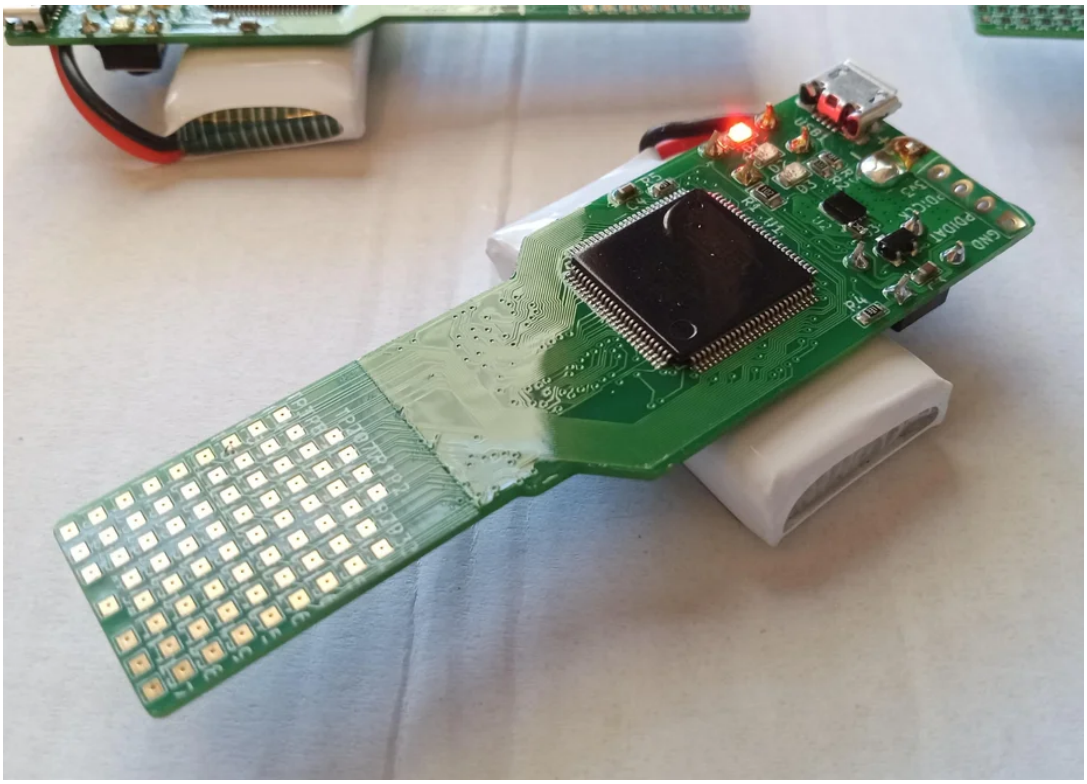
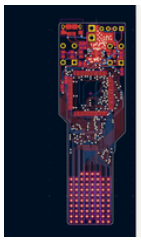
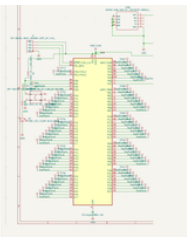
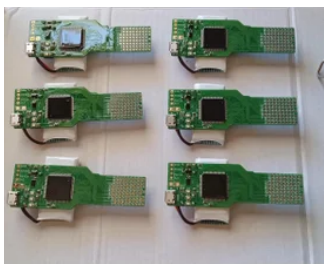
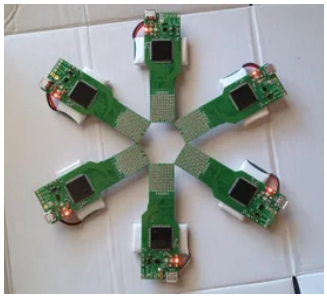
Signal-randomized Translingual Neurostimulator (TLNS Based on PoNS: Portable NeuroStimulator)

By quicksilv3rflash in CircuitsElectronics
Published Oct 29th, 2025
PUBLICDOMAIN

Introduction: Signal-randomized Translingual Neurostimulator (TLNS Based on PoNS: Portable NeuroStimulator)

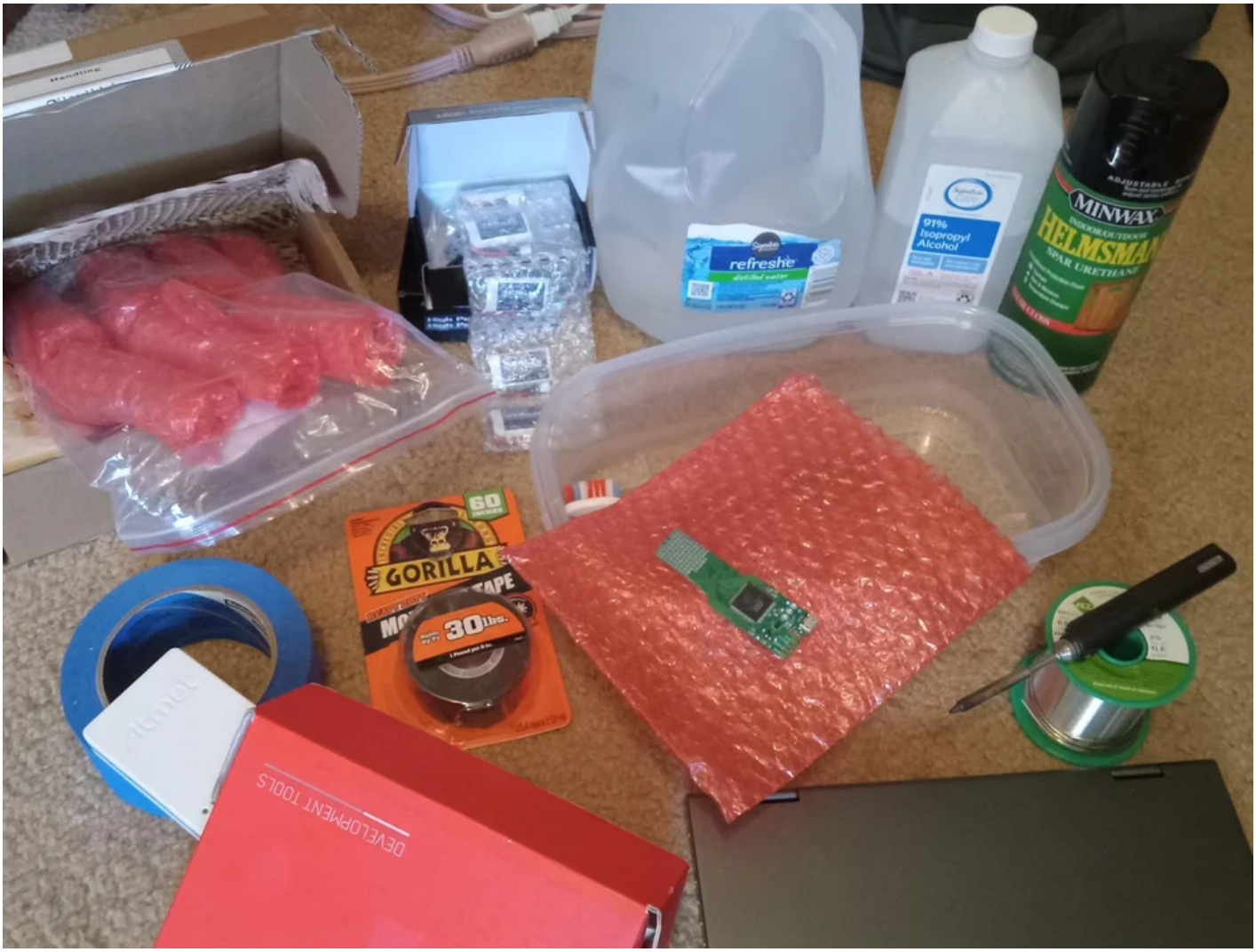






This project was commissioned by a gentleman in Stockholm in order to replicate the functionality of the PoNS ("Portable Neurostimulator") device. My understanding is that there is some evidence that this form of neurostimulation, translingual neurostimulation or TLNS, may be useful in improving attention-related cognitive processes in healthy humans.

It may also be particularly useful when combined with physical therapy, enhancing rehabilitation in those with traumatic brain injury (TBI), stroke, Multiple Sclerosis (MS), and Parkinson's Disease (PD). For additional information, see "Real world evidence of improved attention and cognition during physical therapy paired with neuromodulation: a brain vital signs study".

Kirby ED, Jones CB, Fickling SD, Pawlowski G, Brodie SM, Boyd LA, Venter J, Moser N, Kalsi-Ryan S, Medvedev G, D'Arcy RCN. Real world evidence of improved attention and cognition during physical therapy paired with neuromodulation: a brain vital signs study. Front Hum Neurosci. 2023 Jun 9;17:1209480. doi: 10.3389/fnhum.2023.1209480. PMID: 37362950; PMCID: PMC10289164.

Supplies



Assembled TLNS boards (see ordering instructions below)

Spar Urethane

https://www.truevalue.com/product/helmsman-11-5-oz-aerosol-high-gloss-spar-urethane/

Double-sided mounting tape

https://www.homedepot.com/p/Gorilla-1-in-x-1-67-yd-Black-Heavy-Duty-Mounting-Tape-6055002/308910063

Masking tape

https://www.homedepot.com/p/3M-ScotchBlue-1-88-In-x-60-Yds-Original-Multi-Surface-Painter-s-Tape-3-Rolls-2090-48EP3/305218415

Toothbrush

https://www.amazon.com/GuruNanda-DentalGuru-Gum-Massager-Toothbrush/dp/B0CHGQ4VHY

Distilled or deionized water

https://www.amazon.com/Distilled-Steam-Distilled-Drinking-Appliances-Medical/dp/B0FX43RGCT

Isopropanol (91%)

https://www.amazon.com/Isopropyl-Alcohol-Antiseptic-Topical-Solution/dp/B0FNK4FQLC

Rectangular food storage container

https://www.walmart.com/ip/Rubbermaid-Red-Plastic-Food-Storage-Container-3-Count/16664882

Lithium-Ion cells with built-in protection circuit (450mAh)

https://www.amazon.com/dp/B0CQ71ZS1X

Soldering iron (I'm using a MINIWARE TS100 -- since replaced by the TS101)

https://www.amazon.com/NovelLife-TS101-Soldering-Adjustable-Temperature/dp/B0BLNHB11B

Solder (SN100C alloy, 3% NC601 flux, 0.032" diameter)

https://www.amazon.com/FCT-SN100C-NC601-Clean-Solder/dp/B07RWRSYKK

AVR programmer (Atmel-ICE)

https://www.digikey.com/en/products/detail/microchip-technology/ATATMEL-ICE/4753379

Pin header (1x4, 2.54mm spacing)

https://www.digikey.com/en/products/detail/w%C3%BCrth-elektronik/61300411121/4846827

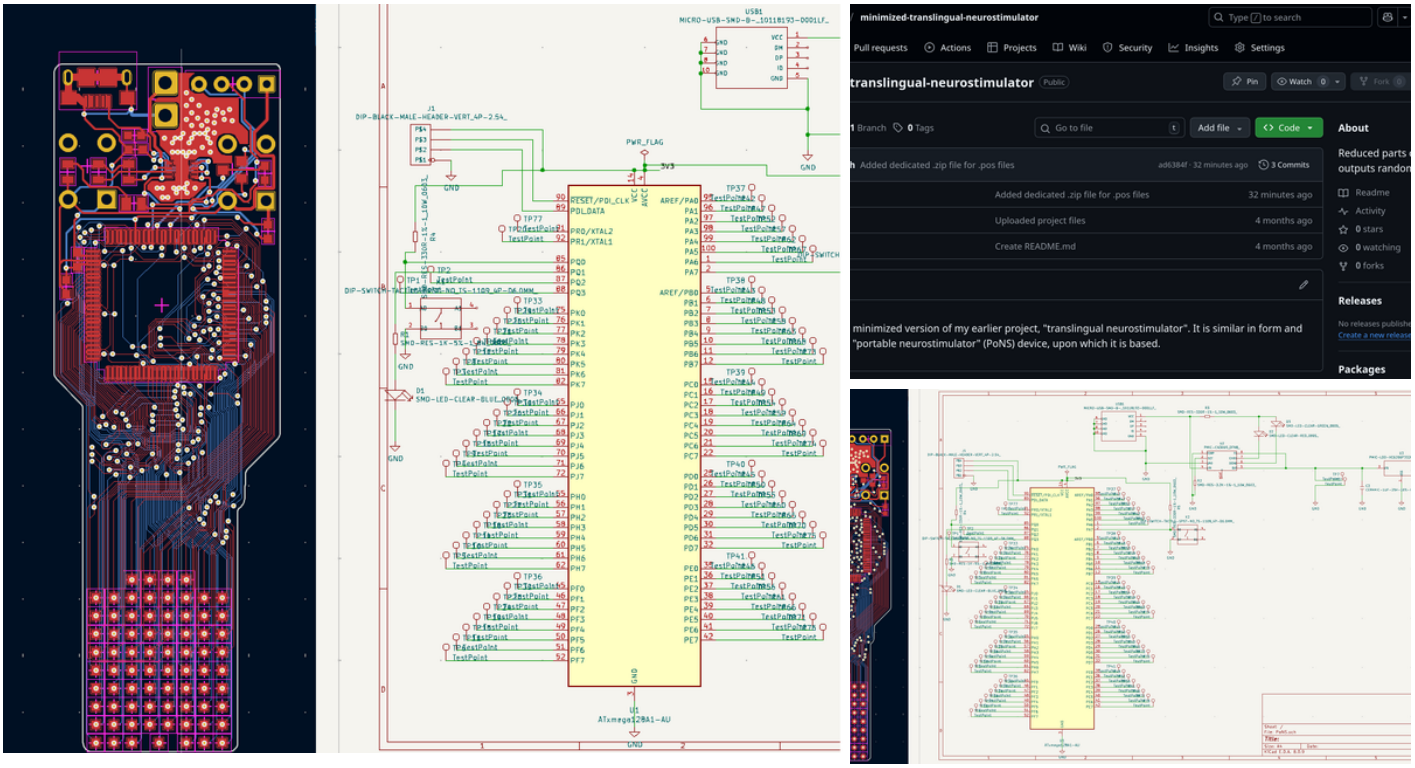
Personal computer (I'm using a CHUWI MiniBook X N100 -- since replaced by MiniBook X N150)

https://www.chuwi.com/product/items/chuwi-minibook-x-n100.html

Atmel Studio (version 7.0.2389)

https://sourceforge.net/projects/gcbasic/files/Support%20Files/ATMELCompilers/ATMEL7Studio_installer-7.0.2389-full.exe/download

Step 1: Design Files



In the interest of completeness, and to ensure that you, dear reader, may modify this project to suit your needs as desired, I am linking to the complete set of project files here:

https://www.neuroplustech.com/minimized-translingual-neurostimulator-main.zip

They are also available via this project's GitHub repository:

https://github.com/quicksilv3rflash/minimized-translingual-neurostimulator

These files were created using KiCad and Atmel Studio. While the most recent version of KiCad, available free here:

https://www.kicad.org/download/

will open the schematic and circuit board design files with minimal complaint*, Atmel Studio has since been discontinued and replaced with Microchip Studio, which was then itself replaced with MPLAB X IDE; I am uncertain of the compatibility of Atmel Studio files with Microchip Studio, let alone with MPLAB X IDE. The most recent (and final) release of Atmel Studio, version 7.0.2389, may be downloaded from here:

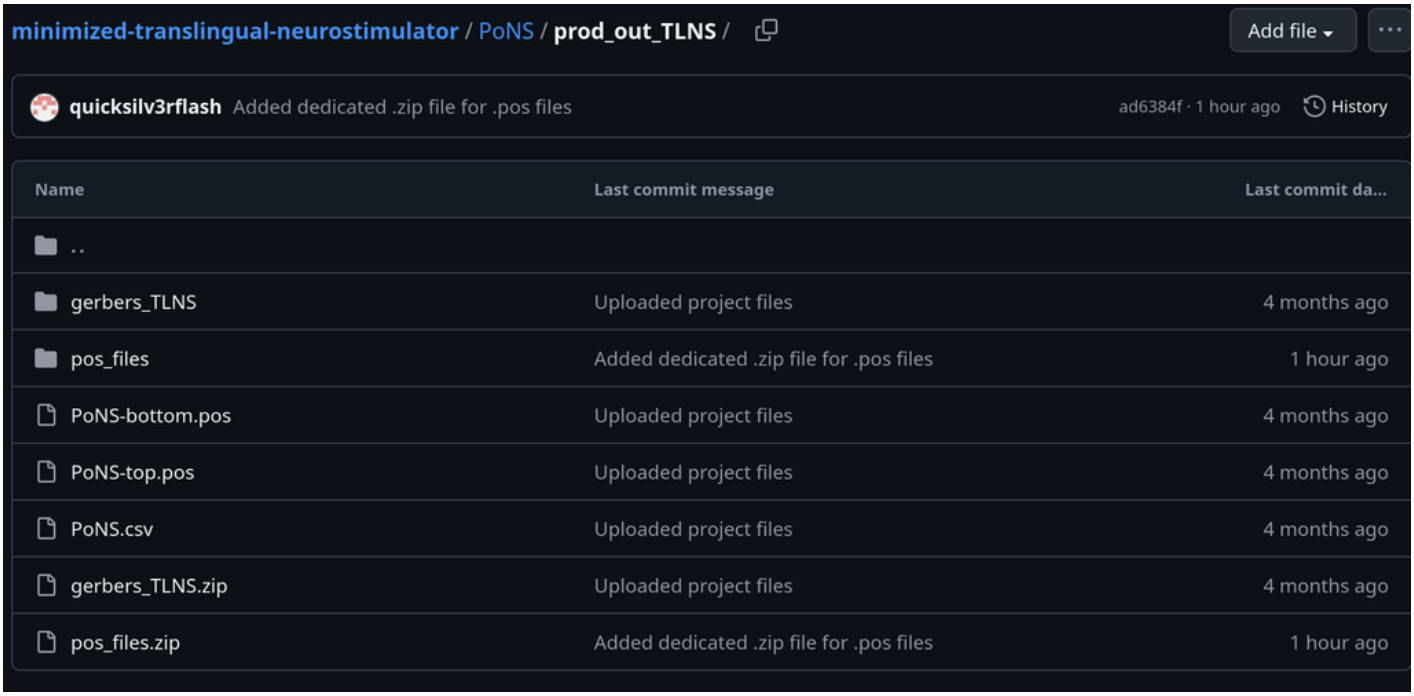
https://sourceforge.net/projects/gcbasic/files/Support%20Files/ATMELCompilers/ATMEL7Studio_installer-7.0.2389-full.exe/download

*I used some components from the SeeedStudio OPL Kicad Library; should you find yourself

needing to make extensive modifications to the design, said library is available here:

https://github.com/Seeed-Studio/OPL_Kicad_Library

Step 2: Circuit Board Fabrication Files



While this design could be hand-soldered, in this case I found it most expedient to use SeeedStudio's FusionPCBA service to assemble the circuit boards. The required fabrication files are listed below.

*gerbers_TLNS.zip: circuit board layout files

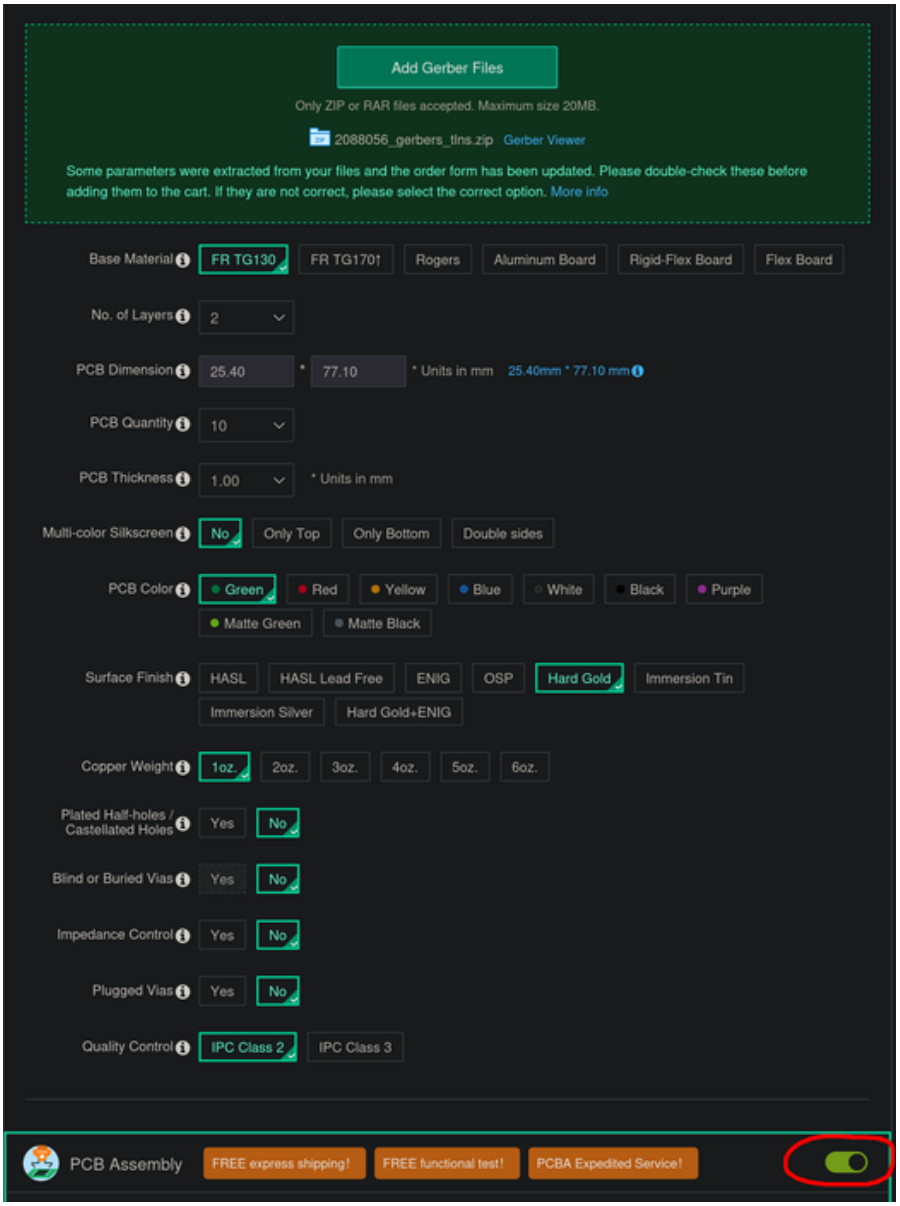
*PoNS.csv: bill of materials

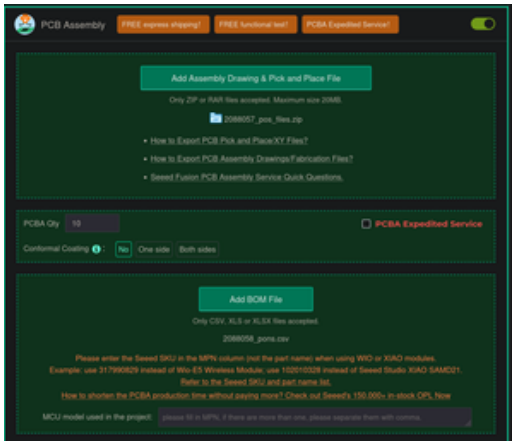
*pos_files.zip: X-Y position files (.pos files)

These files are available at this project's GitHub repository here:

https://github.com/quicksilv3rflash/minimized-translingual-neurostimulator/tree/main/PoNS/prod_out_TLNS

Step 3: Placing the Fabrication Order





This project requires a special surface finish, hard gold, so SeeedStudio's standard PCBA (printed circuit board assembly) service will not be able to produce boards to the required specification. SeeedStudio's standard PCBA will only provide boards with an ENIG (electroless nickel immersion gold) plating. ENIG surface plating ranges in thickness from 0.05 to 0.15 micrometers (50 nanometers to 150 nanometers). Given that a gold atom's radius is only about 135 picometers, that means that, on the thinner side, the ENIG finish may be only 370 atoms thick. For this application, a class 3 surface plating (greater than 2.5 micrometers) is advisable. More information about gold surface plating classes and types may be found here:

https://www.valencesurfacetech.com/the-news/astm-b488/

The reason that a hard gold surface finish is required is because this device runs current between exposed electrodes inside the human mouth, a saline environment. If you've ever tried putting copper wires in salt water and running current between them, you will know that they tend to dissolve, turning into a pile of sludge at the bottom of your container in short order. In an aqueous saline solution, the free charge carriers are ions -- Na+ and Cl- in common table salt -- and so at the positive electrode, for example, Cl- ions are attracted to (and then react with) the electrode. This is an oversimplification, and the actual mechanics become complex to analyze, particularly given the many diverse chemical constituents of human saliva. Plating the exposed electrodes with a noble metal such as platinum, iridium, palladium, or (in this case) gold will prevent them from corroding, at least as long as the user doesn't have an aqua regia-based toothpaste or mouthwash. Using alternating current can diminish the electrochemical etching effect, but only somewhat, such that a suitably thick hard gold finish is required. More on that later.

In any case, you, dear reader, will need to upload the files listed above -- gerbers_TLNS.zip, PoNS.csv, and pos_files.zip, to SeeedStudio's advanced printed circuit board assembly service, linked here:

https://www.seeedstudio.com/fusion-advanced-pcb.html

Unfortunately you will need to make an account with SeeedStudio before they allow you to do this. After having done so, click the "Add Gerber Files" button at the top of the page, and upload gerbers_TLNS.zip. Then, a whole passel of options must be selected:

Base Material: FR TG130

No. of Layers: 2

PCB Dimension: 25.40 * 77.10 (this should auto-fill)

PCB Quantity: 10

PCB Thickness: 1.00 (this is actually up to you, 1mm seems about right to me)

Multi-color Silkscreen: No

PCB Color: Green (this is actually up to you, green is just the standard)

Surface Finish: Hard Gold

Copper Weight: 1oz.

Plated Half/holes / Castellated holes: No

Blind or Buried Vias: No

Impedance Control: No

Plugged Vias: No

Quality Control: IPC Class 2

Under these options, you will see the text "PCB Assembly". To the right of it is a toggle slider. Click it to open PCB Assembly options. A button should appear to "Add Assembly Drawing & Pick and Place File". Click "Add Assembly Drawing & Pick and Place File". Upload pos_files.zip.

Then, more options!

PCBA Qty: 10

PCBA Expedited Service (up to you, but I left it unchecked, as it costs more)

Conformal Coating: No

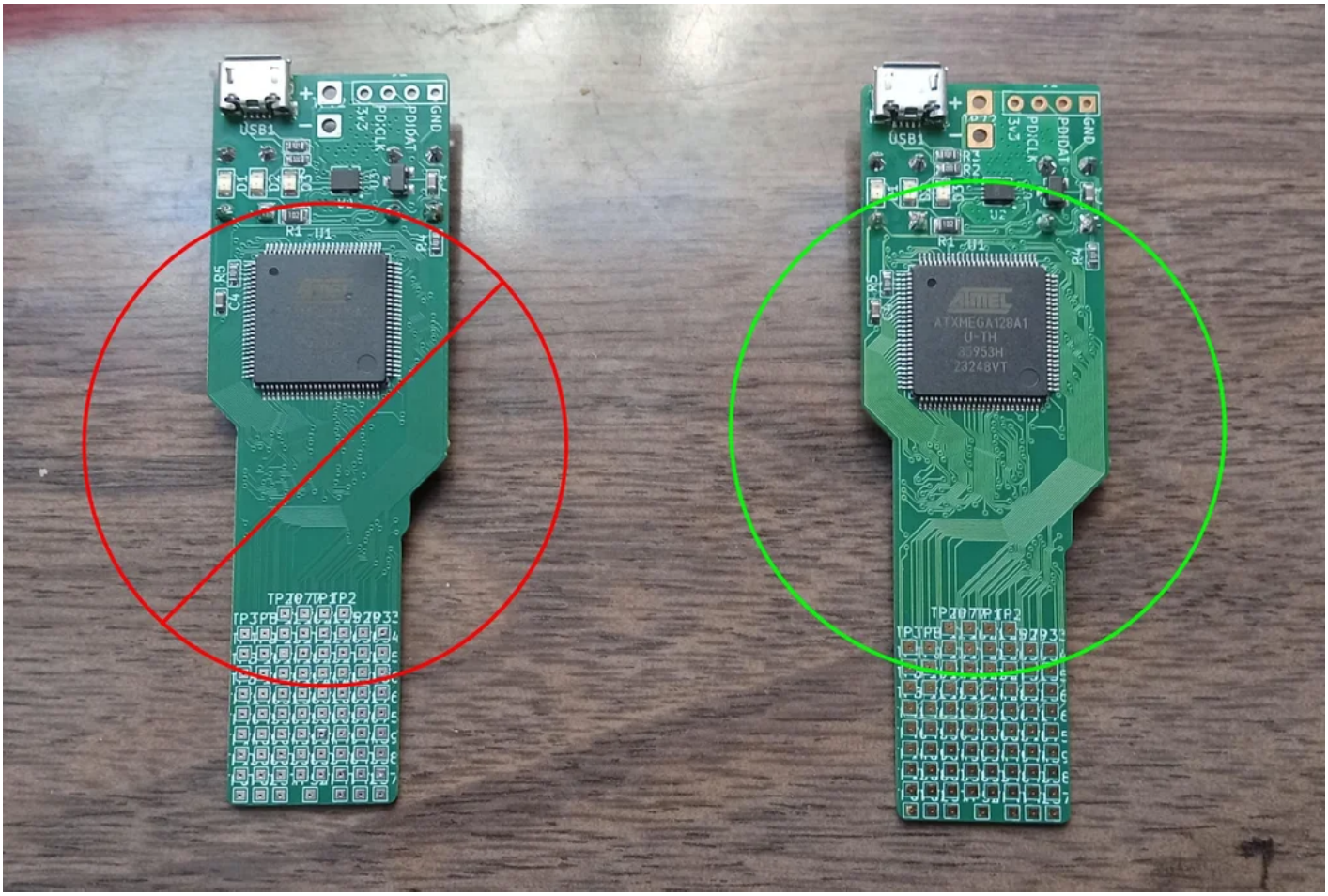
Below this, there will be a button labeled "Add BOM file". Click it, and upload PoNS.csv.

Under the "100% Functional Testing" header, click "Add test plan later".

Then, scroll back up and ensure that all of the above options are set as described. In particular, Seeed's system seems to like to switch the surface finish from "hard gold" to something else after uploading the BOM file. At least, it did so as I was going through all of these steps to take screenshots. Double and triple-check, dear reader!

Then, click the button on the right that says "Send Quotation Request". It'll be a few days before you can even place an order -- their engineers have to look over the design to ensure that they can fabricate it on their end.

Step 4: Potential Fabrication Pitfalls

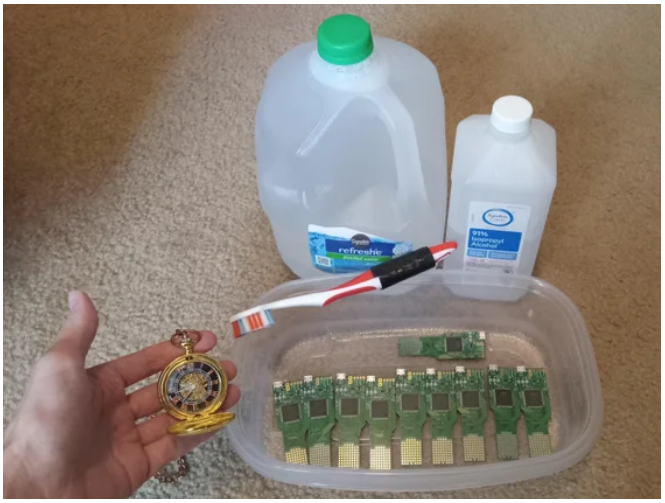


When I received the production order for this design, there was a grave error. The boards were shipped to me with (probably leaded) HASL -- hot air solder leveling -- finish. Very much unsuitable to put in somebody's mouth and dissolve away into their saliva with what amounts to an electrochemical etching process, from the circuit board's perspective!

As a result, I needed to contact the manufacturer and place the order again. The second time, I received boards with the appropriate hard gold surface finish.

Beyond getting boards with the wrong surface finish, you may be contacted by Seeed's engineers requesting that you specify the thickness of the hard gold surface plating. I specified 30 micrometers.

Step 5: Toothbrush Time



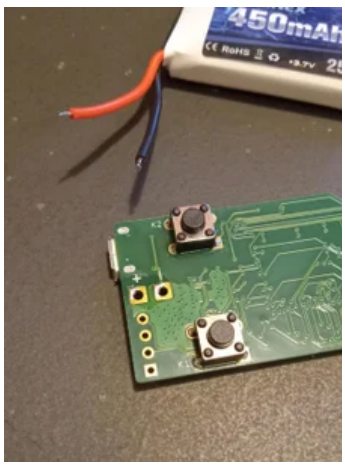
When you have received the assembled circuit boards, verify that the surface plating on the pads without components is correct. It should look like gold. If it's silvery, you've got a problem.

At long last, you are ready to begin the work on your end. Soak the boards in distilled water for 10 minutes, gently scrub them with a toothbrush, drain them, and then replace the distilled water. Allow the boards to soak for an hour. Then, gently scrub the boards with a toothbrush and rinse the boards (and container) with distilled water. Set the boards aside on a clean surface and shake any excess water out of the container.

Next, put the boards in a bath of 91% isopropanol and swish the bath over the boards to ensure that the isopropanol clears away any water. Allow to soak for 30 minutes. Gently scrub the boards with a toothbrush and rinse with isopropanol. Allow to dry for at least 24 hours before moving on to the following steps.

The purpose of this step is to wash away any residual contaminants that the user would otherwise end up eating. While moisture exposure can reduce the lifespan of the LEDs on the boards, I consider it more important to minimize user exposure to leftover manufacturing contaminants than it is to protect the lifespan of the on-board LEDs. LEDs are straightforward to desolder and replace. End users, less so. It is recommended that you somehow distinguish the toothbrush used for this sort of work from your other toothbrushes, whether by cutting a notch in the handle or by wrapping duct tape around it.

Step 6: Preparing for Programming



If you have not done so already, install Atmel Studio 7.

Cut off the connector of one of the batteries. It is very important that you do this *one wire at a time* or it will short-circuit and may catch on fire or explode. Strip the battery wires, taking care not to let them touch, or it will short-circuit and may catch on fire or explode. Tin the exposed battery wires with solder (heat the wire from one side with the already-tinned soldering iron tip, and apply solder to the wire so that it fills in the gaps between the strands). Partially tin the pads labeled + and - on the side of the assembled device board with the buttons, as shown. It's not the end of the world if you fill in the holes with solder by accident, but it will make future steps more difficult. I've found that 350°C is the best temperature for working with the SN100C alloy solder listed above.

Tack the battery to the tinned locations on the board, taking care to connect the red wire to the pad labeled + and the black wire to the pad labeled - (see picture). This temporary power connection will allow the microcontroller (ATXMEGA128A1-AUR) to be programmed.

Step 7: I Really Need to Learn Mandarin

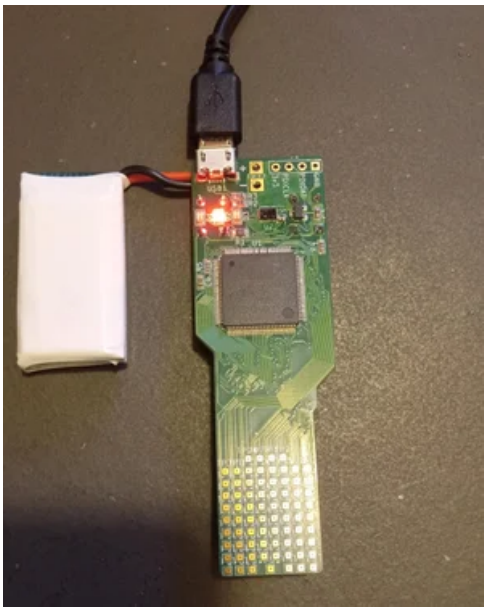
	Function Description
	Temperature Sense Input. Connecting TEMP pin to output in Lithium ion battery pack. If TEMP pin's volt input supply voltage V_{IN} for more than 0.15S, this mea temperature is too high or too low, charging is suspend level is above 46% of input supply voltage for more th state is released, and charging will resume. The temperature sense function can be disabled by gro

This, dear reader, is where these instructions stop being what I thought they would be. I thought to myself, "It's just a straightforward set of instructions! I've thoroughly tested this design! What could possibly go wrong?" But. Reality makes fools of us all. The best-laid plans of mice and men, and all that.

My reading comprehension of Mandarin is exceedingly limited. I've learned that 光 (guāng) means light, that 红外 (hóngwài) means infrared, and that 红光 (hóng guāng) means red light -- as in a red LED, not a stoplight. I've concluded that the name I'll write under -- if ever I get to the point of being able to translate what I've made into Mandarin -- would be 汞光 (gǒng guāng), literally "mercury light". I'm sure you're shocked.

In any case, the point I'm driving at here is that I designed and initially tested this circuit with the CN3065 battery charge controller, which is since no longer in production. The fabrication facility suggested CN3166 as a replacement, which is pin-for-pin compatible and fits on the same footprint. "Sounds great!", I said, having skimmed the supplied CN3166 datasheet, which -- I might mention -- was in Mandarin. Come to now, I am testing the boards, and... and the charge function doesn't work. A -mere- seven hours and twenty minutes of troubleshooting later, I have found the difficulty! In CN3065 (original design), you ground the TEMP pin to tell the chip to ignore the temperature sense function. In CN3166, the TEMP pin must be connected to ground through a fixed 10kOhm resistor. Just imagine my surprise, joy, and wonder at this revelation!

Step 8: The Land of Tweezers and Razor Blades



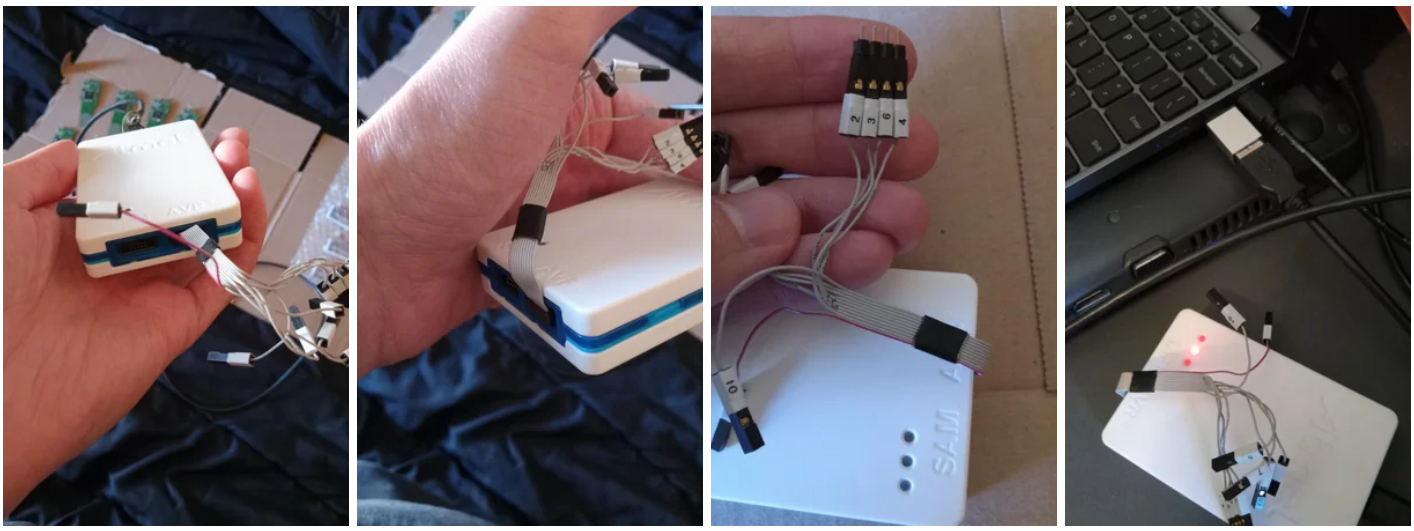
This step is included merely in the interest of completeness, or, perhaps, amusement. You should not need to follow the instructions in this step unless for some reason you have chosen to build the device based on the files in the "PoNS_deprecated" folder, in which case you must also remember that the ordering system will note that 2 parts "need to be checked manually". Purchase links must be pasted in next to each.

U3: https://www.digikey.com/en/products/detail/torex-semiconductor-ltd/XC6206P332MR-G/7386004

U2: https://jlcpcb.com/partdetail/ShangHai_ConsonanceElec-CN3166/C7435233

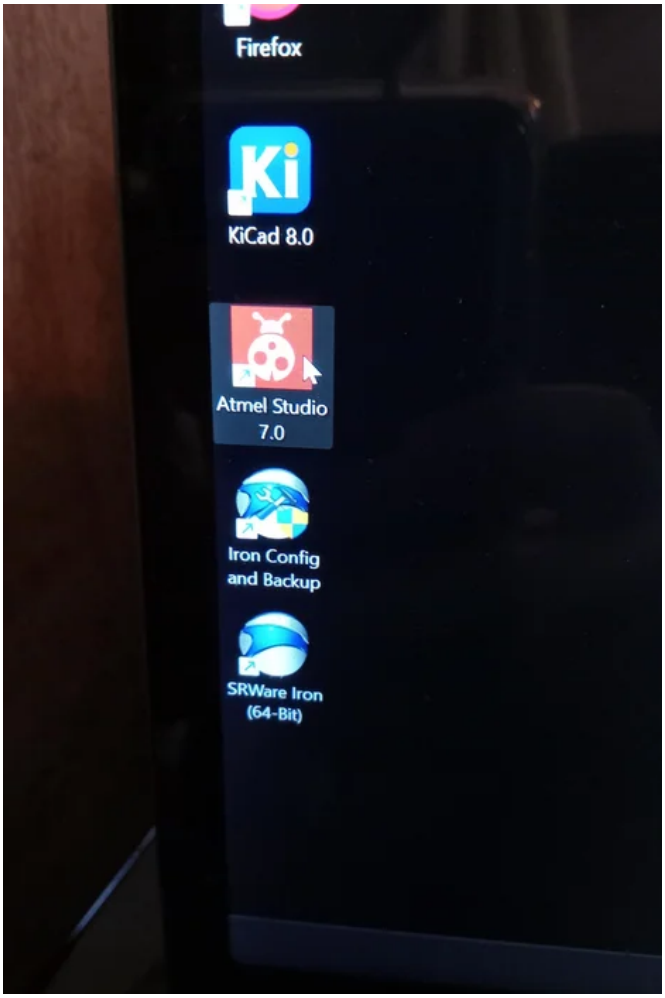
Fixing the problem identified in the previous step requires cutting a trace and adding a fixed 10kOhm resistor in series with the TEMP pin (pin 1) on the CN3166. This involves very careful work with a razor blade -- first to cut the trace itself, and then to scrape away the solder resist on the adjacent ground trace. Next, the soldering iron must be employed in order to tin one terminal of an 0603 (1608 metric) 10kOhm resistor, which is then held with tweezers against pin 1 of the CN3166. Applying the soldering iron reflows the solder and forms a connection between one side of the resistor and pin 1 of the CN3166. The other side of the resistor can then be soldered to the copper exposed on the adjacent ground trace (see pictures). You will not have to do this, as I have since updated the production files to include the required part.

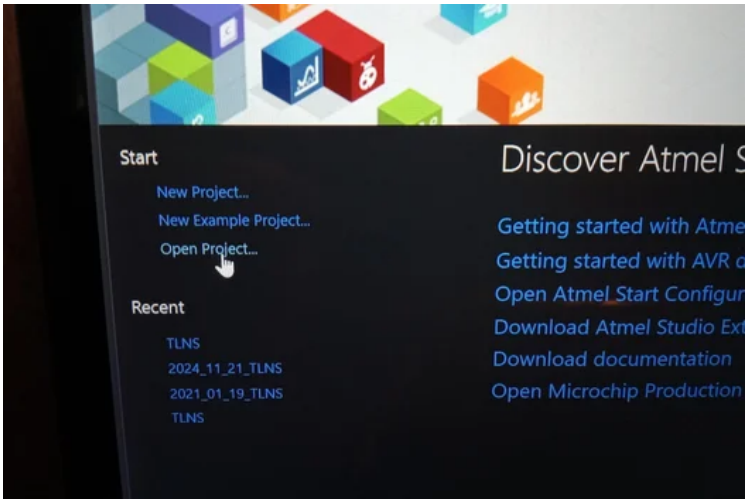
Step 9: Flashing the Code (part 1)

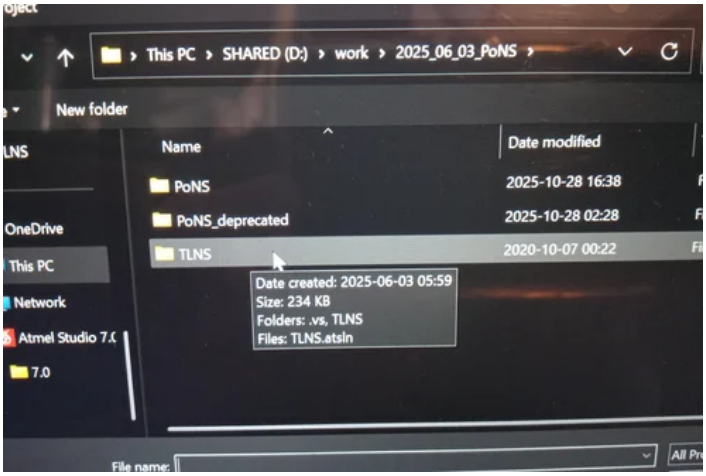


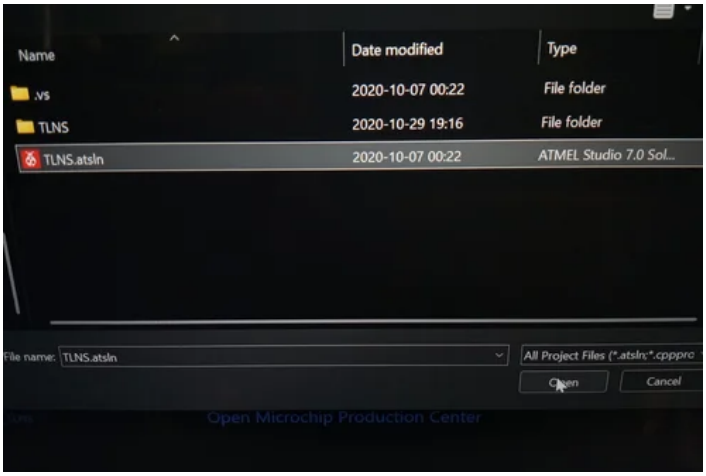
Connect the Atmel-ICE to the 10-pin connector, plugging the connector into the "AVR" port. Plug the pins into the pin header. The pins to be connected to the pin header are 2, 3, 6, and 4, in that order. They are also -- confusingly -- labeled D2, D21, D13, and D3, in very faint black-on-black numbering embossed into the plastic connectors. Connect the micro-USB cable to the Atmel-ICE and the computer. In my case, I needed a USB-A to USB-C adapter, but this will vary depending on the computer you are using.

Step 10: Flashing the Code (Part 2)









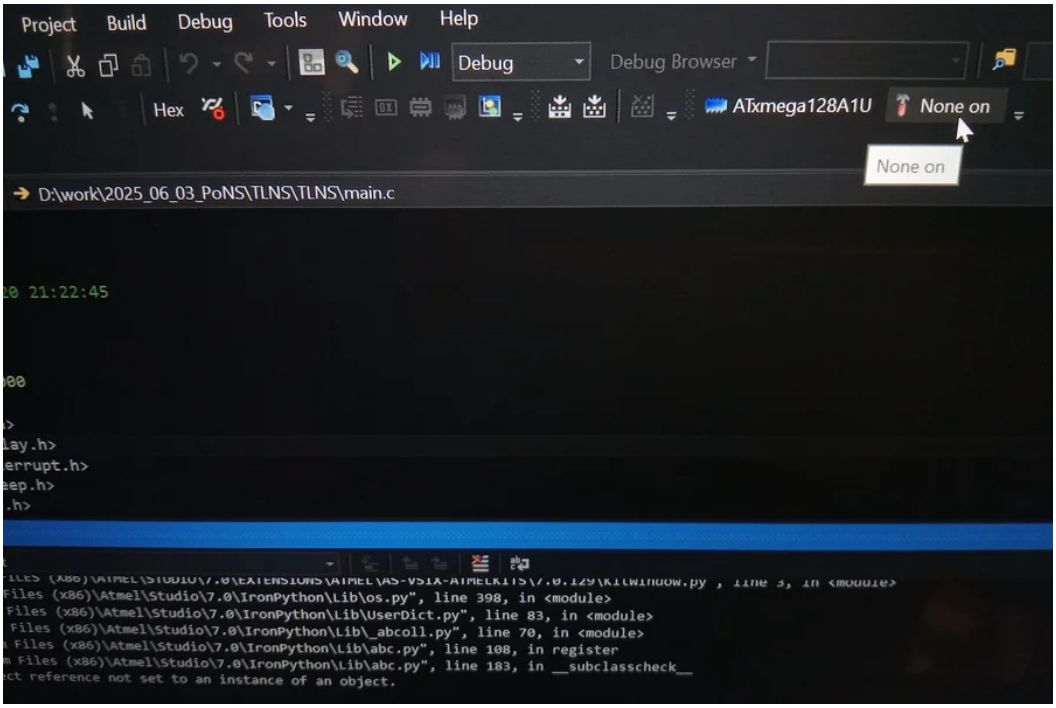
Download the device code. If you downloaded the complete project files from Step 1, it will already be in the sub-directory labeled "TLNS". Otherwise, you will find it here:

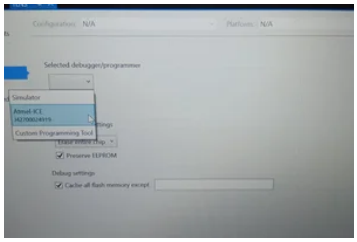
https://github.com/quicksilv3rflash/minimized-translingual-neurostimulator/tree/main/TLNS

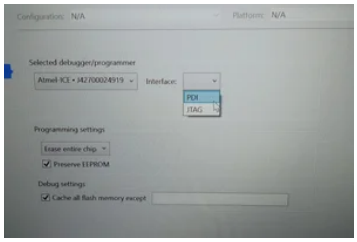
Start Atmel Studio 7.0. Once the program opens, click on "Open Project..." and navigate to

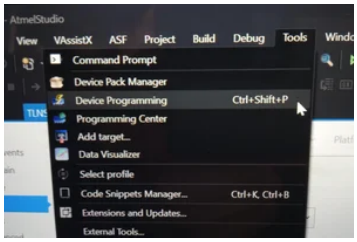
the TLNS folder. Within the TLNS folder, select TLNS.atsln. Once opened, you may have to navigate to -- and open -- main.c within the TLNS folder.

Step 11: Flashing the Code (part 3)







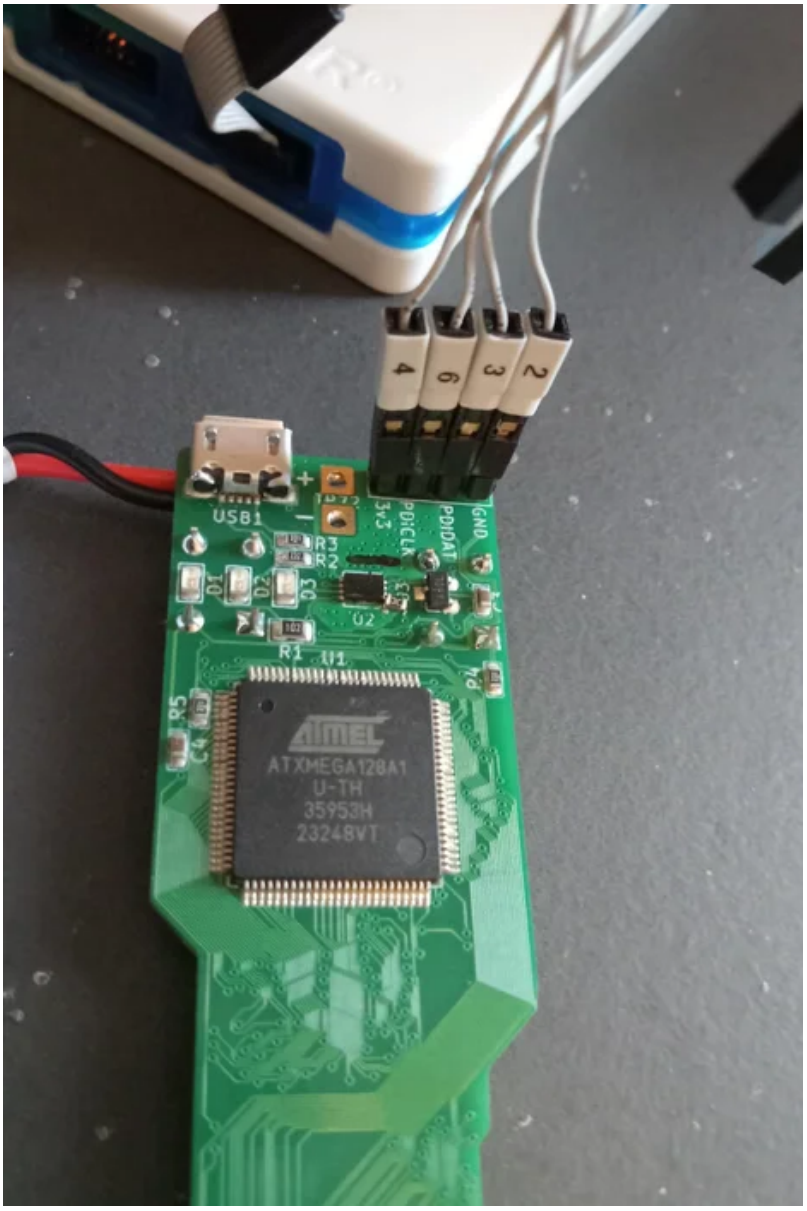


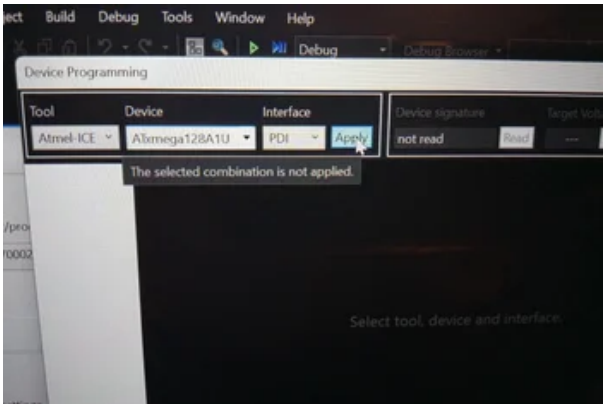
Click the little hammer icon in the top-right of the toolbar. This will open a window labeled "Tool". From the "Selected debugger/programmer" dropdown menu, select "Atmel-ICE". (If it isn't appearing, try closing and re-opening Atmel Studio, or rebooting the whole computer, or using a different micro-USB cable -- though the one included with the kit should almost certainly work.)

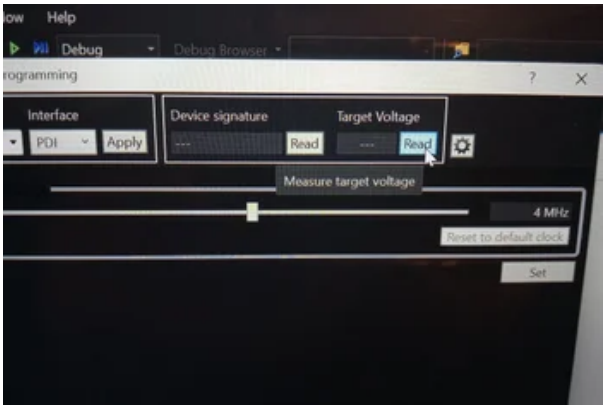
Next, under the "Interface" dropdown menu, select "PDI".

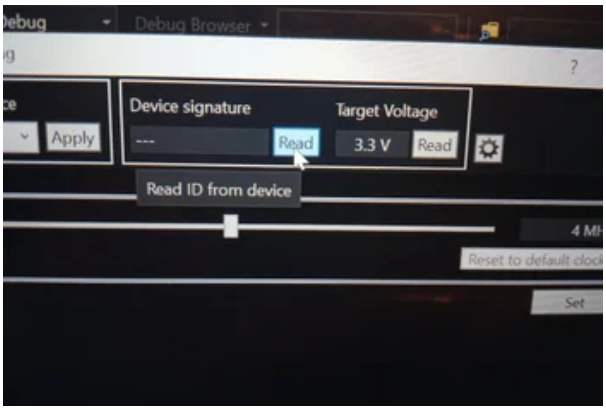
Then, from the top menu bar, select "Tools > Device Programming".

Step 12: Flashing the Code (Part 4)





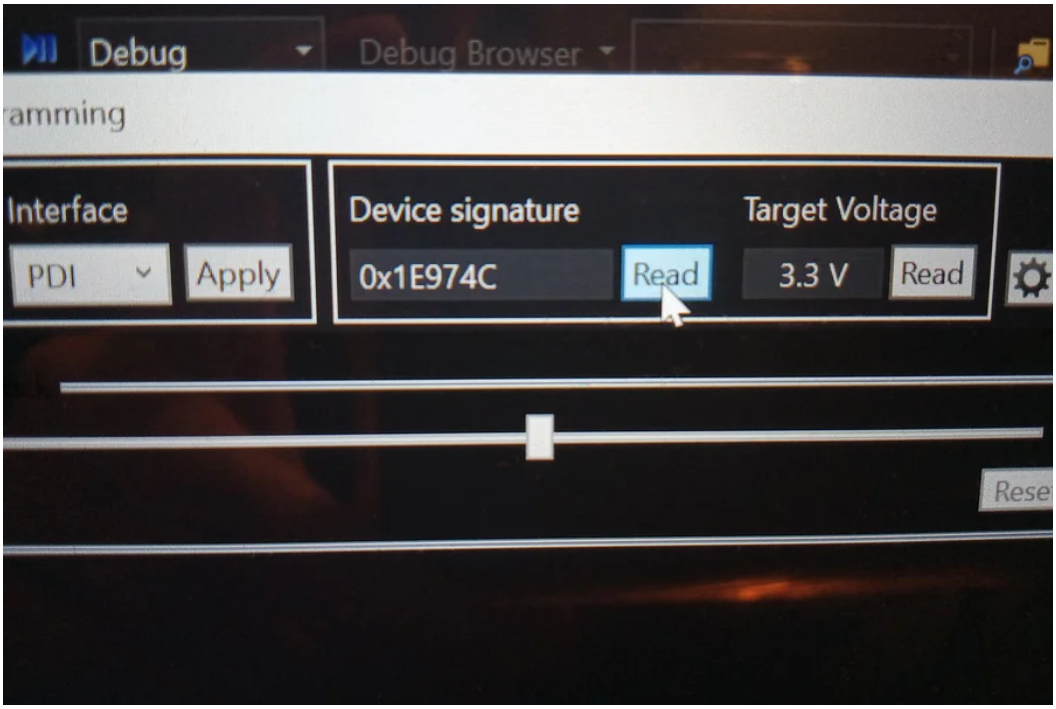


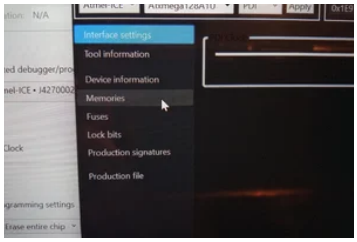


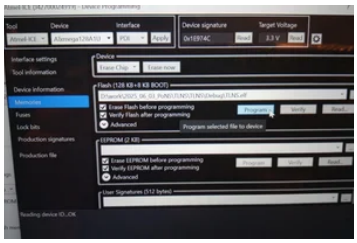
Plug the pin header into the device, as shown. Pin 2 goes to GND, pin 3 to PDIDAT, pin 6 to PDICLK, and pin 4 to 3v3. The green light on the Atmel-ICE should light up. Be advised that the pin header will probably not make full contact with all four of the pads when simply resting in the programming socket; you will most likely need to gently press it at an angle or twist it in order to ensure a complete connection.

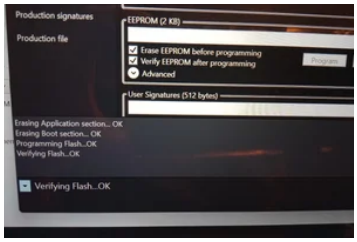
Within the "Device Programming" menu, if they are not already selected, chose Tool:Atmel-ICE, Device:ATxmega128A1U, and Interface:PDI from their respective dropdown menus and click "Apply". Then click the "Read" button (under "Target Voltage"). The reading "3.3 V" should appear under "Target Voltage". This indicates that the Atmel-ICE has read the board's operating voltage correctly.

Step 13: Flashing the Code (Part 5)









Under "Device signature", click "Read". A hexidecimal number should appear. One of my microcontrollers, for example, identified itself as "0x1E974C". This indicates that the data transmission link between the Atmel-ICE and the board is functioning properly.

Then, click "Memories" on the left-hand bar of the "Device Programming" window. Under "Flash (128 KB + 8 KB BOOT)", click "Program".

If all went well, at the bottom of the "Device Programming" window, you should see the following text:

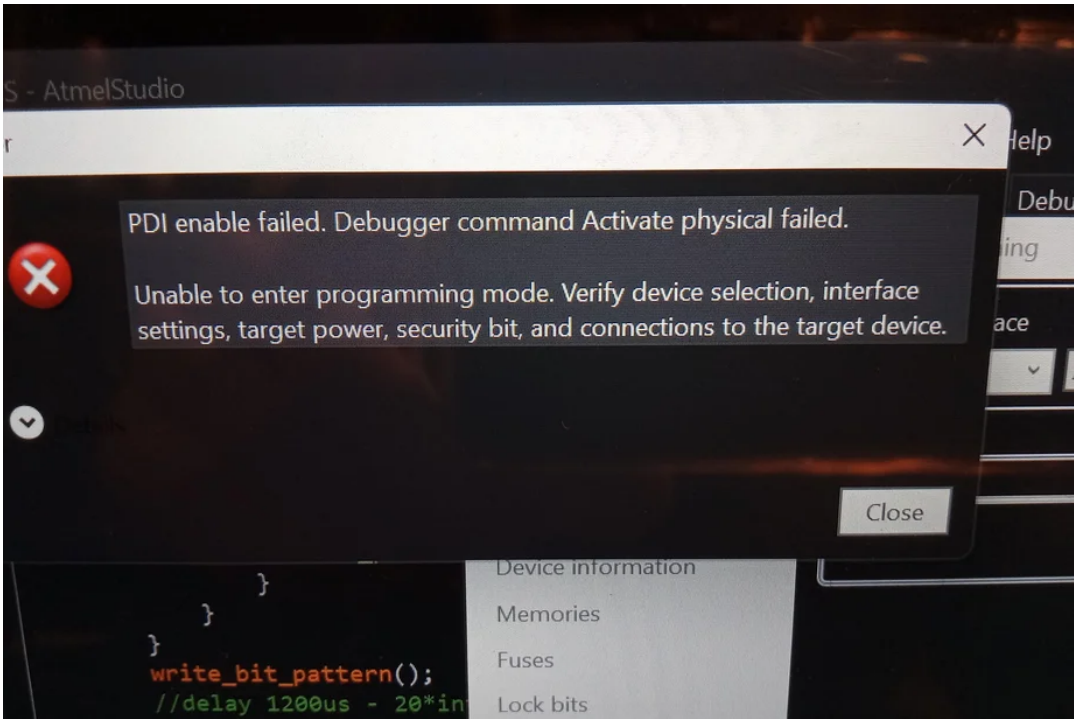
Erasing Application section... OK

Erasing Boot section... OK

Programming Flash... OK

Verifying Flash... OK

Step 14: Potential Code Flashing Pitfalls

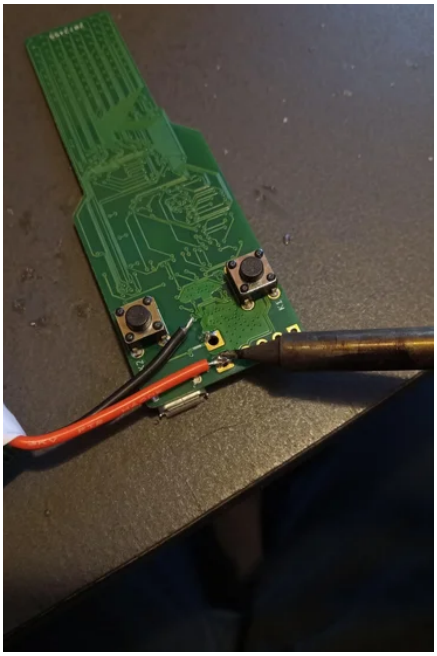


While attempting to program the board, you may see the error "PDI enable failed. Debugger command Activate physical failed. Unable to enter programming mode. Verify device selection, interface settings, target power, security bit, and connections to the target device."

This means -- for this project -- that either:

1. The board isn't properly connected to the programmer. Try gently pressing the pin header against the programming pins. If one direction of pressure doesn't work, try the other one, or try gently twisting the connector (and holding the applied pressure or torsional force during the programming process).
2. The battery isn't properly connected to the board, or it needs to be charged. Verify the battery connections, and try charging the connected battery by plugging the board into a USB power supply and waiting for the charge indicator light to turn green.
3. The board is defective. This is quite possible; one of the ten boards I received simply refused to be programmed at all.

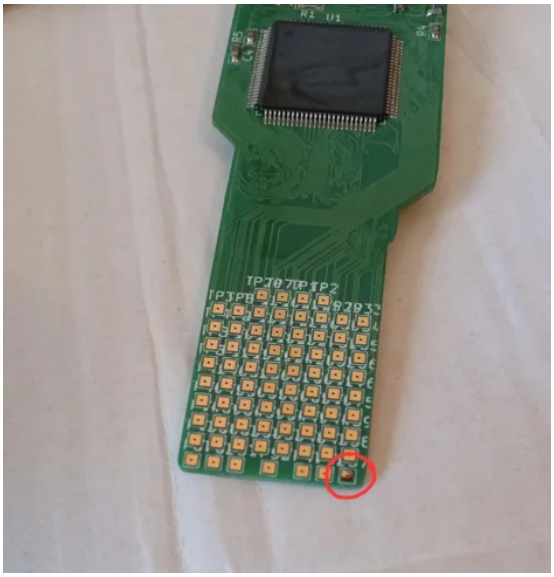
Step 15: Functional Testing (part 1)



Once you've successfully programmed a board, disconnect the programmer. You can finally test the board by sticking it in your mouth, with the electrode grid facing down, touching your tongue. (Yes, prior to shipment I disinfected the boards with isopropanol after doing this test.) You should feel a tingling sensation, which you can increase the intensity of by momentarily pressing switch K1, and decrease the intensity of by momentarily pressing switch K2. Holding either K1 or K2 should turn the device off, and when off, clicking either K1 or K2 should turn the device on. After programming and initial functional testing, desolder the battery.

Two of the ten boards I received failed immediately after this initial testing procedure, no longer turning on.

Step 16: Yield Issues



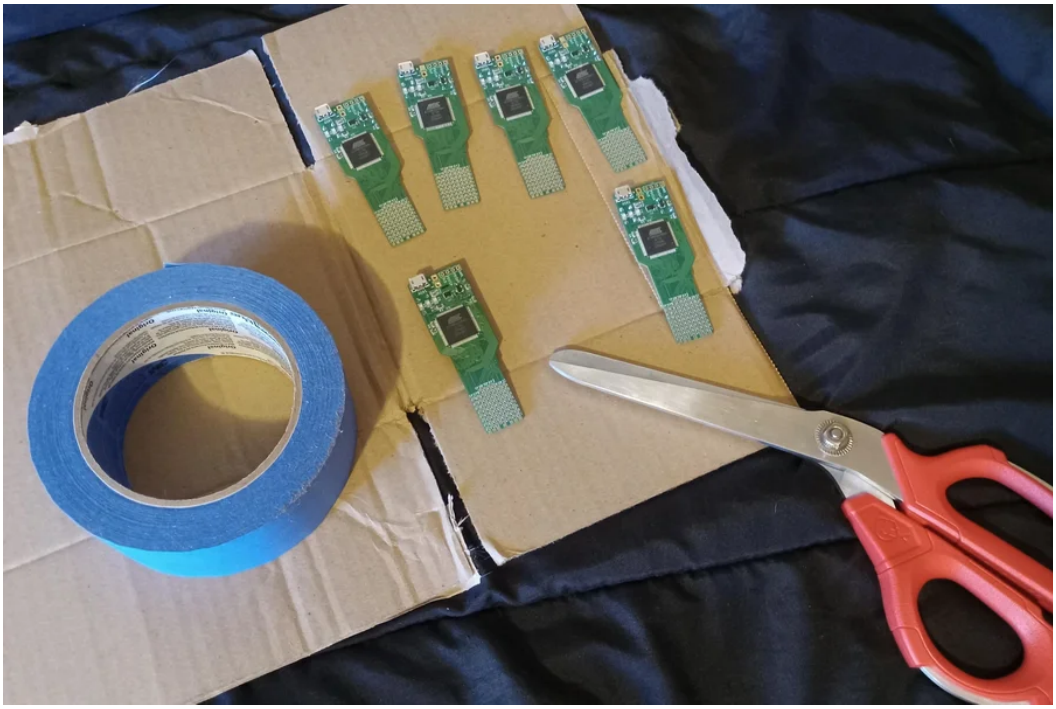
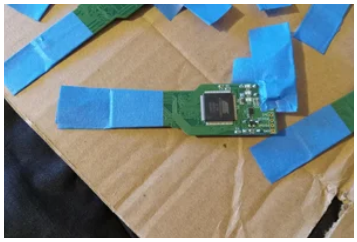
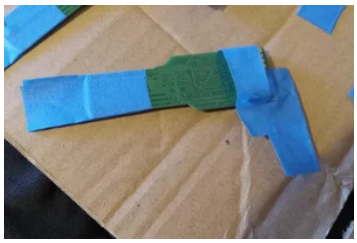
I had some issues with the yield rate of this process. Only 60% of the boards I initially received were fully functional by the final testing step. The failure modes were as follows:

1. 1 board failed during rework to correct CN3065 / CN3166 charge controller issue
2. 1 board refused to accept code at all
3. 2 boards were successfully flashed with code, but then failed immediately during initial testing

One of these losses could be avoided if I were more perfect with the razor blade and tweezers, but three of them appear to be due to defects in fabrication. Note the black pad (circled in red) in the picture above -- etching and gold deposition defects in this particular manufacturing run may be the primary cause of the low production yield.

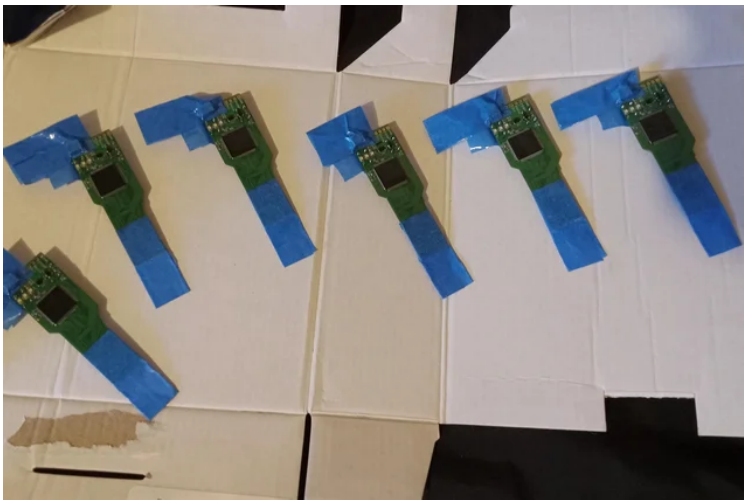
https://hilelectronic.com/what-is-enig-black-pad-and-how-to-avoid/

Step 17: Masking and Coating

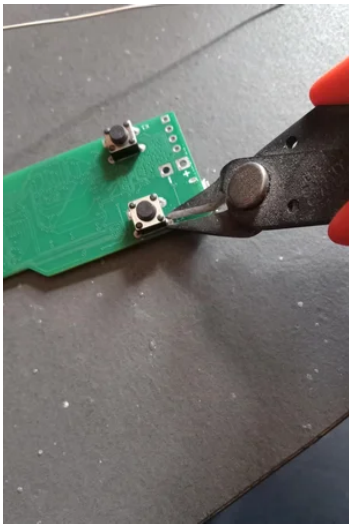
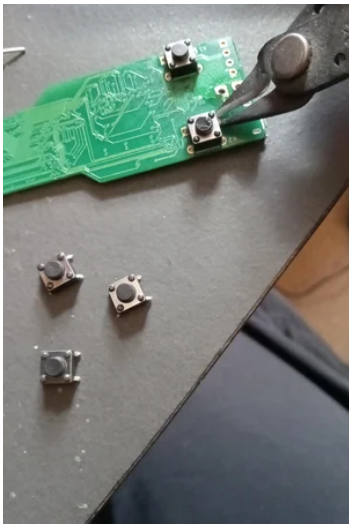
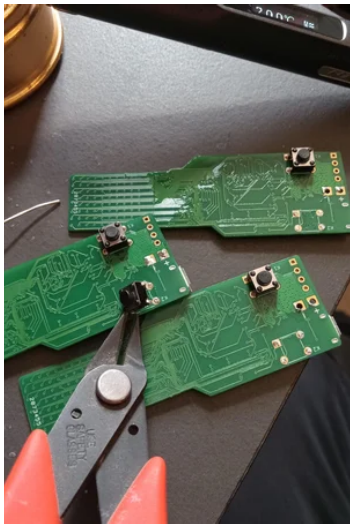
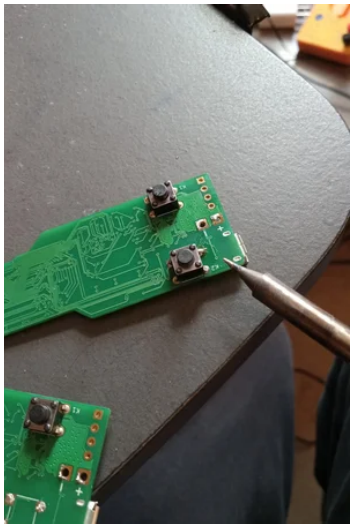

Next, mask the boards with tape, making sure to protect the electrode grid, USB connector, battery connection pads, and switches. In my own masking work, I failed to fully protect the switches; this caused me some later hassle. Apply spar urethane to the side of the board with the ATMEGA128A1 U-TH, ensuring that the board is well-coated. Allow to dry for 10 minutes.

Step 18: Cleaning Up




After allowing the spar urethane to dry for 10 minutes, remove the masking tape. Excess spar urethane can be removed with naphtha (at this point, specifically, as it has not yet fully set). Allow to dry for 24 hours before handling further.

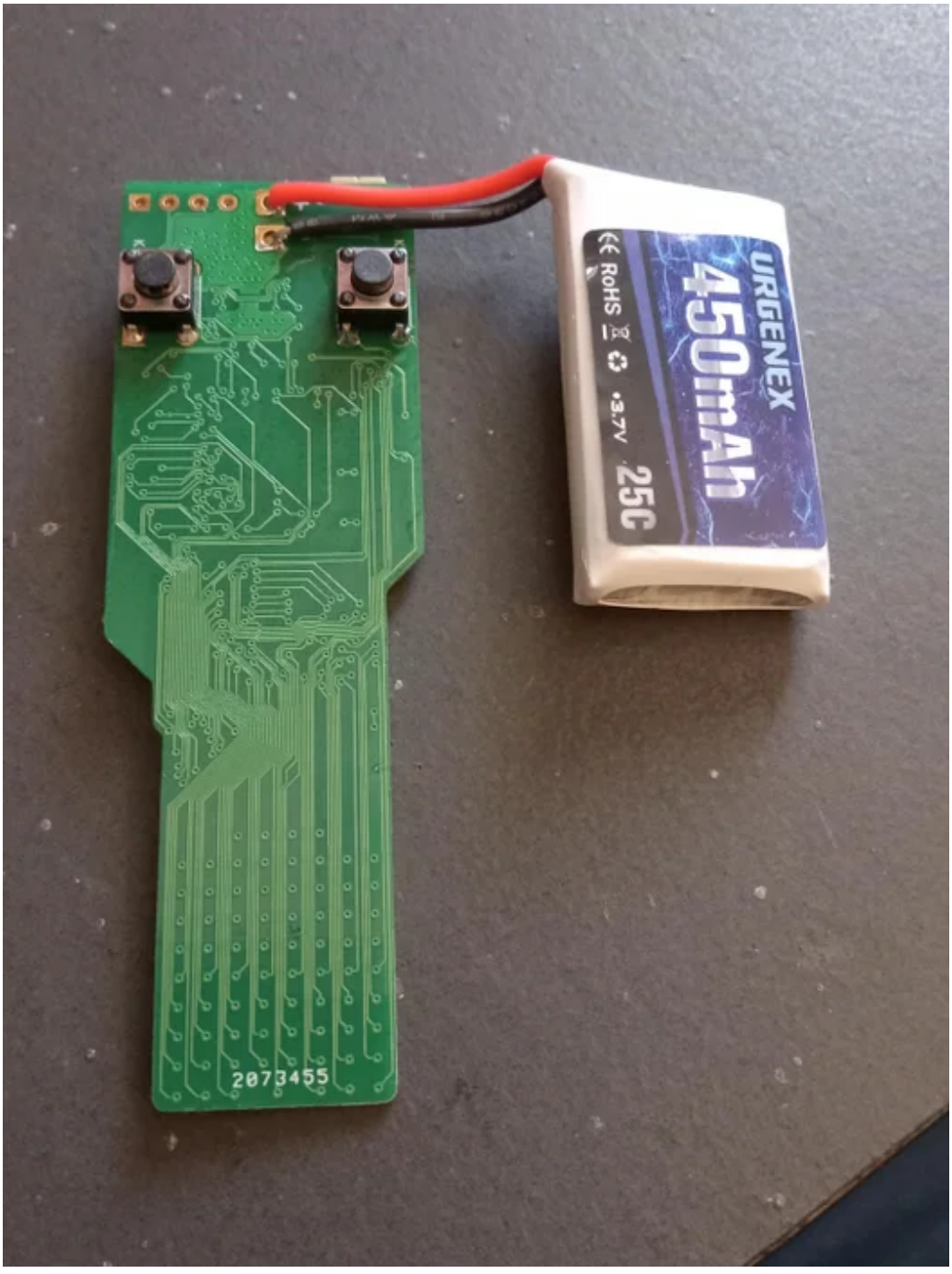
Step 19: More Rework

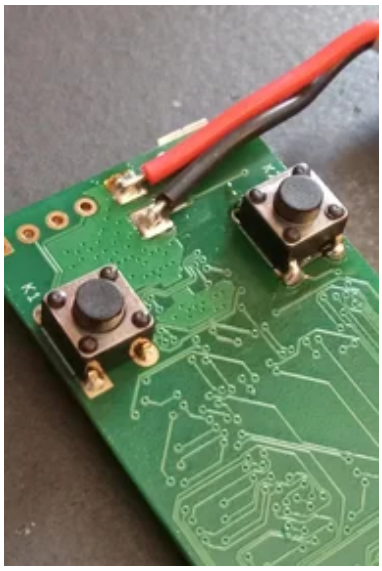





If you, dear reader, are more cautious in your masking of the boards and application of the spar urethane, then this will be another step that you may safely ignore. However, I did not mask off the switches carefully enough, and some of the spar urethane seeped into three of them. I found they were not fully functional after the 24 hour curing time.

To correct this, I cut three working switches off of boards which had previously failed tests, then cut off the three defective switches, and then soldered the replacement switches on in their stead.

Step 20: Attaching Batteries





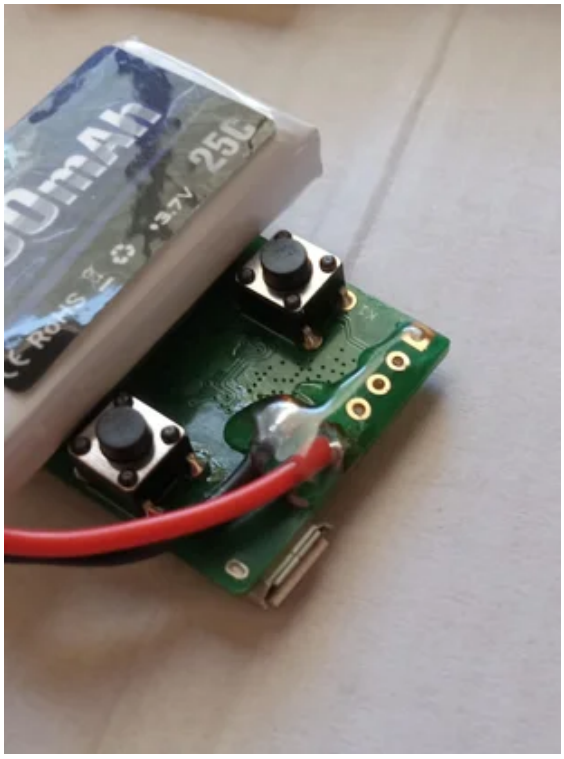
Carefully cut the connectors off of all of the batteries you will be using, *one wire at a time* or they will short-circuit and may catch on fire or explode. Strip about 4mm of insulation off of the end of the battery wires, taking care not to let them touch, or the battery will short-circuit and may catch on fire or explode. Twist the stripped ends of the battery wires to wrap all of the strands in a spiral pattern. Stick the stripped ends of the battery wires into the holes in the power input pads, and bend the wires so that the battery rests next to the board, with the stripped ends of the battery wires still in the power input pads. Note that the red wire will go to the pad labeled "+" and the black wire to the pad labeled "-".

Solder the battery wires into the power input pads.

Cut a piece of mounting tape, about 20mm by 14mm, and stick it to the side of the board opposite the ATMEGA128A1 U-TH, below the switches, as shown. Peel off the protective coating on the exposed side of the mounting tape, and curve the battery wires around (see picture), enabling the battery to be pressed against the mounting tape's exposed adhesive surface.

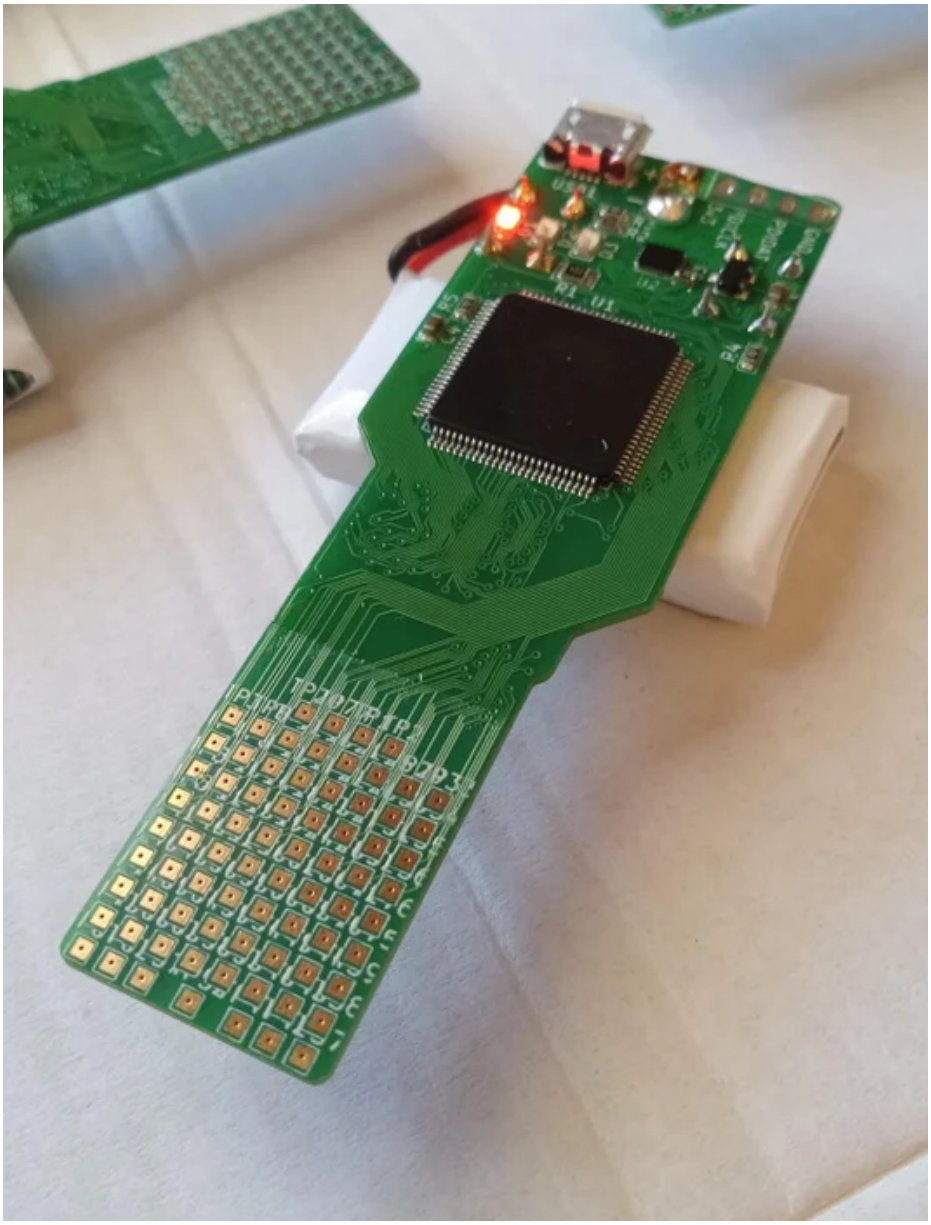
If you have inadvertently filled in the holes in the power input pads with solder in a previous step, you may instead tin the pads and the stripped battery lead wires with solder, then hold them together and apply the soldering iron in order to connect them (see picture). While this results in lower mechanical strength than running the wire through the hole, as the battery is physically attached to the device, it should not matter in this application.

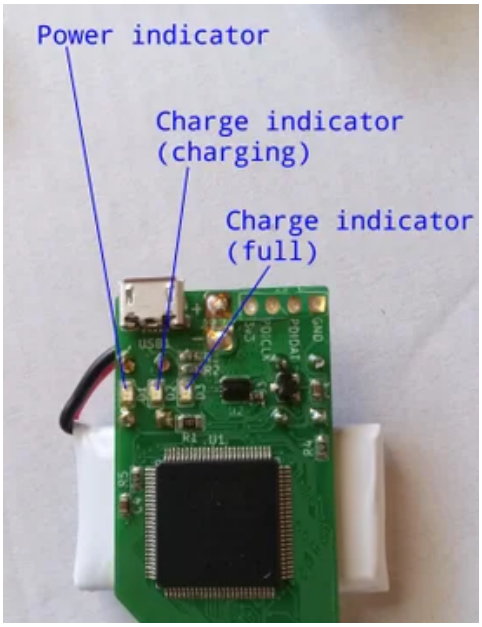
Step 21: A Little Hot Glue Makes Everything Better

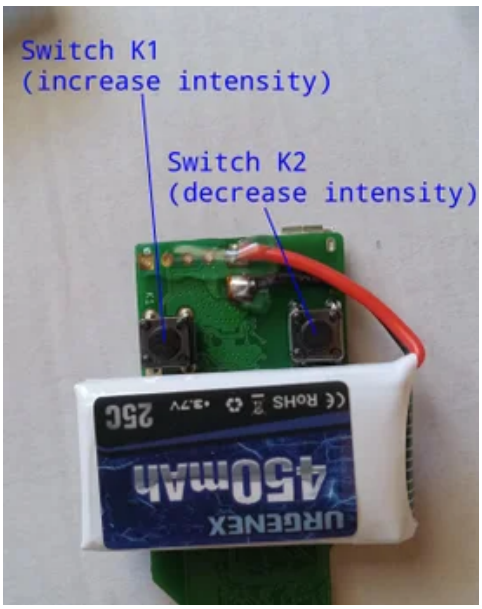


While not strictly necessary, I applied a little bit of hot glue to insulate the programming pins and power connector, and to provide a bit of strain relief for the battery wires.

Step 22: Functional Testing (part 2)



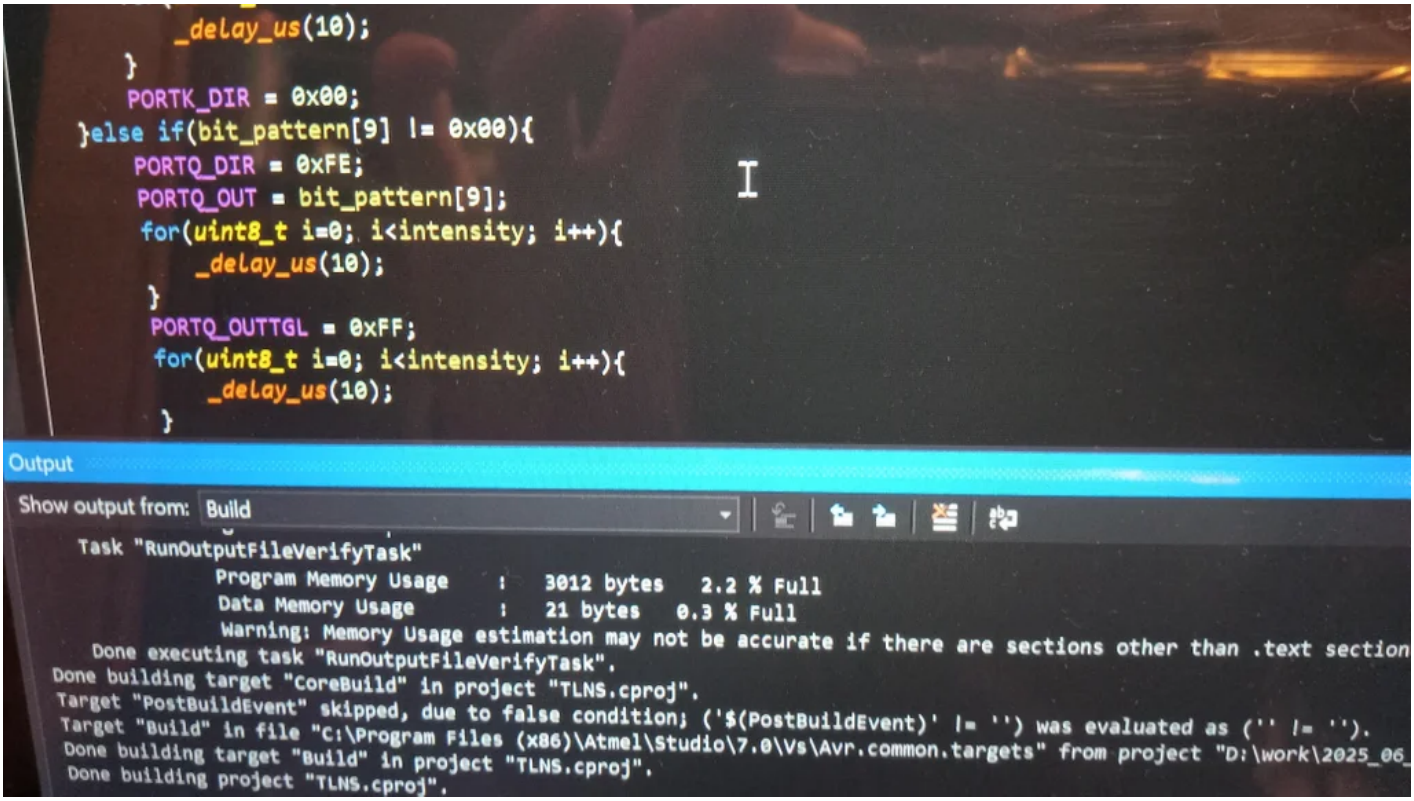




Now that your TLNS devices are complete, it is time to run them through some final tests. First, make sure the charging circuit works. Plugging the device into a 5V USB power supply using a micro-USB cable should cause LED D2 to illuminate, if the battery isn't yet full, and after some time LED D2 should turn off and LED D3 should illuminate instead, indicating that the device is fully charged.

Next, try testing the on/off functionality -- momentarily pressing either switch K1 or switch K2 should cause LED D1 to illuminate, indicating that the device is on. Holding either switch K1 or switch K2 should cause LED D1 to extinguish, indicating that the device is off.

Step 23: How the Code Works



Immersing multiple pieces metal in an aqueous solution of ionic salts and then running current between them causes corrosion. Using alternating current can diminish the electrochemical etching effect, however. This code uses an alternating output pattern to reduce the rate of electrode erosion.

The electrodes are activated in a random pattern, and an array is used to keep track of which electrode is currently active. Each loop of main(), the device randomly chooses an electrode to be activated, and then pulls it to +3.3V, while pulling the other electrodes associated with the same output port down to +0V. The device then waits a little bit (how long exactly depends on the intensity setting), and then toggles the state of all of the pins on the currently-active port, such that the activated electrode is pulled down to +0V and all of the other pins on the same port are pulled up to +3.3V. The device then waits for another intensity-setting-dependent period of time. This ensures symmetrical alternating current flow between the electrodes. All ports other than the one containing the currently-active electrode are kept in a high-impedance state, preventing any current from flowing.

Because the ports are not all the same number of pins, the randomization function has been adjusted in order to ensure that each output electrode has an equal chance of being activated on any given cycle.