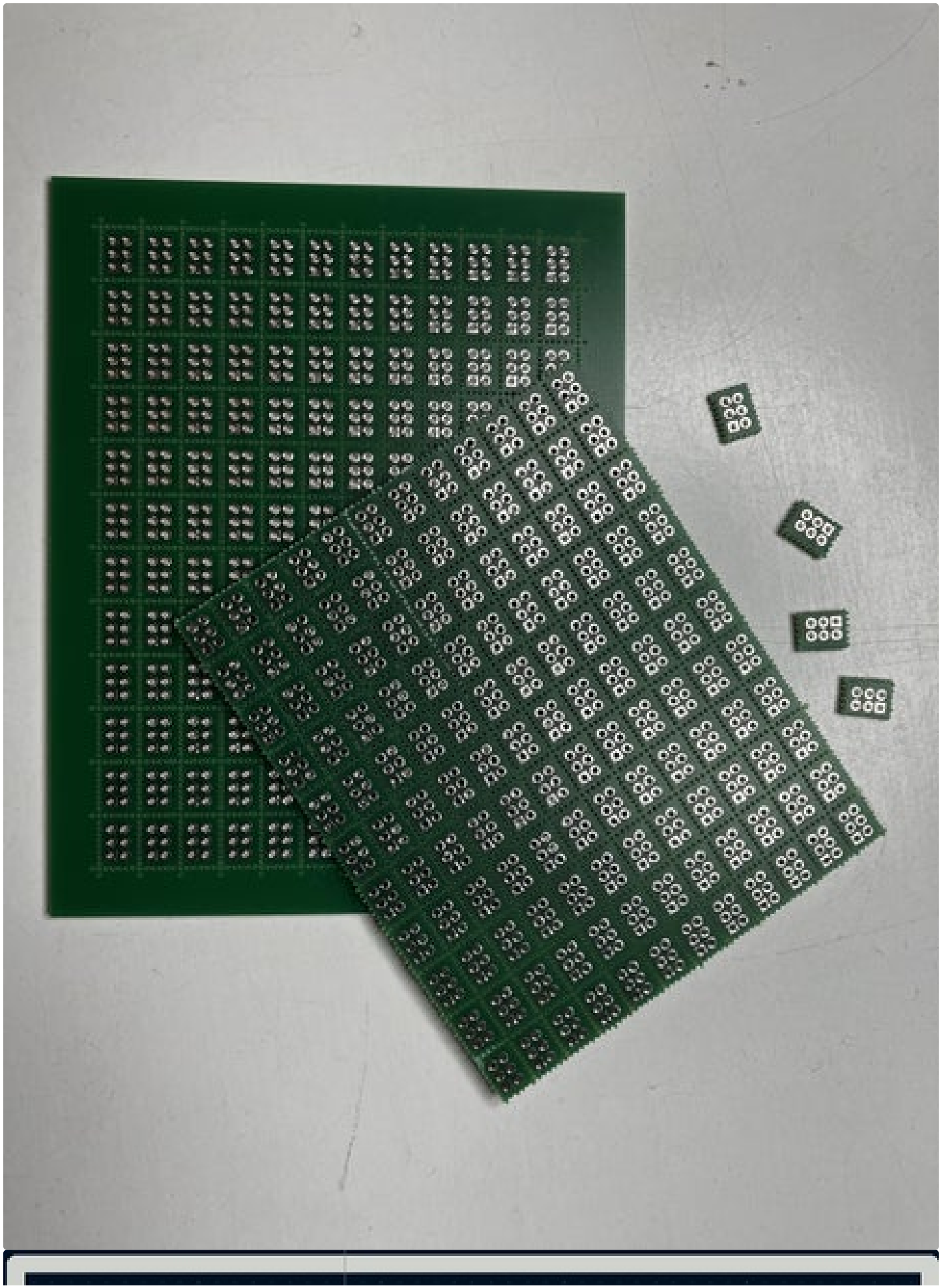
JazzHands

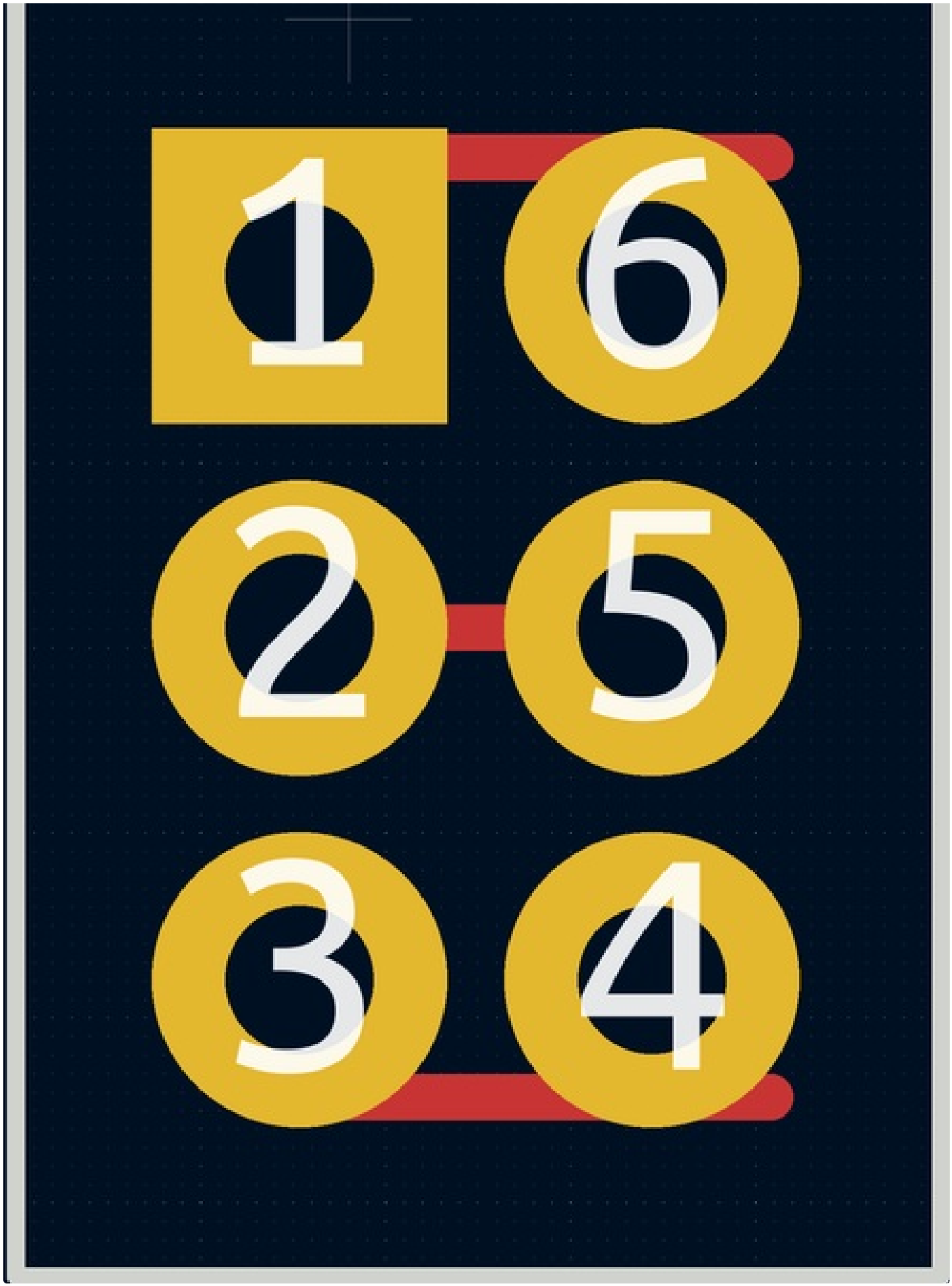
The SensorBoard is well... the board where the sensors will be connected to. It holds the ESP32 microcontroller running the software, the MIDI output connector and the expression pedal input.

The Sensor Panel is a panelized PCB having 144 individual sensors. More than enough!! You'll need only 20, so share them with your friends in need!!

I might upload a new single PCB holding both the SensorBoard and enough Sensors in a single PCB. But for now, this is how it's done. The Sensor PCB is nothing more than some kind of stripboard with internal vertical connections. So if you are creative, you can easily make this yourself from stripboard or protoboard. I decided to make a dedicated PCB because it's just a little smaller than the standard dimensions of such protoboard. Have a look on the picture to see the simple internal connections. If you're working on a soprano saxophone, I strongly advise you to use the SensorPCB.






1
6
2
5
3
4

Step 2: Choose Your Hall Switches Wisely

Hall effect sensors are great little sensors. They'll go active when a magnetic field is detected. It's the type of sensor that's typically used to get you're smartphone in sleep mode whenever you close the magnetic lid. But those sensors are too slow for what we're after. Of course you want to play Charlie Parker solos with a fat synth don't you?

The type we will be using is much faster and I've been told were used in the automobile industry for measuring the RPM of the motor to be displayed on the speedometer on the dashboard. 3000RPM? No problem. Are you a robot saxophone player with super speedy titanium fingers? We got you covered!

Hall effect sensors come in 2 types: digital switches and lineair analog sensors. Since we are measuring only wether your key is open or closed, we'll be using the digital switch ones. The lineair sensors will not work with the board (you'll know why when we're soldering the SensorBoard).

Every hall effect switch will work. Just make sure you're ordering a THT (Through Hole Technology) package that looks like a standard transistor. The official name is TO-92. Modern SMD sensors can be as small as 0.4mm... not exactly suited for manual soldering!!

Also, very important is to check the datasheet for speed!! We want the fast ones, not the ones for opening your laptop, tablet or phone!!! They are a bit cheaper, but they will be too slow for your rapid fingers.

In the picture you can see a screenshot from the datasheet of a Texas Instruments DRV5023. You can see the output delay time is in the microsecond ballpark and the rise and fall times are expressed in nanoseconds. That's what we want.

These magnetic switches are usually unipolar, which means they'll switch on when a positive magnetic field is detected. This is the South pole of a magnet.

The other types are bipolar (switch on +, switch off -) and omnipolar (switch on when magnetic field + or - are detected). Do not use those!! Stick to the classic unipolar hall effect switch.

Another important feature to check is the operating voltage. Should be okay, never saw a TO-92 package that's incompatible, but please double check your sensor will operate at 3.3V.

Want to keep it simple? Just order the Texas Instruments DRV5023 in a TO-92 package. Another great choice is the one made by Allegro Microsystem. They have a very informative document here. A good read!

You'll need at least 20 of them. If you want to expand to even more sensors (if you just love cables on your sax...), the SensorBoard supports as many as 25 hall effect switches on the PCB and if you're really creative you can find a way to get even more sensors going on the current system (more on that later).

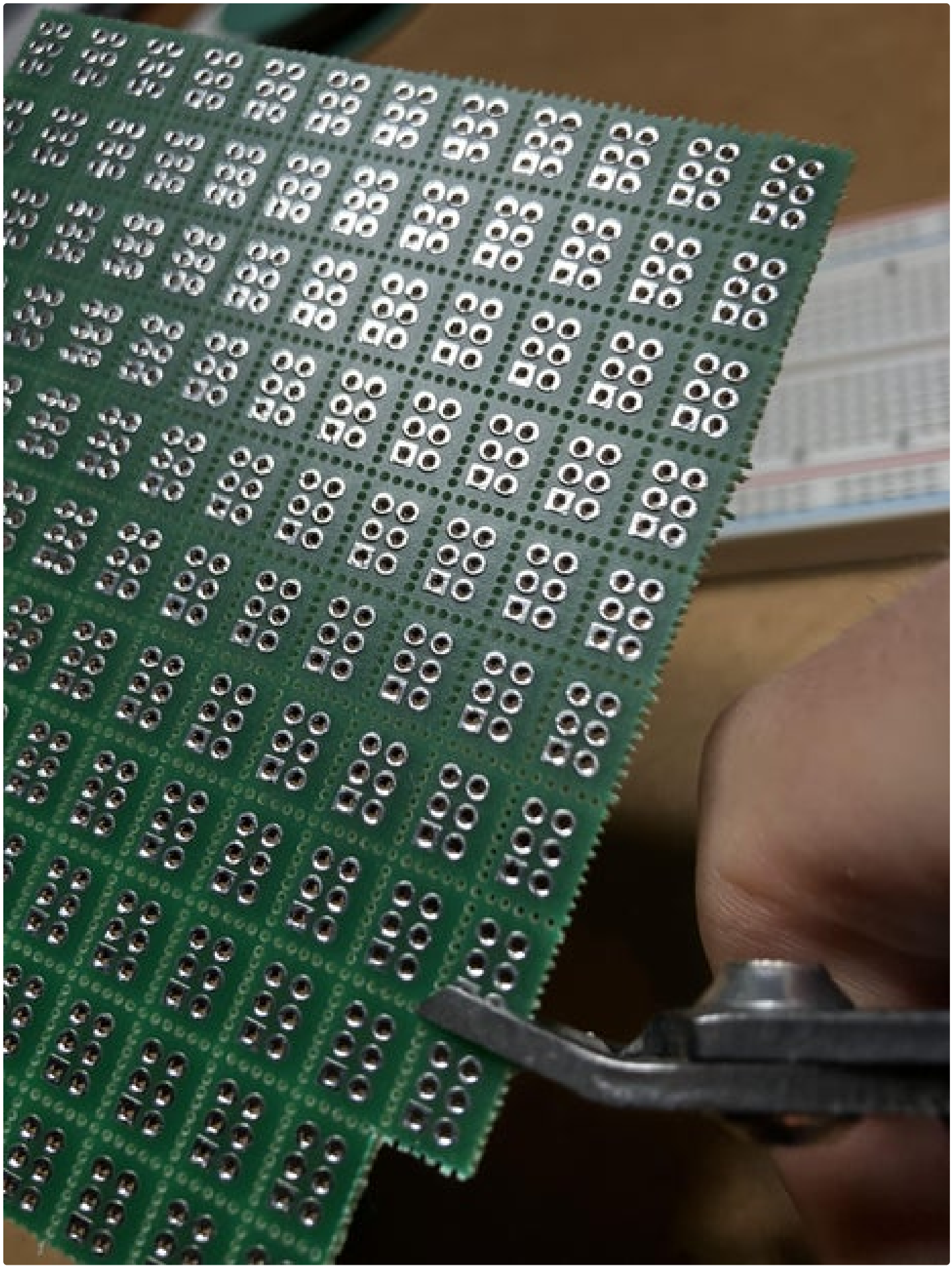
6.6 Switching Characteristics

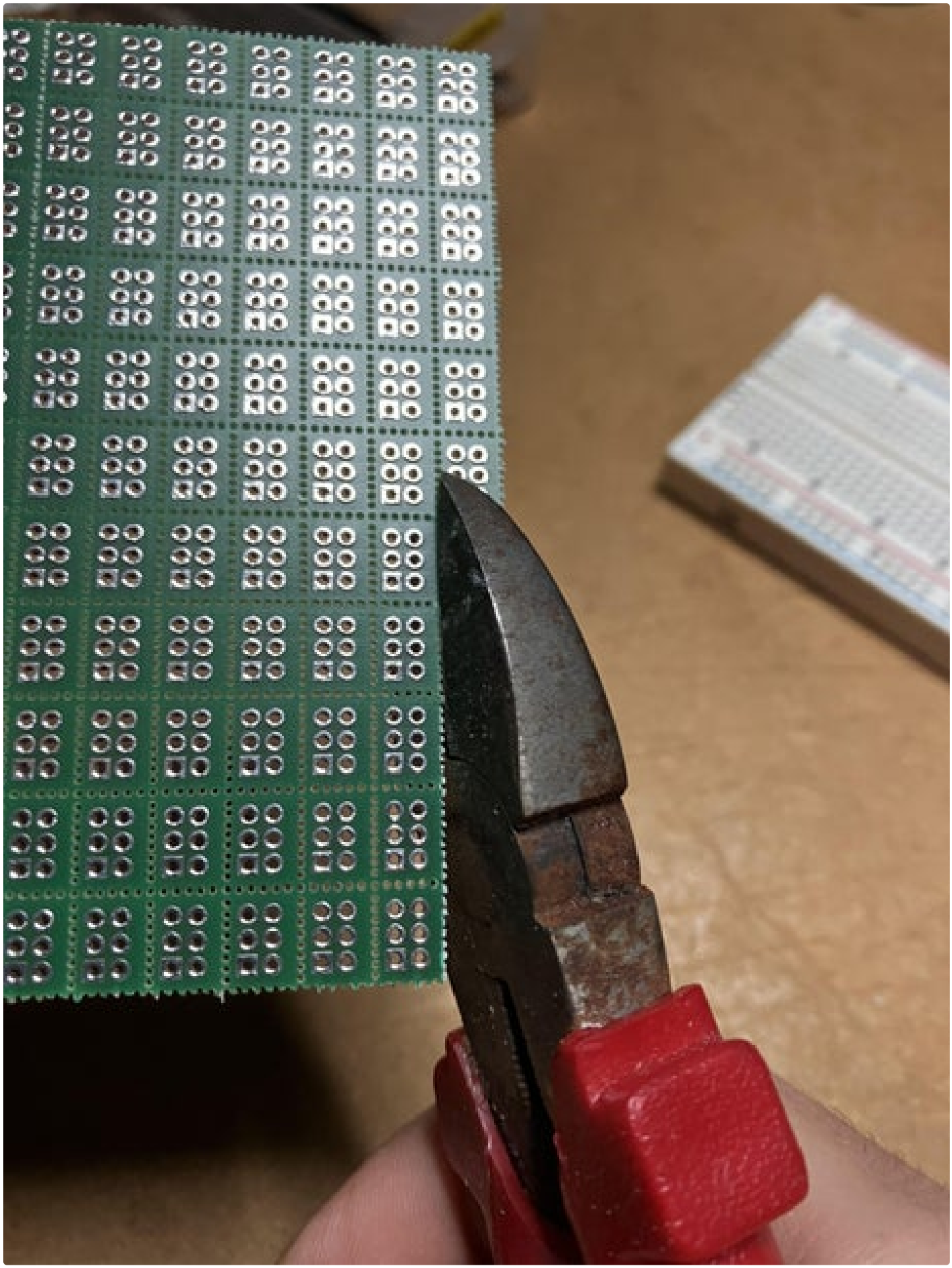
over operating free-air temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
OPEN DRAIN OUTPUT (OUT)						
t_d	Output delay time	B = B_{RP} – 10 mT to B_{OP} + 10 mT in 1 µs		13	25	µs
t_r	Output rise time (10% to 90%)	R1 = 1 kΩ, C_O = 50 pF, V_{CC} = 3.3 V		200		ns
t_f	Output fall time (90% to 10%)	R1 = 1 kΩ, C_O = 50 pF, V_{CC} = 3.3 V		31		ns

Step 3: Cut the Sensors

To get started, we'll start with the sensors. If you are using the Sensor panel PCB, you'll see the 144 individual sensor PCB's are separated with so-called 'mouse bites'. Start with cutting them using a pair of pliers.



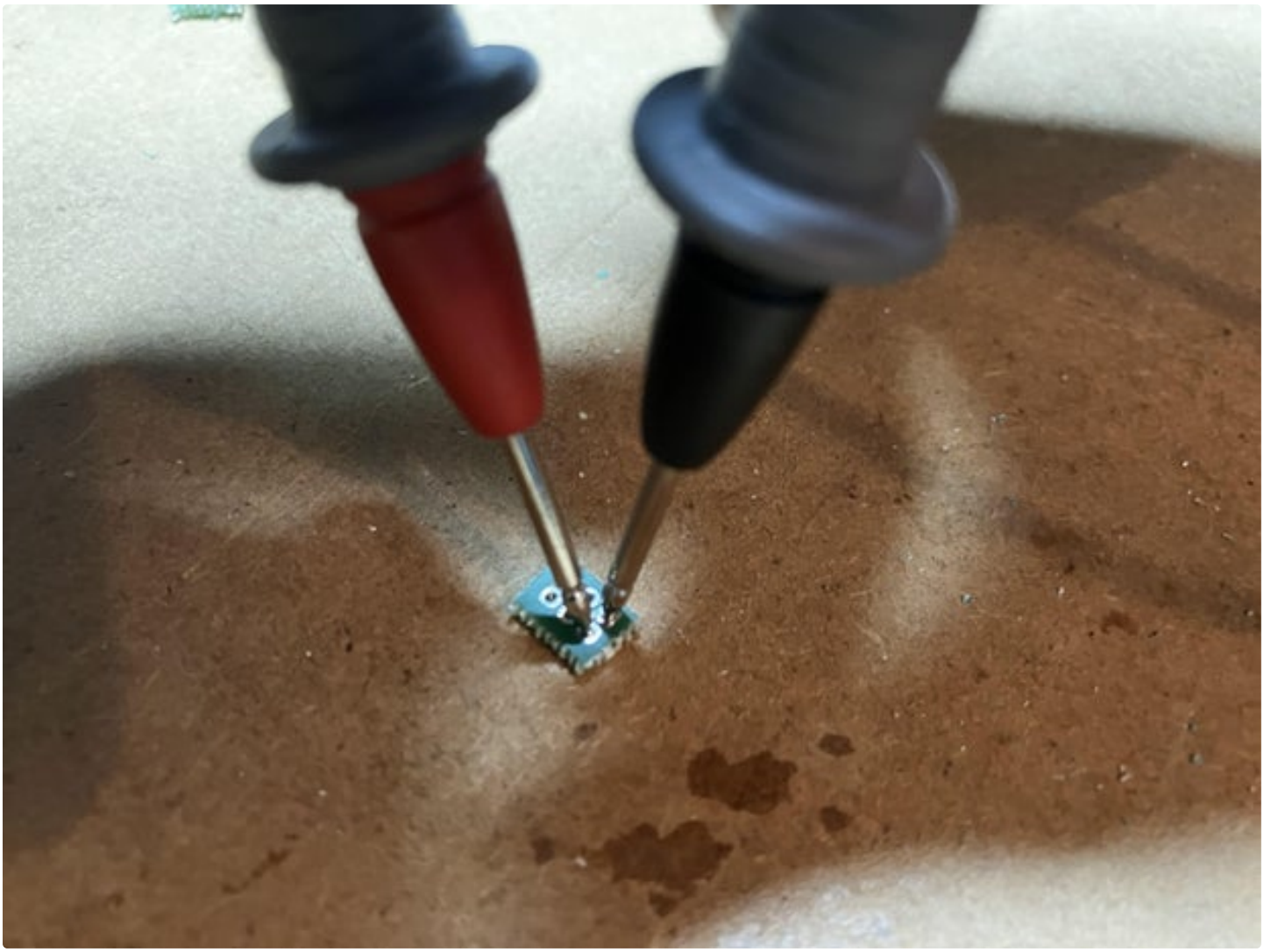


Step 4: Check the Sensor PCB

I know... You want this thing working. But you've got to believe me on this: better safe than sorry! The 'mouse bites' are kind of stubborn and such a tiny PCB is quite vulnerable. So after cutting it to pieces, let's do a quick check for conductivity.

Place your multimeter pins on the overlying connection pin holes. Place your multimeter in DC mode (the one with a line and a dashed line) and look at the display.

It should short and display a stabile decisive 0. If it's doing something else, get rid of that tiny PCB at once!! Don't throw it in the bin but be a decent human being and recycle it properly!!




SELECT
OFF

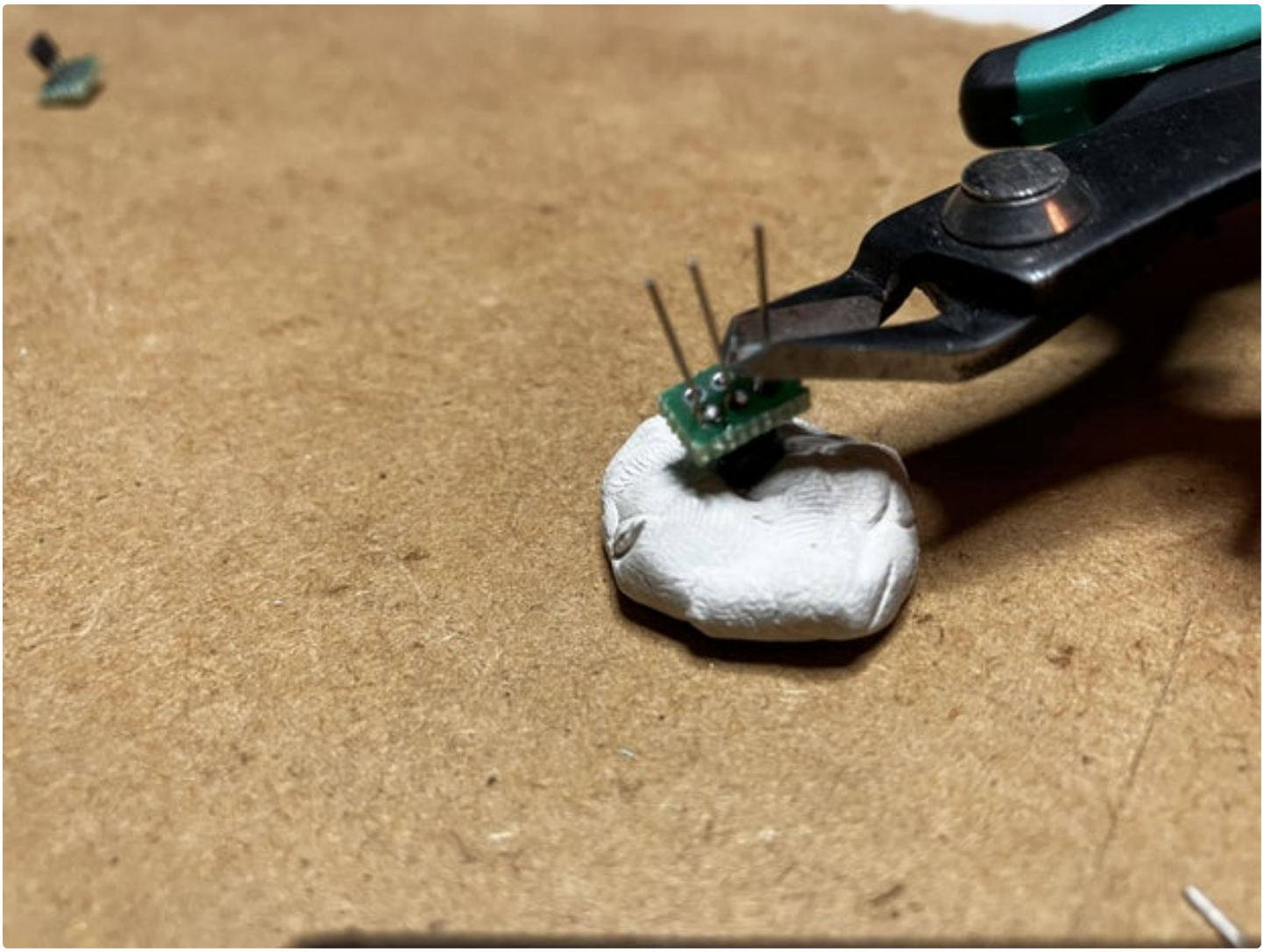
Step 5: Solder the Sensors

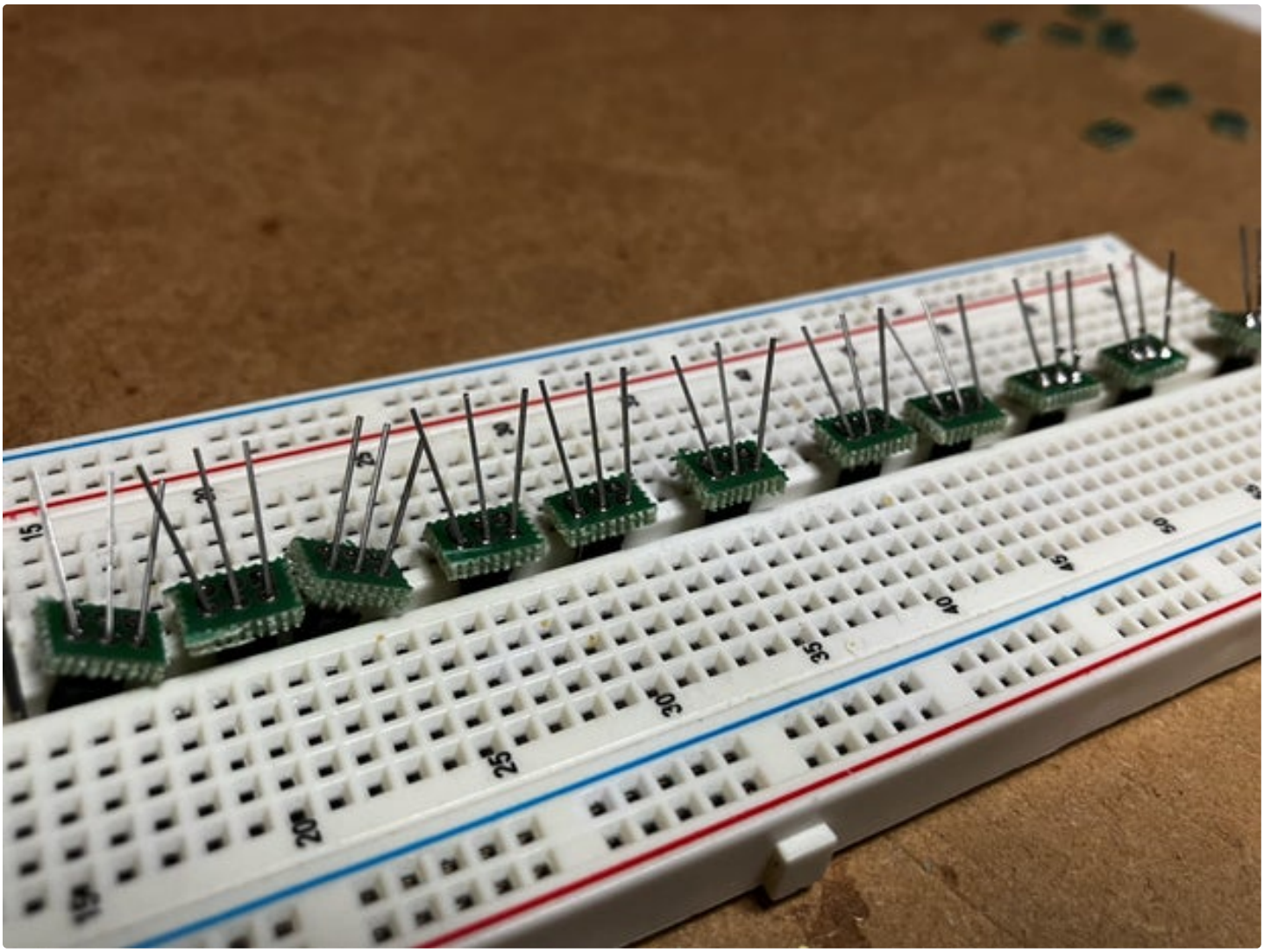
Alright, let's get busy soldering the hall effect switches to the small PCB's.

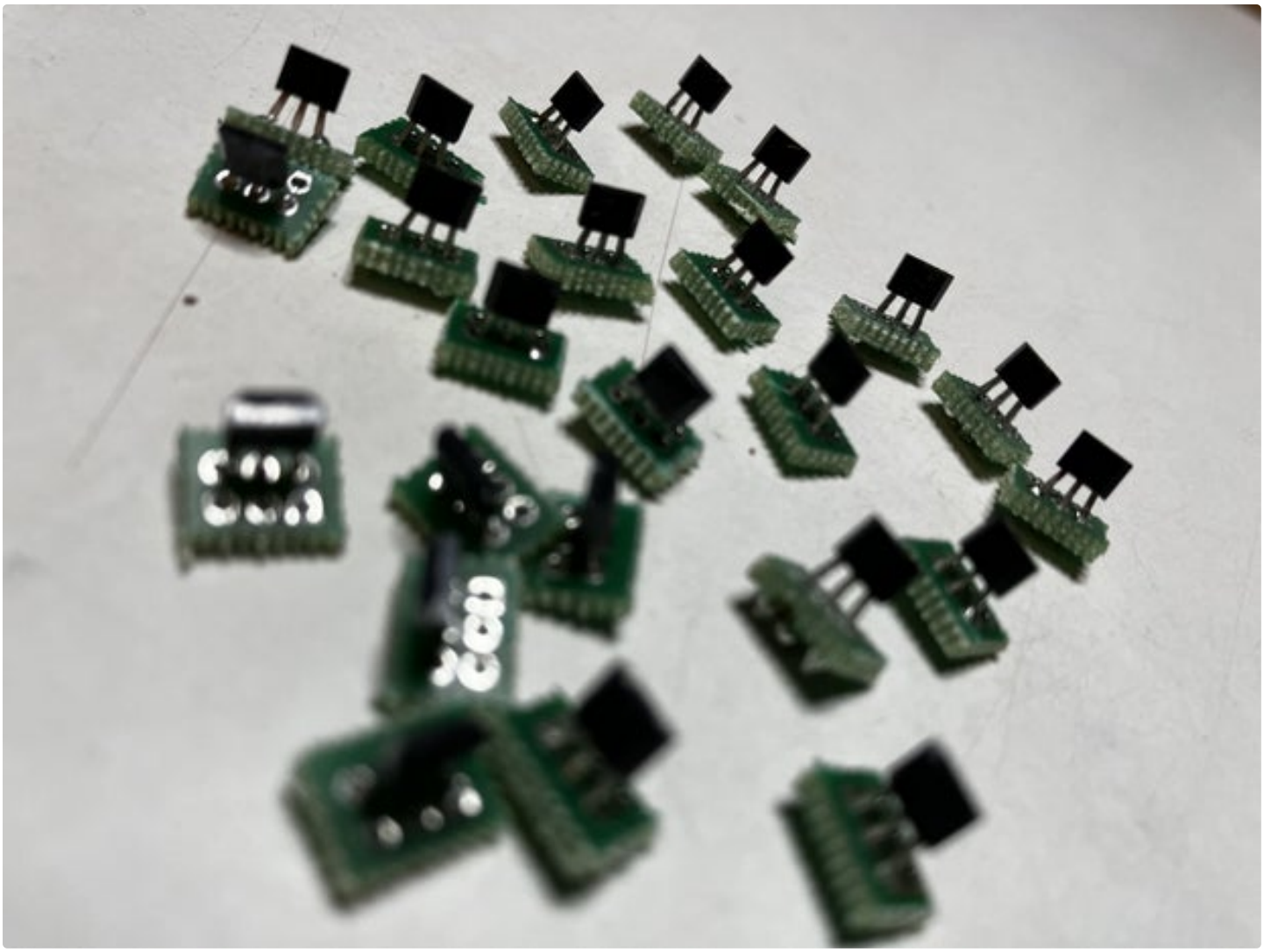
It really doesn't matter how you are holding the PCB. Just make sure the sensor side (small side) is facing outward.

You'll need at least 20 of them to get a fully working system. It's a good idea to solder some extra. It's small and we're not done soldering yet!

I use sticky putty a lot. It's super handy if you're working alone and need some support. Maybe that's why Pritt calls them buddies?







Step 6: Prepare the Cables

Now let's prepare the cables. I made the project with two different types of cables. You could go for slightly thicker cable. Cables could be stranded or solid core. Standard stranded wire of 28AWG can be used, that's what I did in my first version, but your sax will be literally covered in cables.

Three cables are needed for each sensor, sporting a whopping 57 cables on your horn. Agreed, thicker cables are easier to handle while making the system, but in the end they are far more difficult to handle later on when the system is fully installed. They will be more difficult to hide and make the (already quite) vulnerable system more prone to defects.

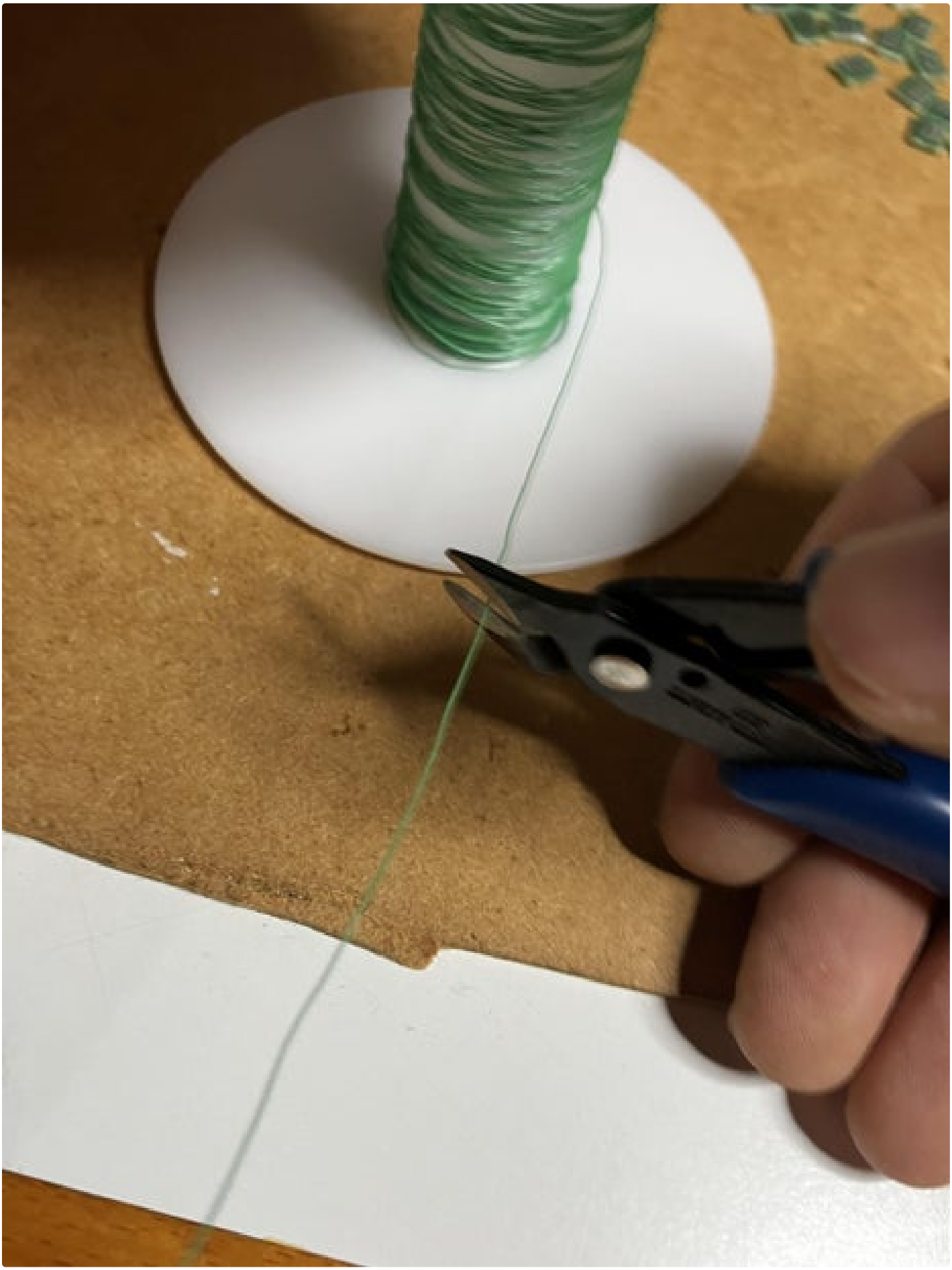
Small stranded cables are easier to hide and can be easily breaded too, resulting in a more elegant look. But... it's a bit more work. I guess for a baritone saxophone you can make your life a bit easier and go for 28AWG stranded cables. If you happen to find very thin solid core wire that's flexible enough you can try that too. Please let me know where to get those!!

So I ended up using 38AWG stranded wire. It came on spools. Use 3 different colours, so you'll be able to know what's what when attaching them later. As you can see on the pictures there's actually 4 colours. Don't mind, it's just from another build where I couldn't get the same colours again from my local supplier.

Cut the wire in pieces. I used pieces of about 60cm each. But it really depends on what type of sax you're working on. I'm working on C-Melody saxophones, just to be ... well different.

You can get away with shorter cables on altos and sopranos or longer on tenors and baritones. Just stating the obvious here of course. Make sure you can reach the region of the bell of the sax from the highest keys. If you're short on cable, you can make some longer pieces and some shorter. To keep things simple, I'm making them all around 60cm and cut them later (I'm recycling excess cable in other projects).

Cut as many pieces of cable in each colour as you have sensors. A minimum of 20 cables of each colour is needed.

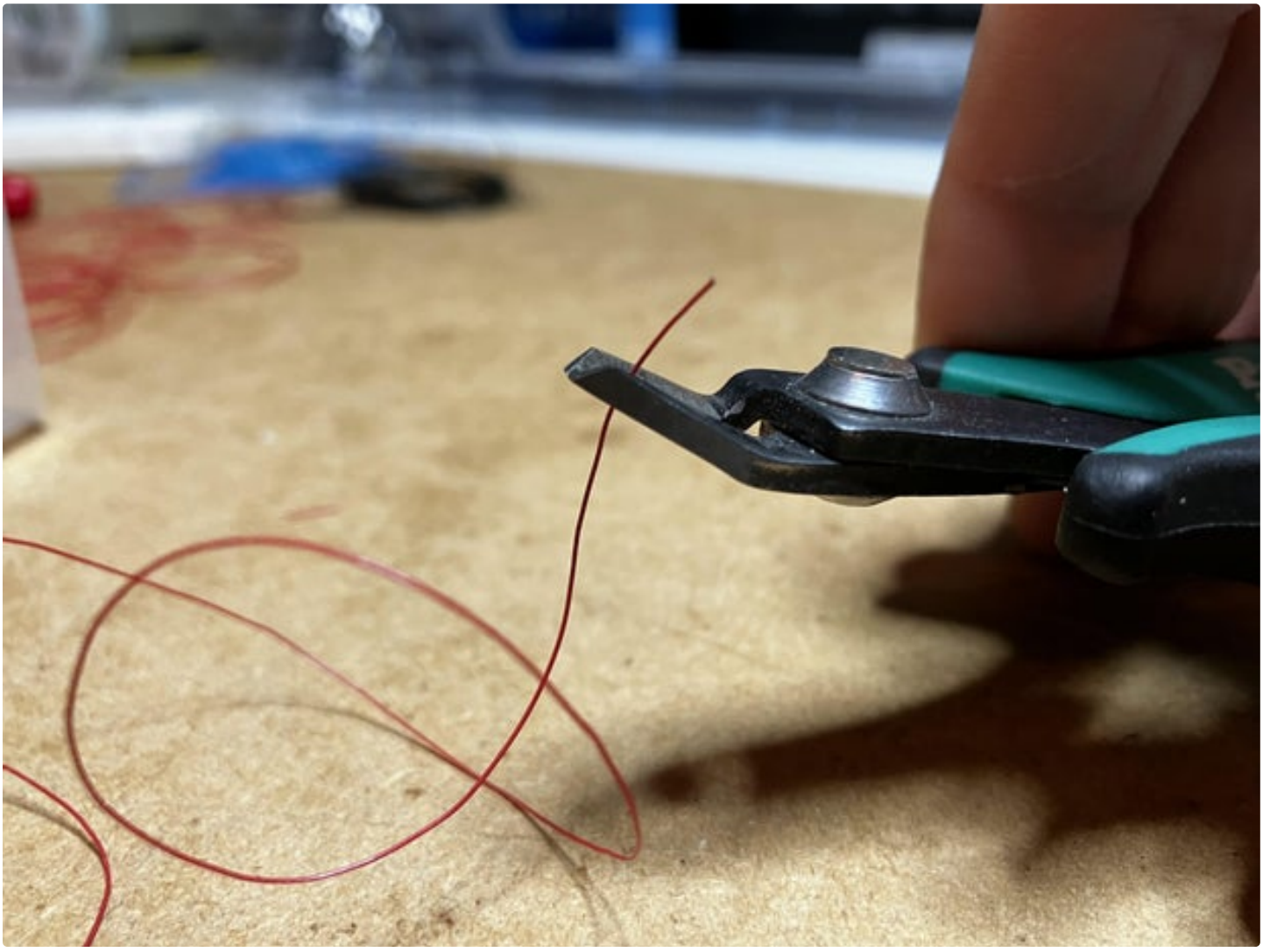


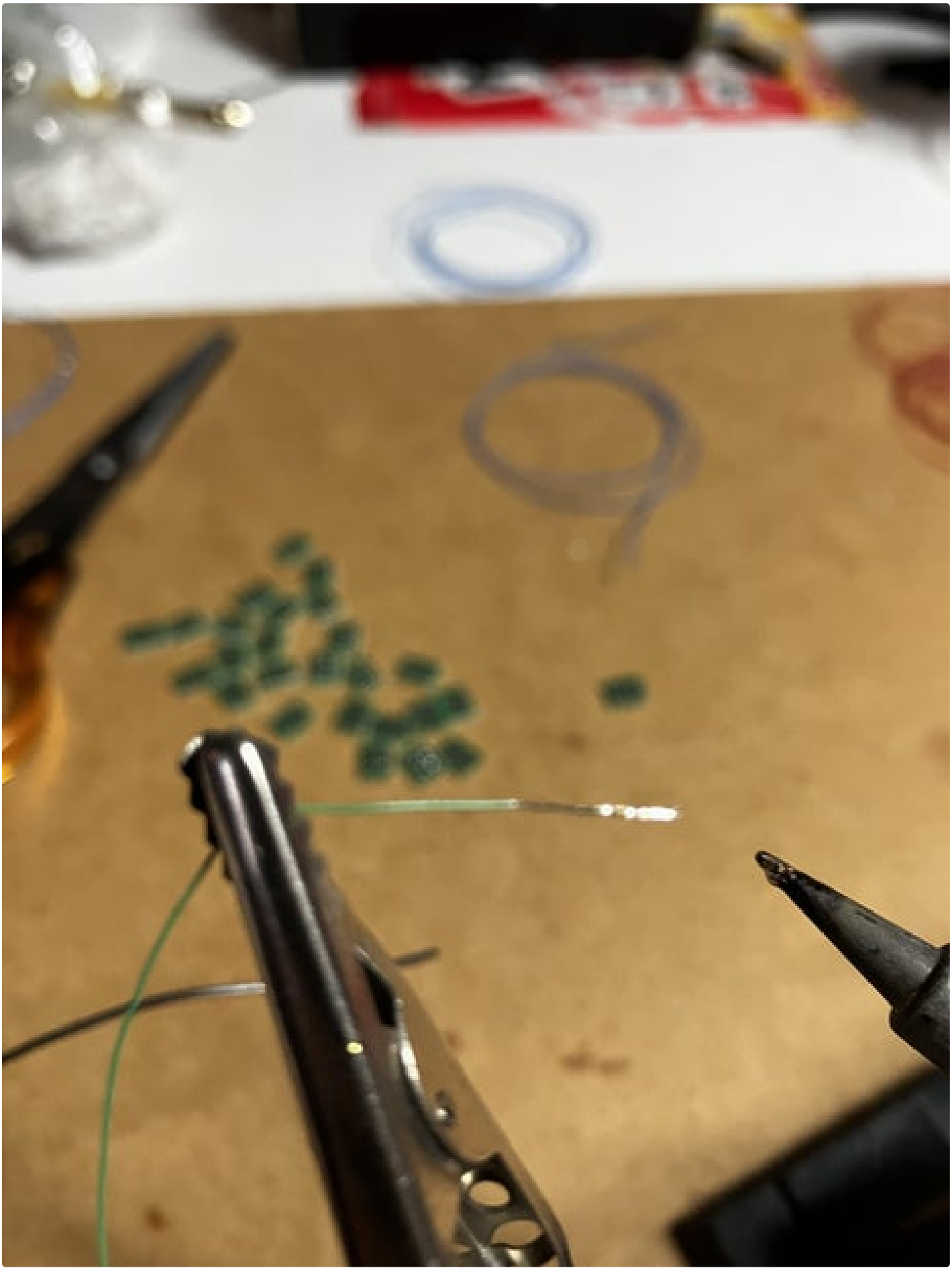
Step 7: Stripping and Tinning the Cables

Now let's strip all the cables of their insulation on both ends. 1cm should be enough. If you are using thicker cable, use stripping pliers to make your life easier.

Unfortunately, I had some bad luck finding such a handy tool for these extra thin wires. I bought a special tool for this, but unfortunately it didn't work well. So I've done it the hard way using cutting pliers. If you accidentally cut off a piece of wire, don't bother, if you're sticking to 60cm or more, you'll have more than enough left.

After stripping, it's important to tin the exposed stranded wire before going to the next step. Do not attempt to solder stranded wire to the sensor PCB. You'll end up with a lot of shorts, as I had to learn the hard way. One defect in one of your sensors and... your system will stop working properly (unfortunately).





Step 8: Soldering the Cables to the SensorPCB's

Now it's time to solder the cables to the small SensorPCB's. If you look at the SensorPCB from above, with the hall effect switch pointing upwards (to the north if you want), the 3 vacant pin holes from left to right are 3.3V, GND and OUTPUT.

Choose your colours as you want, but be consistent in your choice! A reasonable choice would be red for the voltage source, black for GND and some other colour for the sensor output.

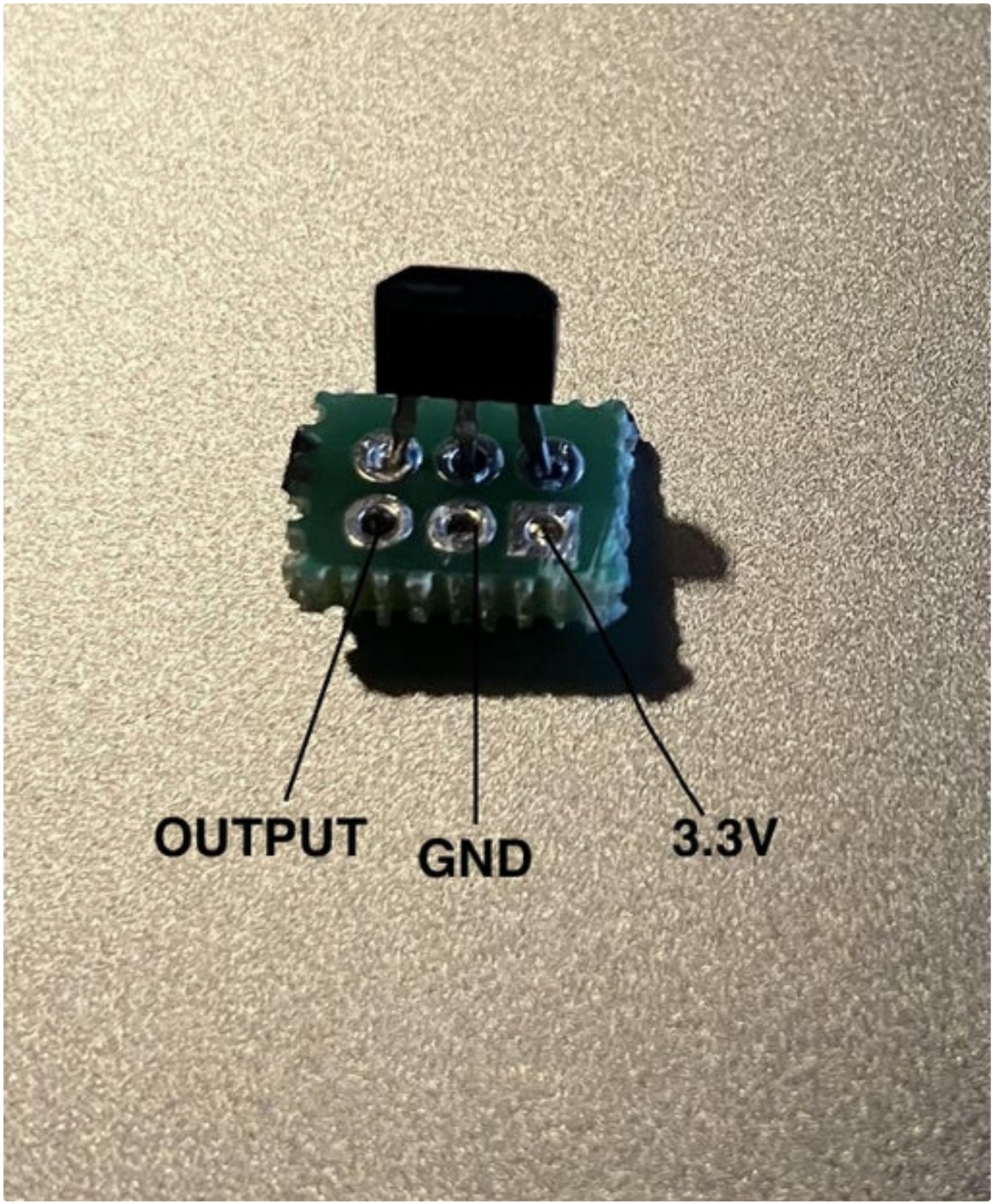
Now put the tinned end of the cable through the corresponding hole and solder it in place. It's very small, so make sure not too drink too much coffee before doing this...

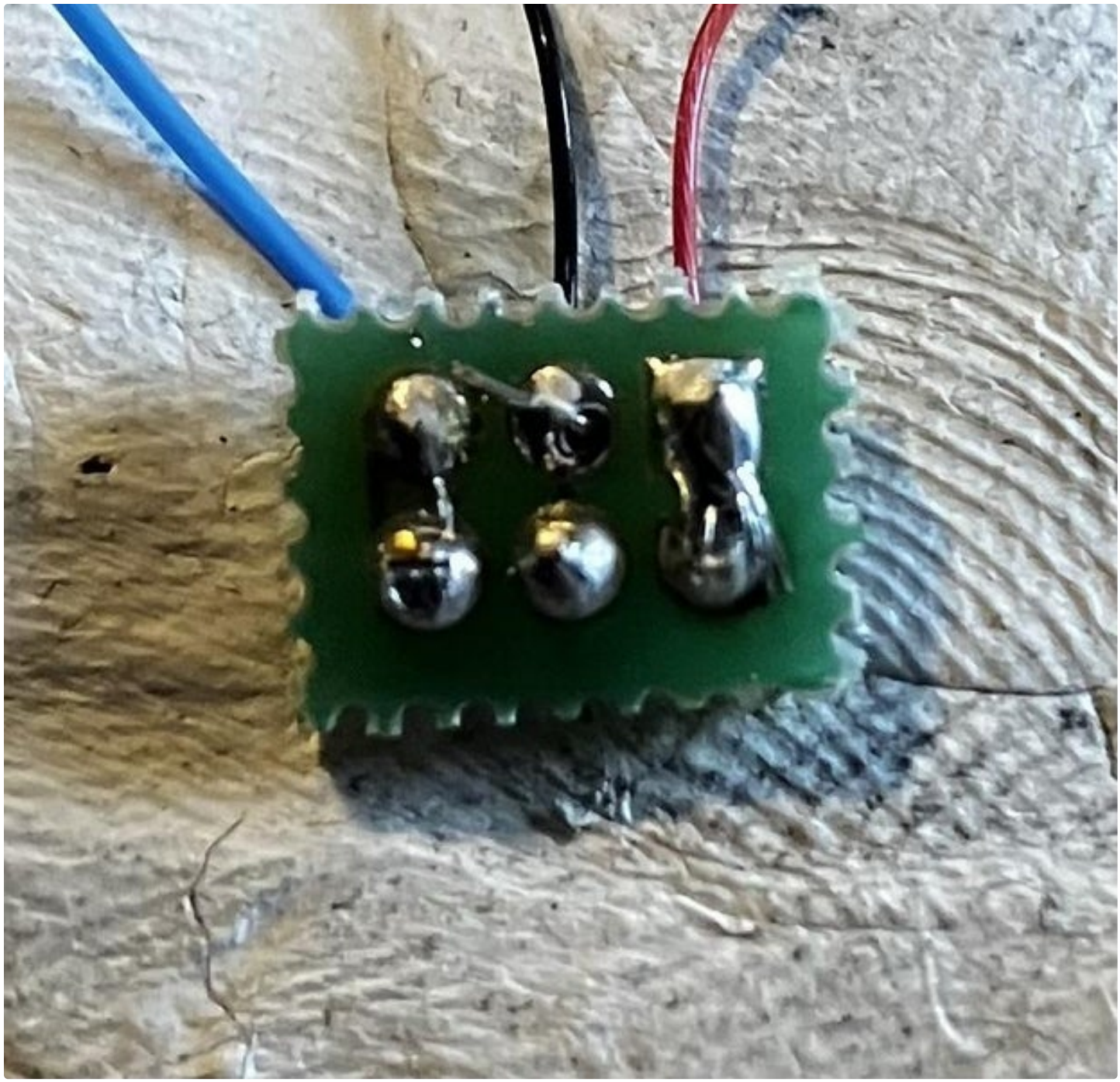
The pin holes on the PCB are connected in overlying fashion. So you just have to make sure no short is created on adjacent pins.

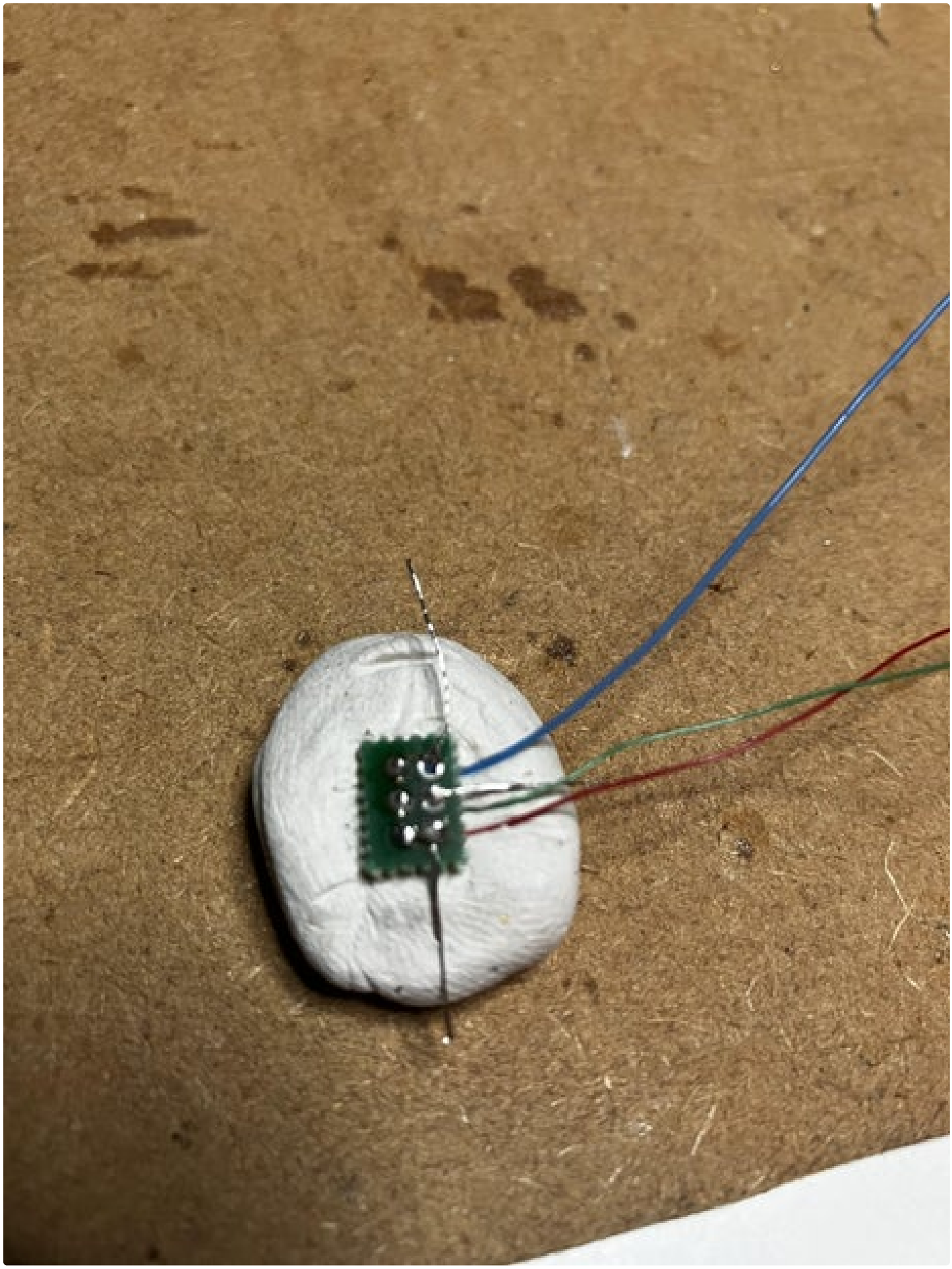
If you are not using the SensorPCB but chose to make the sensors out of standard protoboard, you can make a solder bridge from the sensor pins to the corresponding cable.

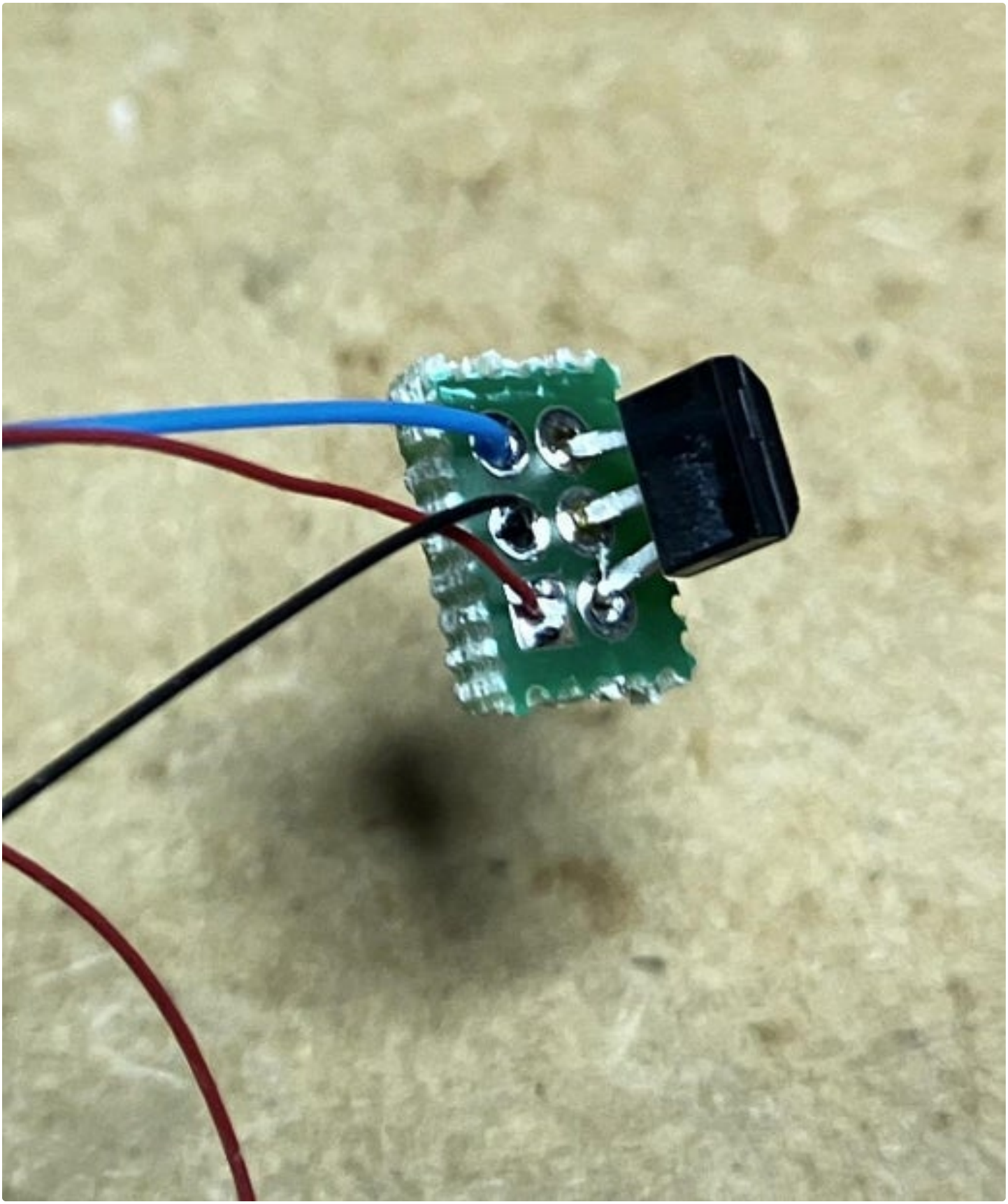
If you're really headstrong I suppose you could also solder the cables directly to the pins of the hall effect sensor, without any PCB or protoboard. It definitely could work, but remember, this whole thing is quite vulnerable and you don't want cables breaking off when putting them in place. You'll have to start all over again! I did warn you.

I'm again using some sticky putty to keep everything in place. If your tinned cable end is a little too big, it's also a good idea to fold the tinned edges to the sides on the sensor and 3.3V pins, to further avoid making shorts.


OUTPUT
GND
3.3V



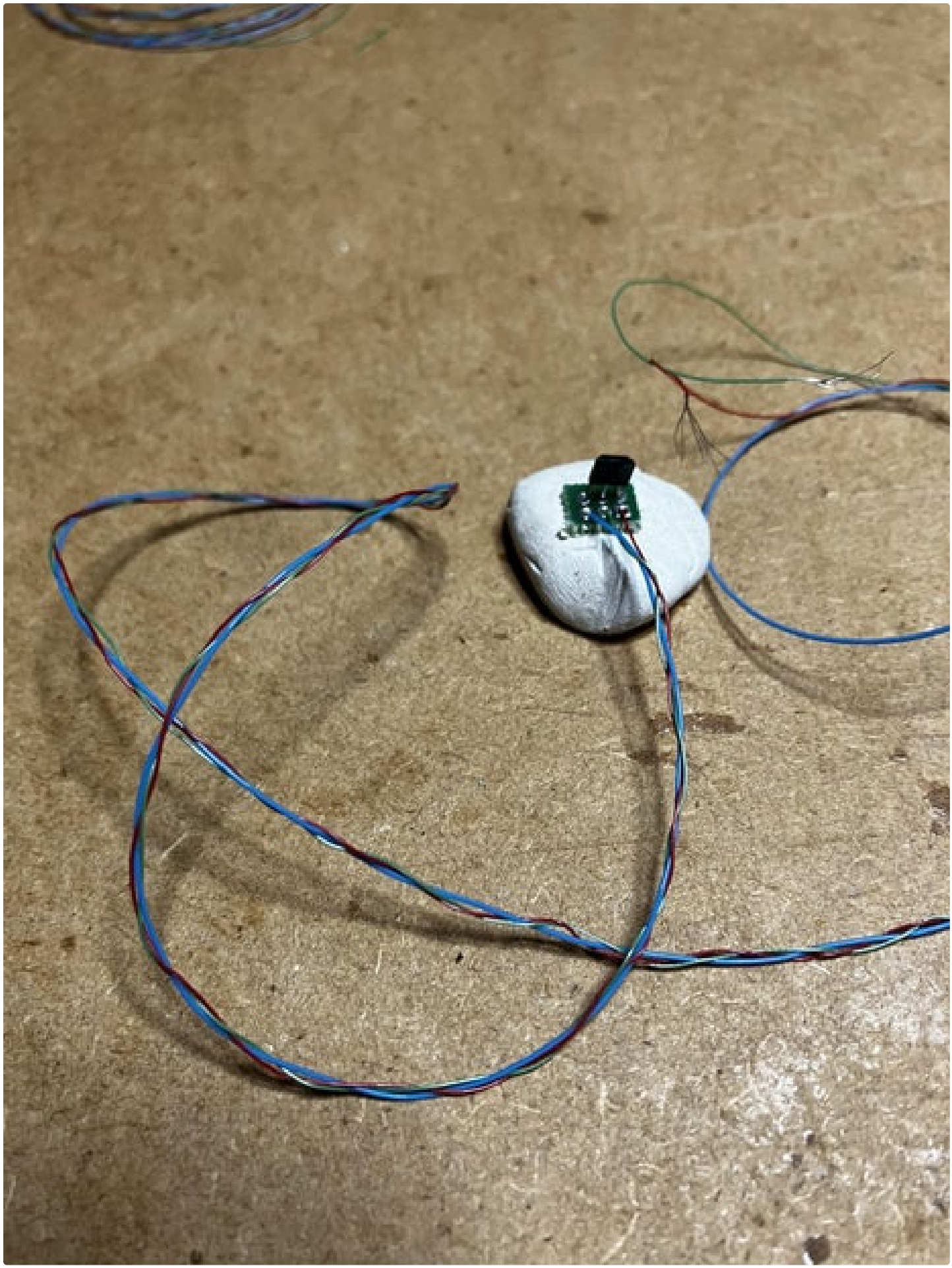




Step 9: Braiding the Cables of the Sensors

With the individual sensors being ready and made, let's tidy things up by braiding the cables.

I'm trying to be serious here. I know you know how to braid with three cables. Go for it!



Step 10: Making the Octave Plates

What is an electric wind instrument without a lot of octaves at your disposal? Let's expand the hybrid sax with three more octaves! I drew inspiration from both the Akai EWI system and the Yamaha WX series wind controllers.

The ESP32 Thing has capacitive touch enabled inputs, so let's make good use of them. As you might know, the Akai EWI series of windcontrollers uses capacitive touch as its main technology for the readout of the finger positions.

The octave roller mechanism is nothing short of genius in my humble opinion. Getting such a system on a traditional saxophone is beyond my skillset, but we can make good use of touch control.

The Yamaha WX windcontrollers have a nice system for selecting octaves with minimal buttons and this is what I went for, expanding the octaves in the lower register with 3 octaves using only 2 extra touchplates. So to make it clear:

- We will install a hall effect switch on the sax near the iron rod controlling you acoustic octave. In this way, the MIDI notes will be transposed an octave higher when pressed, just as expected on a saxophone. Because of all the sensors installed, it will be fairly easy to program your standard or alternate fingerings of choice to get to the top tones of the instrument. In this way, the complete natural range of the saxophone is covered.
- We will install 2 extra touch sensitive metal plates underneath the left thumb rest to address lower octaves. These will follow the Yamaha WX paradigma where touching 1 plate will result in transmitting MIDI notes an octave lower, touching 2 plates simultaneously will transmit notes 2 octaves lower and touching only the 2nd plate will give you note values of 3 octaves lower.

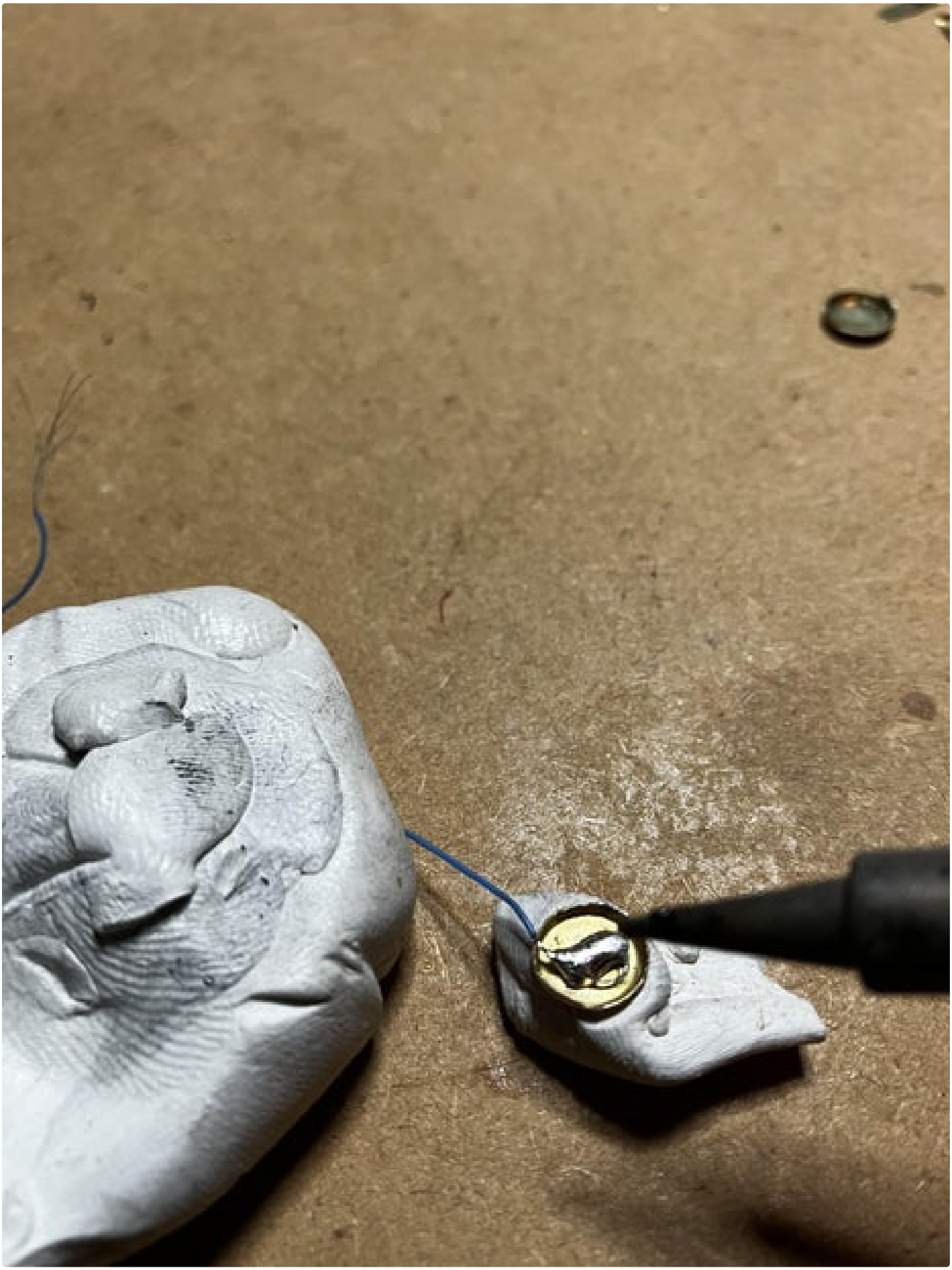
Playing the octaves will feel quite natural if you are used playing Akai EWI or Yamaha WX series of wind controllers.

So let's make them. It's easy!

You will need 2 pieces of conducting metal to be used as the touch plates. You can use whatever you fancy. I used metal (copper?) paper fasteners. When you cut the ends it makes a nice round metal circle, perfect for our use case.

Cut another piece of wire, strip its insulation on its end. Do not tin it, but spread the strands a bit across the surface and solder it to the rear of the metal plate and your done! Sometimes it's easy. At this point you deserve it, really.


APLI
100 u. / 12 mm.
ENCUADERNADORES
PAPER FASTENERS



Step 11: Making the SensorBoard

Now let's go on with soldering the SensorBoard. This is the heart of the system, running the software converting the sensors to MIDI. It's a good practice to start with the smallest components first. In this case the resistors.

If you are using the resistor array 16 DIP package (quite handy) you'll have to solder only three of them. If you're using individual THT resistors acting as pullups for the sensors, you'll need to solder as many as you have sensors. Well actually one less. One less, because the octave sensor will also be a hall effect switch, but we will solder this later to another pin (with an internal pullup resistor).

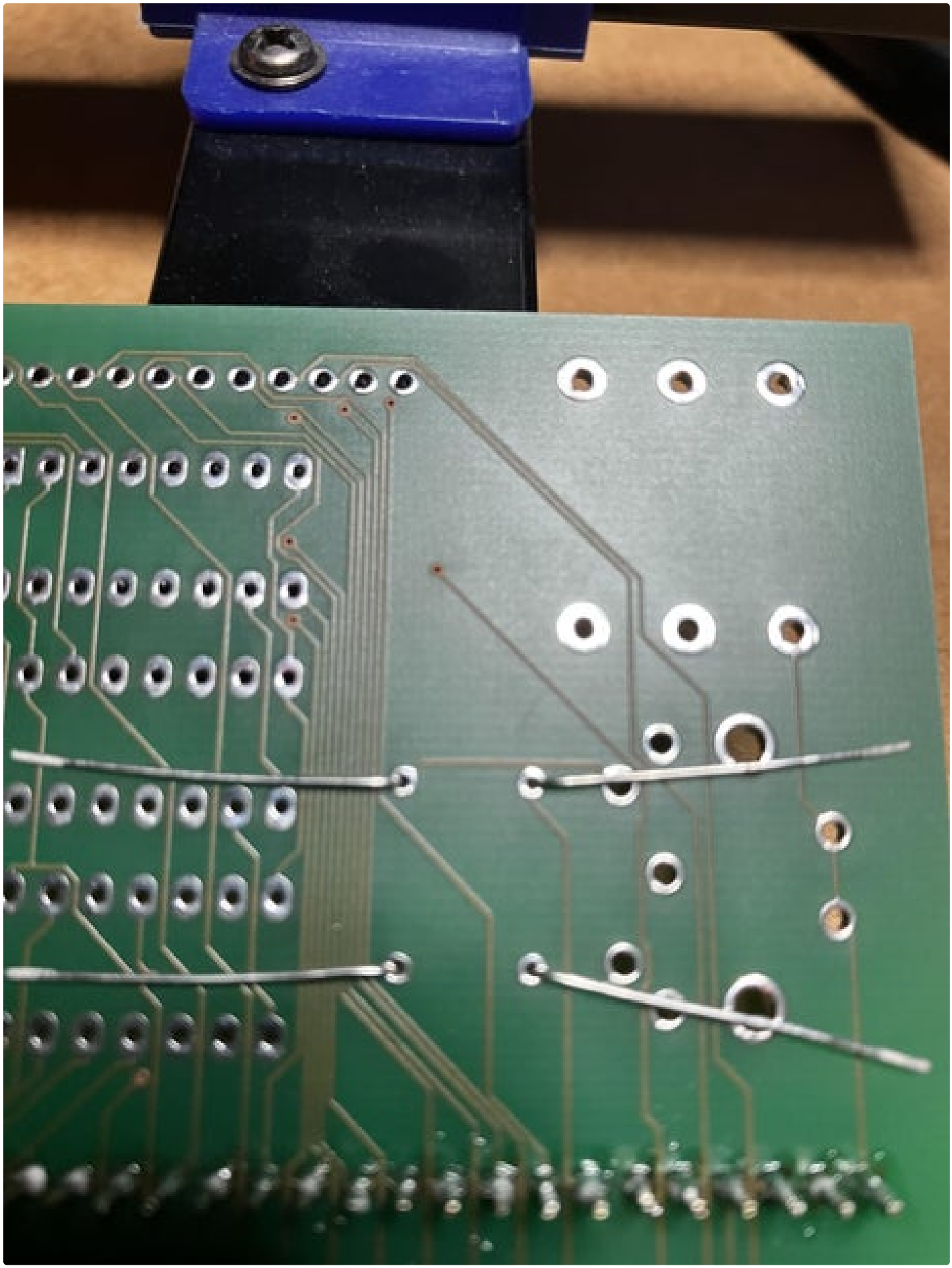
In this guide I will suppose you are using the resistor array 16 DIP chips. Let's go.

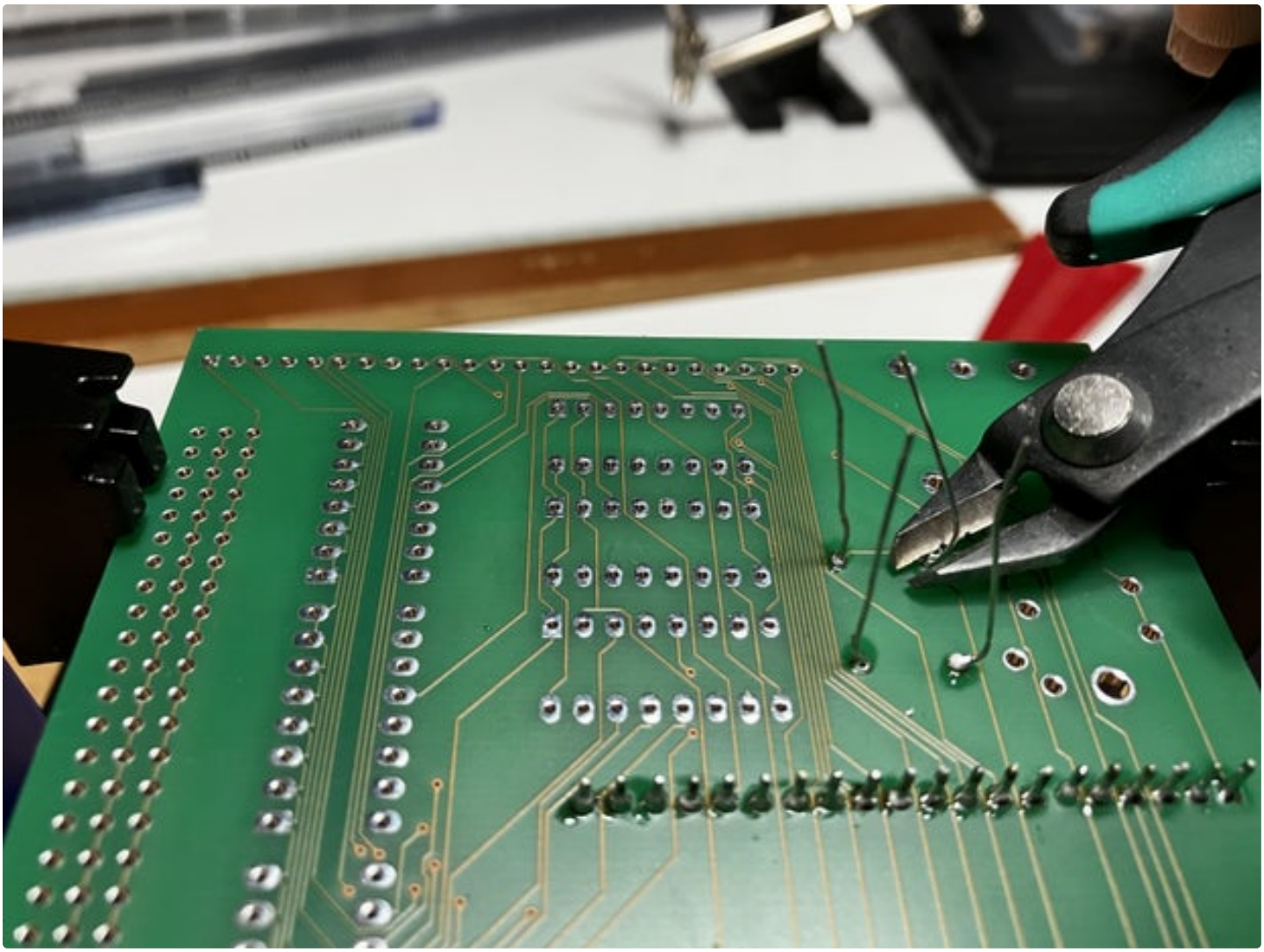
- Solder the 220 Ohms resistors and cut the remaining leads
- Solder the header to the "ESP32 Thing". It's a good idea to fit the header pins in the board first and solder the pins to the ESP32 after that, to avoid crooked pins that don't fit the board afterwards.
- Solder the ESP32's newly soldered pins to the board. If you're the more prudent type of person, you can also use female headers. In this way you can safely remove the microcontroller from the board afterwards. I just soldered the Thing to the board. Basta! *(Actually the female headers were out of stock...)*
- Continue soldering the 16 DIP sockets to the board. I strongly advice to use these sockets. You might get away without them, but the chips might become too hot and get damaged when soldering them directly

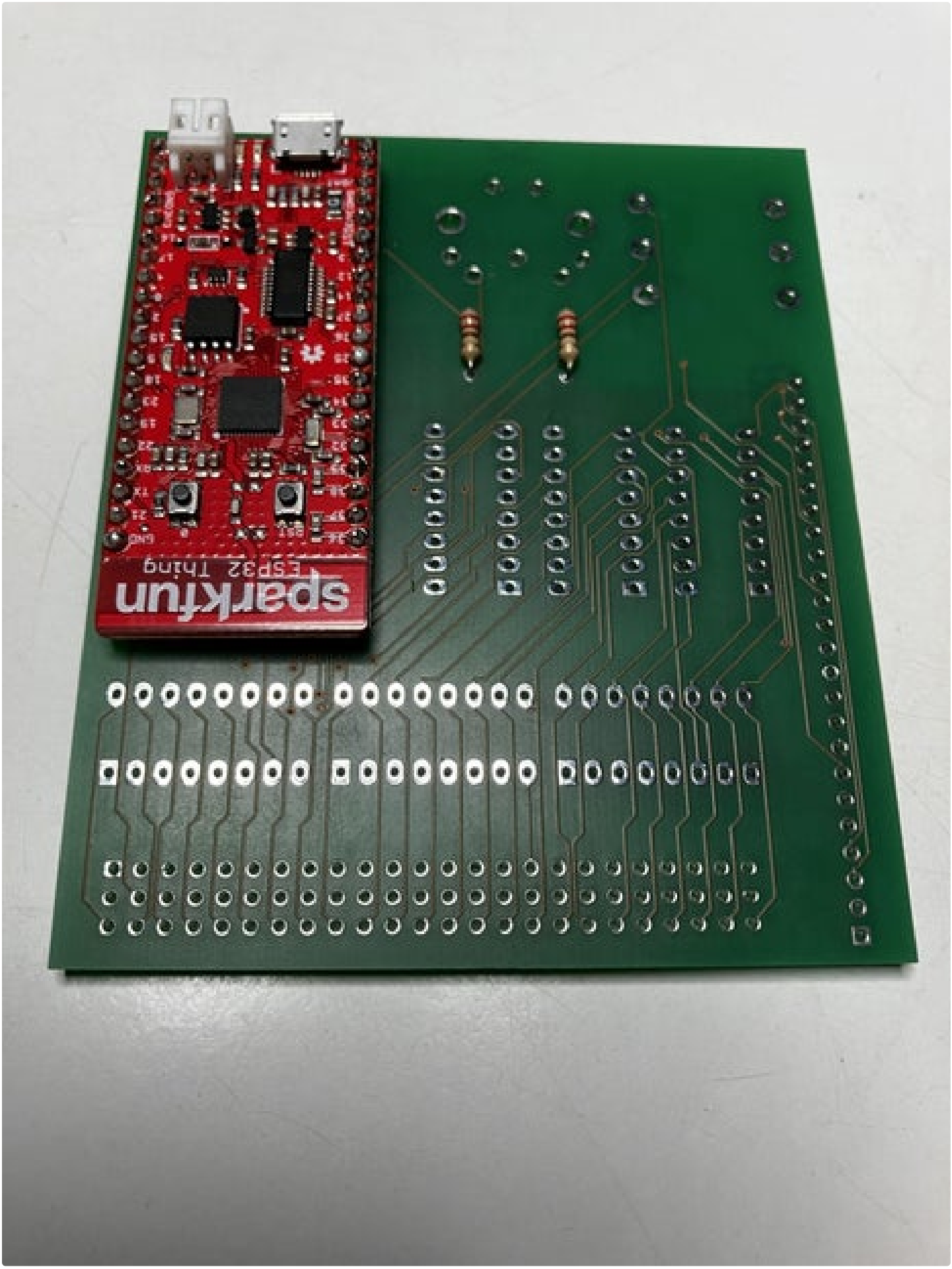
to the board. If you somehow don't have enough sockets for the chips, it's probably relatively safe to solder the resistor array chips without sockets. These are located above the microcontroller. The ones on the side are the shift registers. I strongly advise you to use sockets for these. If you experience trouble putting them in place for soldering, use some adhesive tape.

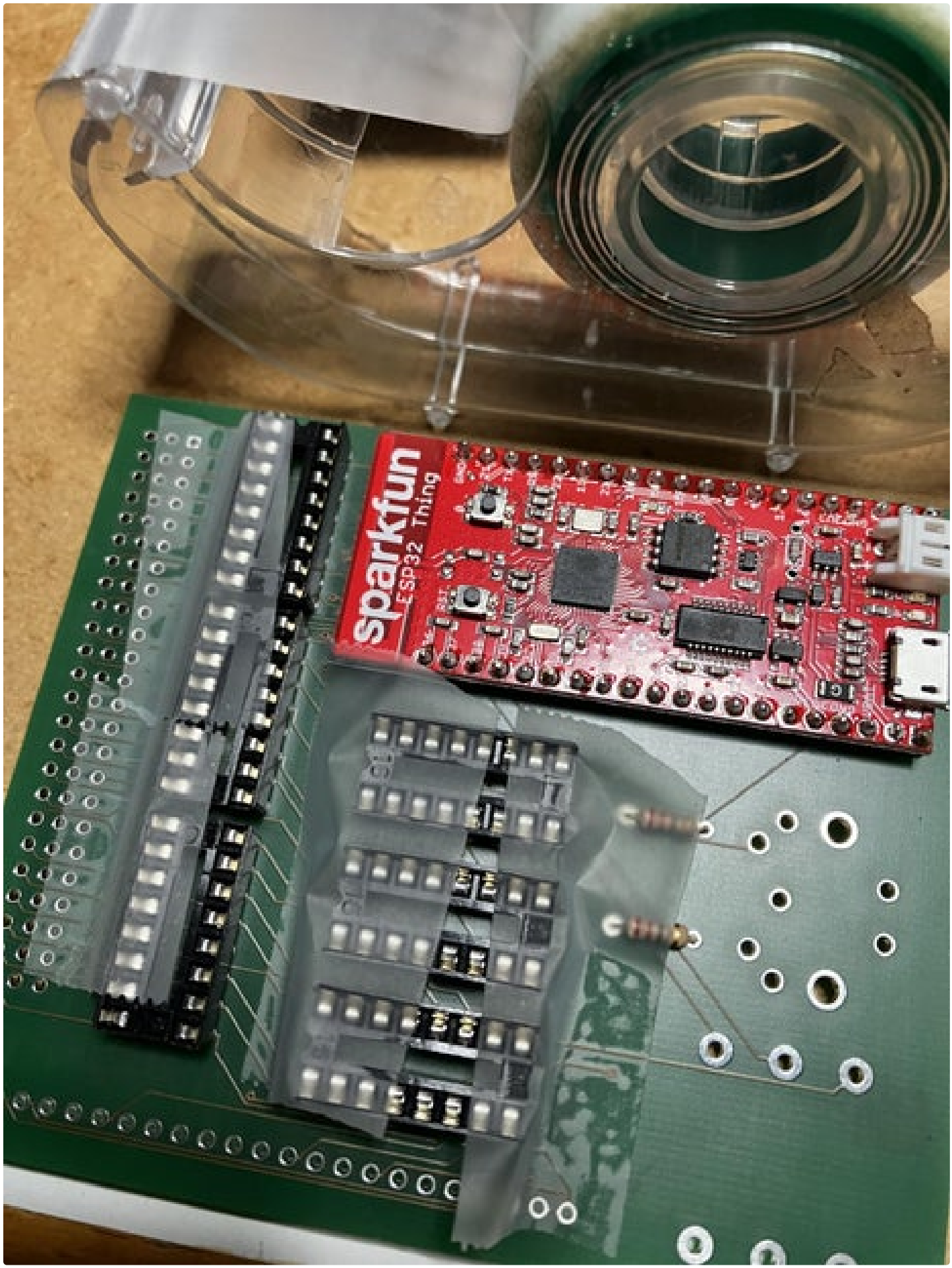
- Go on with the TRS socket. Make sure you are using TRS sockets and not TS sockets. They will not work for the expression pedal. You can recognise them having three metal plates (TRS) in stead of two (TS).
- Finally solder the MIDI socket in place.
- Install the IC's. The resistor array chips are located above the ESP32, the shift registers are on the left of the microcontroller.

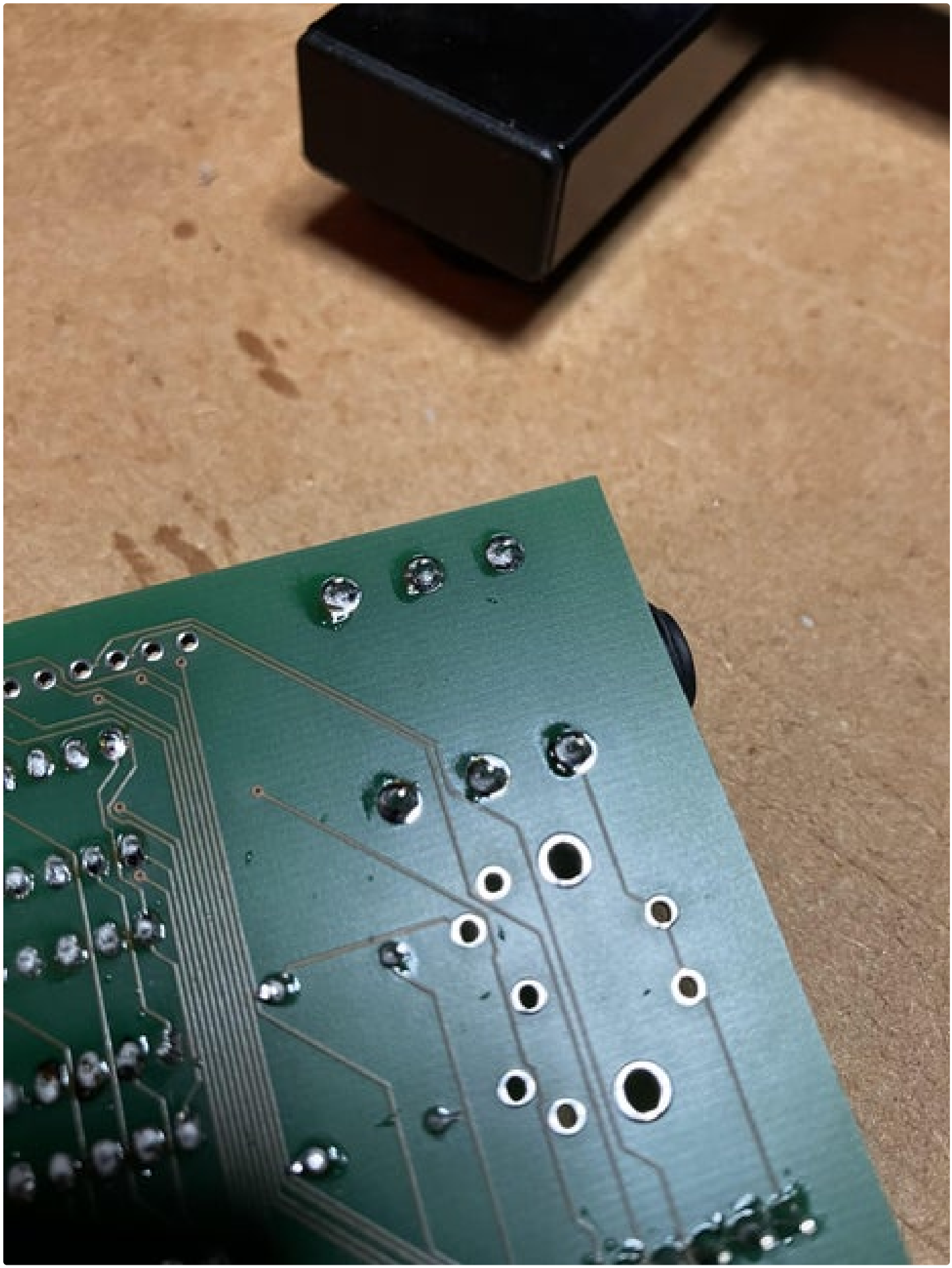
You're done with the SensorBoard. Good work!! Pat yourself on the back. Drink a cup of tea. Chill out.

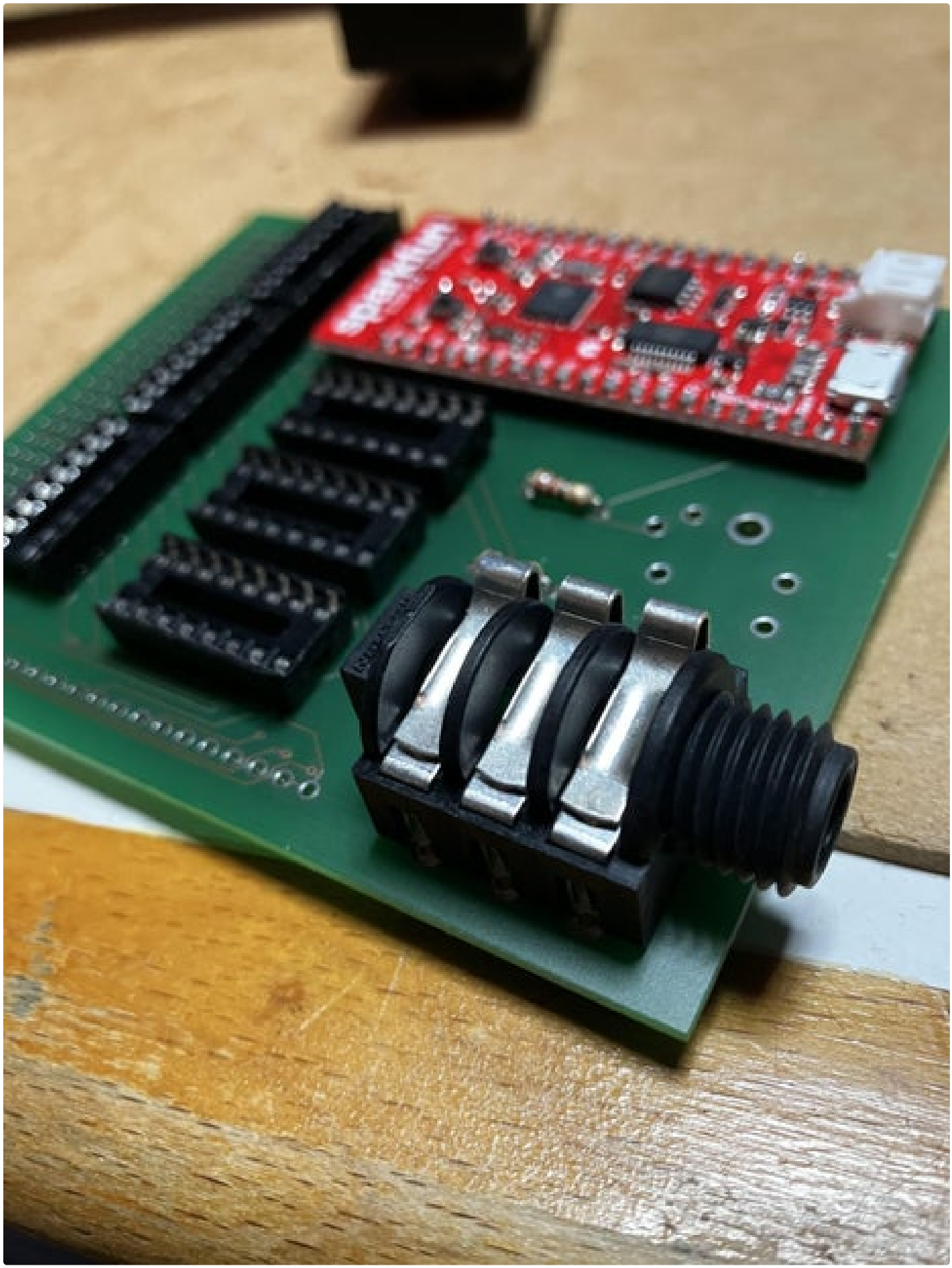


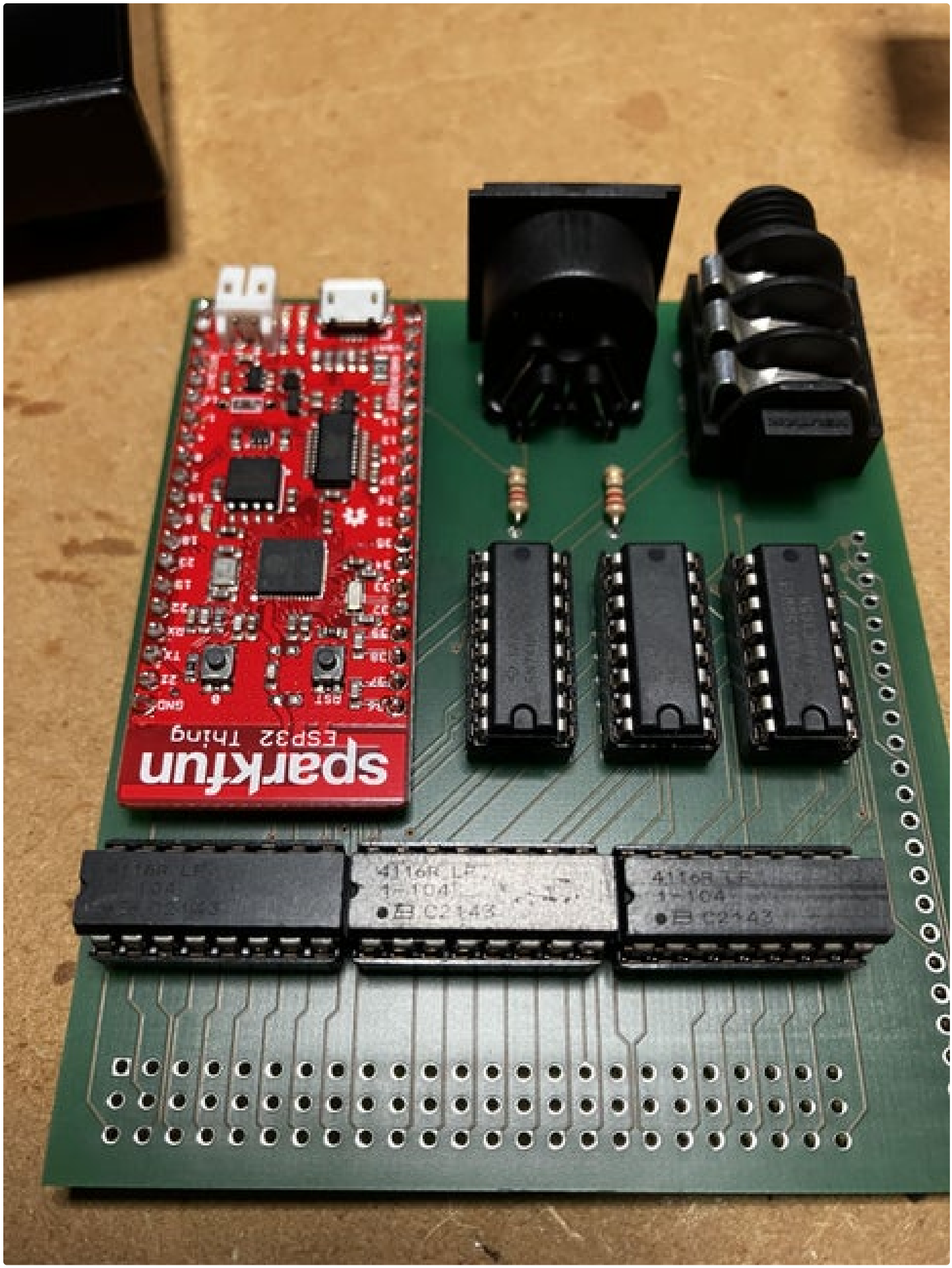



sparkfun
ESP32 Thing


sparkfun
ESP32 Thing






sparkfun
ESP32 Thing
4116R LF
1-104
C2143

Step 12: Installing ESP32 for Arduino

The ESP32 is a great microcontroller. It's fast, relatively cheap and has great features such as BLE MIDI, WiFi capabilities, capacitive touch sensor inputs and more goodies. The Sparkfun 'ESP32 Thing' is a handy board around this chip by Sparkfun and is completely open source hardware. I chose to work on this microcontroller because of the built in BLE and also because I'm planning to make an app to make customising MIDI implementation a breeze. I'm not there yet, but I'll surely keep this instructable updated.

The ESP32 has it's own IDE, but to keep things simple we'll be programming our board with Arduino IDE. In order to make the Thing compatible, you should install some libraries in the Arduino IDE. It's pretty straightforward if you follow the steps presented here:

https://learn.sparkfun.com/tutorials/esp32-thing-hookup-guide/

here:

Installing ESP32 in Arduino IDE (Windows, Mac OS X, Linux)

or here:

https://github.com/espressif/arduino-esp32/blob/master/README.md

Step 13: Installing Libraries for Arduino

In this project we will be using two additional libraries: MIDI.h and BLEMidi.h

Please install these too. It's pretty easy:

- In Arduino IDE go to tools/Manage Libraries
- Search for MIDI I/Os for Arduino and click install
- If you don't find it, check here

- Now again in the tools/Manage libraries search for ESP32-BLE-MIDI
- Click install
- If you don't find it, check here

Ready to go!

Step 14: Testing the Expression Pedal

Let's first test the microcontroller.

Plug the ESP32 Thing in your computer. After a second or so, the light should start blinking. This is the standard patch which is loaded on the microcontroller.

First let's test the expression pedal.

The system is made for a standard expression pedal like the Roland EV-5. This pedal is actually a potmeter of 50kOhms with a mechanic system built around it. So any pedal with an internal resistance around 50k should do. Some brands state 10k is more standard for expression pedals, but this just isn't true. I think the Roland pedal is still pretty standard.

Anyway the EV-5 will work out of the box. If you use another pedal type you might have to add another resistor in series somehow. I had some bad luck with a 'Mission Engineering EP-1' with 10k. It needs a separate TRS cable. If you use a cable with Neutrik connectors, you can in fact desolder the connection and put another resistor in series on the ring connector. I'm planning an update of the SensorBoard PCB where you can install an optional resistor to maximise compatibility. But for now: stick to the Roland or similar. You can get away with resistance values between 30 and 70kOhms approximately.

Let's test!

- Download the Arduino code files and open Pedaltest.ino in the Arduino IDE.
- Plug in the pedal and connect your SensorBoard to the computer with a USB cable.
- Make sure you select the right COM port in Arduino IDE
- Upload the program to the ESP32

Step 15: Testing the Sensors

Now it's time to test your previous soldering job on the SensorPCB. I know, it's getting tedious, but again better safe than sorry. I'm not going to say this again, but if one sensor fails to work, the saxophones MIDI output will be faulty on every note!! Fortunately the sensors themselves are pretty sturdy and we'll make them waterproof later on with the Sugru. So it all depends on you initial build quality. So let's test them properly!

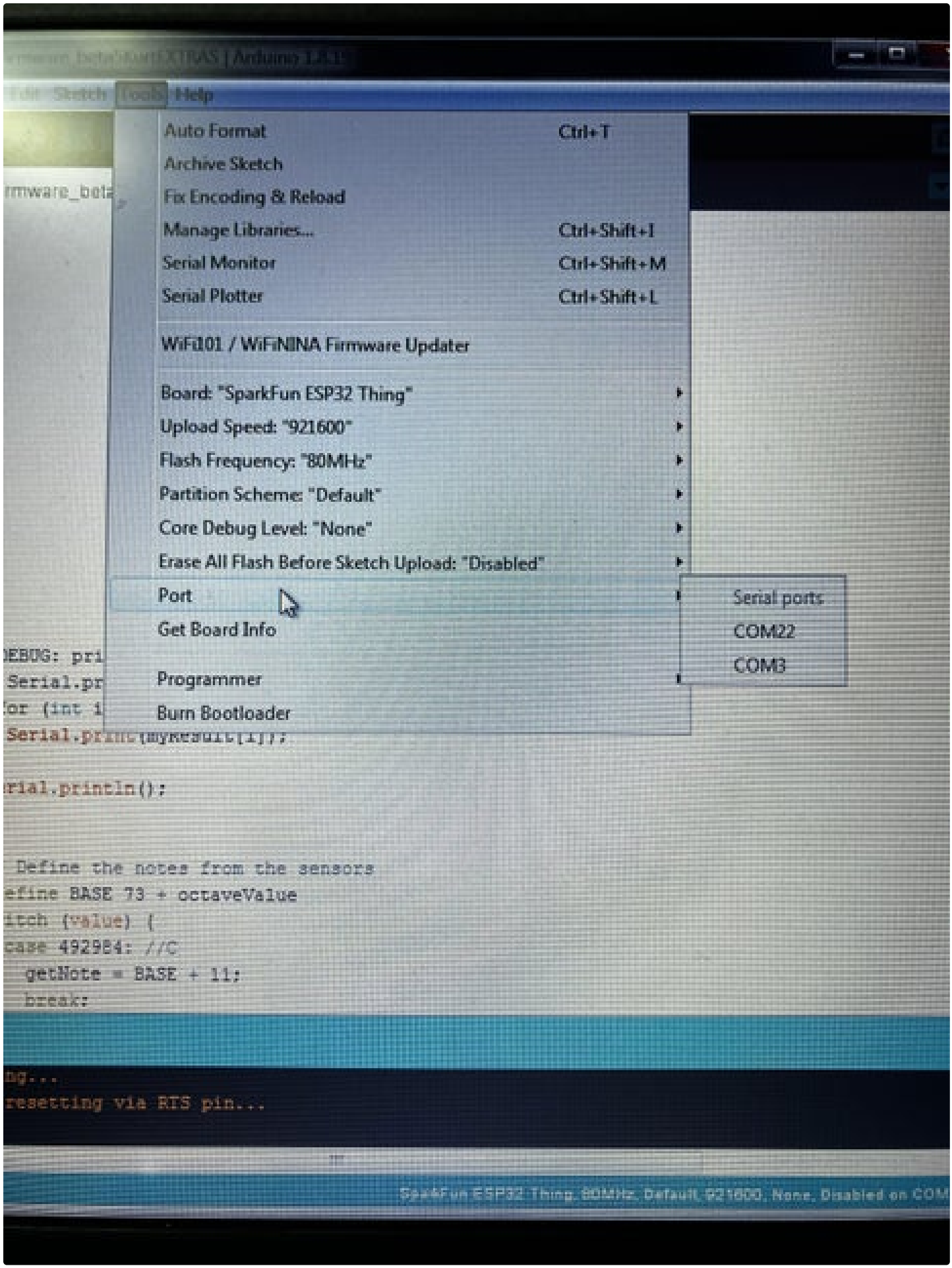
- Download and open the TestSensor.ino file. Upload it to the board. Uploading make take longer than expected if you are used to working with Arduino, Teensy or similar boards. This is due to the conversion process needed to work with ESP32 in the Arduino world. Be patient!
- Take your sensor
- Use the alligator clips to clamp the stripped end of the cables

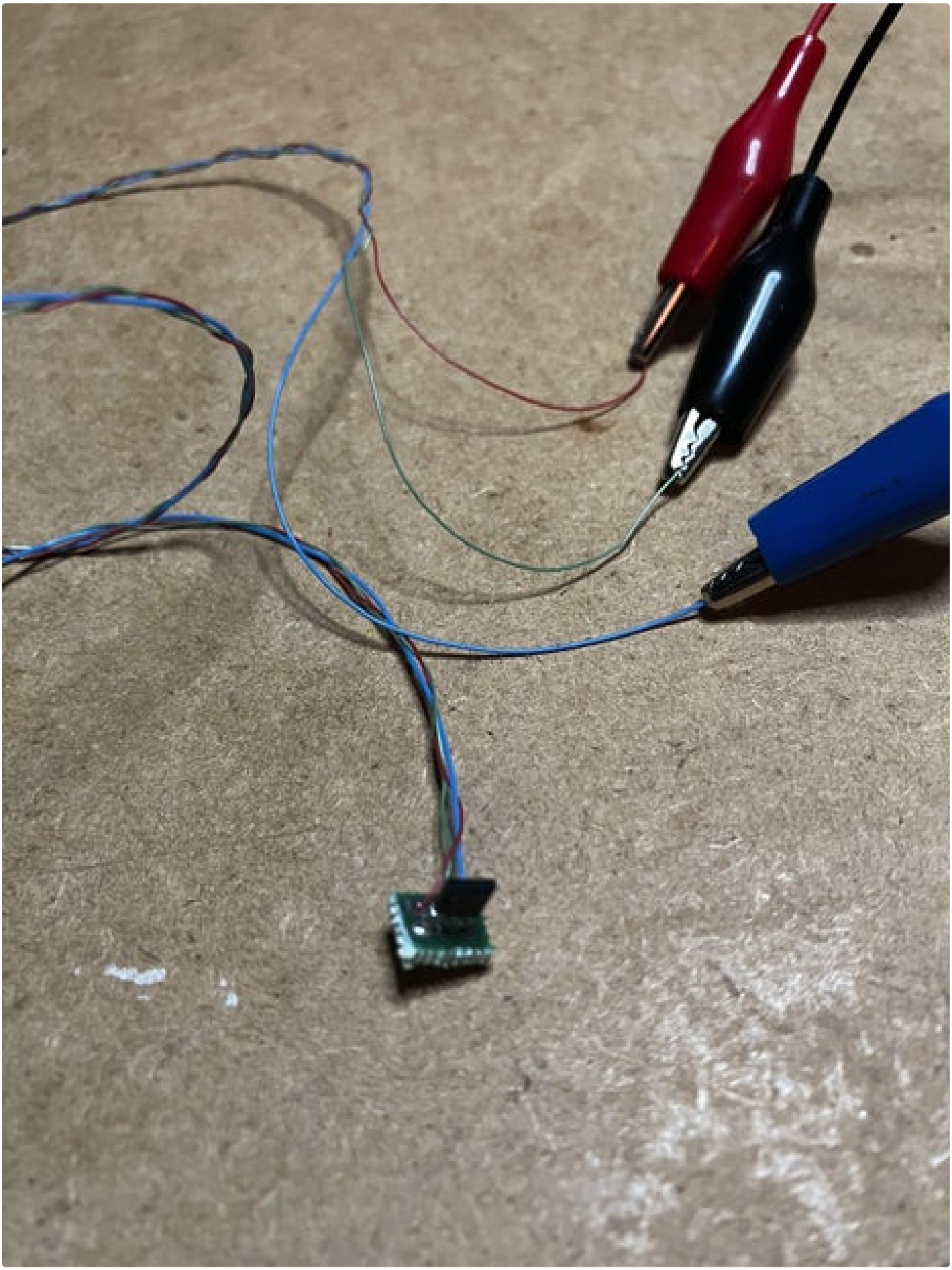
If you are looking at the SensorBoard with the USB, MIDI and TRS connection facing down, the three rows of pin holes at the top of the board are the inputs for the sensors. They are numbered from RIGHT to LEFT. The top row is the actual sensor input, the second row is the GND connection and the 3rd row is the voltage supply (3.3V).

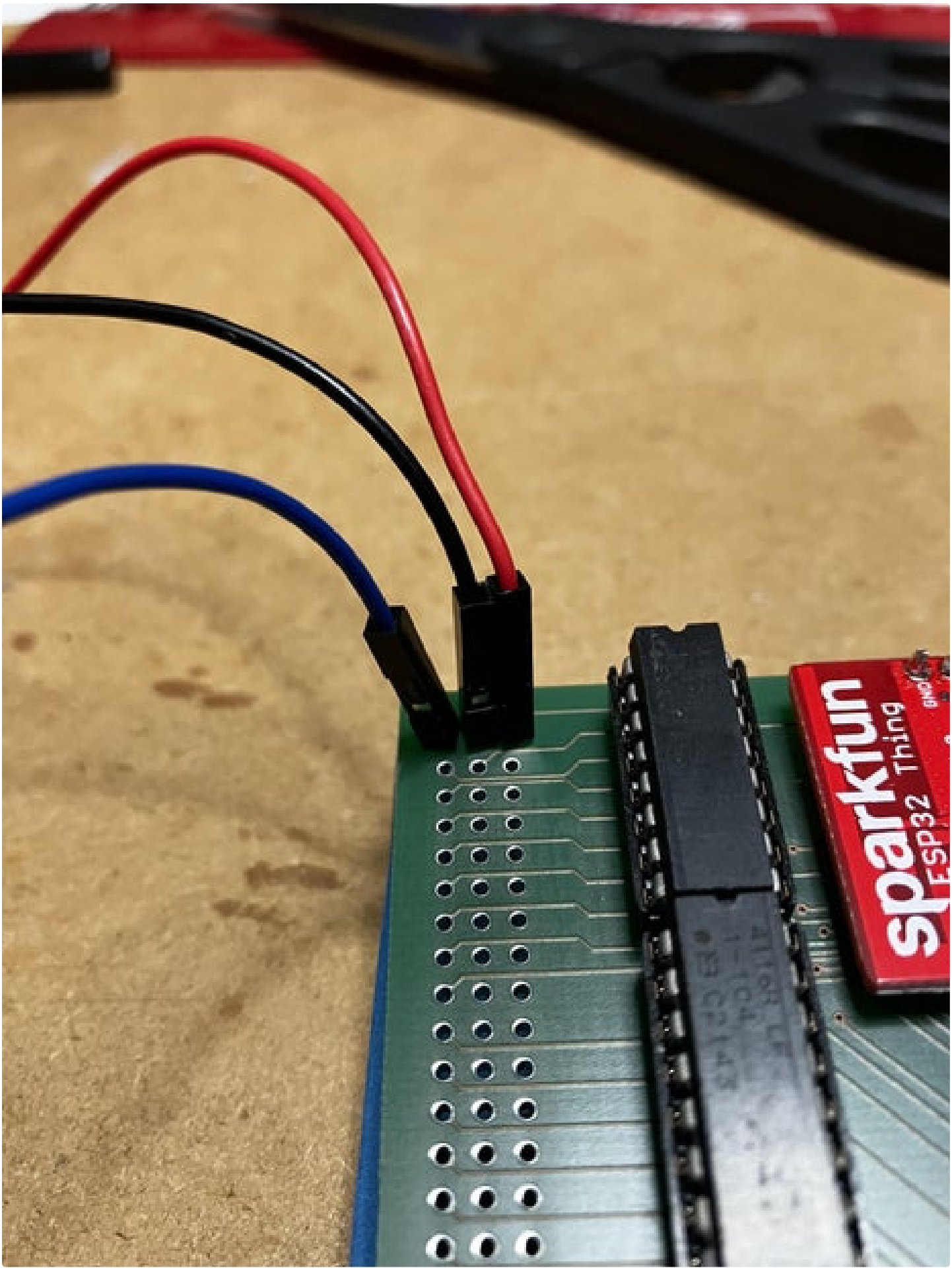
- With the TestSensor sketch running, open the Serial Monitor in the Arduino IDE (tools/Serial Monitor).
- Make sure the Serial Monitor is running at 115200 baud rate. If not, change it!
- If you see strange characters running on your Serial Monitor going from left to right, don't panic! just push the left button on the ESP32 Thing (reset) and you'll be fine.
- You should see a row of 24 zeros running down the Serial Monitor. If you see something other than zeroes, you probably made a mistake while soldering the board. Check your solder joints. If you have this problem, try to find and correct your soldering mistake. If you cannot find any, I'm afraid you'll have to make another SensorBoard...
- If you only see zeroes, congratulations, we can now start testing the sensors. Use the male end of the wire connected to the sensor and connect them to the SensorBoard as shown on the picture. Make sure the top row is your sensor, second row GND and third row +3.3V.
- Make sure the metal pin is touching the pin on the SensorBoard. I use my right hand to keep them in place. Take your magnet and move it towards the sensor. The sensor should turn 1 on the Serial Monitor

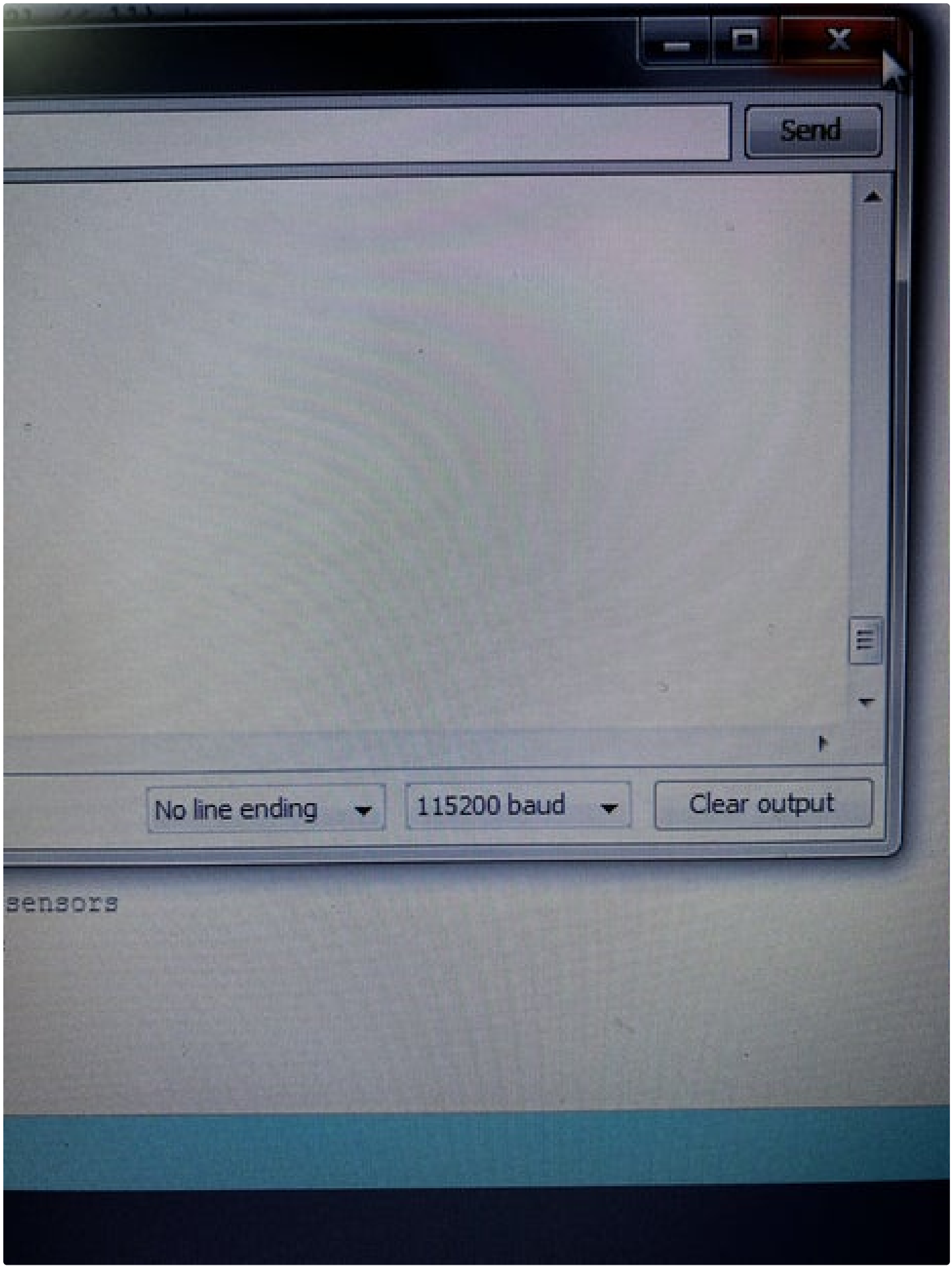
on the corresponding line. If not, change the position of the magnet. The hall switches will respond only to the south pole of the magnet, so figure out which side is what. I used metal tweezers to stick the magnet on and turned it around to check which side is switching the sensor.

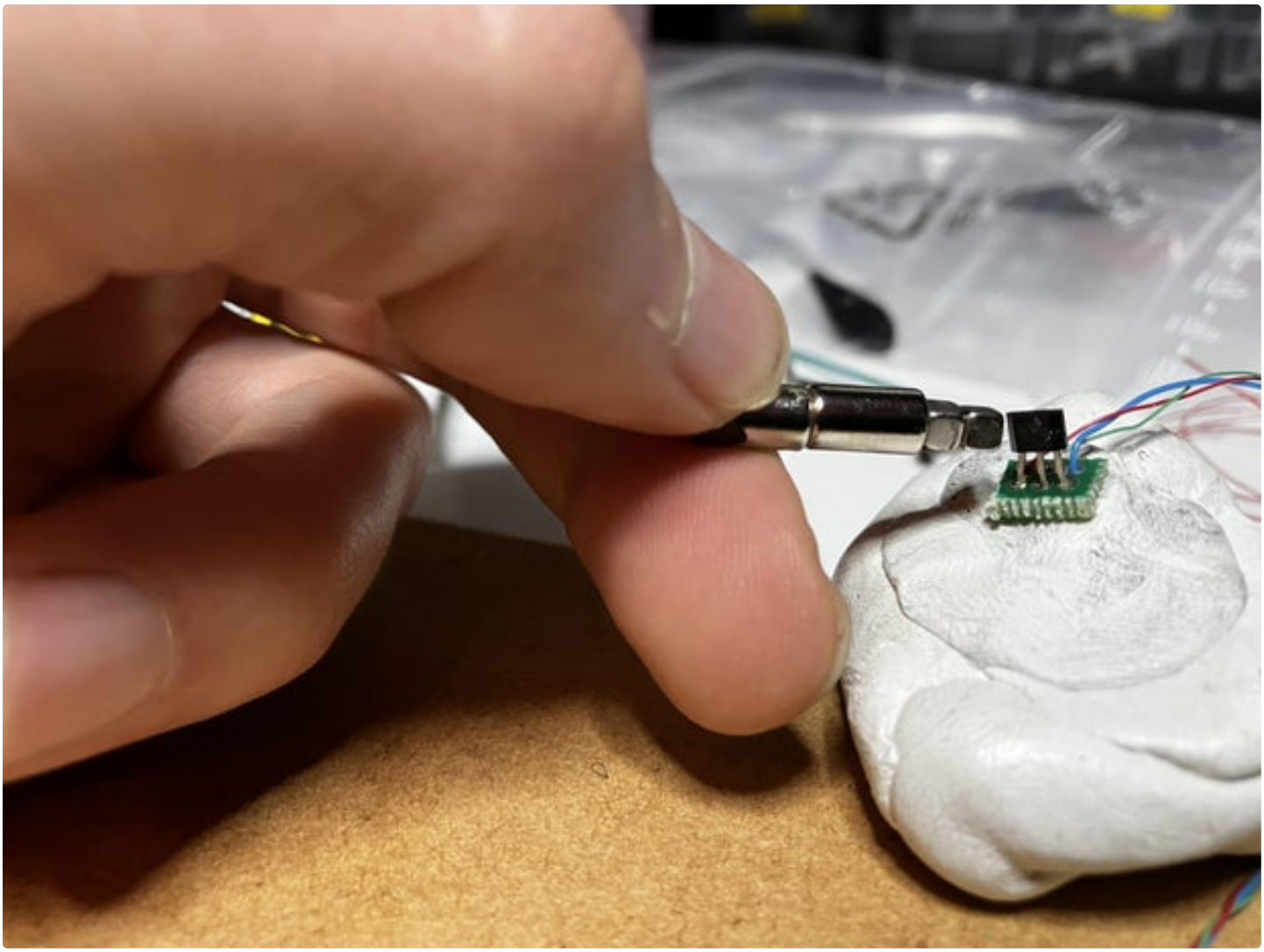
- As you can see, there are 24 inputs, from right to left holding the SensorBoard with the connectors facing down. The zeros on the Serial Monitor correspond to these inputs. First check wether all inputs on the board work. You can do this very easy by testing all the pins like you did the first. So the second pin should correspond to the second zero from the right on the Serial Monitor and so on.
- Once you made sure all the inputs on the SensorBoard are working, check all the sensors you made. Of course you don't have to check all the pins again, but make sure all your individual sensors on the SensorPCB are working properly. If you have a faulty sensor, get rid of it immediately in a responsible way. Of course you can reuse your wires and cut your sensor for future use, but the SensorPCB will be lost...
- If you have at least 21 working sensors on your hand, you're on the safe side!

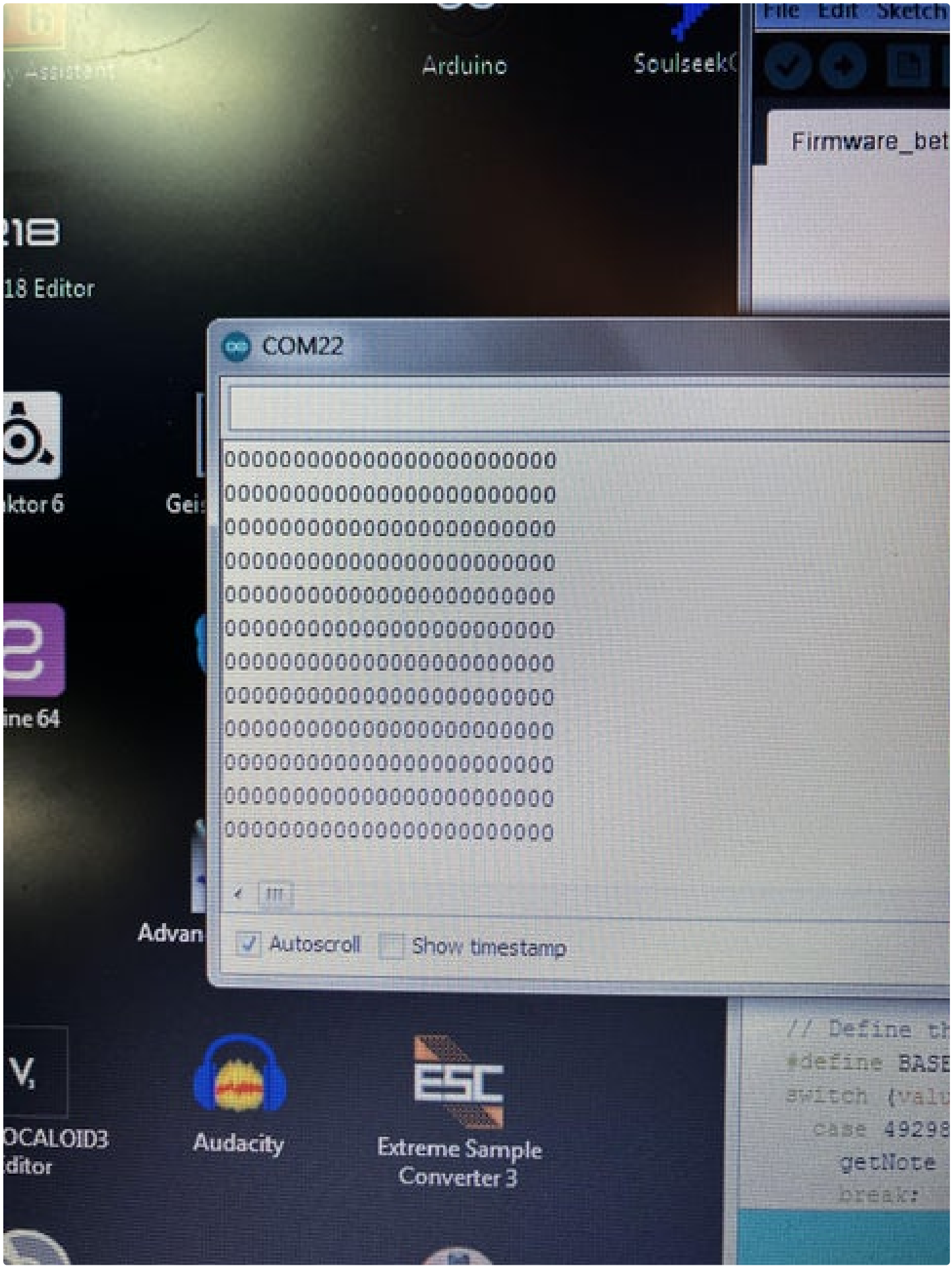

Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "SparkFun ESP32 Thing"
Upload Speed: "921600"
Flash Frequency: "80MHz"
Partition Scheme: "Default"
Core Debug Level: "None"
Erase All Flash Before Sketch Upload: "Disabled"
Port
Get Board Info
Programmer
Burn Bootloader
Serial ports
COM22
COM3
rial.println();
Define the notes from the sensors
efine BASE 73 + octaveValue
itch (value) {
case 492984: //C
getNote = BASE + 11;
break;
resetting via RTS pin...

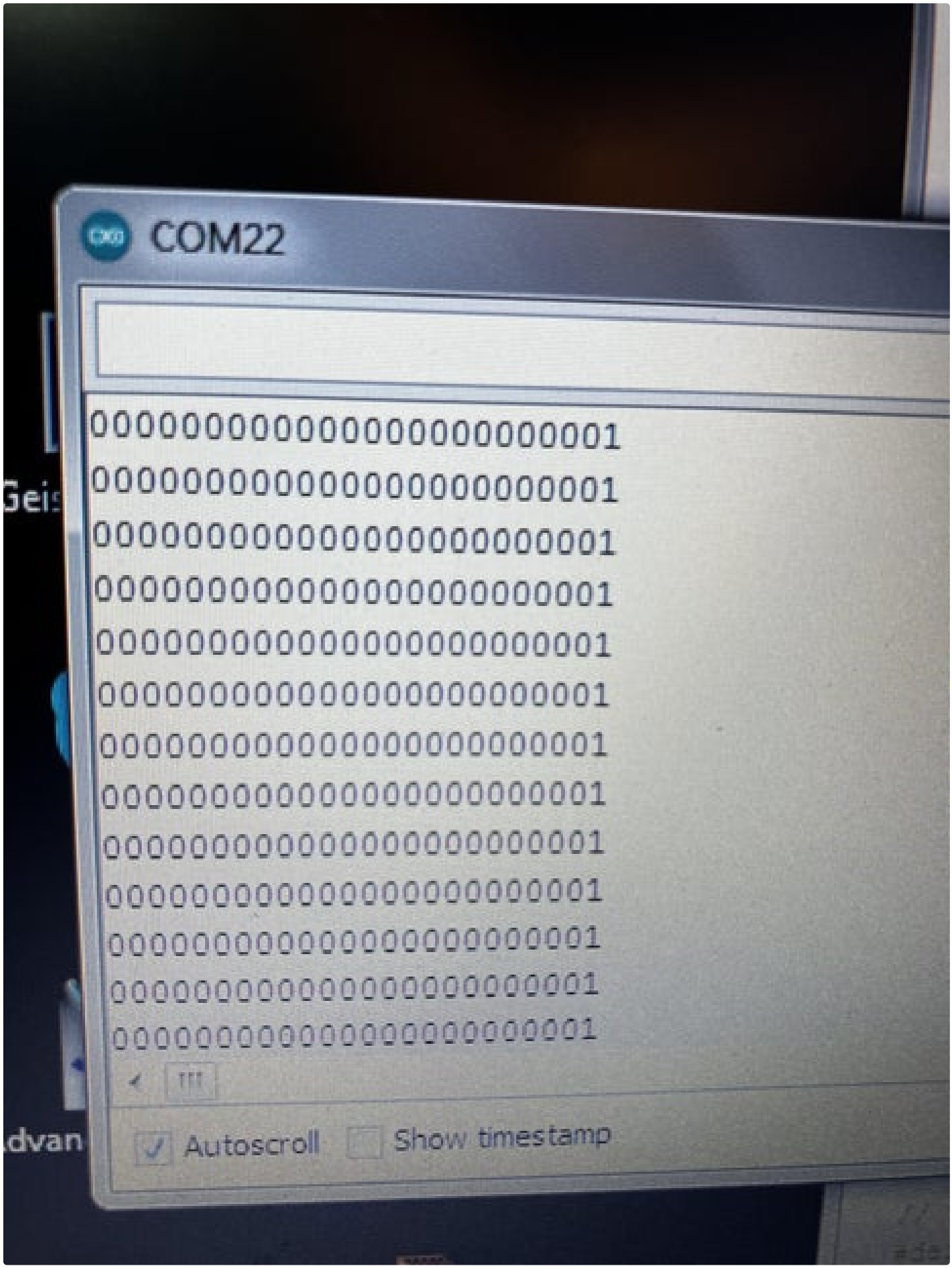


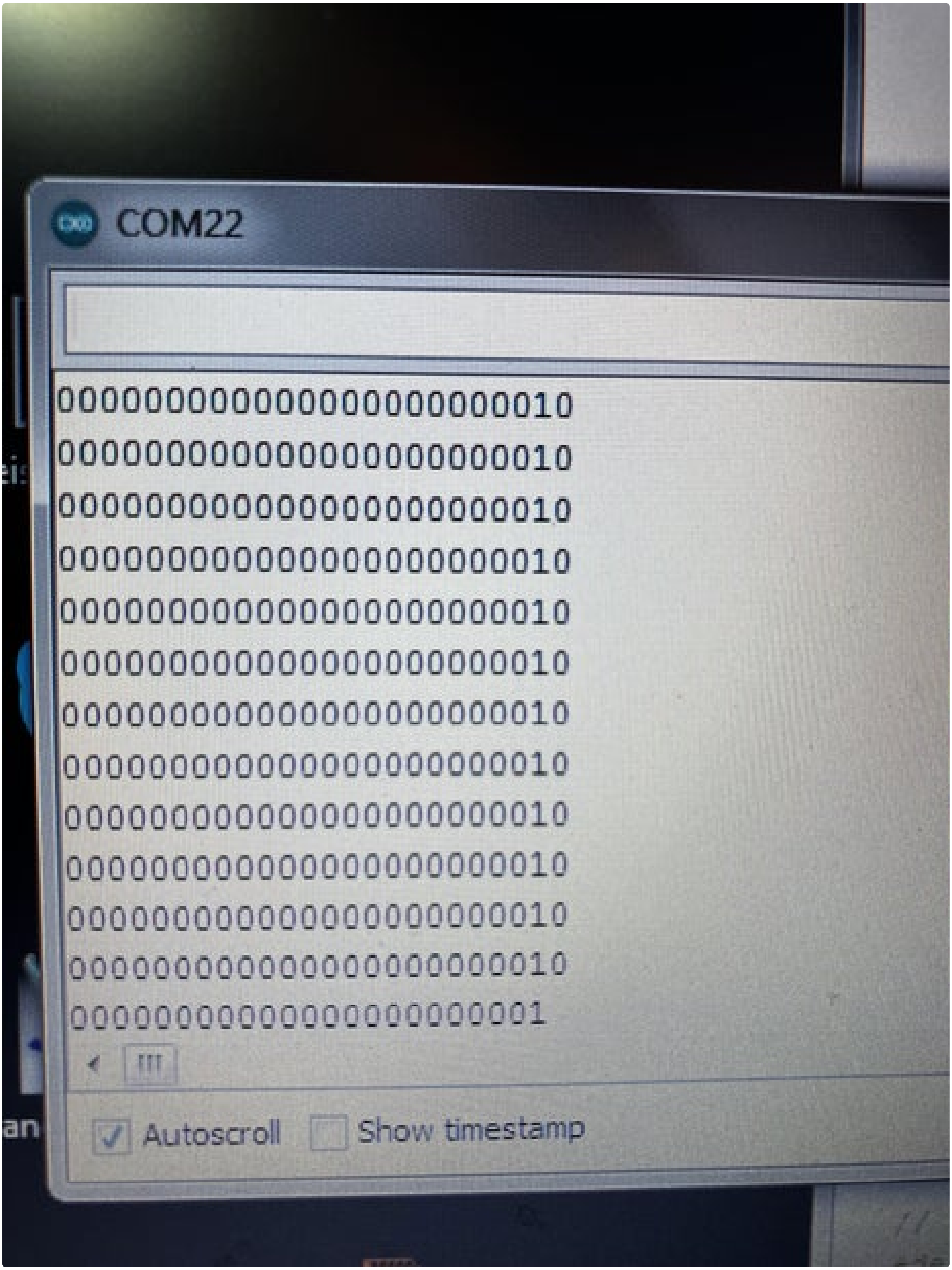

sparkfun
ESP32
Thing


Send
No line ending
115200 baud
Clear output
sensors




Arduino
Soulseek(
File Edit Sketch
Firmware_bet
18 Editor
ktor 6
Gei
ine 64
Advan
COM22
Autoscroll
Show timestamp
Audacity
Extreme Sample Converter 3
OCALOID3
ditor


COM22
00000000000000000000001
00000000000000000000001
00000000000000000000001
00000000000000000000001
00000000000000000000001
00000000000000000000001
00000000000000000000001
00000000000000000000001
00000000000000000000001
00000000000000000000001
00000000000000000000001
00000000000000000000001
Autoscroll
Show timestamp


COM22
00000000000000000000010
00000000000000000000010
00000000000000000000010
00000000000000000000010
00000000000000000000010
00000000000000000000010
00000000000000000000010
00000000000000000000010
00000000000000000000010
00000000000000000000010
00000000000000000000010
00000000000000000000010
00000000000000000000001
Autoscroll
Show timestamp

Step 16: Putting the Sensors on the Sax

Let's glue the sensors to the body of the sax. We will be using a nice putty glue called 'Sugru'. When you open the package, you have about an hour to put everything in place, before the paste starts to harden. Sugru is non toxic, sticks to the saxophone body pretty good and can be easily and safely removed if necessary. This was the whole idea: to make the process reversible without leaving marks and stains on the horn.

So don't be afraid if you are working on an expensive saxophone, you can just pull them off if you apply moderate force. Should the sensor get loose unintended, you can go for something stronger like 'Bison all purpose glue'. Ask your saxophone luthier for help choosing a stronger adhesive if you care about not damaging the lacquer.

Apart from sticking the sensor to the body of the sax, we will also wrap the sensor in Sugru, making them water resistant.

To start: take a small piece of Sugru and knead it to a ball of approximately 3cm. Wrap the sensor almost completely in putty, leaving only the black box uncovered. Make sure you have enough thickness on the bottom part though. I had to learn it the hard way that not enough Sugru on the bottom means electrical short. Most saxophones are electric conductors... yeah I know... I went for the soldered option (see later steps) with the solid core wire first and had to start all over again... That was quite frustrating. So make sure the bottom of the sensor has enough Sugru on it to isolate it from the sax!!

On the pictures you can see I removed all the keys of the sax. This is not necessary. I tried both ways on different saxophones and I can conclude it's not worth the trouble! So you can safely work on a working sax in good condition. Beware, the G# key will be hard to reach. But it's definitely doable.

You can proceed this way:

- Start with the low B-flat key. Cover the sensor in Sugru leavig the black box of the sensor uncovered and stick it to the body near the hole.
- Go up to B, C , C# and so on. Do not forget the side keys!
- To finish: attach the last hall sensor on the octave key. Be careful, many saxophones have different octave systems. Make sure you put the sensor near a part which always moves when pushing the octave key! Make sure to press middle B, A, G and F and look closely at the mechanism. The idea is of course the sensor will be activated every time you press the octave key. So again: be careful and choose a good spot!!

As you can see, the G key will close two holes at the same time. The idea is you only need to put a sensor on the holes you need. This also means you'll need 2 sensors for the middle B-flat: 1 for the side key, and another for the small key in between middle B and A if you want to be able to use both options. Good luck!

When you're finished, let the Sugru dry out for at least 12 hours.

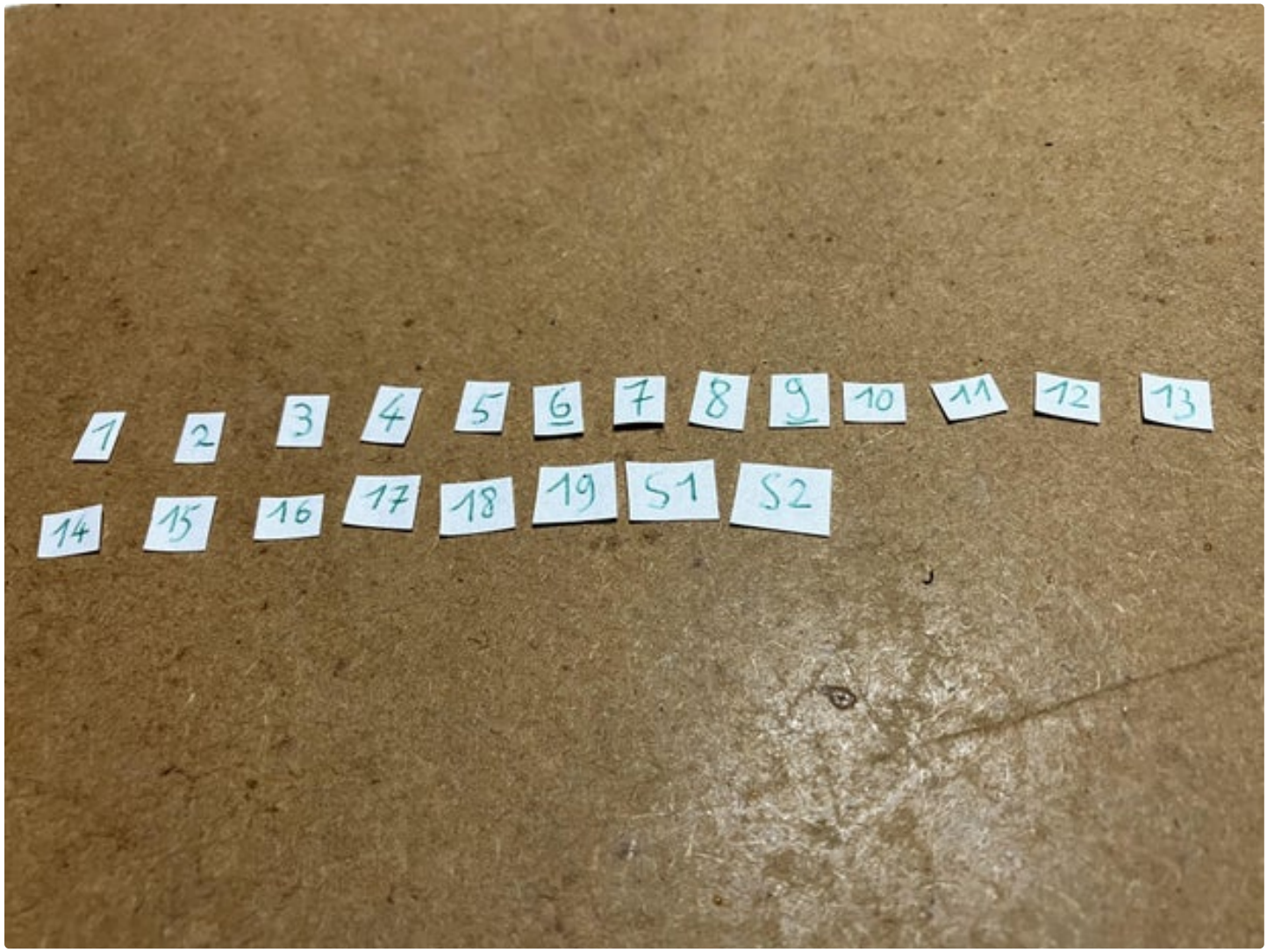

BLACK
tear here

Step 17: Installing the Octave Plates

We will also be using Sugru to support the capacitive touch octave plates to the saxophone. Make a nice, comfortable shape with the Sugru underneath the left thumb rest on the body of the saxophone and push the metal discs in. Done!

Step 18: Paper Numbers to Make It Easy

Take a piece of paper and write numbers 1 to 20 and S1, S2. I wrote the numbers on both sides. Cut them into tiny pieces. We are going to stick them on the cables, so you can easily identify your sensors when you sax is going to be covered in cables. It's just handy.



Step 19: Support the SensorBoard

Now we will make some kind of support for the SensorBoard to be able to attach it safely to the bell of the saxophone.

Since this is a first proof of concept model, I did not provide a proper casing yet.

Learning to design in 3D is next on my list and it's my hope I'll be able to finish a proper casing soon. Meanwhile, whith this guide going public, if anyone reading this would feel inclined to design something. Please do share. That would be fantastic!

So for now let's make something rudimentary that's just sturdy enough to support the SensorBoard. You can come up with a better solution, just make sure the soldered connections are not touching the bell in any way. Yes... short circuit!

Luckily enough the ESP32 has sufficient protection for low voltage shorts, so you wont break it, but still this will result in

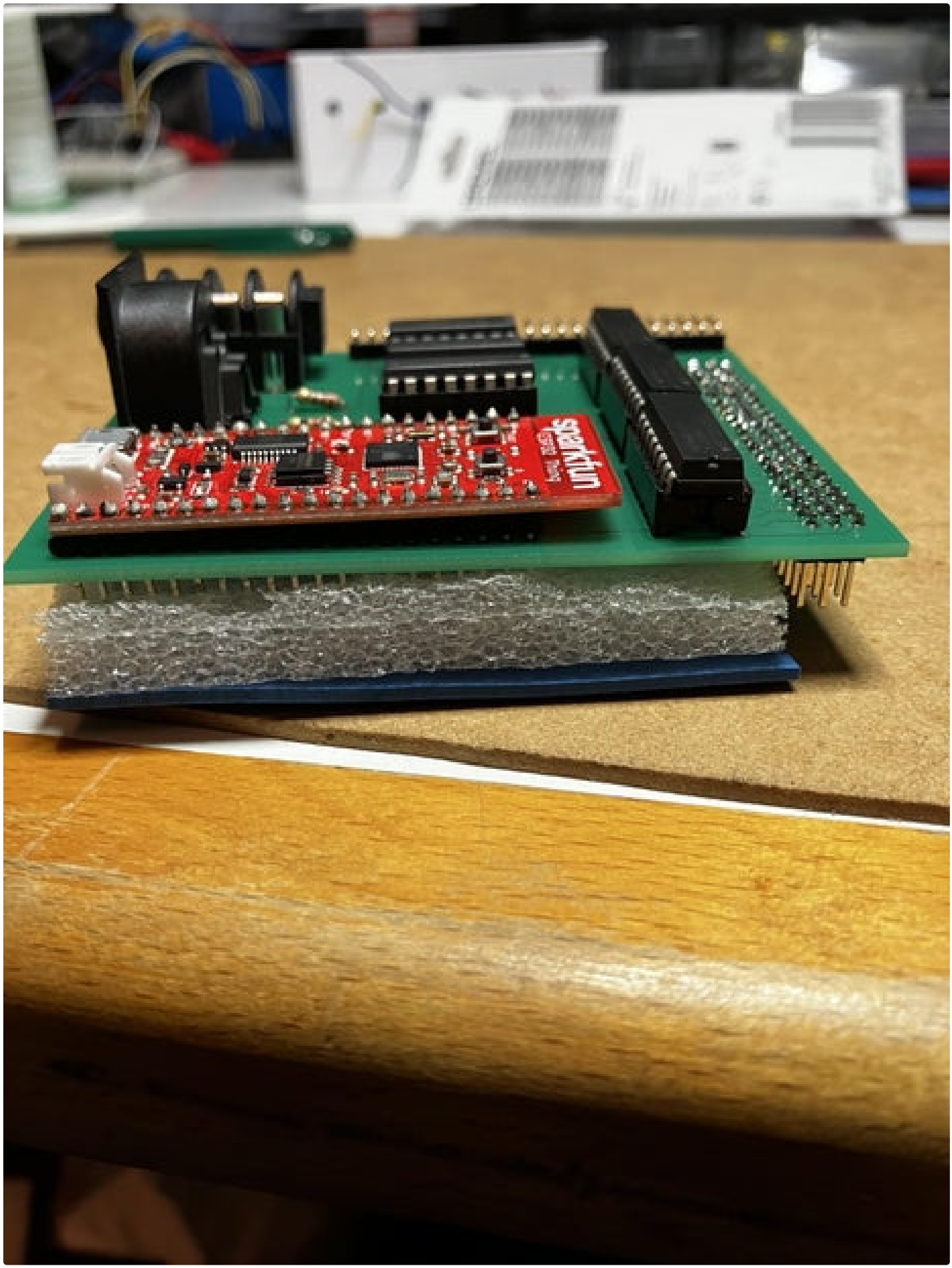
the board failing to power up for as long the short is detected. If this happens, usually this looks like the blue led powering up for a short amount of time, followed by a brief flash of yellow light from a tiny LED next to the USB power connection.

Here is what I did:

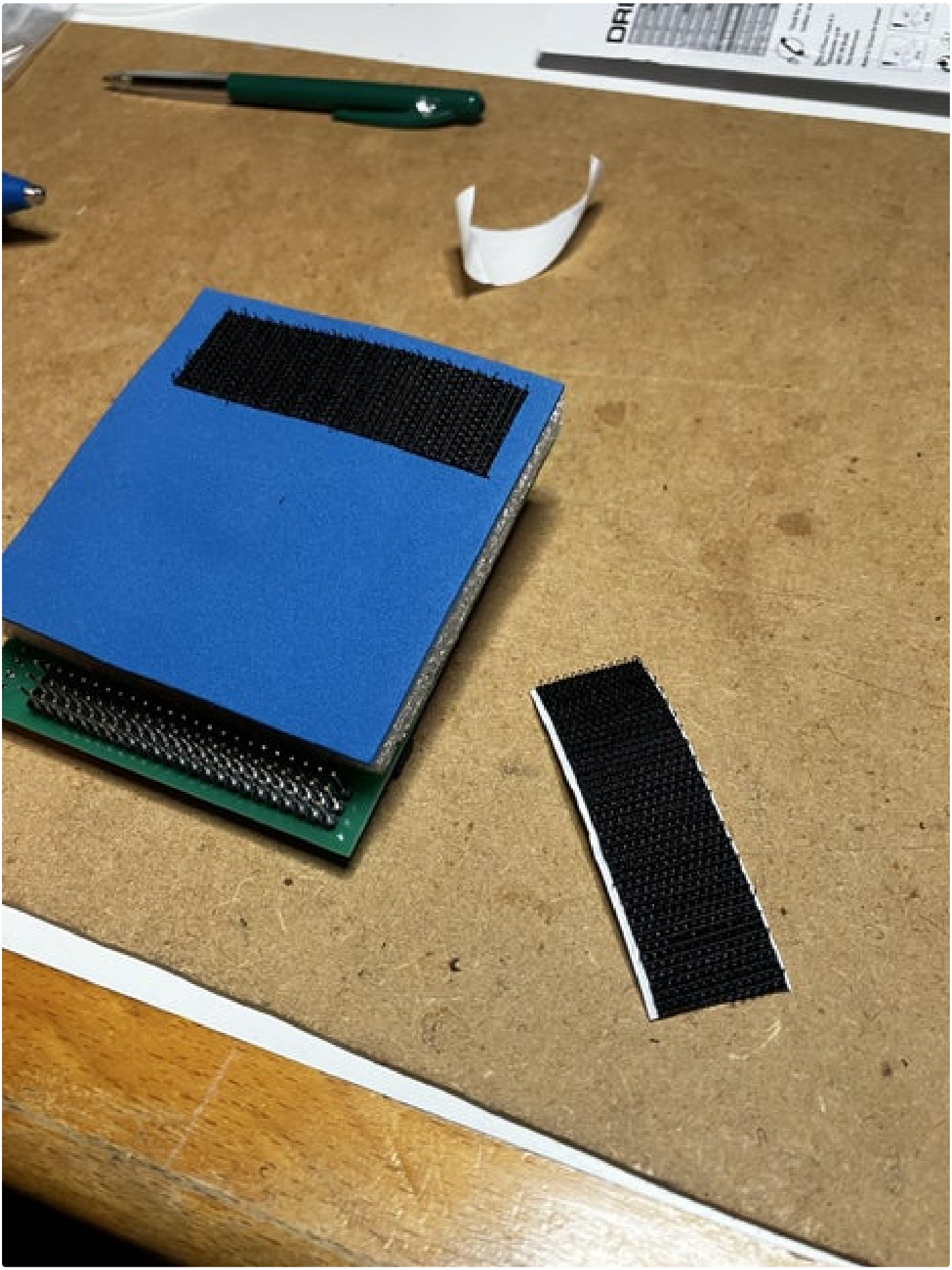
- take a piece of flexible styrofoam and cut it so it's covering the bottom side of the SensorBoard PCB. Leave enough space for the sensor connections.
- Cut another piece of foam or cardboard. I used the type found in shops supplying arts & crafts utilities.
- Use hot glue to stick them together.
- Use hook and loop tape (better known as "velcro") and attach it to the bottom of the board with the styrofoam/cardboard.
- Take the other side (loop?) and let it sit loosely on top of the "hook" side of the tape.
- Push everything firmly against the bell of the saxophone.

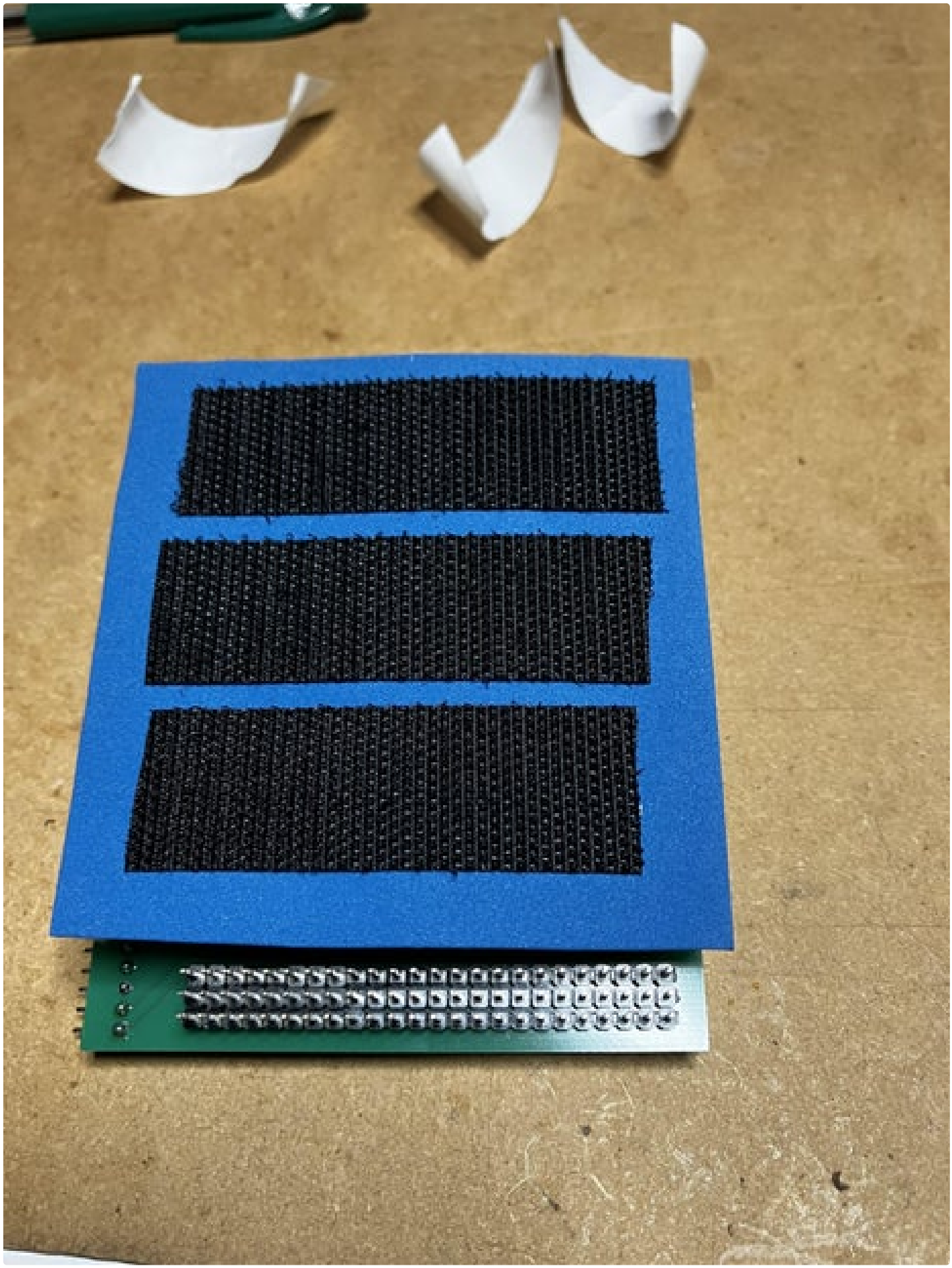
Now you should be able to attach or detach the SensorBoard without too much hassle.

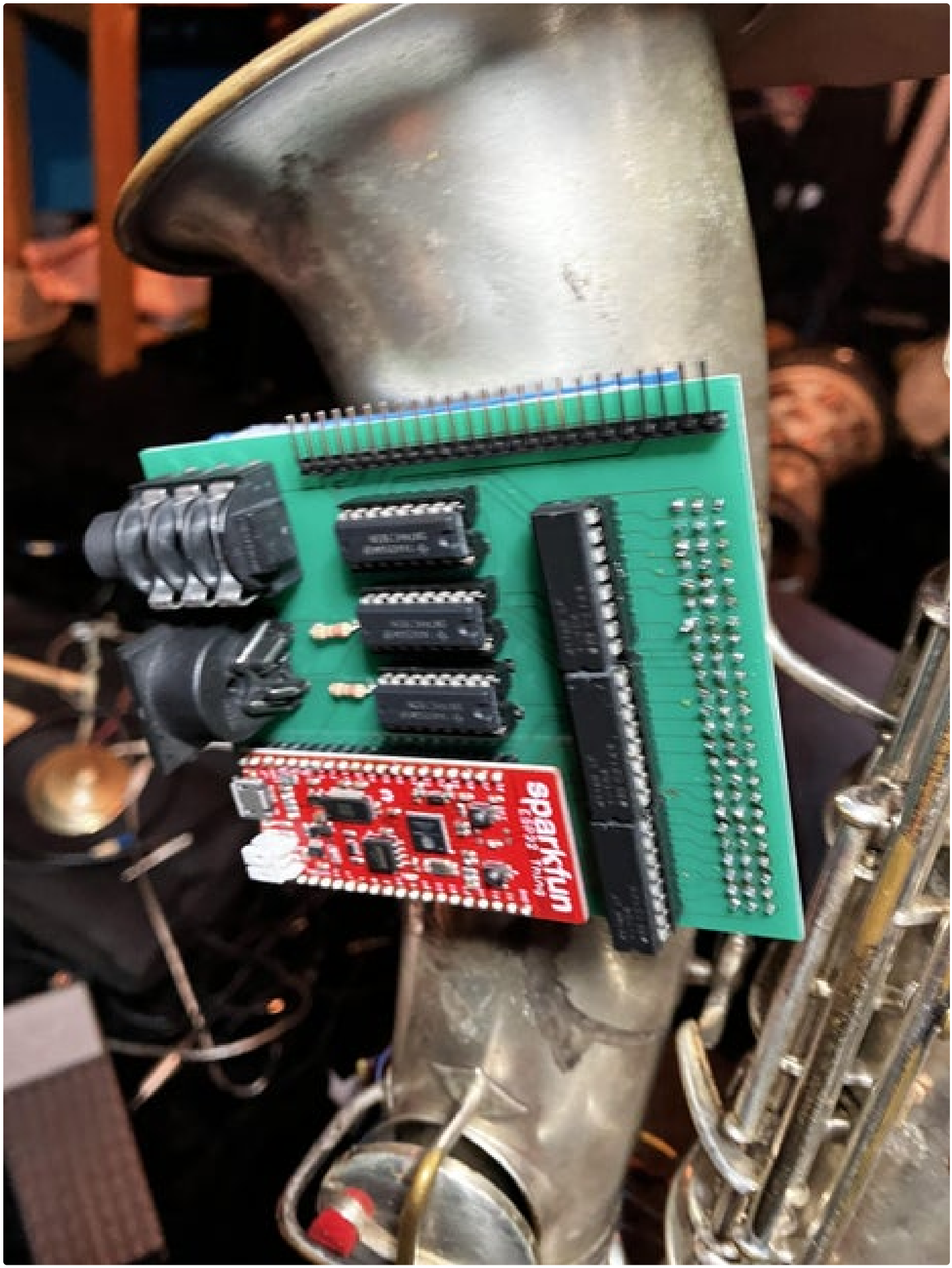

DREMEL


SparkFun


tesa
ON&OFF
Klettband, selbstklebend
Hook & Loop Tape, self-adhesive
Nastro adesivo attacca e stacca
UNIVERSAL
10.000 x
2,5 m x 20 mm
black






sparkfun

Step 20: Choosing Sensor Cable Pins

For this next step you have some choices. We want to connect the individual sensors to the board. These are the options I tried:

- **OPTION 1:**If you are using somewhat thicker cable and have a DuPont connector crimp tool at hand, you could crimp your own connectors. I think my crimp tool was too cheap or something, but I ended up with bad connectors. Yikes. Short circuit. Start over again. No thanks. If you are familiar with making good DuPont connectors, this option is by far the fastest.
- **OPTION 2:**If you are using the very fine very fine 38 AWG cables and feeling sure about your sensors, you might want to solder the sensors directly to the board. This is the most reliable. But I also had some bad luck soldering the small cables to the input. I strongly advice you to solder some regular 28 AWG solid core wire to the end of the sensors and soldering these to the board. I think this is the most reliable, but one mistake somewhere and you can start all over again. Although I have successfully done this, I do not advice this, because I also messed up one time, having indeed to start all over again. Finding the problem and reworking the SensorBoard PCB is a pain, if at all possible, so... I finally went for another option :
- **OPTION 3:** Get some decent pre made female DuPont connectors, cut them, strip them and solder them to the sensor wires. You'll end up with a system you can debug and expand if necessary!

Step 21: Option 2: Soldering Sensors to the SensorBoard

If you are going for the first option, I assume you know better what you are doing than me. So off you go and continue reading from option 3!

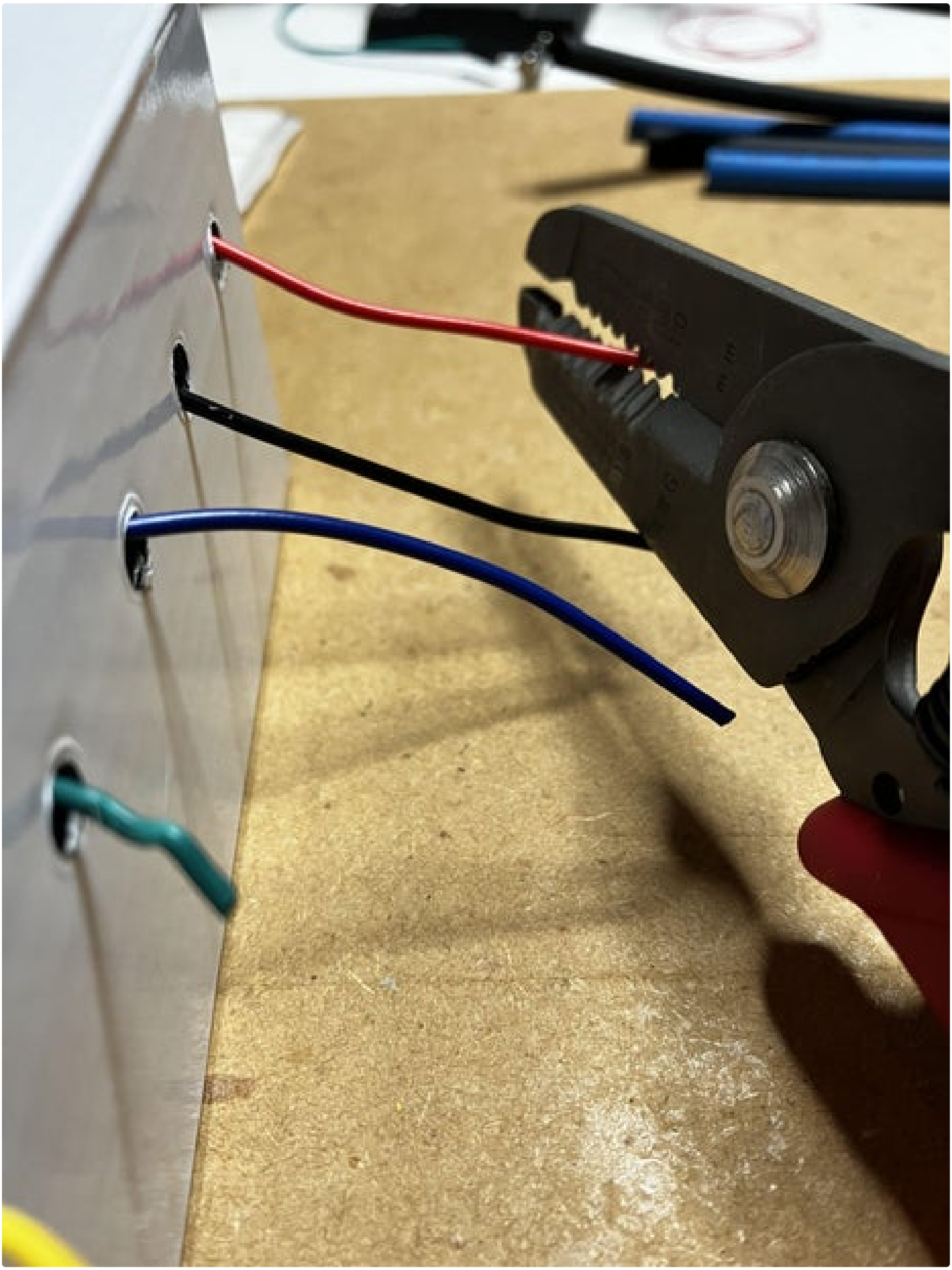
If you want to make a solid version with everything neatly soldered. Pretty brave! Let's go.

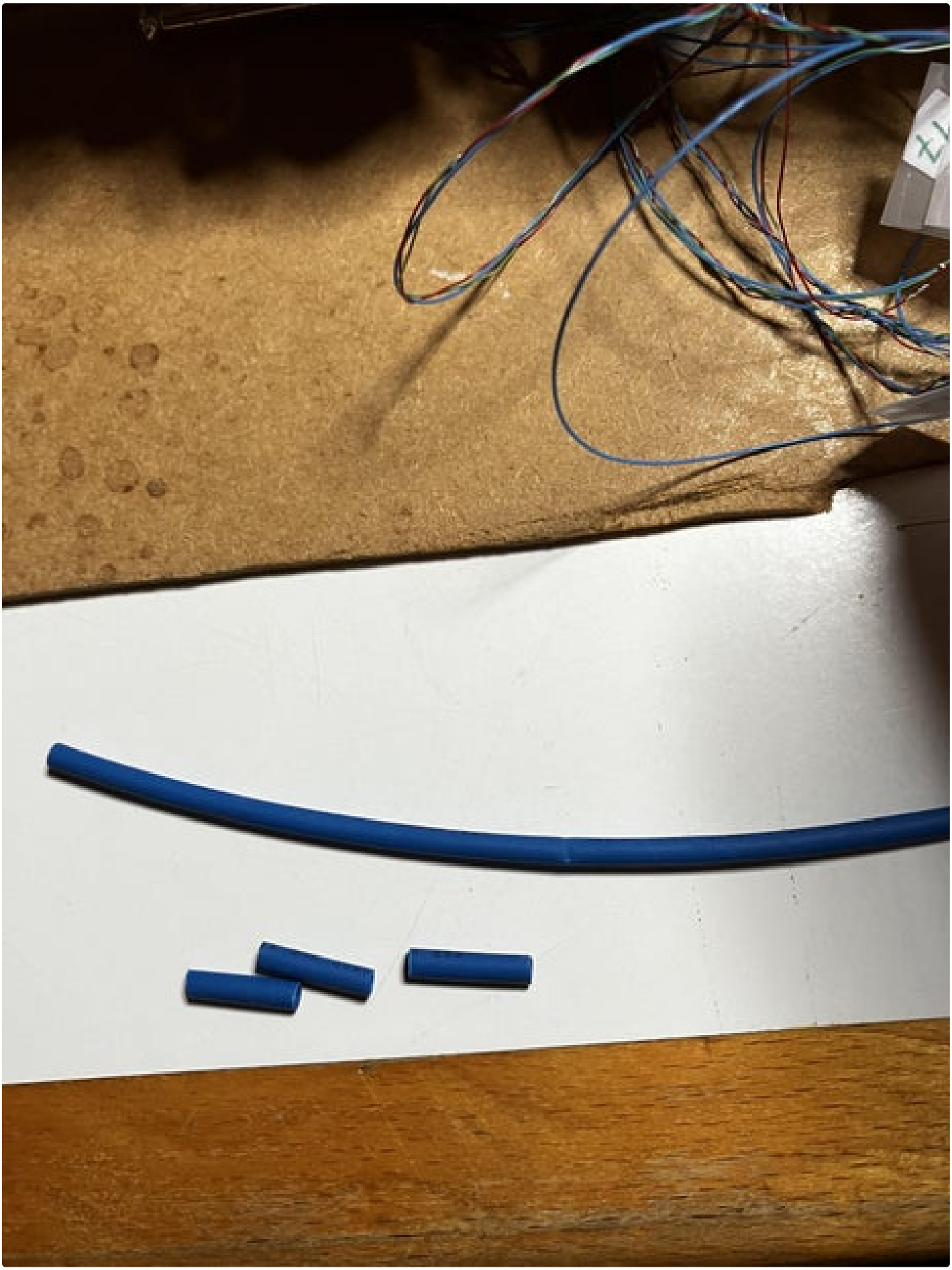
Again two options:

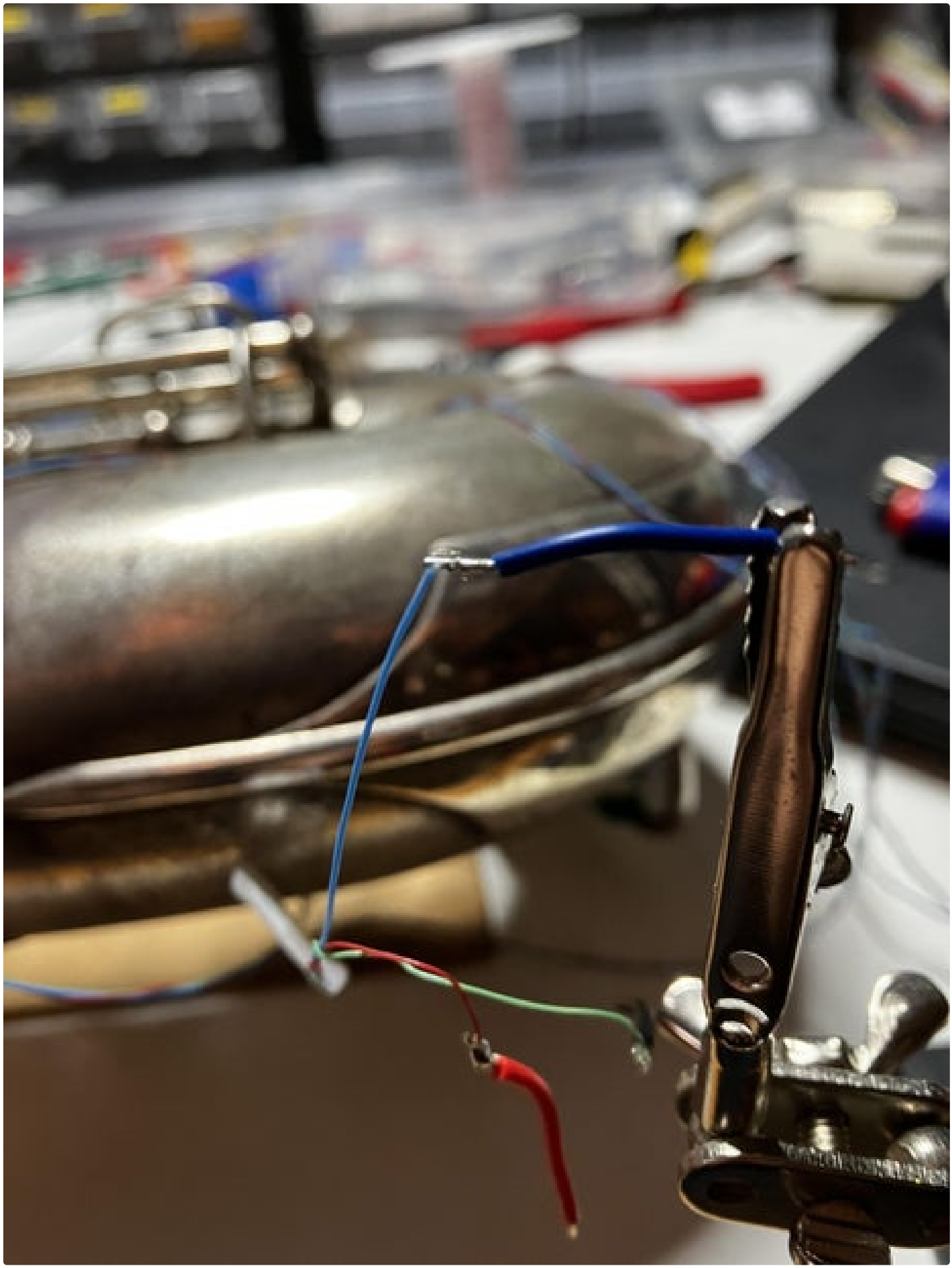
- **1:** Tin the wires and solder them to the board. When looking to the SensorBoard with the connectors facing down, the first horizontal row of sensor pins (closest to the ESP32) is 3.3V, next the GND and the Sensor on the top row. I guess this is the fastest, but also the most dangerous. I would advice against this, except if you are using solid core wire. In that case, of course you don't have to tin the cables. Again not too much coffee and happy soldering!

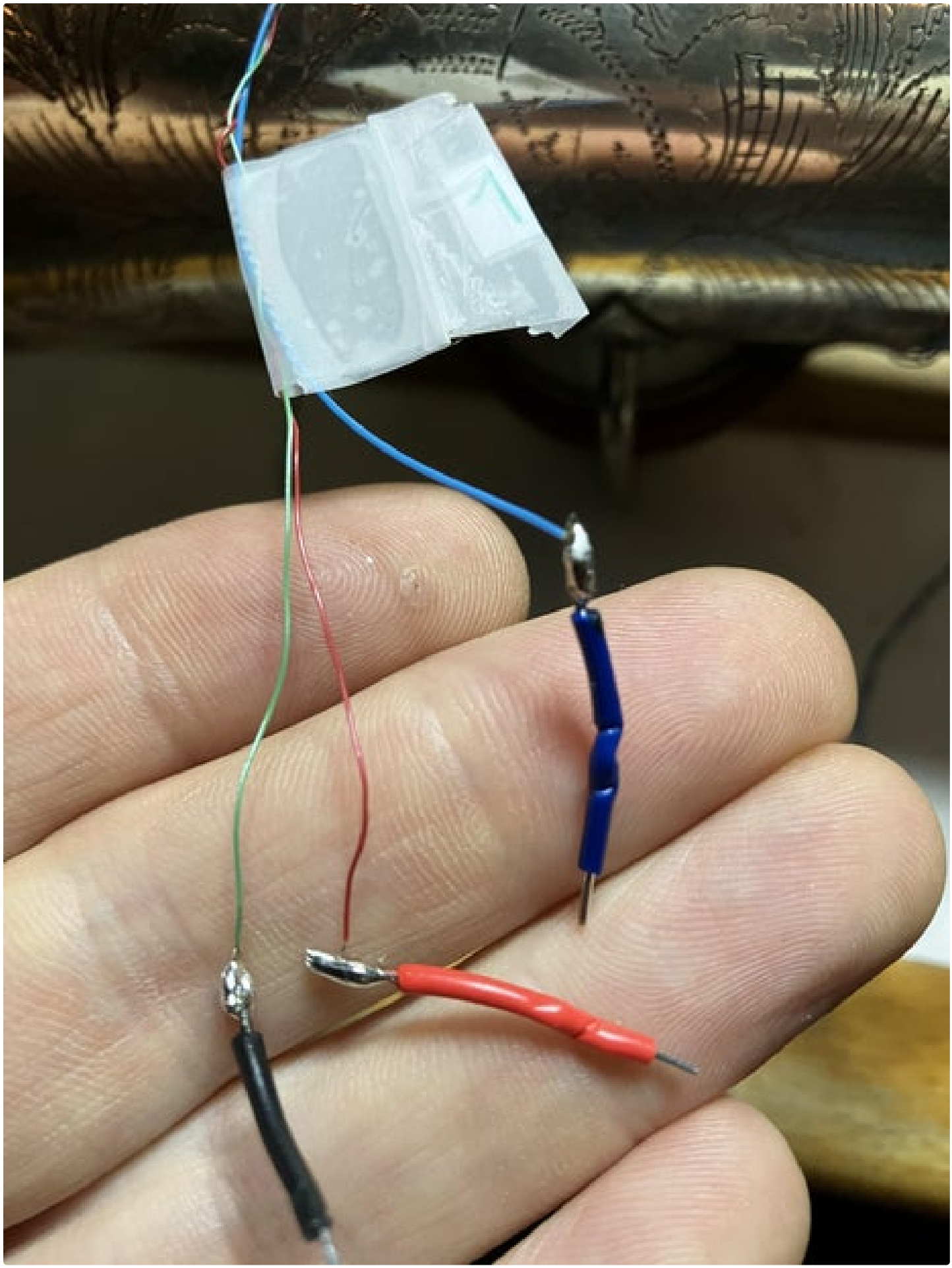
- **2:** Cut 3 x 20 pieces of 28 AWG solid core wire in 3 colours of about 3 cm. Strip them on both ends. Wind the stranded cable around the exposed solid core and solder the two wires together. Use a piece of heat shrink tube and shrink it around the cable.
- With some clear adhesive tape, tape the paper number corresponding to the sensor near the end of the wire.
- Solder the cables to the board. Sensors 1 to 20 are soldered on the pins above the ESP32 (with the connectors facing down).
- Solder the hall octave sensor to the pins on the side of the board. The pins are located on 2 (GND), 3 (3.3V) and 4 (Sensor). I'm counting the pins starting from the top of the TRS pedal input connector.
- Solder the other two octave capacitive touch switches on pins 21 and 22 on the side of the board. That's

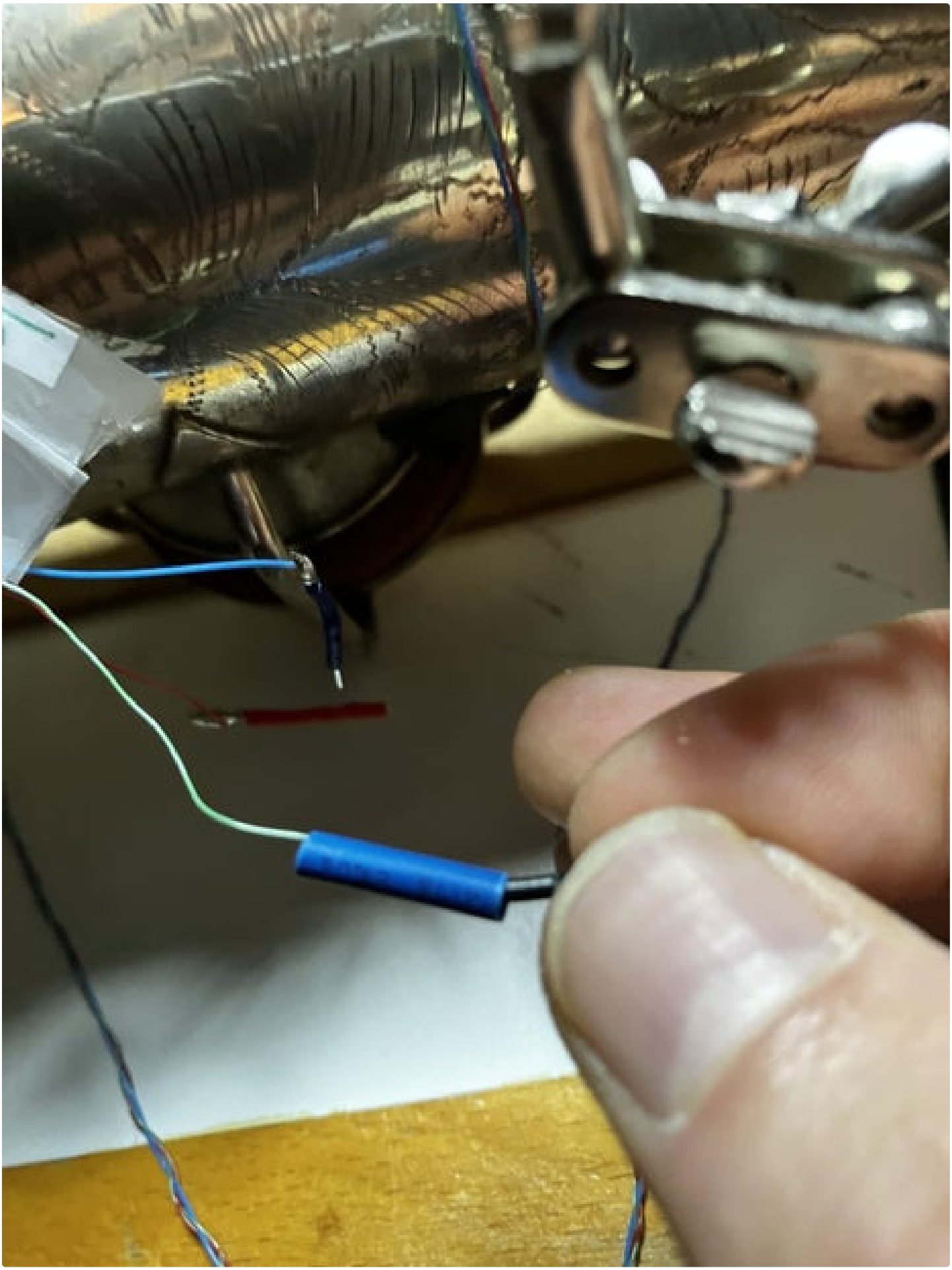
pin 3 and 4 if you're counting from top to bottom.

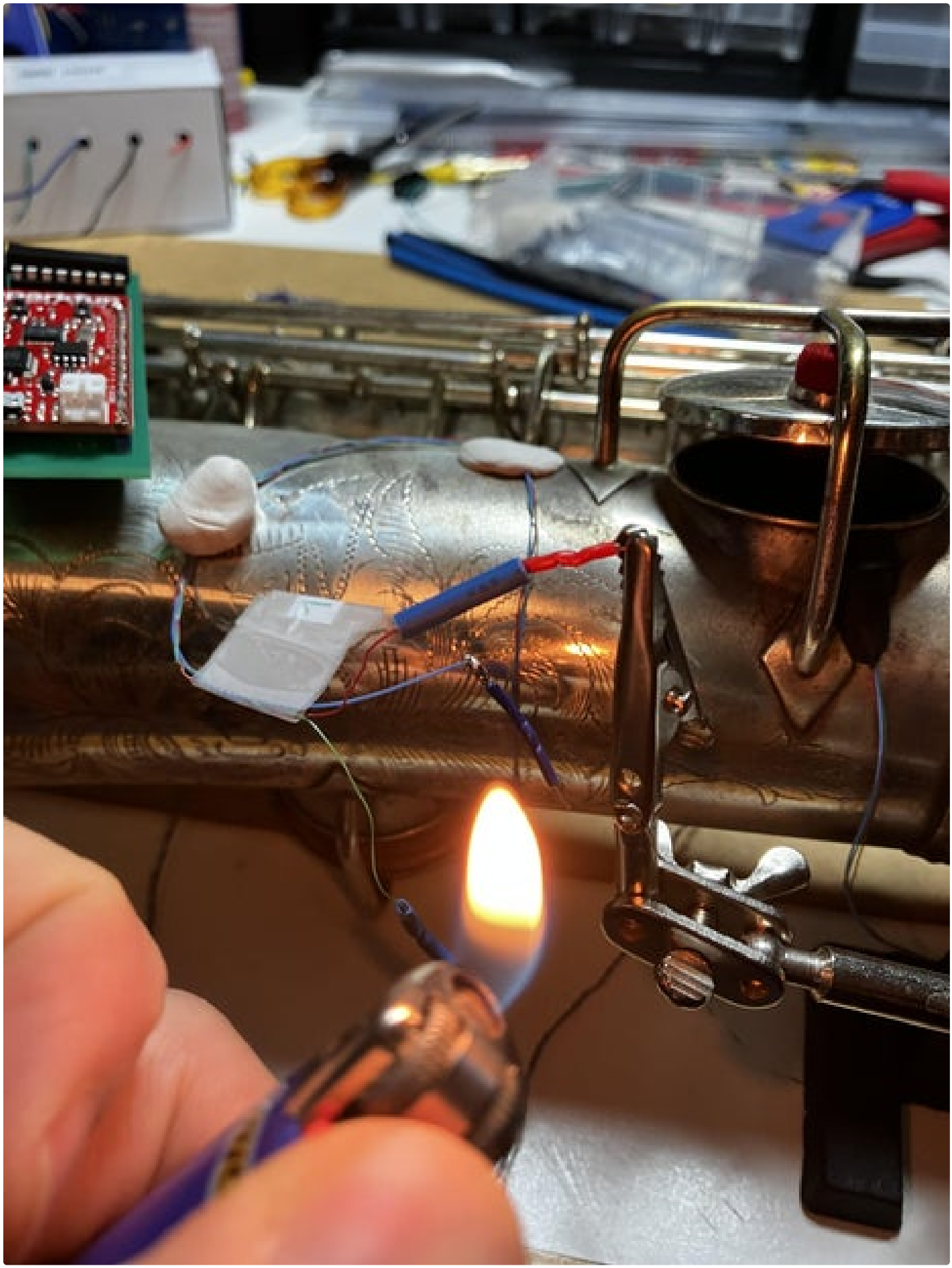


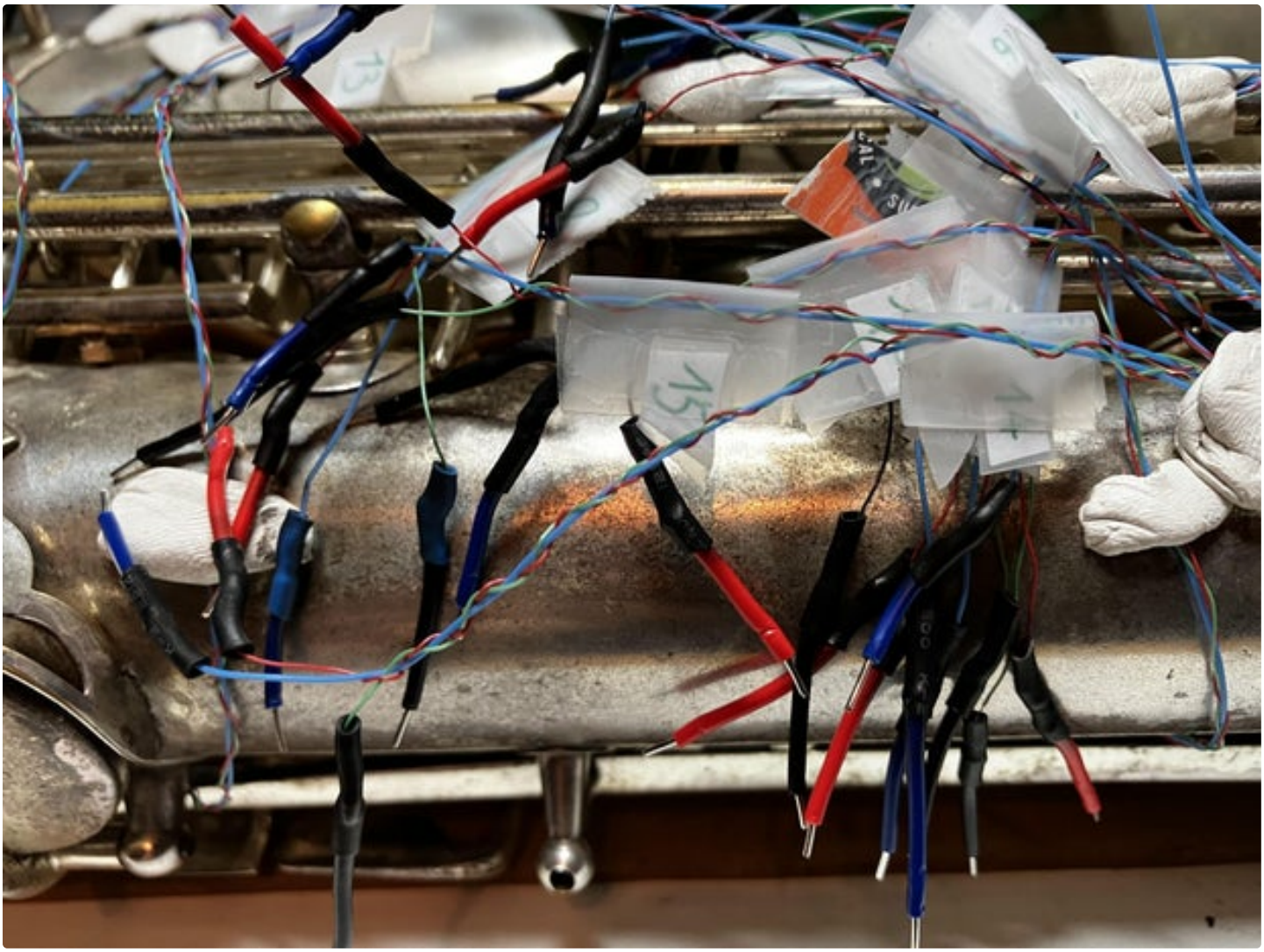


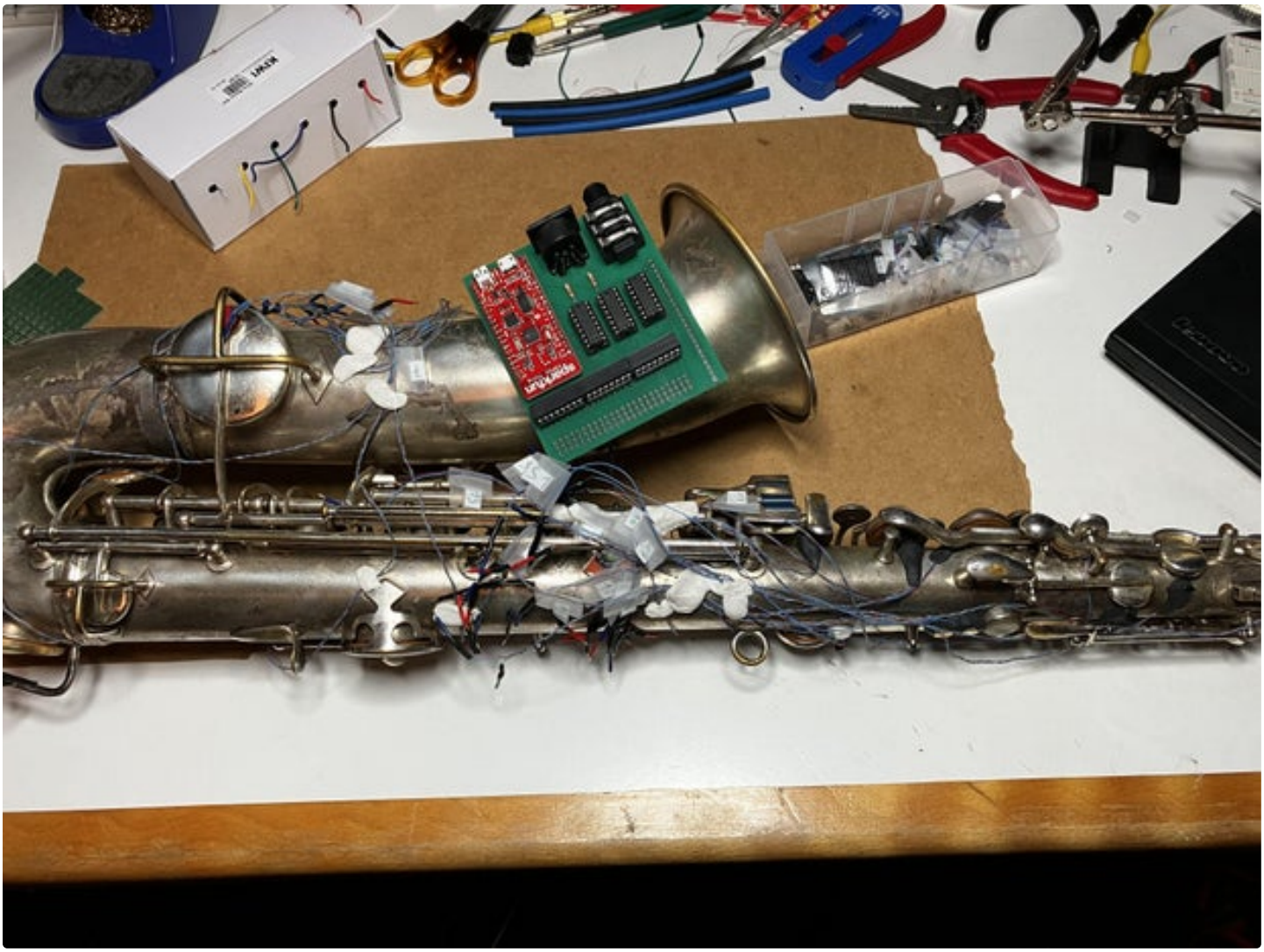


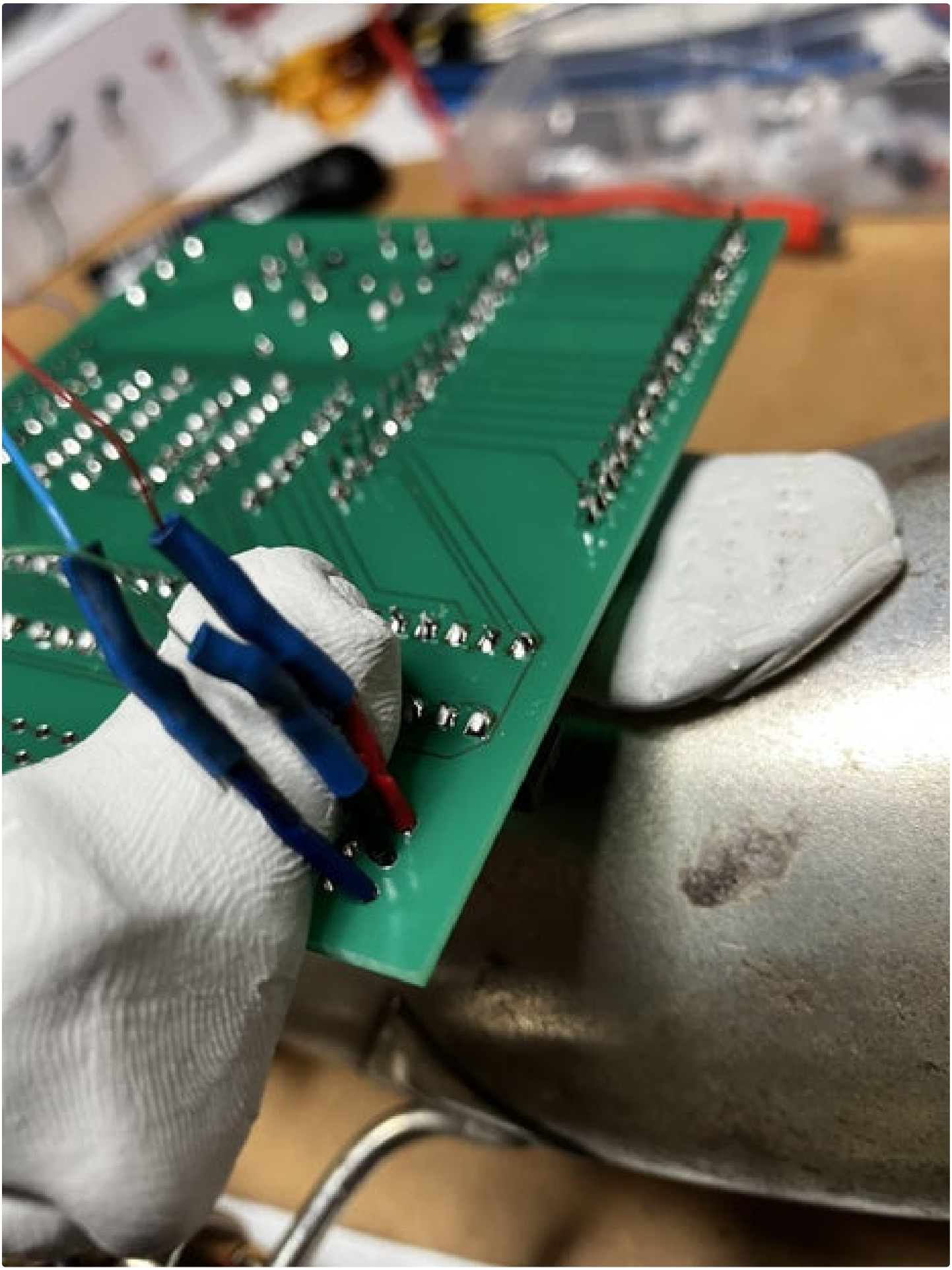


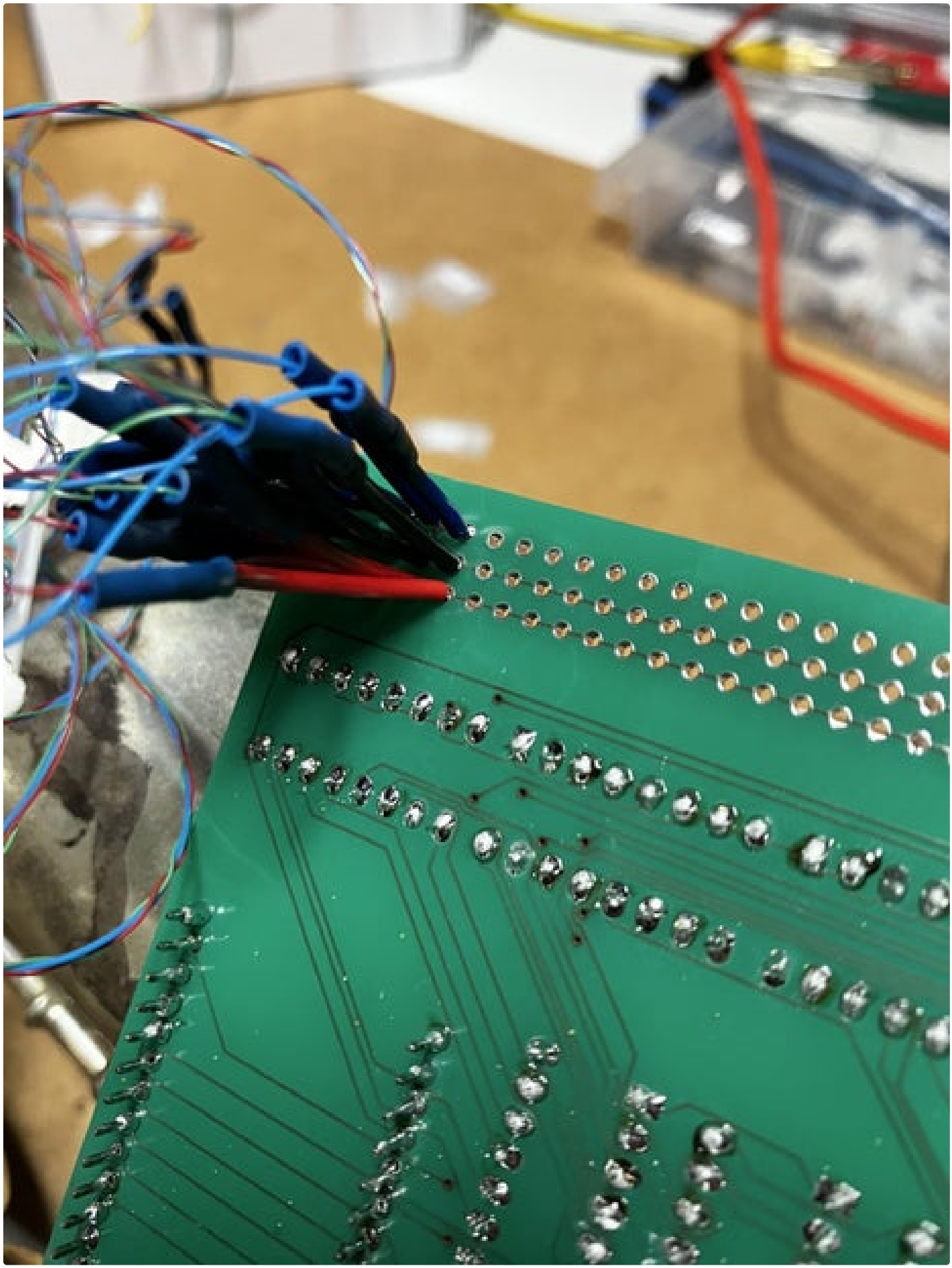


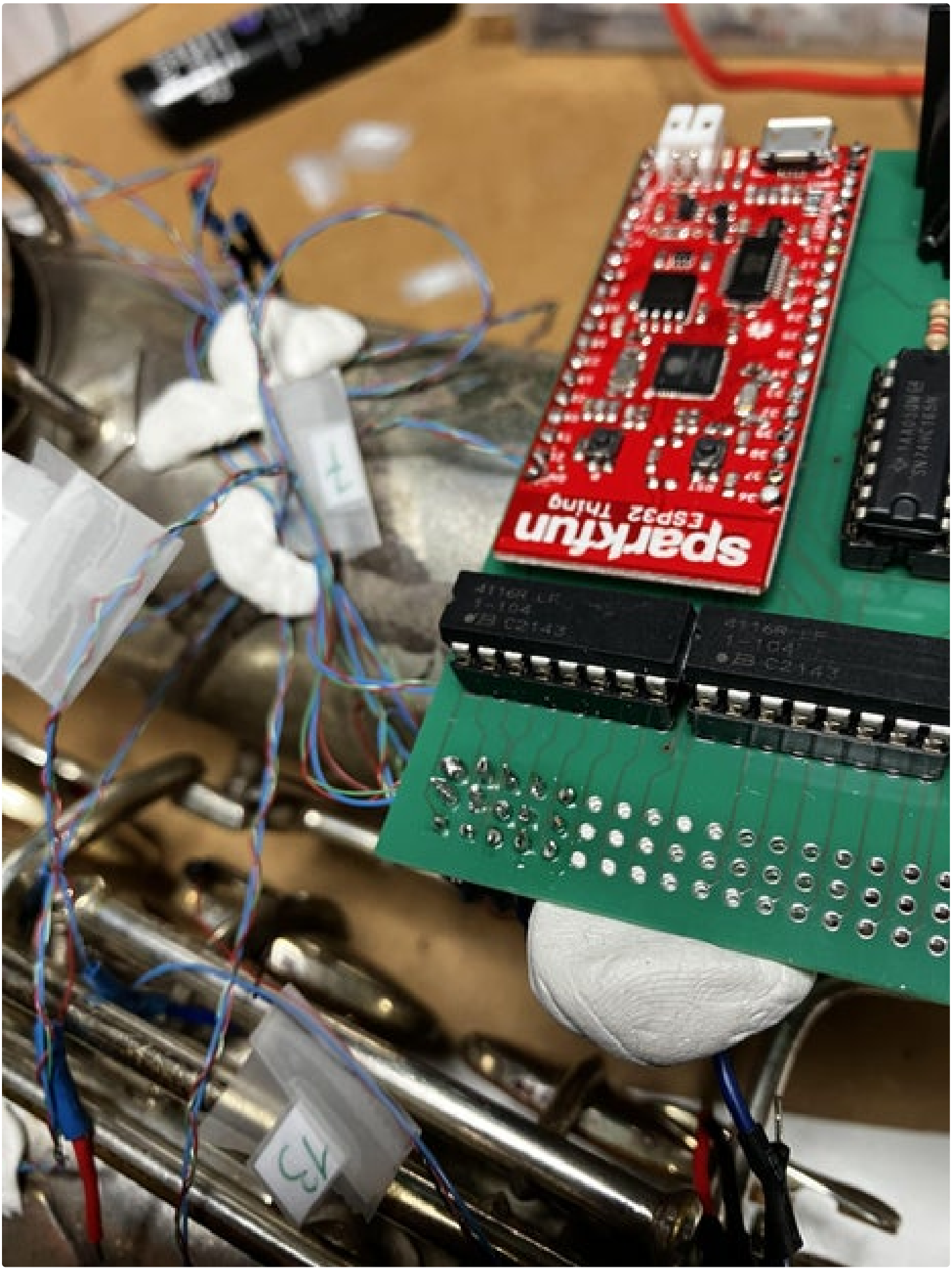


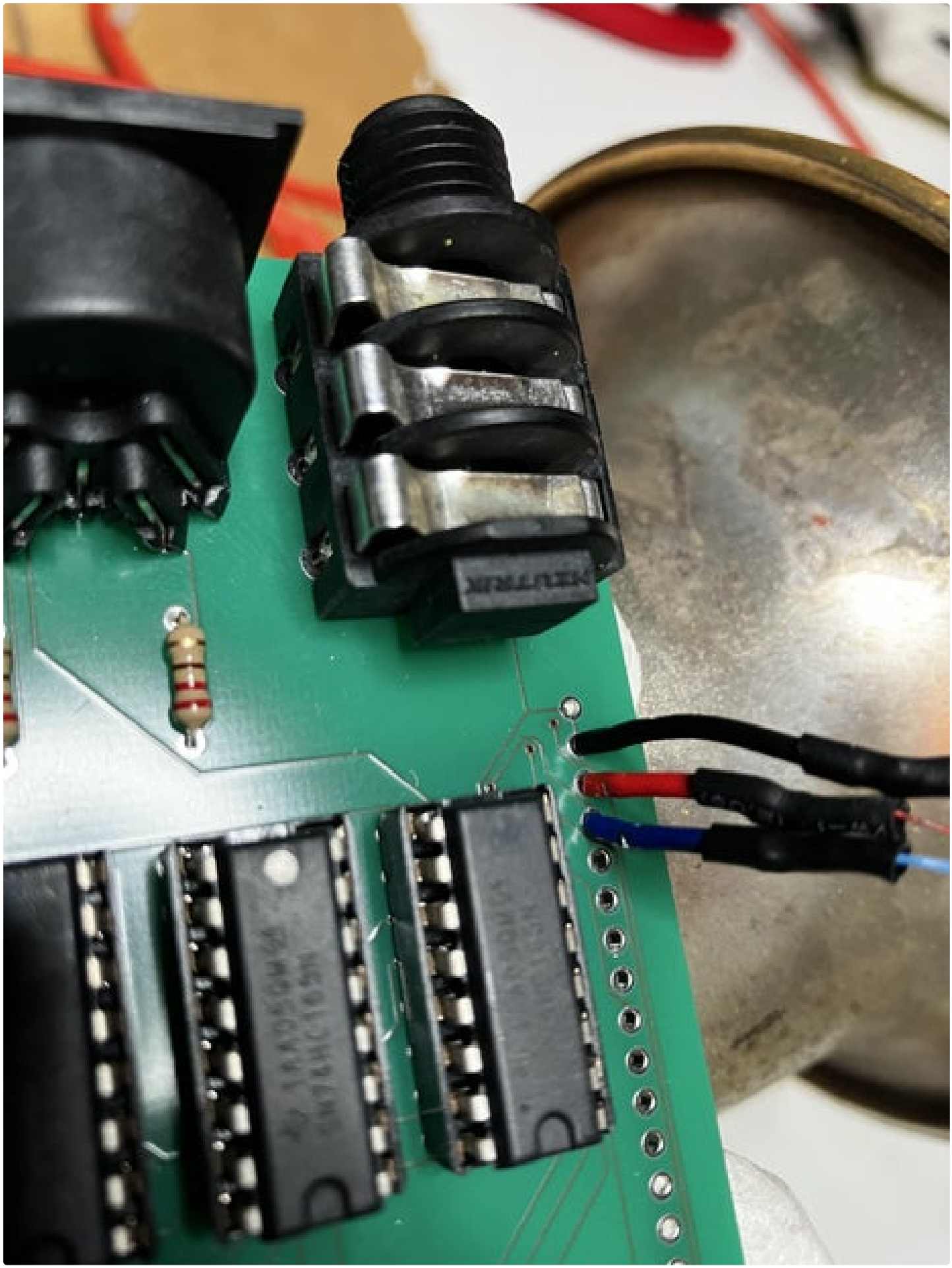


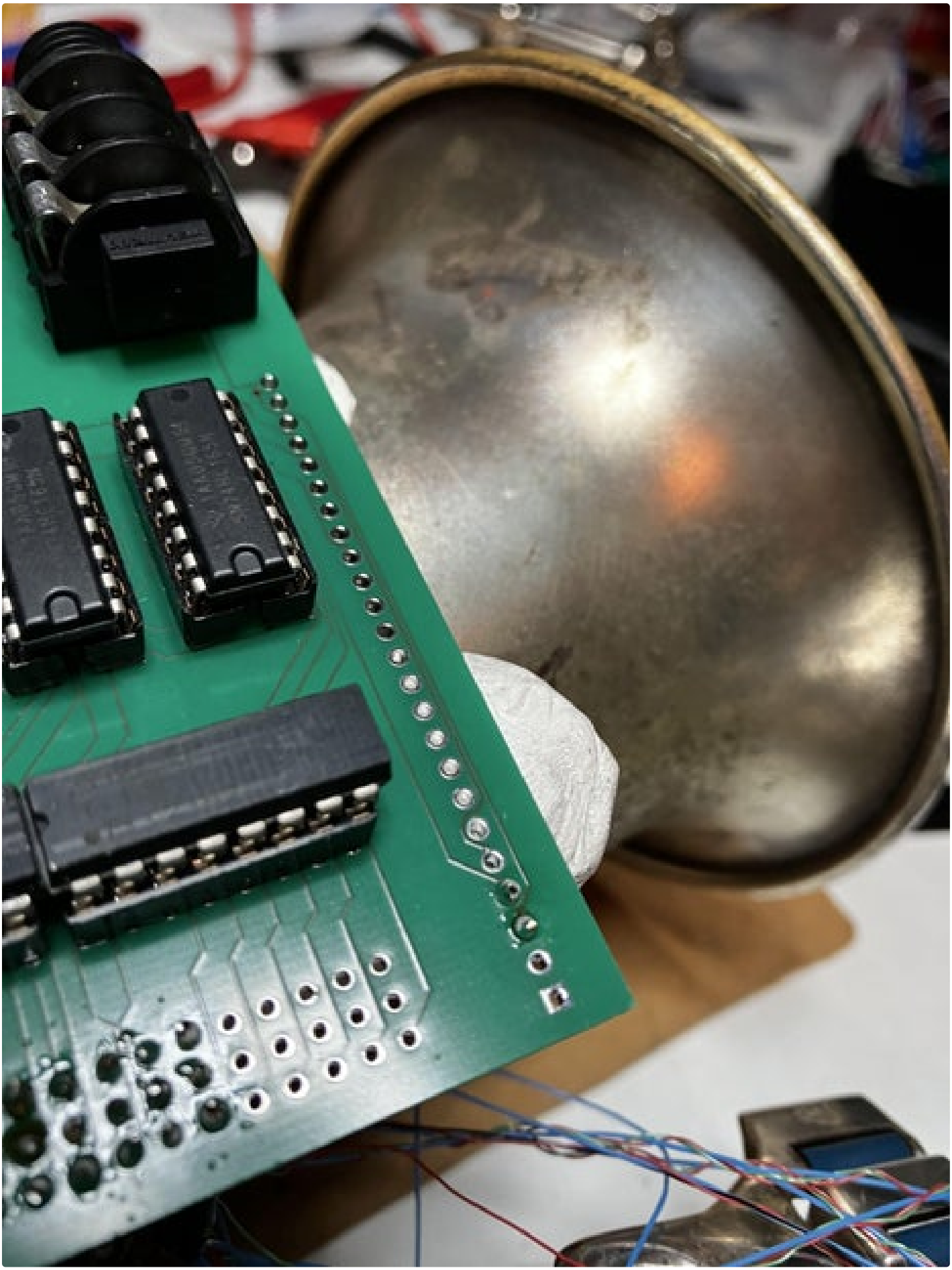






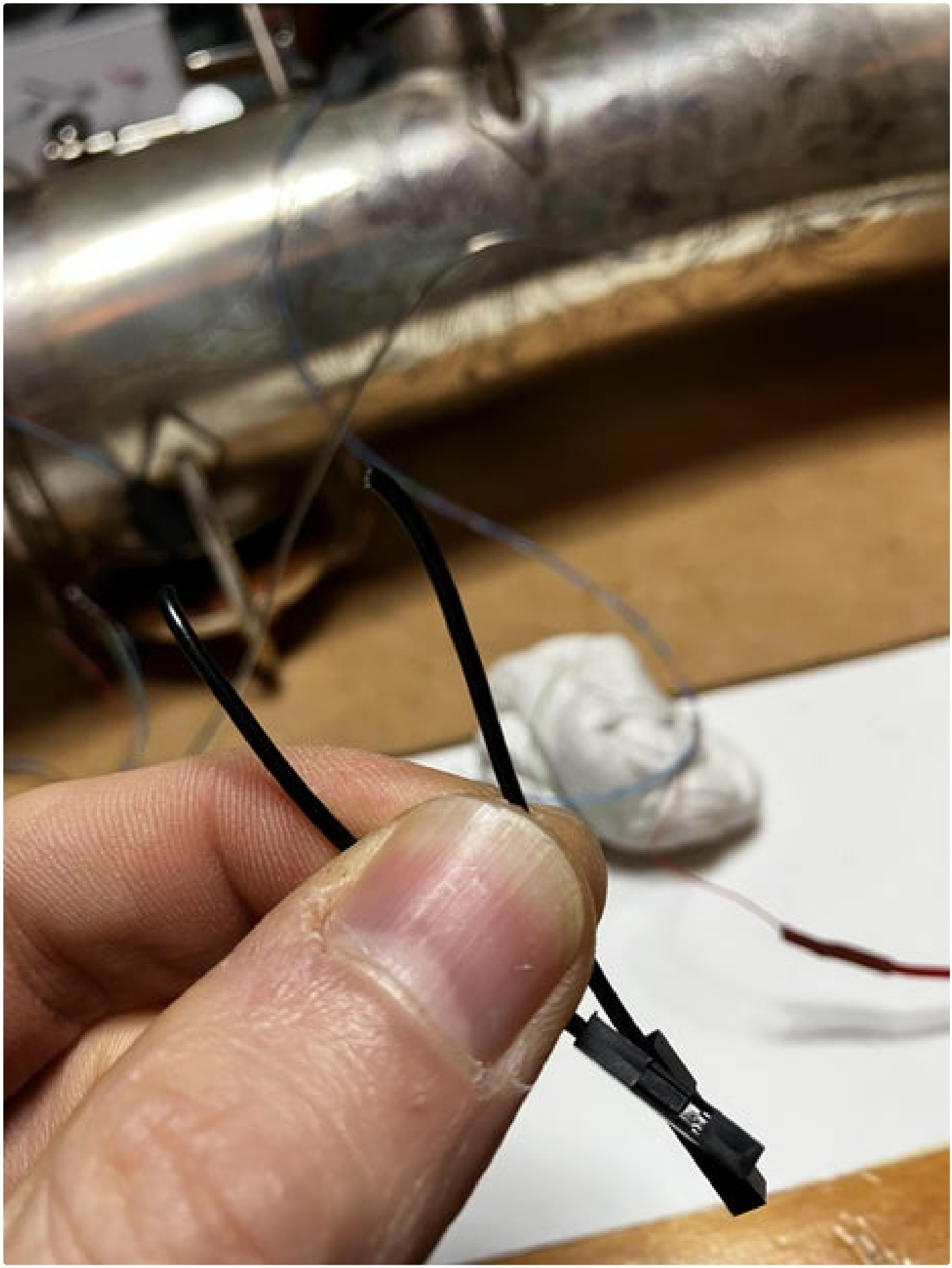

sparkfun
ESP32 Thing

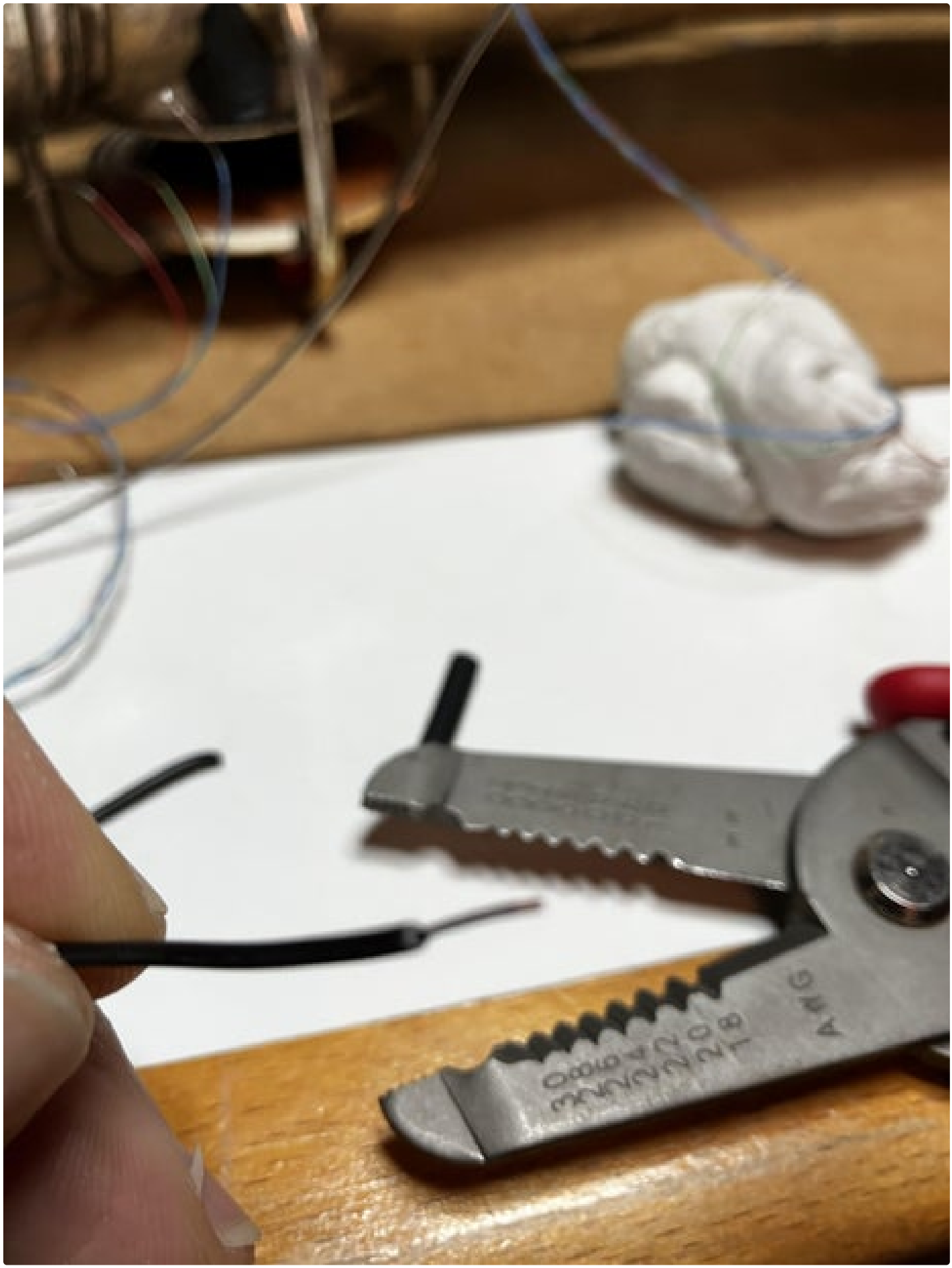


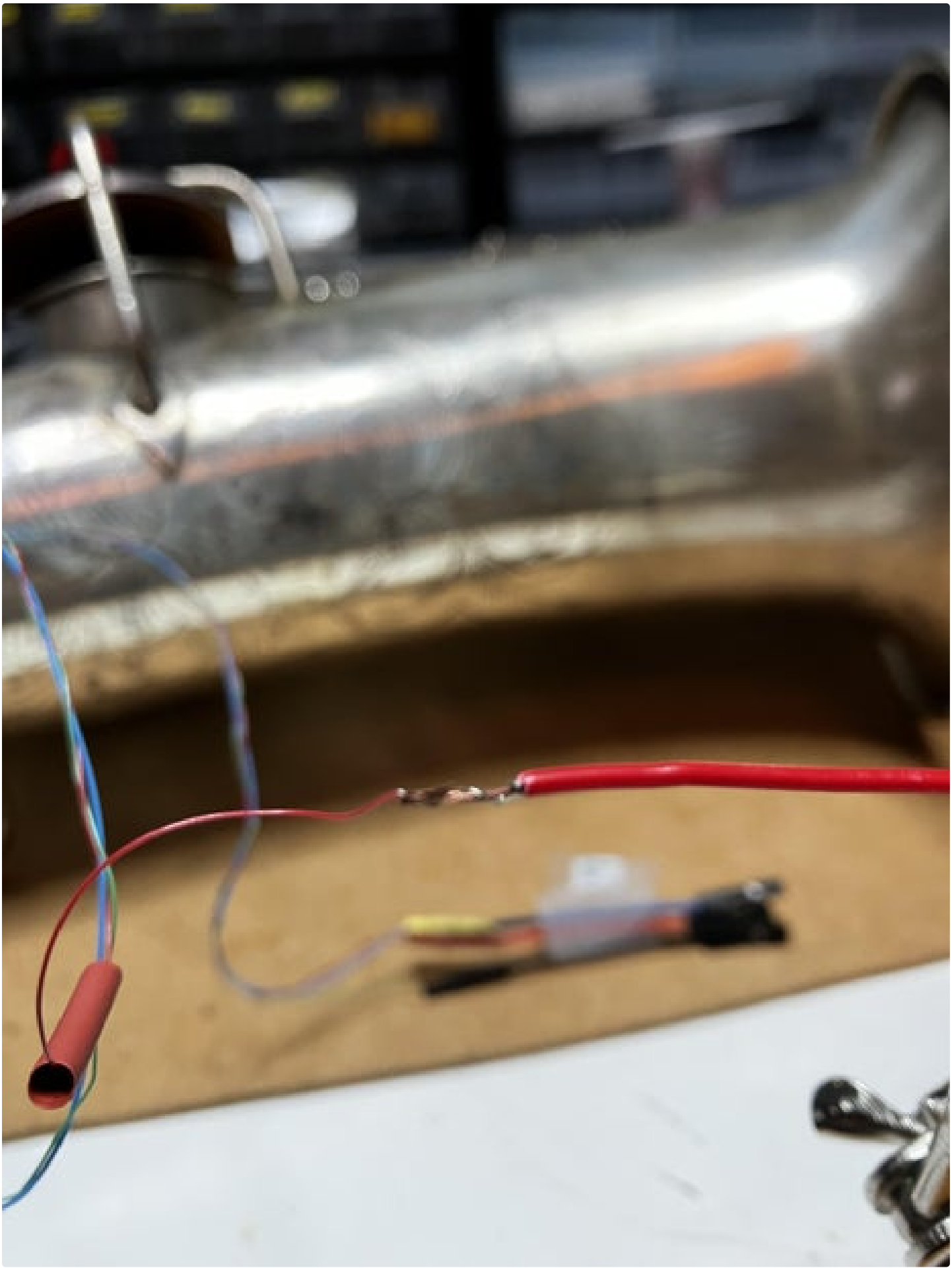


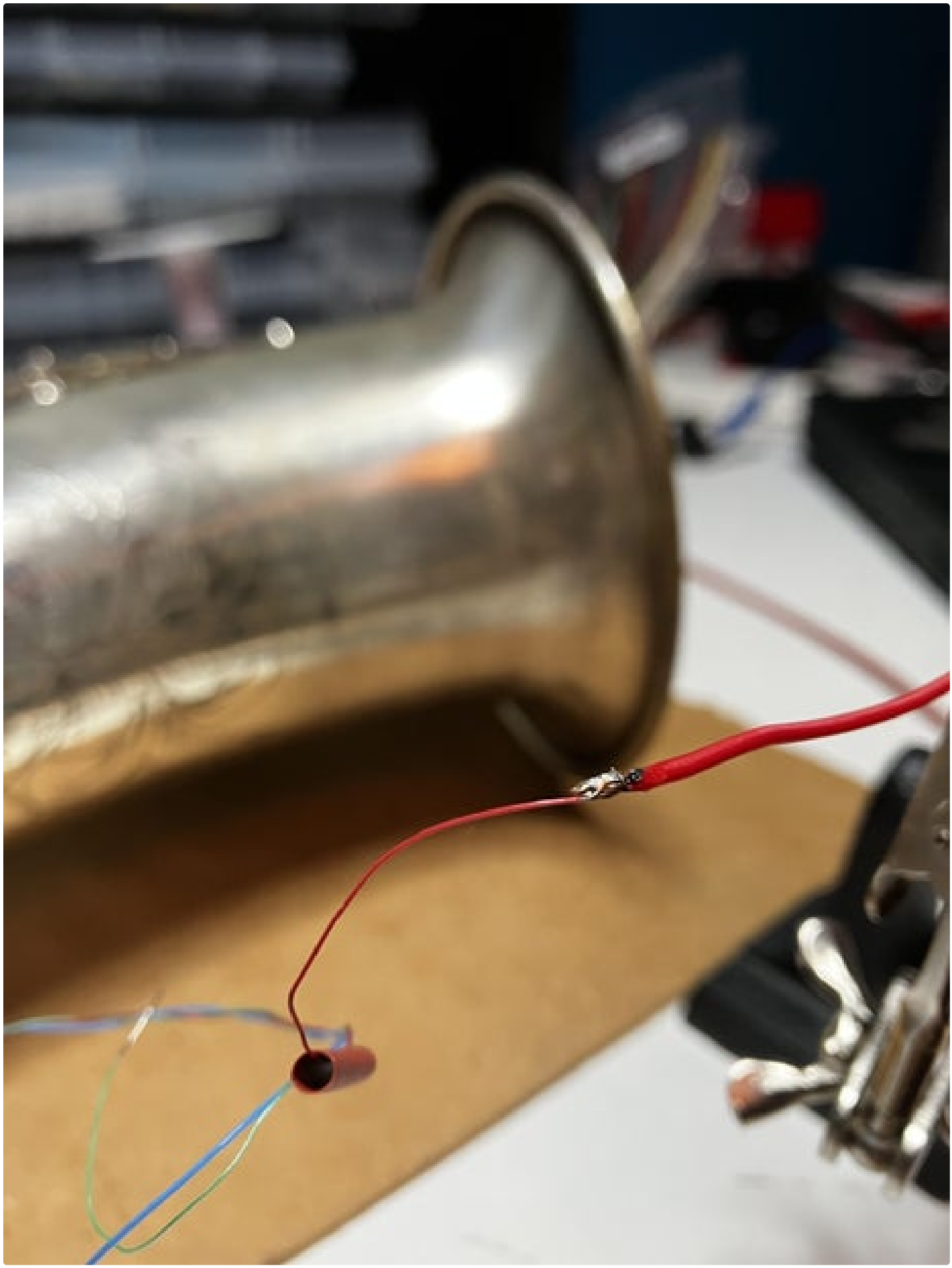
Step 22: Option 3: Soldering Female Connectors to Sensors

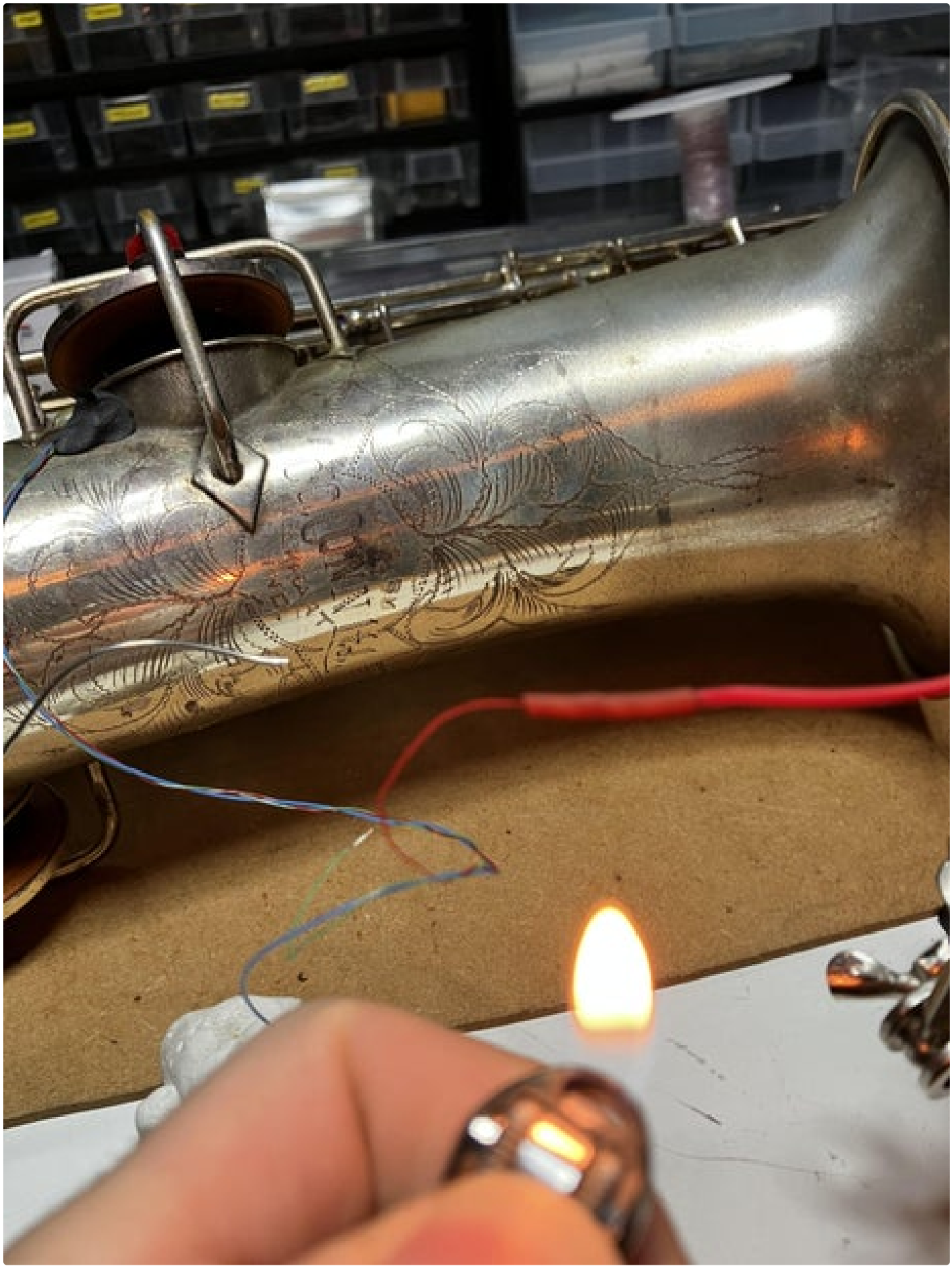
- Cut your pre-made female DuPont-style connectors to a suitable length and strip the wire. You'll need 3x20 pieces. Wind the stranded cable of your sensor around the cable with the connector. Use a piece of heat shrink tube and shrink it around the cable tightly. It's again a good idea to follow some logic colour code for 3.3V, GND and Sensor wires. I used a strict colour code for the heat shrink tubes, since my DuPont cables came in different colours. Use som clear adhesive tape to stick a number to the cable, giving your head some space to relax and preventing you not to become too confused with all these unconnected cables.
- Carefully solder pin headers on both the left side as the top part of the Sensorboard (looking at the board with the connectors facing down). It's a good idea to first put the pins in a breadboard, to make sure your pins are soldered straight. This is especially important for the top 3 rows used to connect the sensors. It's a tight fit, so any crooked pin will result in difficulties connecting the sensor to the pin. For the pins on the side I used angled header pins to give some more room to the sensors connected to the side of the board. I also soldered the pin headers on the top 3 rows facing down, hiding and protecting the connections a bit more.
- Connect the sensors to the board. The first row of pins on top of the board are the sensor inputs, the second row in the GND input and the third row, closest to the ESP32 is the voltage source.
- Connect the octave hall sensor to the pins on the side of the board. The pins are located on 2 (GND), 3 (3.3V) and 4 (Sensor). I'm counting the pins starting from the top of the TRS pedal input connector.
- Connect the two octave capacitive touch switches on pins 21 and 22 on the side of the board. That's pin 3 and 4 if you're counting from top to bottom.

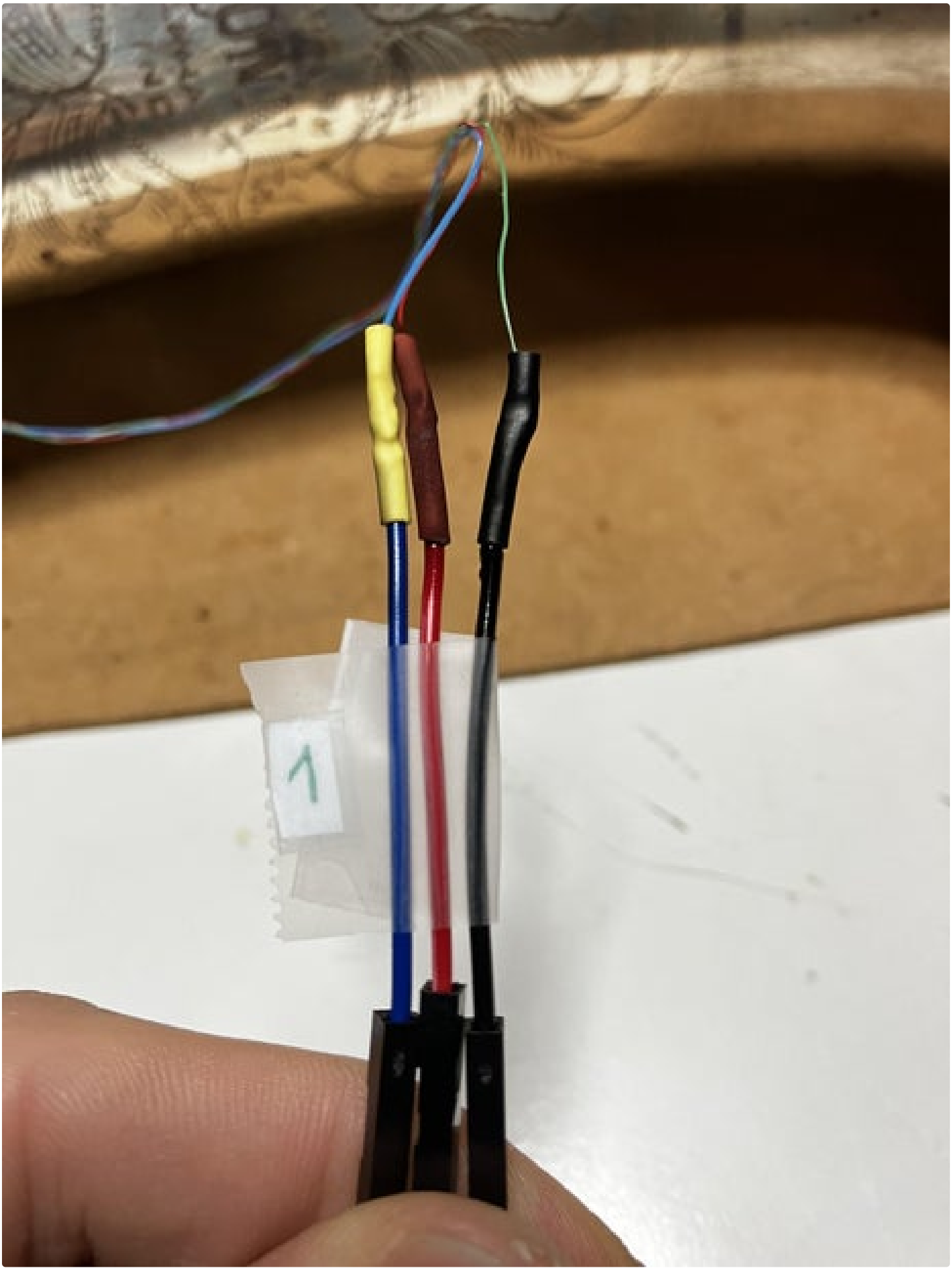


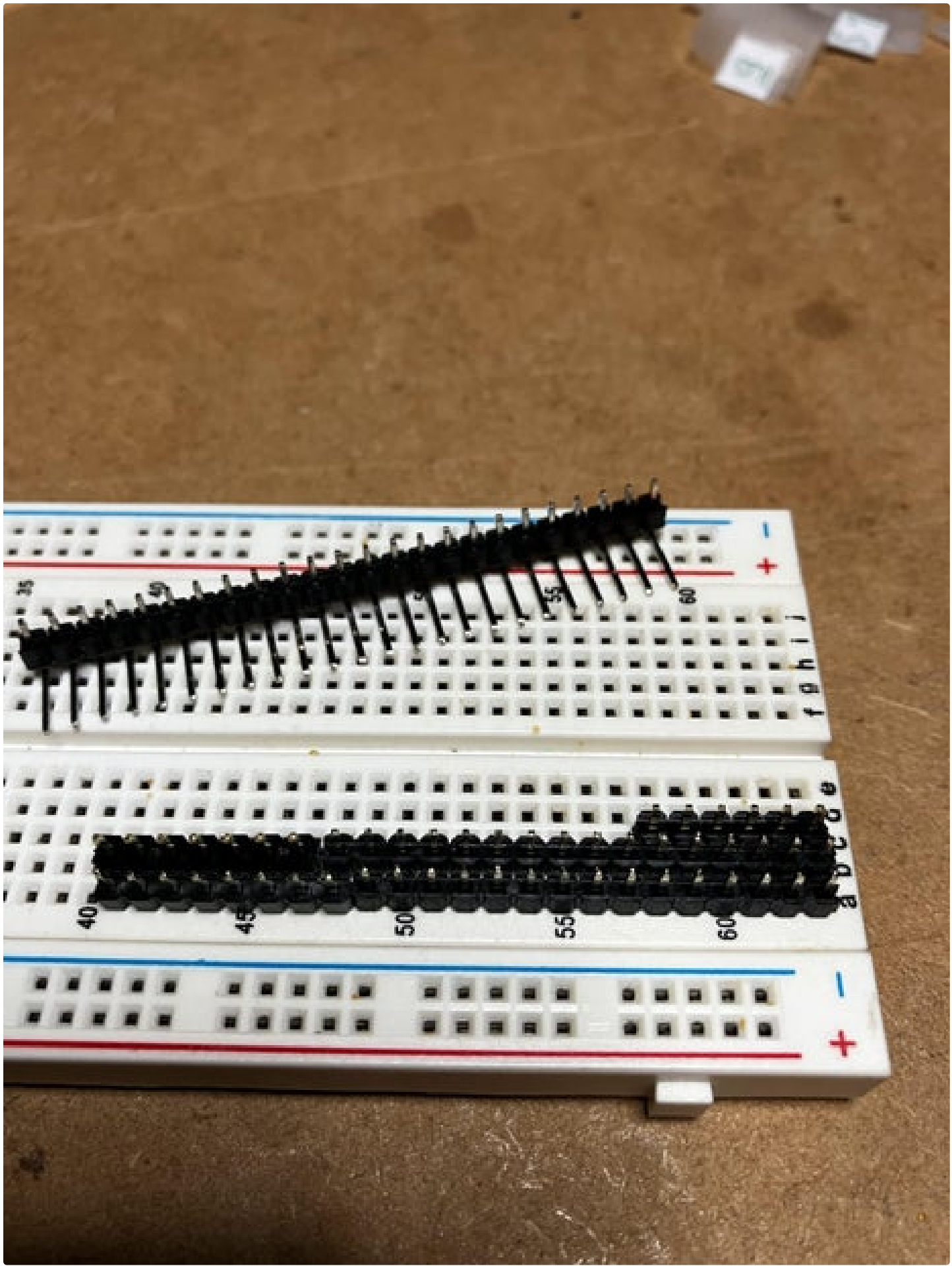

AWG

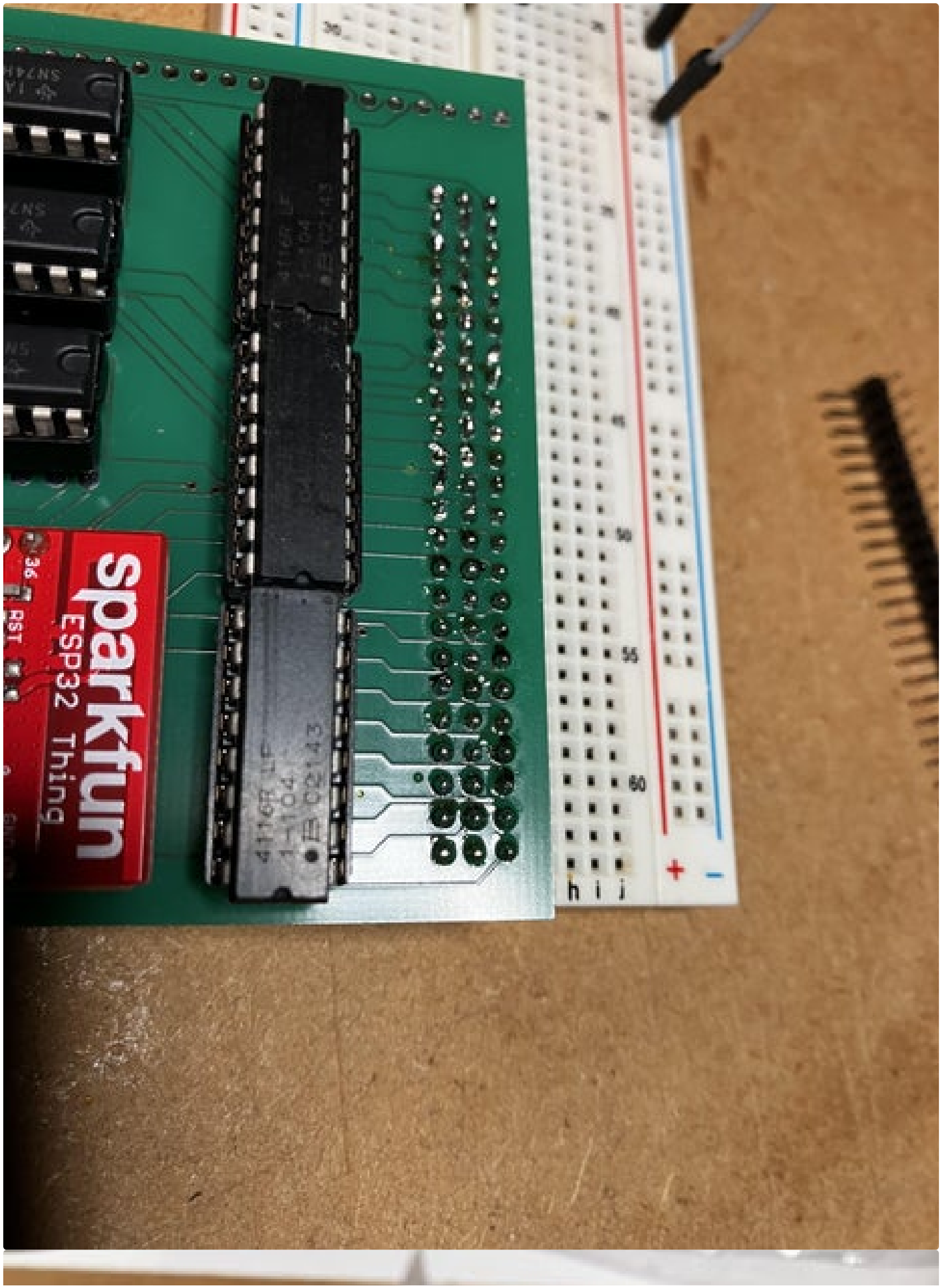


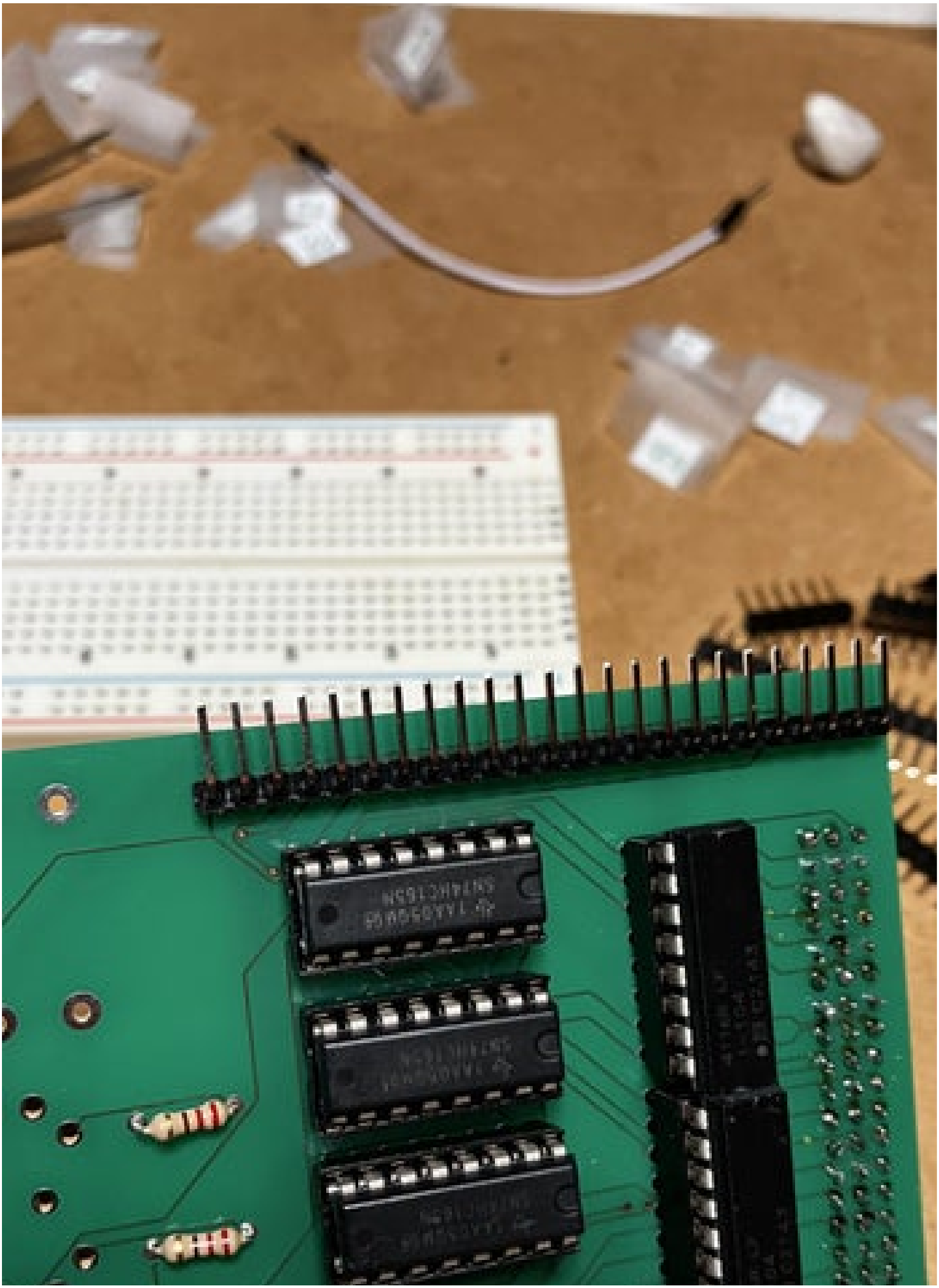


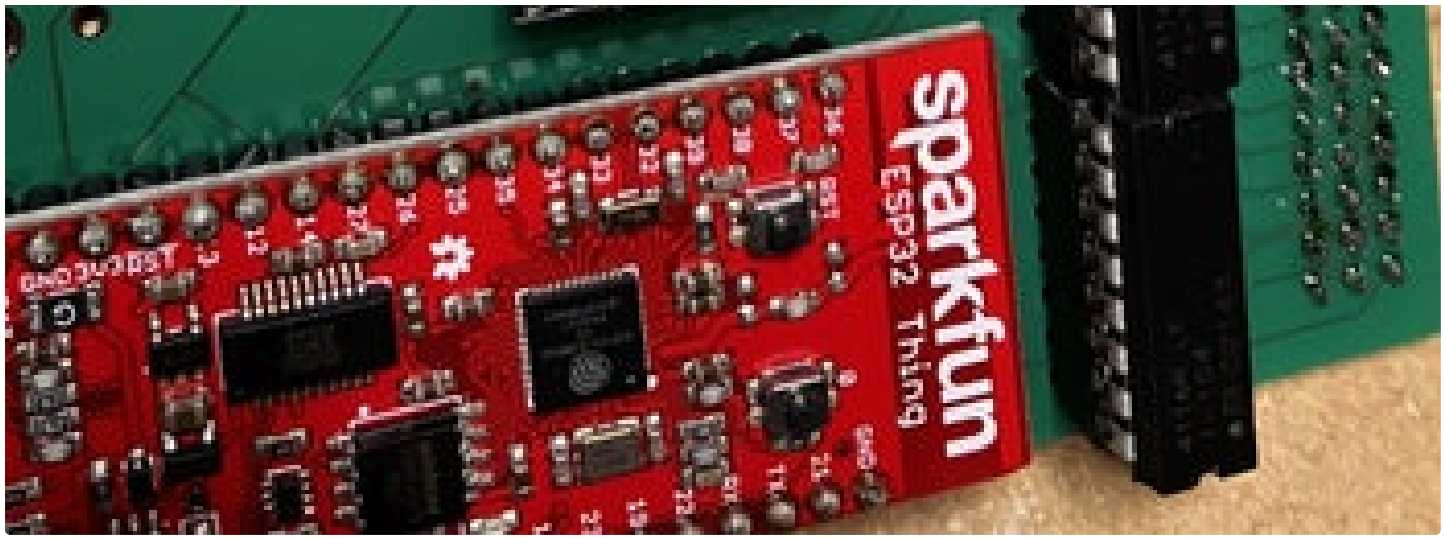


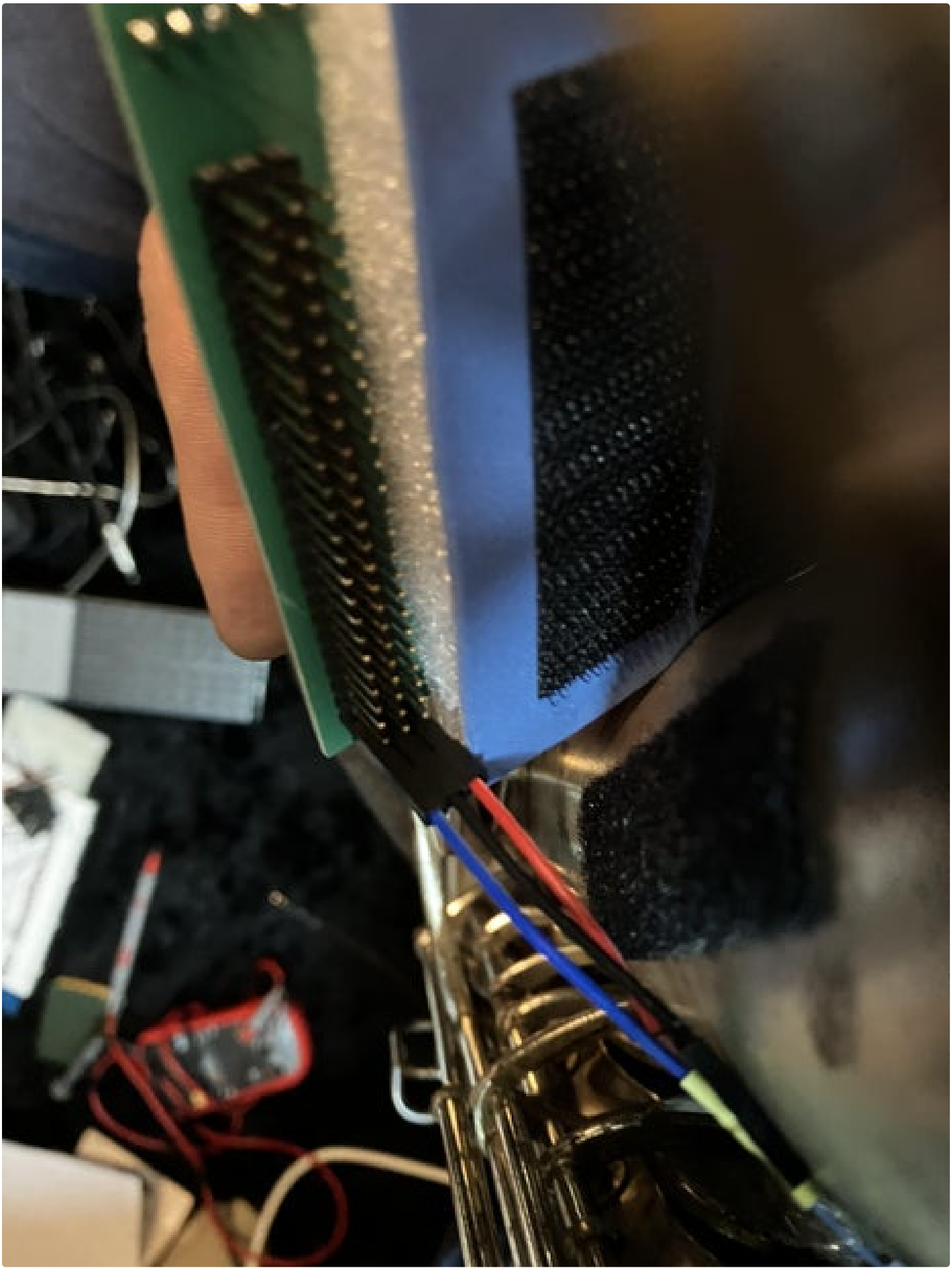

1

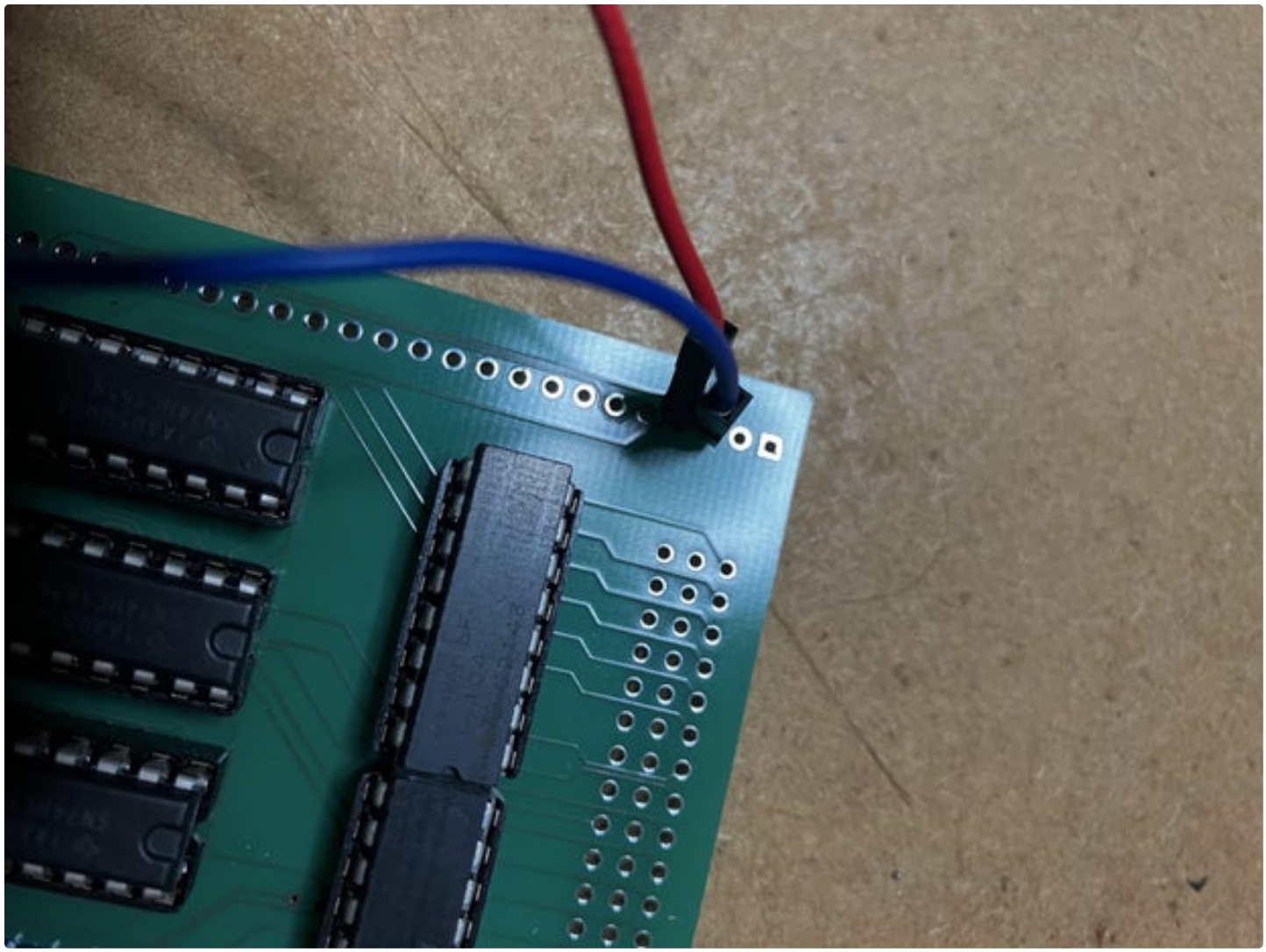

40
45
50
55
60


sparkfun
ESP32
Thing
RST
h i j
+
−




sparkfun
ESP32
Thing

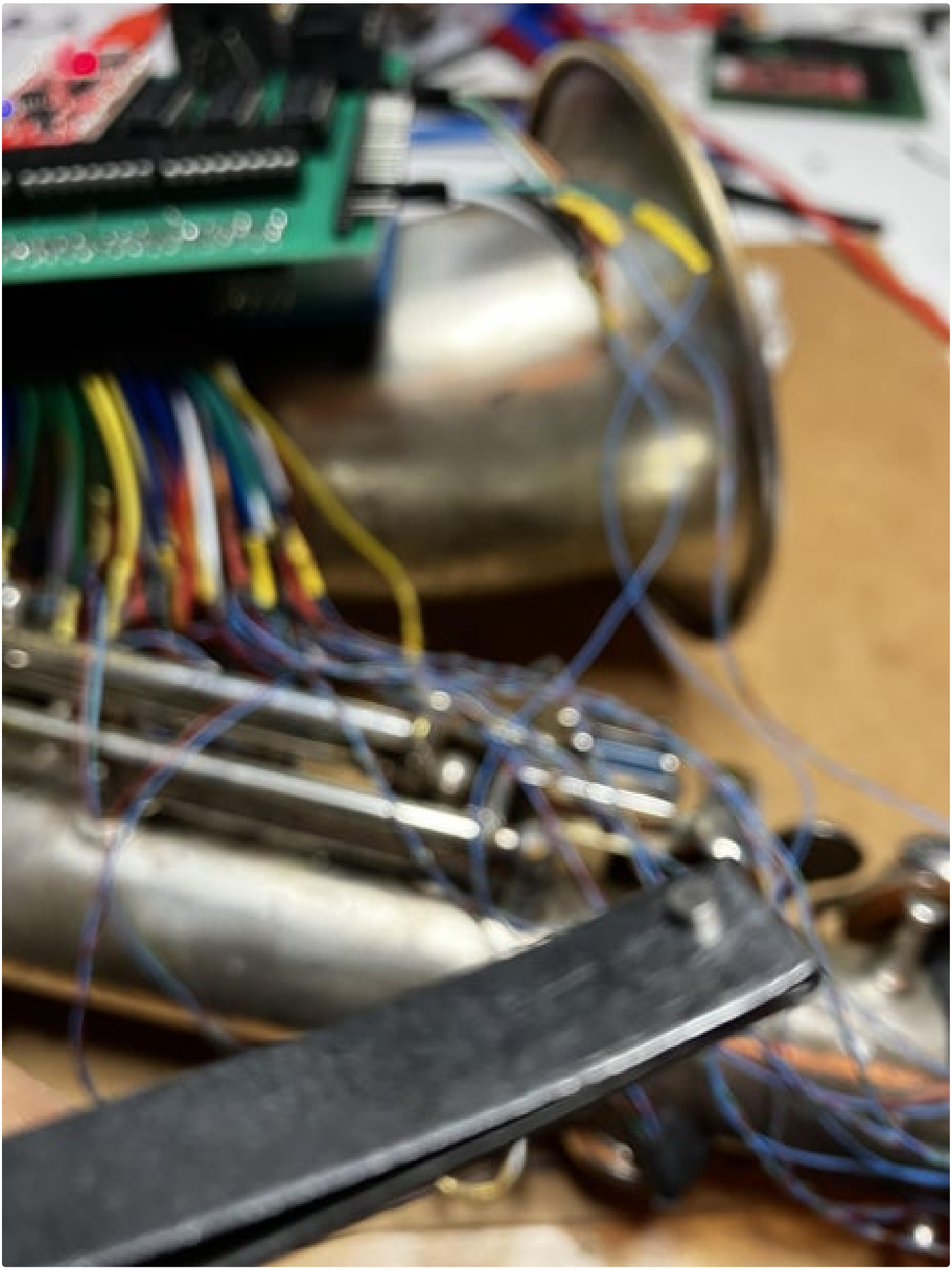


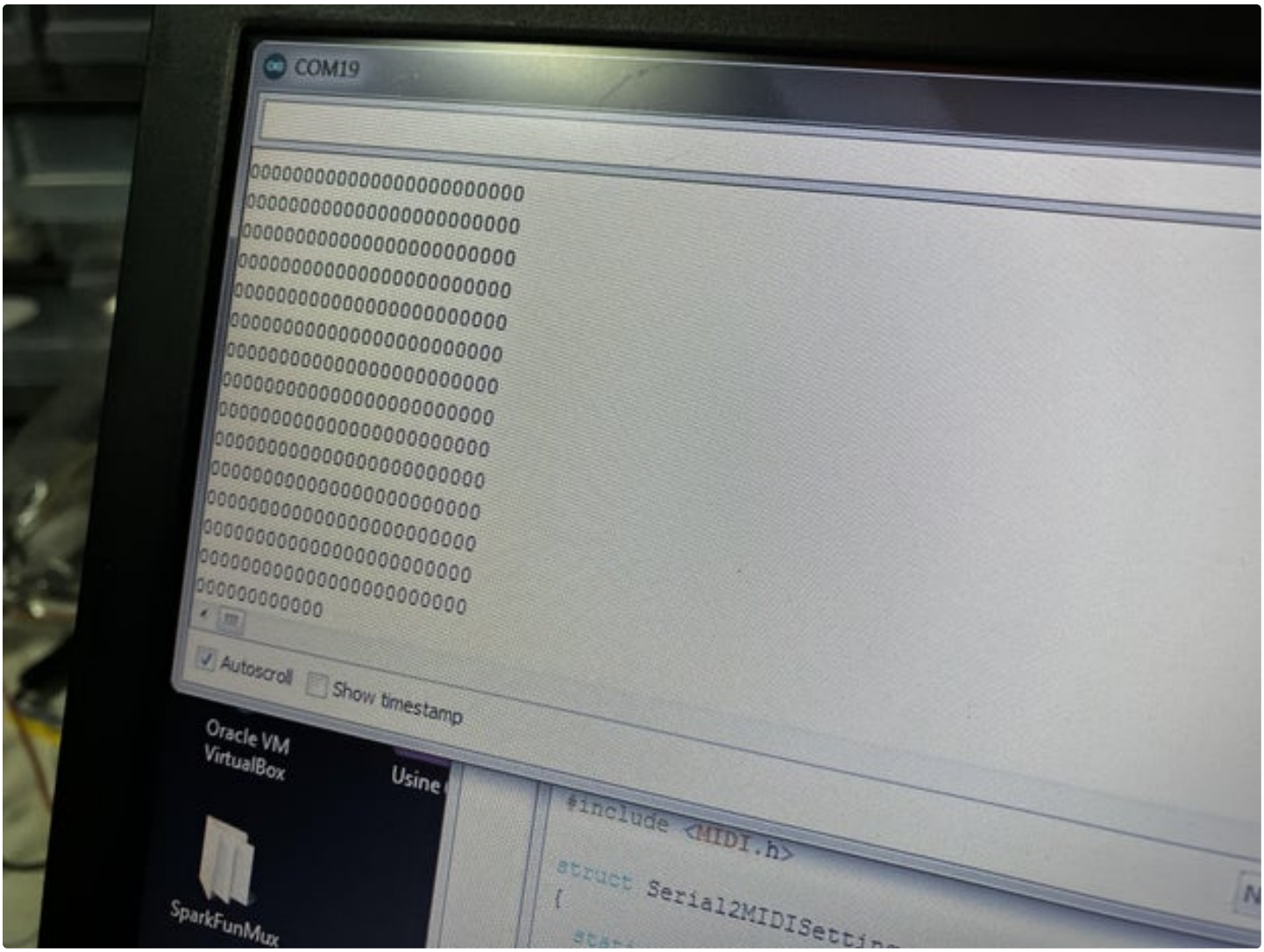


Step 23: Testing the Connected Sensors

Hooray! You are almost done with the hardware. Before sticking the magnets to the keys, first let's test the connected sensors on the body of the sax. We will use the sensortest.ino file again to check wether every sensor is working properly.

- Open the Arduino IDE and load the sensortest.ino file. Flash it to the board and open the Serial Monitor. If you see some strange characters: push the reset button on the ESP32 and wait for the line of zeroes to appear.
- Take your magnet and attach it to an iron object like a pair of tweezers to comfortably reach the attached hall effect sensor switch.
- Briefly move towards the sensor with the magnet. Make sure the south pole of the magnet is facing the sensor. You should see the corresponding zero on the Serial Monitor changing to 1 while approaching the sensor with magnet. When you're sure every sensor is working proceed as follows:
- Open the octavetest.ino file and flash it to the board. Again open the Serial Monitor and you should see **'octave hall switch off'** being advertised in the window. Hold the magnet in proximity to the octave hall switch and the text should change to **'octave hall switch on'**.
- Touch the capacitive octave plate and **'octavesensor1 is working'** should be advertised.
- Touch the second capacitive touch plate and **'octavesensor2 is working'** should appear on the screen.





Step 24: Attaching the Magnets With Sugru

With all the sensors working, it's time to glue the magnets to the keys of the sax.

- Open the sensortest.ino sketch again and run it. Open your Serial Monitor.
- Take a piece of Sugru adhesive putty and put the magnet into the paste. Make sure you have the right side of the magnet bulging out. Carefully stick it to the key of the sax. Make sure the magnet is positioned perfectly so whenever you close the key, the Serial Monitor reads 1 on the corresponding position.
- Keys like the middle G# and the upper palm keys will open the keyholes rather than closing them, so make sure they read 1 in resting position and 0 when opening. That's pretty darn logical!
- When all your magnets are put in place and are working properly, put the sax down on it's side to prevent gravity from deforming your putty while drying. Let it dry for some 12 hours. Sweet dreams, tomorrow you will have a working Hybrid Saxophone System!

Step 25: Programming Your Notes

With your hardware ready, let's program your new baby! The way your sensors are being read out to covert them to a corresponding MIDI note, is very well explained by the great great Gordon Good. While initially dreaming about how to make this system, I was figuring out the most elegant way (for humans) would be to represent the different key combinations as one big digital number. To my astonishment I found out about the Gordophone which *exactly* implemented this in this way. Since he's also one of the good (what's in a name) open source afficionado's, I was able to reuse a lot of his code. Here is a link to a dearly beloved hybrid sax prophet <3.

In short the idea is to read the

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 as the binary representation of 1,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 as the binary representation of 2,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 as the binary representation of 3,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 as the binary representation of 4,

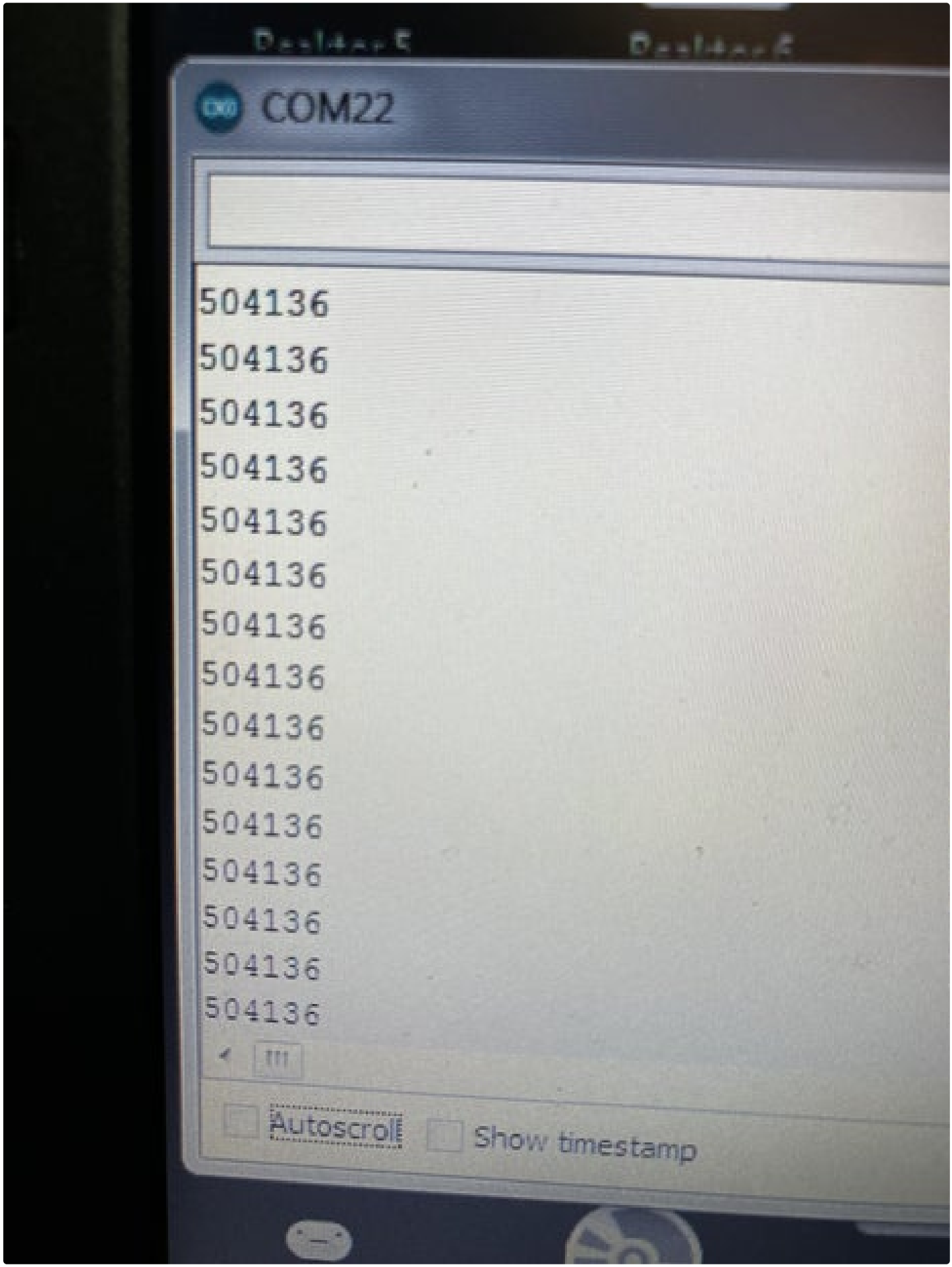
and so on, up until

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 as the binary representation of 34511010.

On Good's blog there is an interesting discussion on several other techniques resulting in the same result, but I'm sticking to this way, because it's super easy to edit, grasp and it's been proven that, while slightly slower then other methods, the digital representation mode is already fast enough on a modern microcontroller to outperform even the fastest finger movements possible, for even the most fit homo sapiens sapiens. So there you go!

Since you might have various reasons of having used slightly different sensor pin inputs for your hall switches, or even deliberately want different notes being triggered than the actual notes (kudos to the avant garde freaks) and want these to be hard coded in the system, let's go over the method of programming your instrument.

- Open the values.ino sketch and load it onto your board. Open the Serial Monitor and look out for the values, this time represented in base 9.
- Finger your notes and write down every corresponding value. When you are working home alone, like me, it proved pretty useful to record a voice memo of the note and the corresponding value. So I started with fingering my low Bb, looking at the screen and reading out loud the corresponding number. Just take note of every key fingering position you want to use to send MIDI. That's right, you can use some unused keys or invent new fingerings to map them to MIDI commands. In the Firmware.ino all notes are covered, as well as values for Program Change UP and DOWN. I used side keys 1 and 2 for going up and down the programs of my favourite synths.
- Open the Firmware.ino file (finally!!) and look in the code for the portion that states **'case 123456:'**. As you can see, the whole list of possible notes is there, along with the "cases" for Program Change UP and DOWN. Take your list, or listen to your recording and fill in all the values in the program.
- Save your work and upload it to the board.
- Connect your pedal.
- Connect your MIDI cable to the DIN connector to a MIDI capable device's MIDI input. Or connect via Bluetooth. Note that when connecting to Bluetooth, make sure your pedal is connected first. Because of the nature of expression pedals, the board will reset and you will have to connect again. Why? Well the TRS jack connection has 3.3V on it's ring, which will short the board while connecting it. If some real engineer can fix this. Please inform me or fork this project. Would be great!
- If all goes well. You are ready to go. Congratulations, you've made yourself a Hybrid Sax! Haribol!


COM22
504136
504136
504136
504136
504136
504136
504136
504136
504136
504136
504136
504136
504136
504136
504136
504136
Autoscroll
Show timestamp

Step 26: Going Further

As stated, this system is a fully working Proof of Concept model. Let's say it's a first try of a novel take on windcontrollers. Well... wind controllers?

As you can see, no attention is given on the actual wind control, as typically seen in wind controllers (duh).

The fact is, I actually made a version featuring an analog envelope follower, based on the schematics of the Sparkfun Sound Detector.

A piezo pickup can be used to translate your sax vibrations and convert them to a usable breath controller. But this means you will not be able to control the breath separately from your acoustic playing, which is in fact less interesting for me, for now.

So I started experimenting with a proper breath sensor, using both a freescale pressure sensor and the fabulous DIY optical breathsensor by KontinuumLAB. They both work fine.

I'm trying to wrap my head around a system that could potentially be used as a breath sensor, while actually playing out loud (with an envelope follower) and that could also double as a breathcontroller while blowing through the mouthpiece without making sound.

I'm not there yet, which means for now, two separate controllers will need to be installed in addition to the already big mess of cables on your beautiful instrument.

For me this is not an elegant solution at all and in fact quite impractical to play. And of course, now I want to play too!!

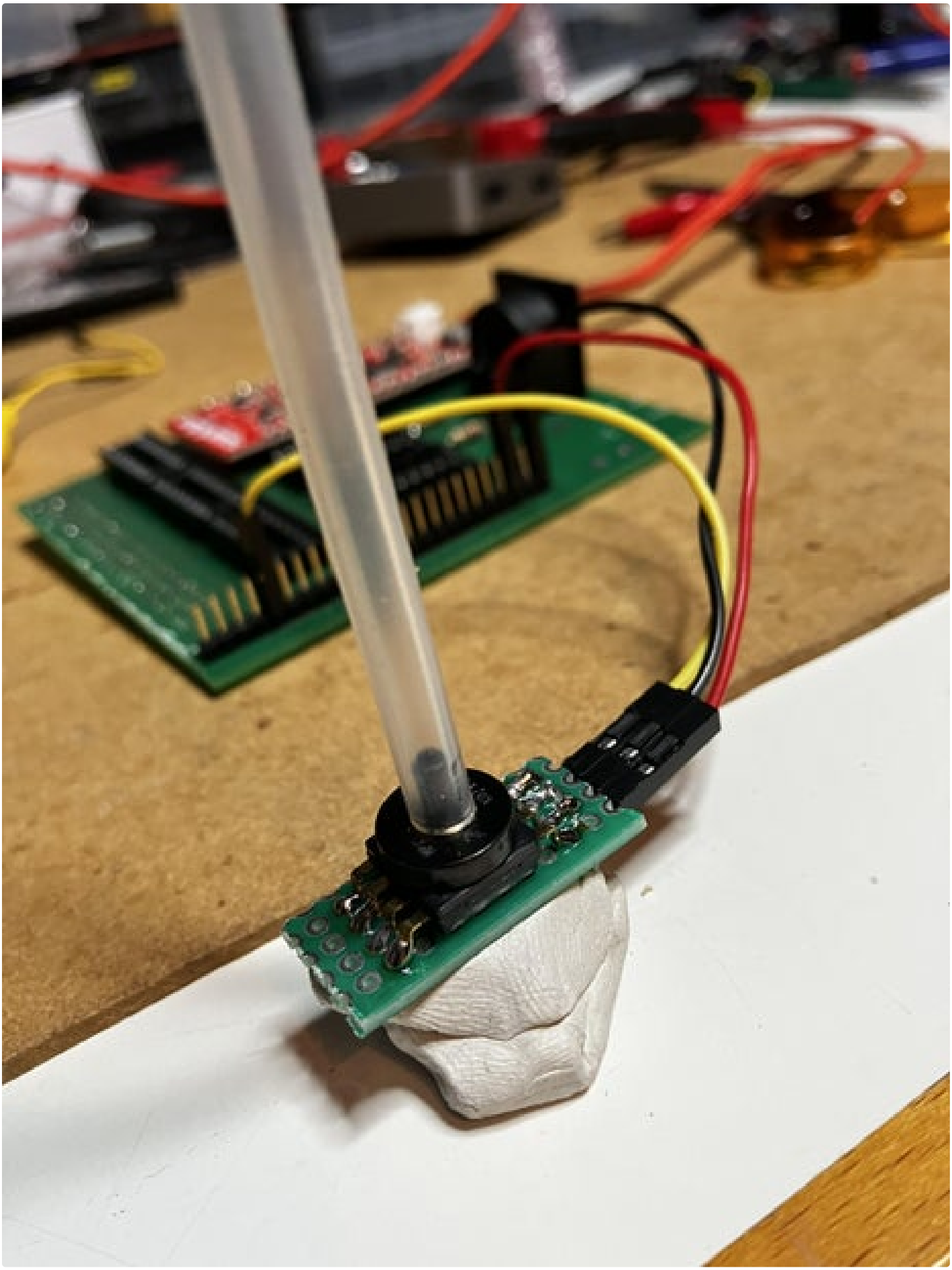
So the foot pedal has the advantage to be able to control the level of your MIDI signal separately from your playing out loud on the horn, which is actually pretty interesting from an artistic point of view. Of course this comes at the price of being less expressive, as your foot is well... not exactly your painstakingly trained airflow.

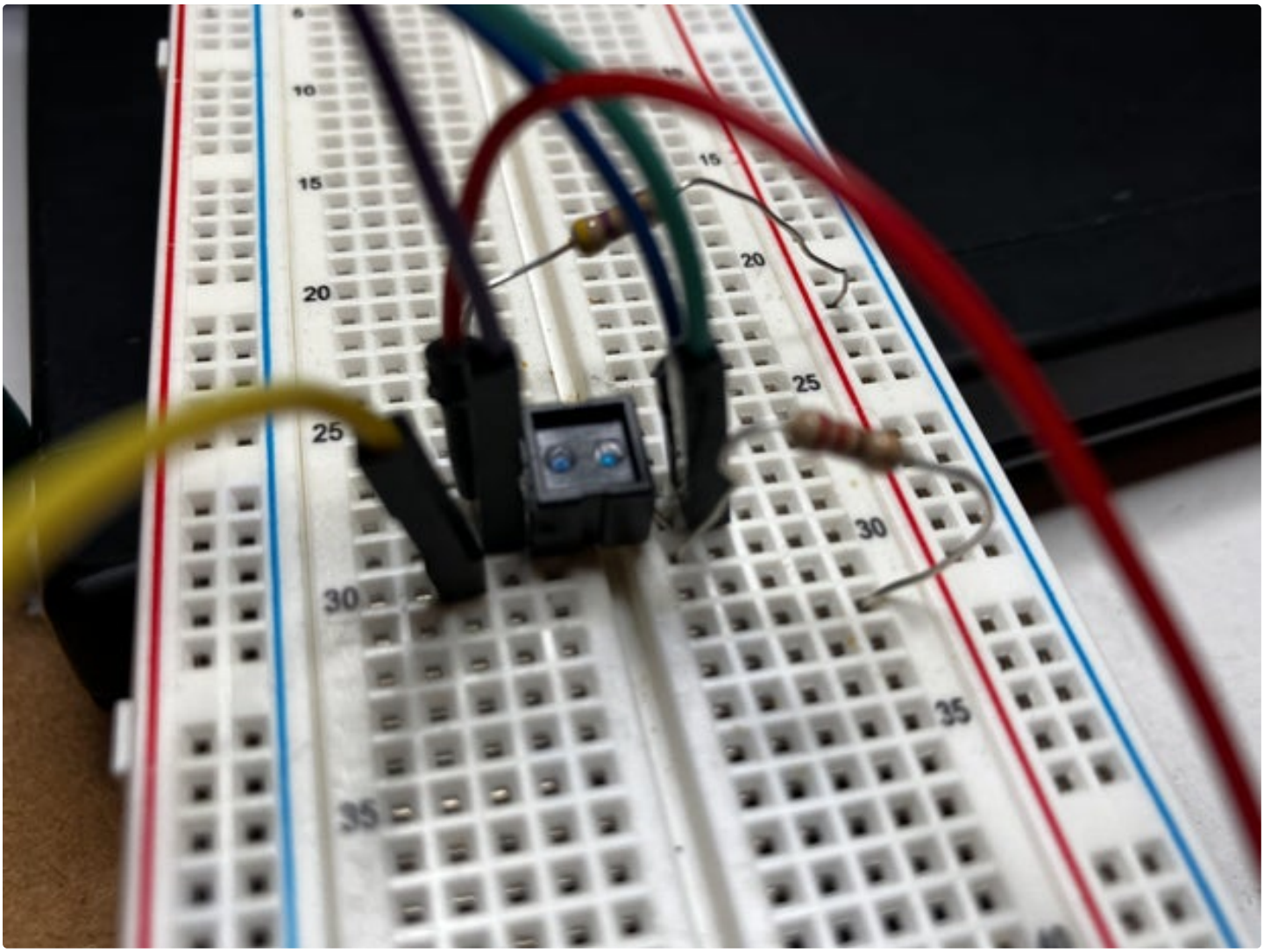
The good news is the board has still some pins available to accommodate future updates in this regard. And the real nice deal is, you can do it yourself if you're up to it and can't wait for my next update!

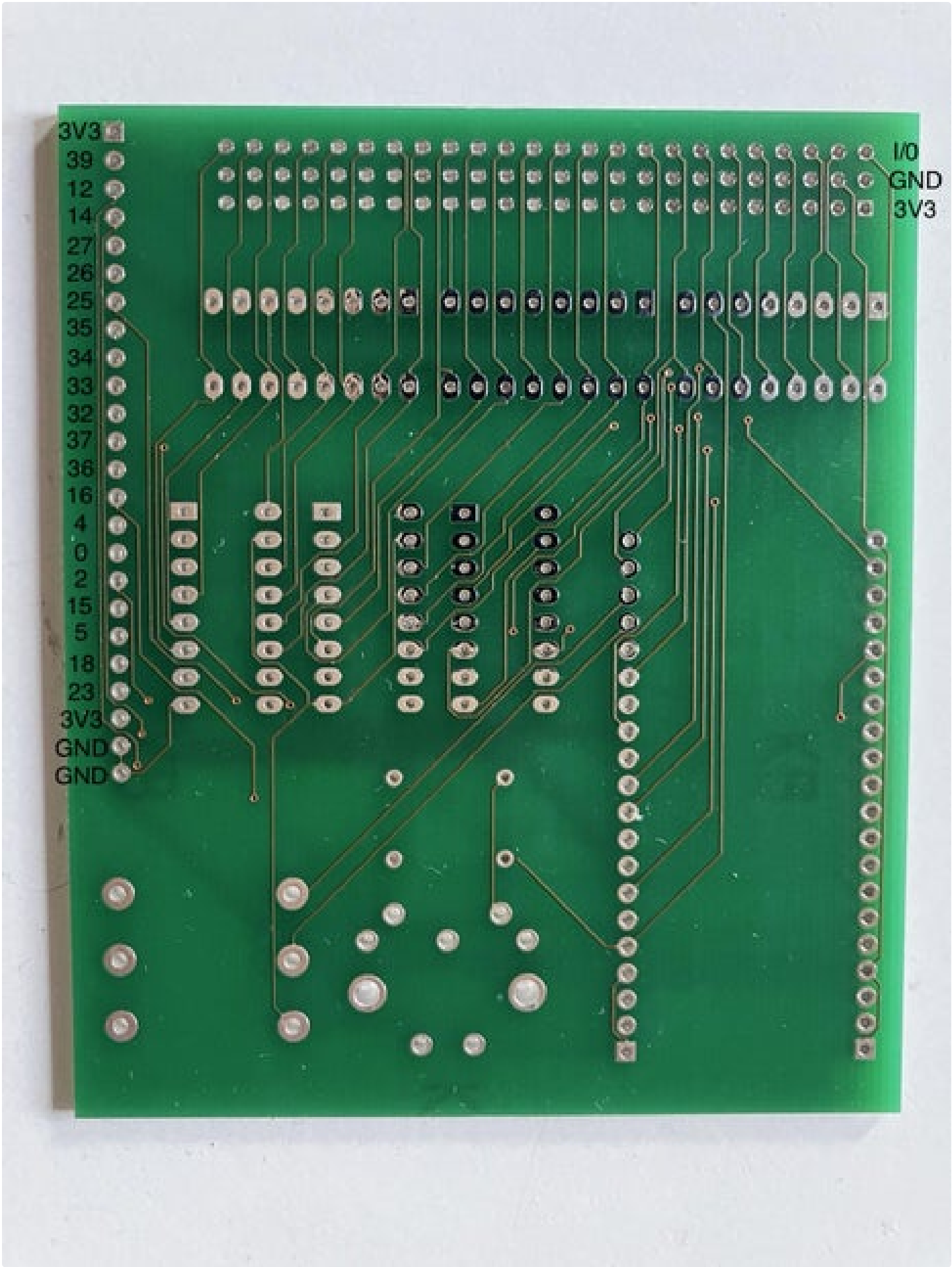
All the maker files are available, including all KiCad project files, so please go nuts and keep me updated too (or not, whatever you prefer).

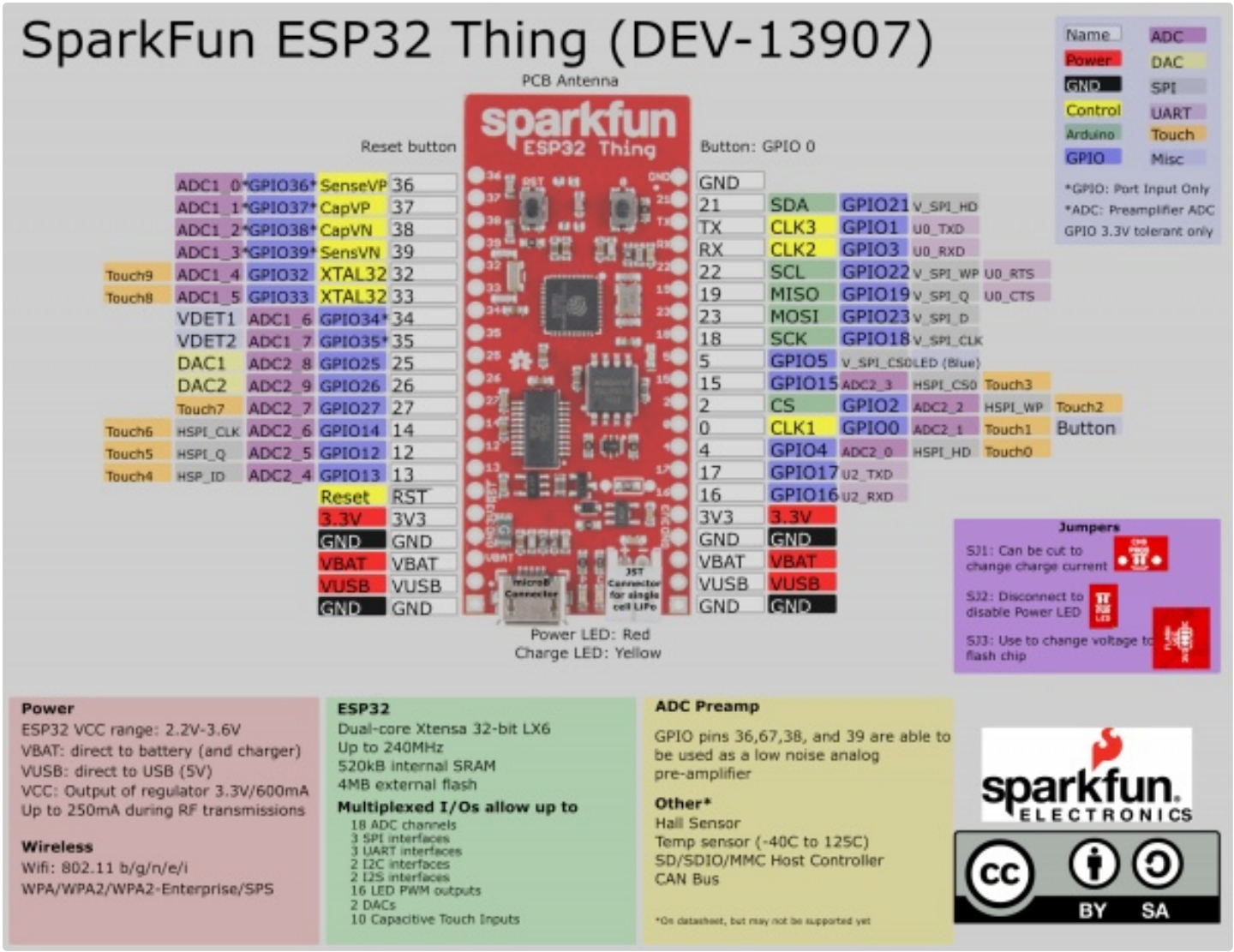
This project has been kindly supported by IDLab, a division of the University of Antwerp. Big thanks to Steven Latré and Kurt Van Herck for making this possible.

PS: I did manage to include proper breath control and I will update this instructable at the proper time. Now just give me a break, I'm doing a gig first to show off my work. Stay tuned!




10
15
20
25
30
35
15
20
25
30
35


3V3
39
12
14
27
26
25
35
34
33
32
37
36
16
4
0
2
15
5
18
23
3V3
GND
GND
I/O
GND
3V3


SparkFun ESP32 Thing (DEV-13907)
PCB Antenna
Reset button
Button: GPIO 0
sparkfun
ESP32 Thing
Name
Power
GND
Control
Arduino
GPIO
ADC
DAC
SPI
UART
Touch
Misc
*GPIO: Port Input Only
*ADC: Preamplifier ADC
GPIO 3.3V tolerant only
Power LED: Red
Charge LED: Yellow
Jumpers
SJ1: Can be cut to change charge current
SJ2: Disconnect to disable Power LED
SJ3: Use to change voltage to flash chip
Power
ESP32 VCC range: 2.2V-3.6V
VBAT: direct to battery (and charger)
VUSB: direct to USB (5V)
VCC: Output of regulator 3.3V/600mA
Up to 250mA during RF transmissions
Wireless
Wifi: 802.11 b/g/n/e/i
WPA/WPA2/WPA2-Enterprise/SPS
ESP32
Dual-core Xtensa 32-bit LX6
Up to 240MHz
520kB internal SRAM
4MB external flash
Multiplexed I/Os allow up to
18 ADC channels
3 SPI interfaces
3 UART interfaces
2 I2C interfaces
2 I2S interfaces
16 LED PWM outputs
2 DACs
10 Capacitive Touch Inputs
ADC Preamp
GPIO pins 36,67,38, and 39 are able to be used as a low noise analog pre-amplifier
Other*
Hall Sensor
Temp sensor (-40C to 125C)
SD/SDIO/MMC Host Controller
CAN Bus
*On datasheet, but may not be supported yet
sparkfun ELECTRONICS
BY SA