
Designing with Kicad

This is not a Tutorial

This is just a collection of ideas and suggestions

- You won't become an expert kicad user by reading this
- Moreover, you may as well disagree with me

The web is full of tutorials: just pick one

- If unhappy pick another

Or you can run the application and read help

- As a general rule, I tend to prefer real documentation
- If documentation accompanies the program, then it is the best one

We may hold a tutorial lesson later if deemed useful

Overview of Kicad

Kicad is a Free Software application to design PCBs

- It covers both schematic design and PCB drawing
- It's getting pretty complete over the years
- Unfortunately, it means it's getting a little more complex too

The main schematic tool is called "eeschema"

- Schematic files, called "*.sch", are ASCII texts
- It relies on a library of components
 - ♦ There is a rich default library of components
 - ♦ More components are found online, with a free license
 - ♦ You can design yours, as usual

The PCB design tool is called "pcbnew"

- The file is called "*.kicad_pcb", and it is an ASCII text
- It relies on a library of footprints
 - ♦ There is a rich default library of footprints
 - ♦ More footprints are found online with a free license
 - ♦ You can design yours, as usual

Data Flow

The main output of eeschema is the netlist

- A schematic is turned into a description of "nets"
- In order to draw a PCB you must import this information

The main output of pcbnew is a set of gerber files

- They are text files that describe each layer of the device
- "Edge cuts" is the perimeter of the device
- "Cu" means "cuprum", i.e. copper. You have front back and inner layers
- The "silk screen" is the set of (usually white) text and drawings
 - ♦ Sometimes you avoid the silk screen on production runs
 - ♦ An alternative, you can place text on the copper layers too
- "Solder mask" is the (usually green) insulating film.

Additionally pcbnew can create "production" files

- They are useful for mechanized mounting,
- The "paste" layer is a laser-cut metal film, to place the soldering paste
- The BOM (bill of material) is a CSV file describing your components
- The position files are used to instruct the pick-and-place tool

Tools and Common Workflow

Eeschema includes a "library manager" tool to manage components.

- If also offers a graphical editor to make your own components
- Additionally, each component must be associated with a footprint
 - ♦ This is done while placing components
- The "cvpcb" tool is an alternative, separate tool, now discouraged

Pcbnew, similarly, has a "footprint manager". And an editor.

So the standard workflow looks like this:

- eeschema: you draw the schematic diagram, assign footprints and save
- pcbnew: it reads the netlist and footprints, you route the PCB and plot gerber files
- gerbv[iew]: you check the output (gerbview is in kicad, gerbv is from another EDA suite)
- email/web: you ship the files and wait for the PCB to be delivered

Finally, mounting

- You can buy components and turn on the soldering iron
- Or, at extra cost, you can buy, pack and re-ship for mounting (with BOM and POS)
- Or, at even more cost, your manufacturer does everything for you
 - ♦ You must be most precise with your BOM file

The design workflow for the lone developer

For my own projects, I prefer a different workflow

- **1- Lay out the schematics with the most important components**
- **2- Draft mechanics and place core components**
- **3- Add more items to the schematic diagram**
- **4- Adjust mechanics if needed**
- **5- Place and route the new items**
- **6- Goto 3**
- **7- Add anything optional that may be useful and fits mechanics**
- **8- Place and route these extra things**
- **9- Goto 7**
- **10- Remove unused "options" before the mounting step**
 - ♦ **You really don't want solder alloy on your free pads**

Everything must be performed with later steps in mind

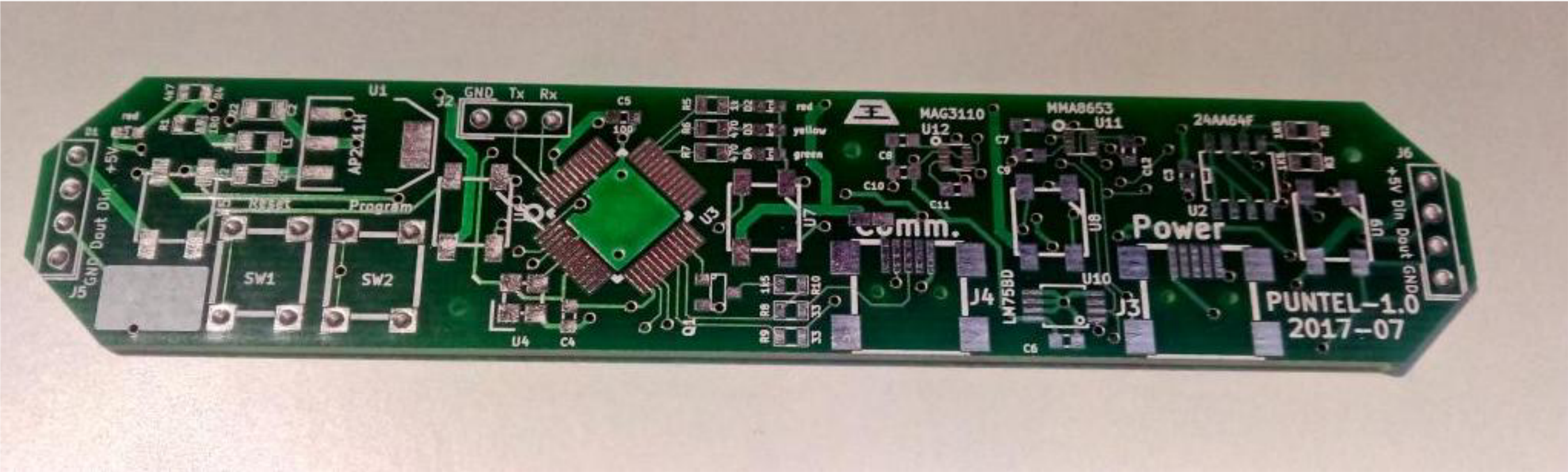
Other ideas worth considering

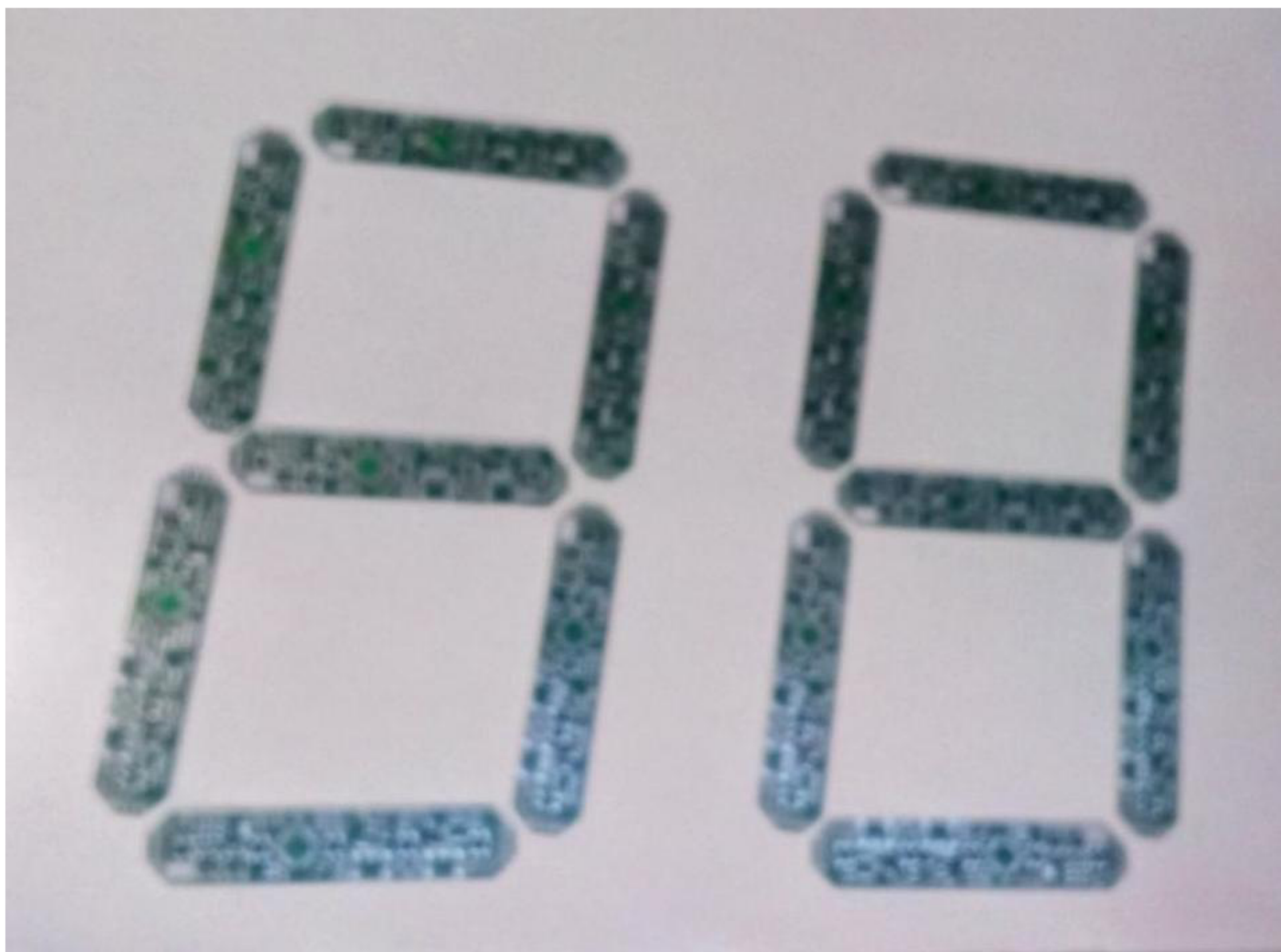
Be friendly to your software mate (who might be yourself :)

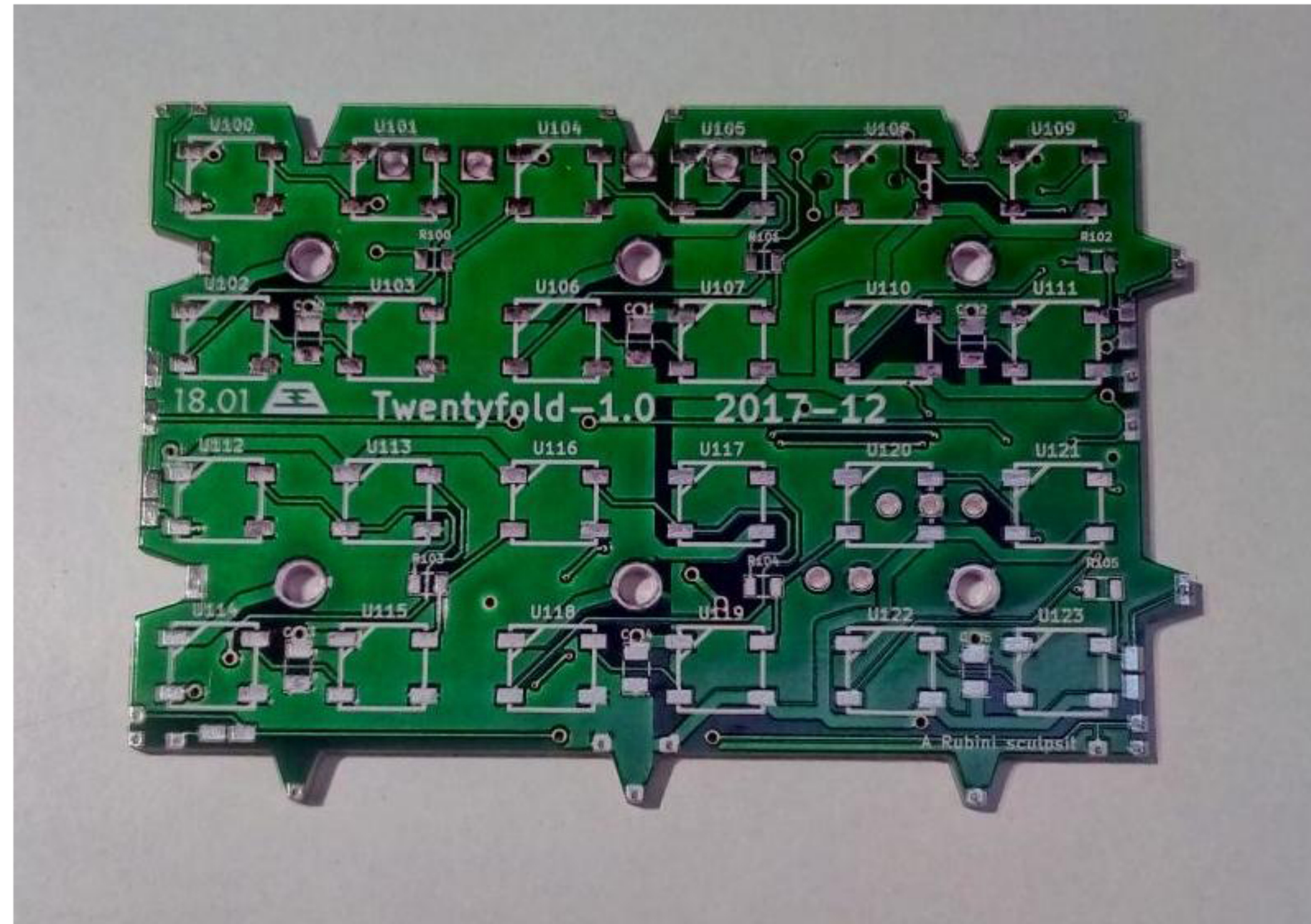
- **Add I/O channels if possible**
- **Provide means for logging**
- **Add test-points whenever possible**
- **Remember debugging (jtag and/or something creative)**
- **If it fits, add new components to test**
 - ♦ **you might save the cost of an evaluation board later on**

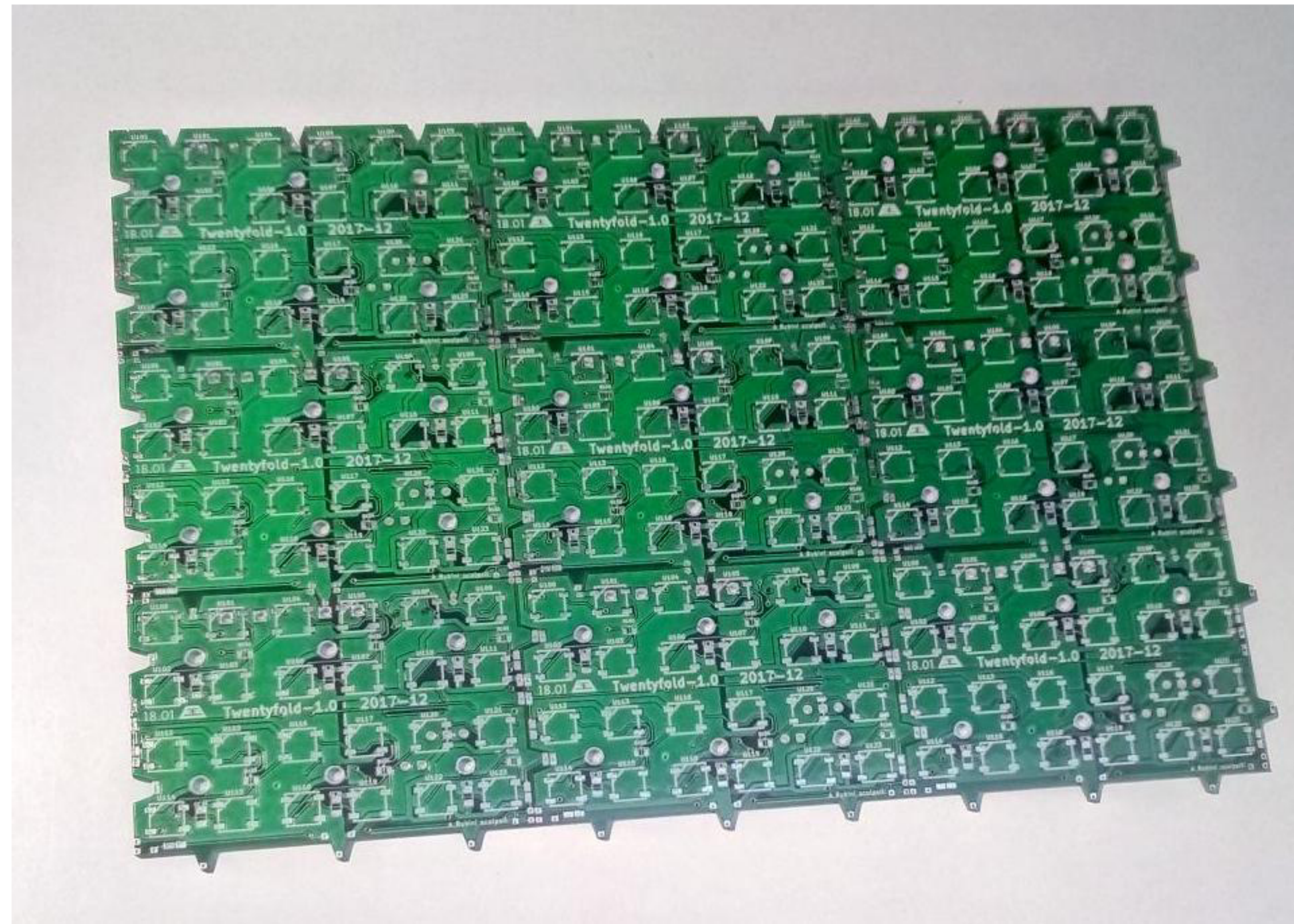
Surprise your client, by guessing their next moves

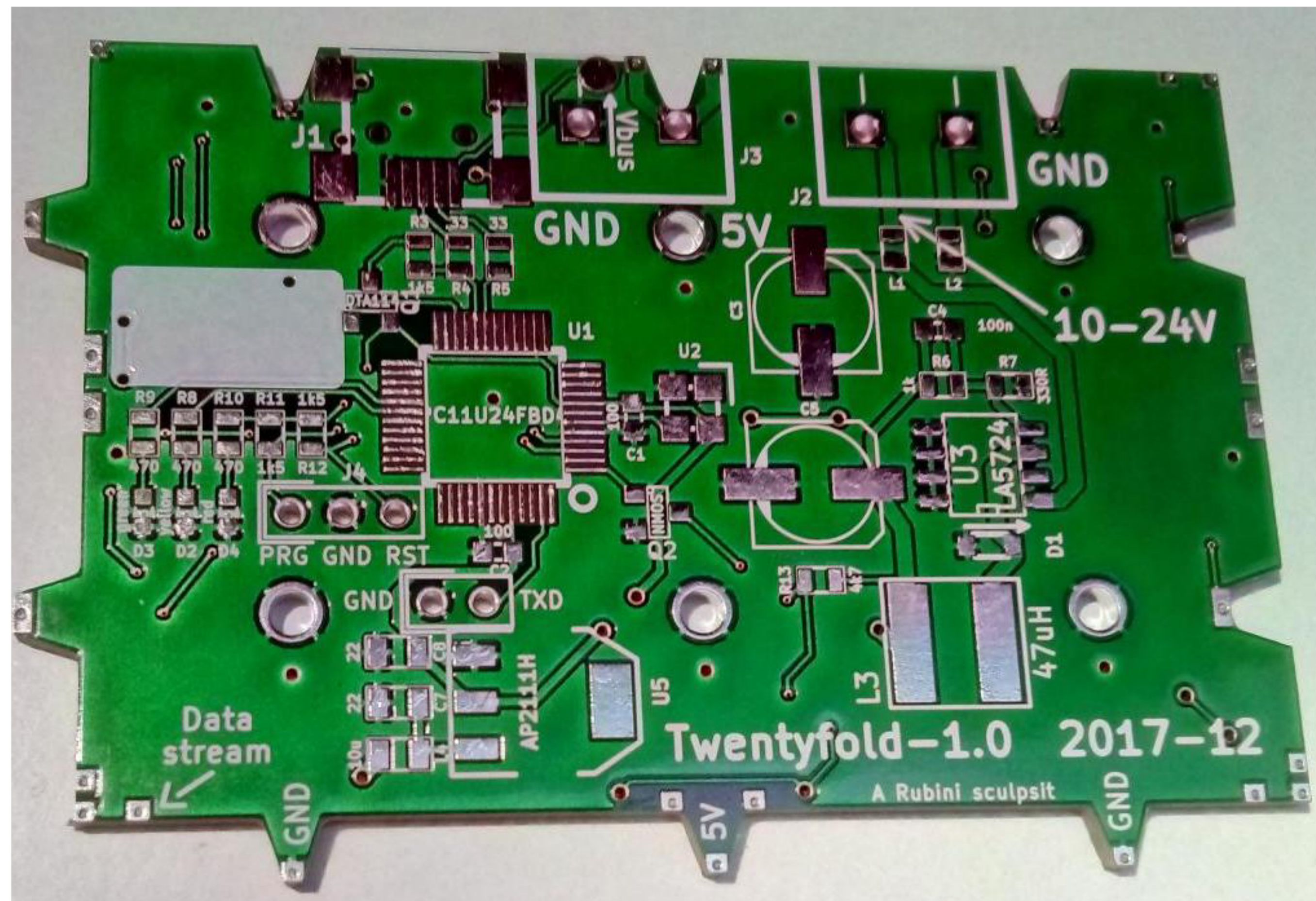
- **They didn't ask for storage, but you know better**
- **Won't they need a thermometer sooner or later?**
- **Doesn't the application benefit from a light sensor?**
- **And why not, always offer a UEXT expansion**

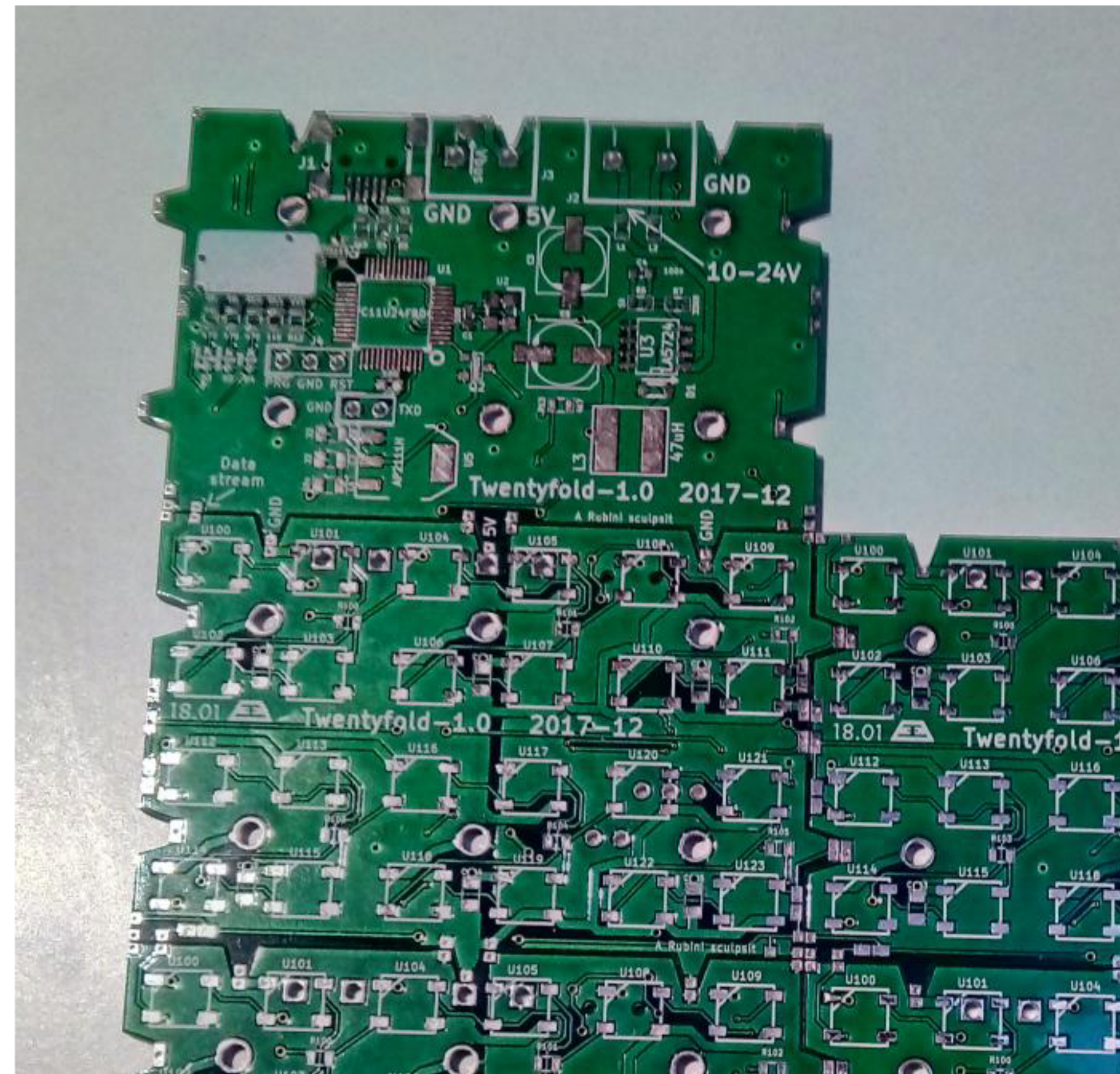


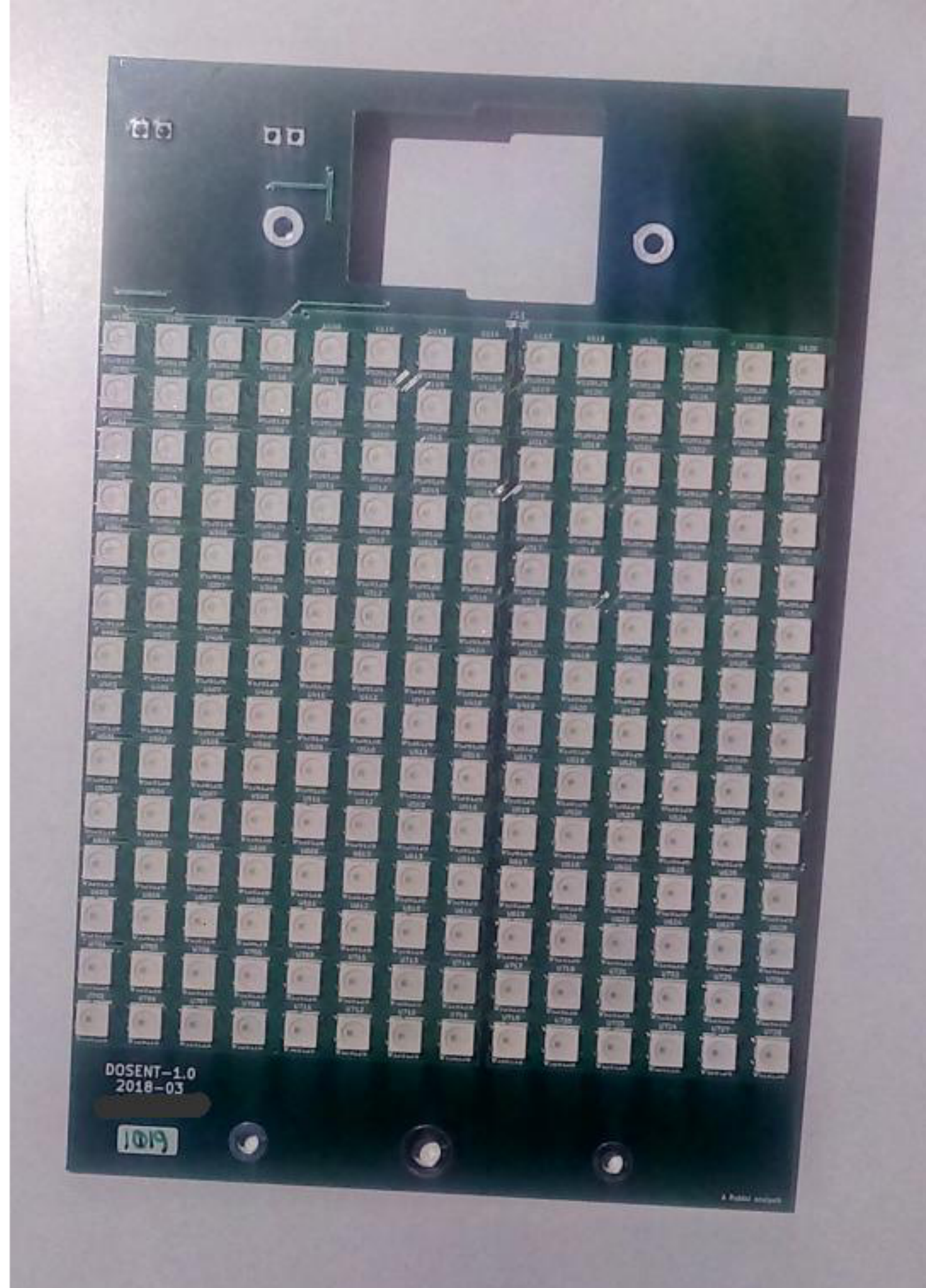


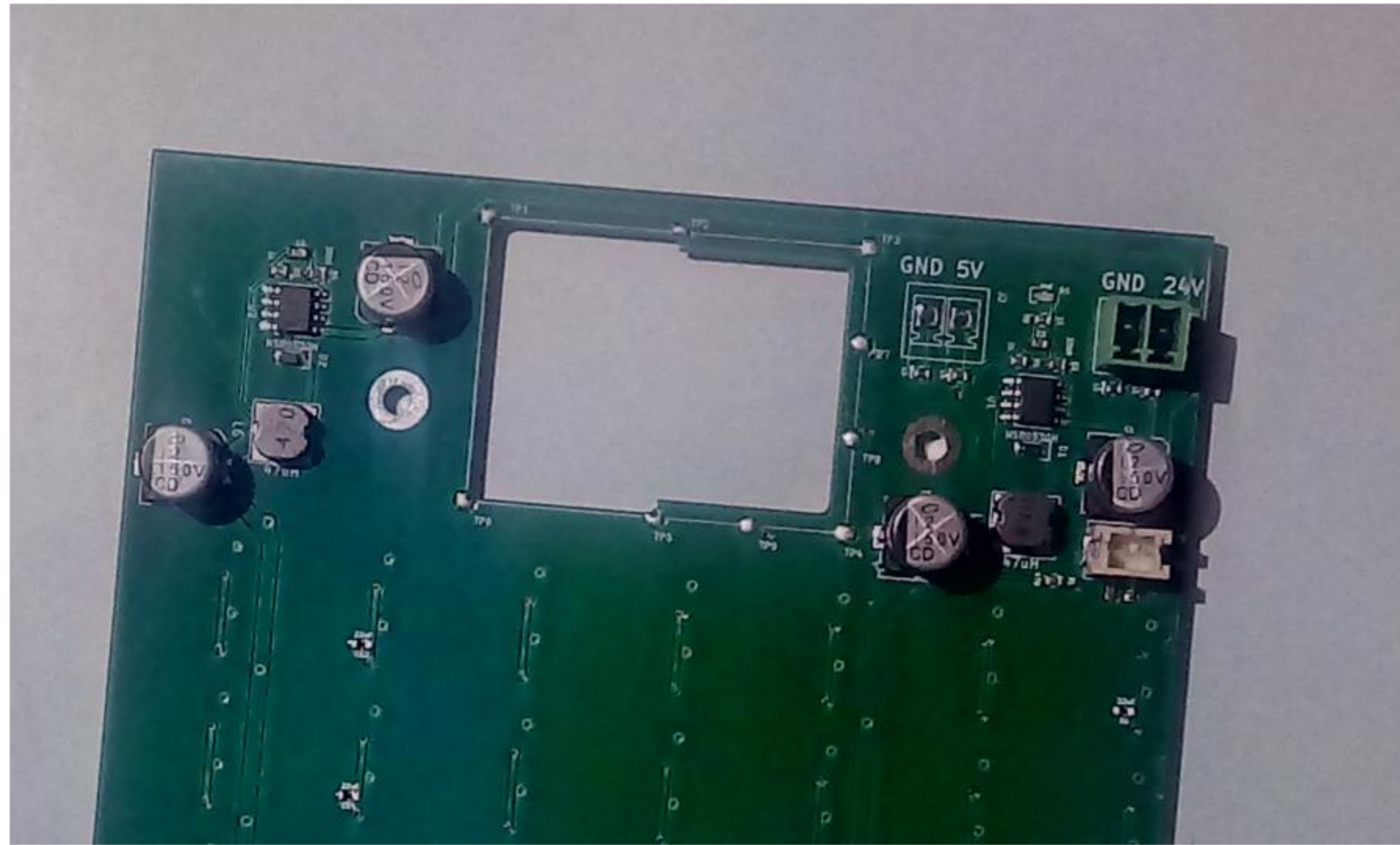


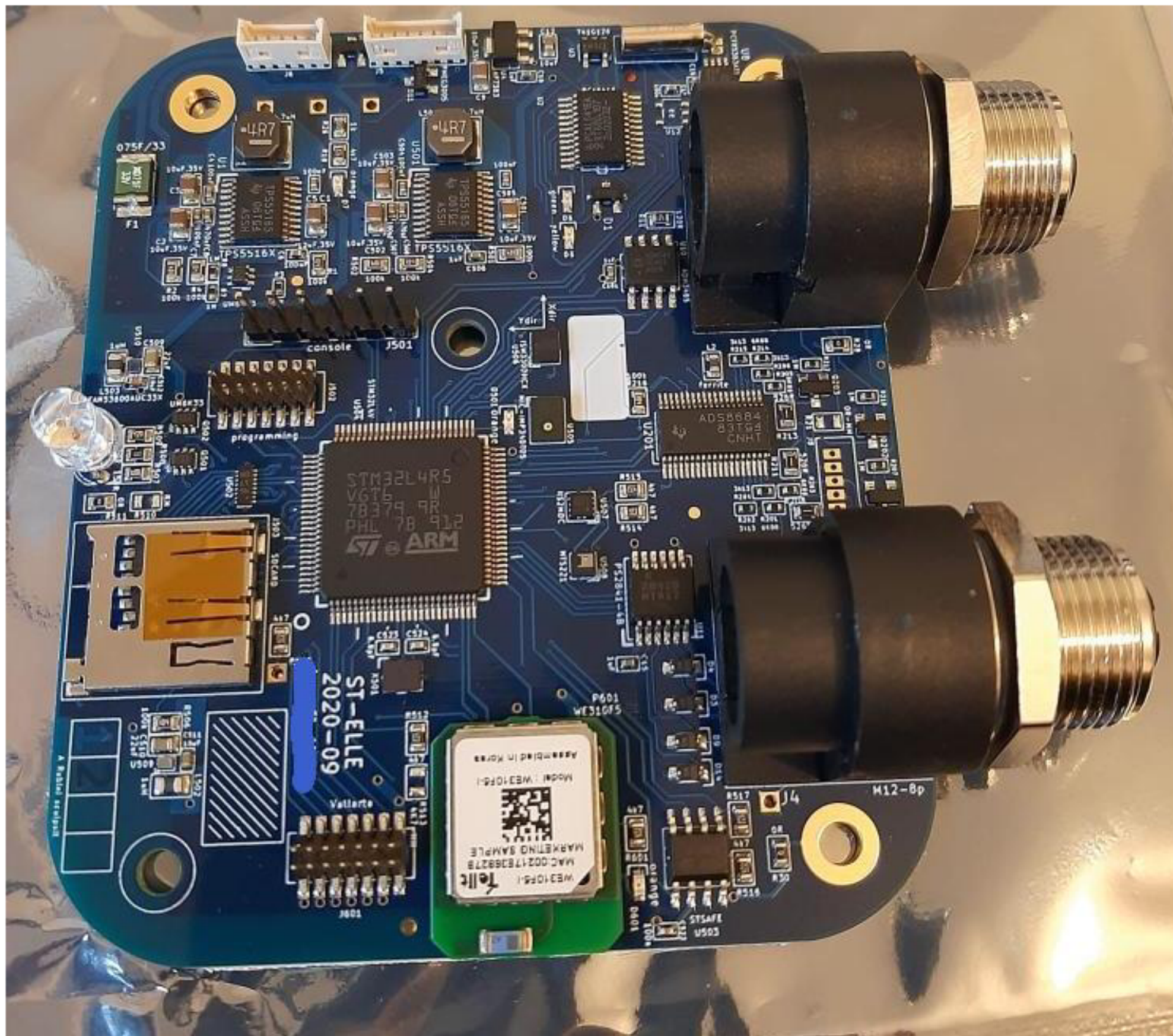












But, also, there is the real world

Consider power consumption

- **Your power supply must provide enough current**
- **Your tracks, connectors, external cables must be big enough**
- **Are you battery-driven? Hmm... interesting....**

Consider heat

- **Your components get hot and must dissipate**
- **Your power supply gets hot and must dissipate**

Consider mechanics

- **Coordinate with the mechanical engineer**
- **Try to ease maintenance, and component replacement**

Consider noise and interference

- **Switching or linear supplies?**
- **Beware of capacitive coupling between signals**
- **Beware of high impedance critical signals**
- **With high currents, consider the ground return path**

Suggestions from a frequent user (1/2)

What follows may look trivial, but these items are usually forgotten

In schematic drawing

- **Make drawings small, or you'll use up too many pages (hard to manage)**
- **Default resistor, inductor, etc may be too big: make yours**
- **Use smaller-than-default text: 0.04 or 0.03 inches is best (40 or 30 mils)**
- **Use labels rather than wires, so you can change pin assignment while routing**
- **Add diagnostic leds for your microcontroller -- you can always not mount them**
- **Use spare pins for board identification**
- **Always include a power-on led**
- **If it fits, add self-test circuitry**
- **Be generous with 0R resistors for options and tests**
- **Be generous with 0R1 or 1R0 resistors as amperometers**
- **If they fit, provide multiple alternative components**

Suggestions from a frequent user (2/2)

In PCB drawing

- **Place the pin1 marker outside the component**
- **Write component values outside the component**
- **Place the grid origin in a meaningful place (and mark it!)**
- **Always place pin-strip devices on a 2.54mm grid**
- **Place mounting holes at an even distance (e.g, 5mm grid)**
- **Add extra "mechanical" copper for all connectors**
- **Make pads bigger than suggested, to help DIY soldering**
- **Use square-angle routing between different layers**
- **Add fiducials, just in case -- your mounting company will thank you**
- **Bless the board with a unique name, funny if possible**
- **Provide a "sticker" area for the serial number**
- **Proudly sign your device**

My personal suggested sizes

- **References and values: 0.5mm (0.1 tick). Bigger text is a waste of pcb space**
- **Tracks: 0.2mm, 0.4mm, 0.8mm, 1.6mm**
- **Clearance: 0.018mm**
- **Grid: 1mm (or 0.5mm) for components, 0.25mm for tracks**
- **Vias: 0.8/0.6 or 0.5/0.3; bigger for power. 0.4/0.2 may be an issue in manufacturing**