Using versatile power tools for testing embedded system efficiently

Lars Sjödahl
@LarsSjodahl

My background

- R&D since 2001
- From Sweden (live in Sweden)
- M Sc in Engineering Physics
- Slipped into testing

My early (narrow) view of testing

- Waterfall, possibly many small rapids.
- Code is tested against spec. Pass/Fail...
- Digits of precision, some is necessary so more is better, and spending time on it is good.
- If it can be done in SW, it will run faster and be more efficient.

Problems?

- Observability:
  - Black boxes
  - Heisenbugs
  - Sleeping test code

Problems?

- Affecting the system from code:
  - Chains of “information transduction”
  - Messy non linear world. Batteries, sensors.

Unfortunate solution

- Do manual tests with some variations.
  - Extremely tedious
  - Limited perception and attention.
  - Time-consuming, requires 1 person.

Unfortunate solution

- Do manual tests with some variations.
  - Extremely tedious
  - Limited perception and attention.
  - Time-consuming, requires 1 person.
  - “Arbitrary” (human), is not random
- Forced to rationalize why skipping some tests is acceptable...

My wish-list

- To affect sensors, switches, and environment around Device-Under-Test, FROM CODE.
- To do identical repetitions for statistics
- ...or vary execution variables as desired.
- Watchdogs to watch DUT for me.
- Exact timing of input and output.

“Pushback” from management

- “We don't have the budget for it. Turnkey systems are expensive, and building from scratch takes too much time.”

“Pushback” from management #2

- “We'll only run this test for a short time, so it's not worth putting too much time into something specialized that we'll never use again.”

“Pushback” from management #3

- “Automation won't test all the parts, like some manual tests do, so it's no replacement.”

My solution

- It's called the “Arduino”
- Development board
  - based on micro controller with >10 GPIO-pins
  - it's “open hardware” licensed: legal “copies”
  - comes with free IDE
  - tons of built in examples to get started.

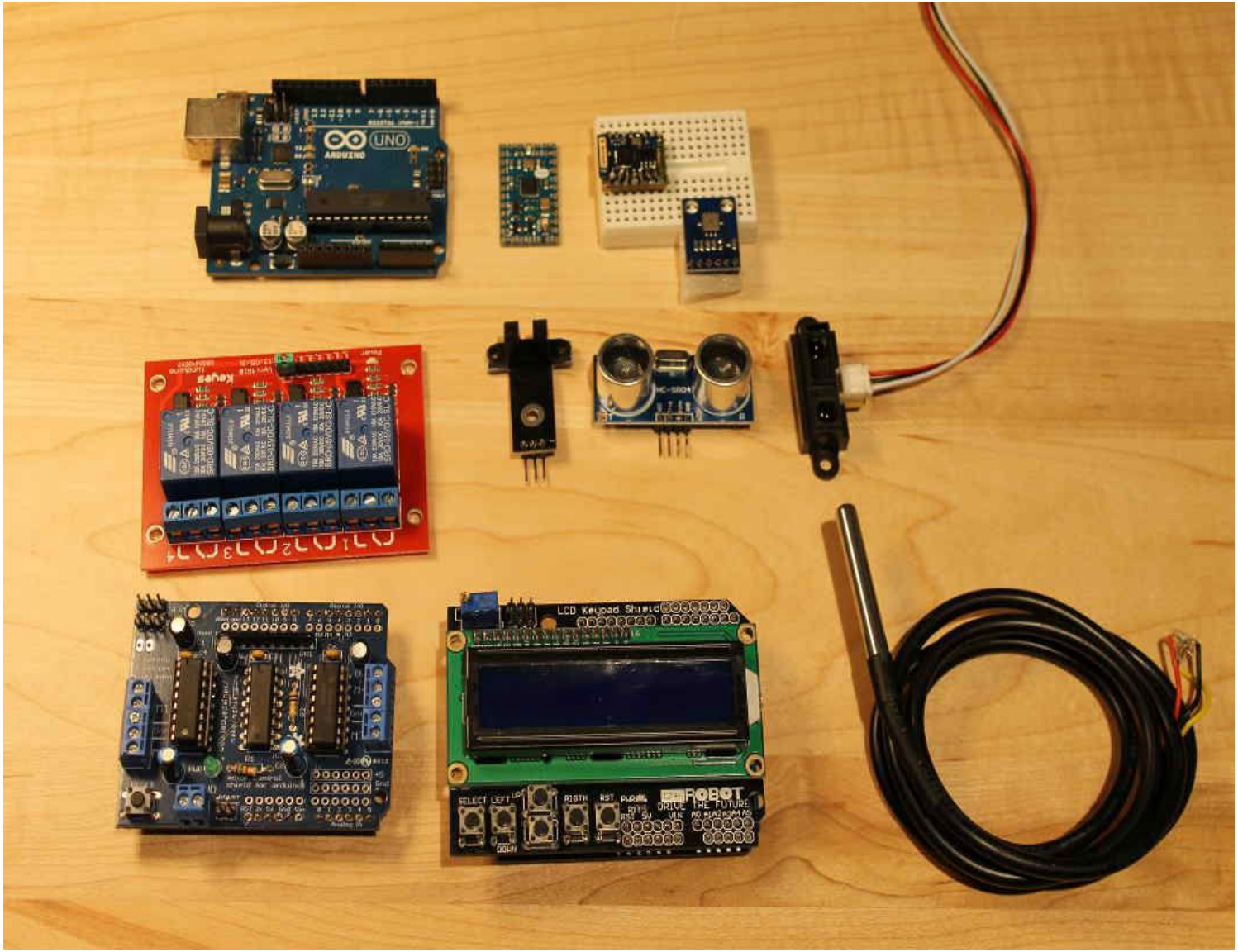
Objections from testers?

- “Oh no, not another *%&! tool to learn. No way!”
- “My boss don't want us to spend even that much.”

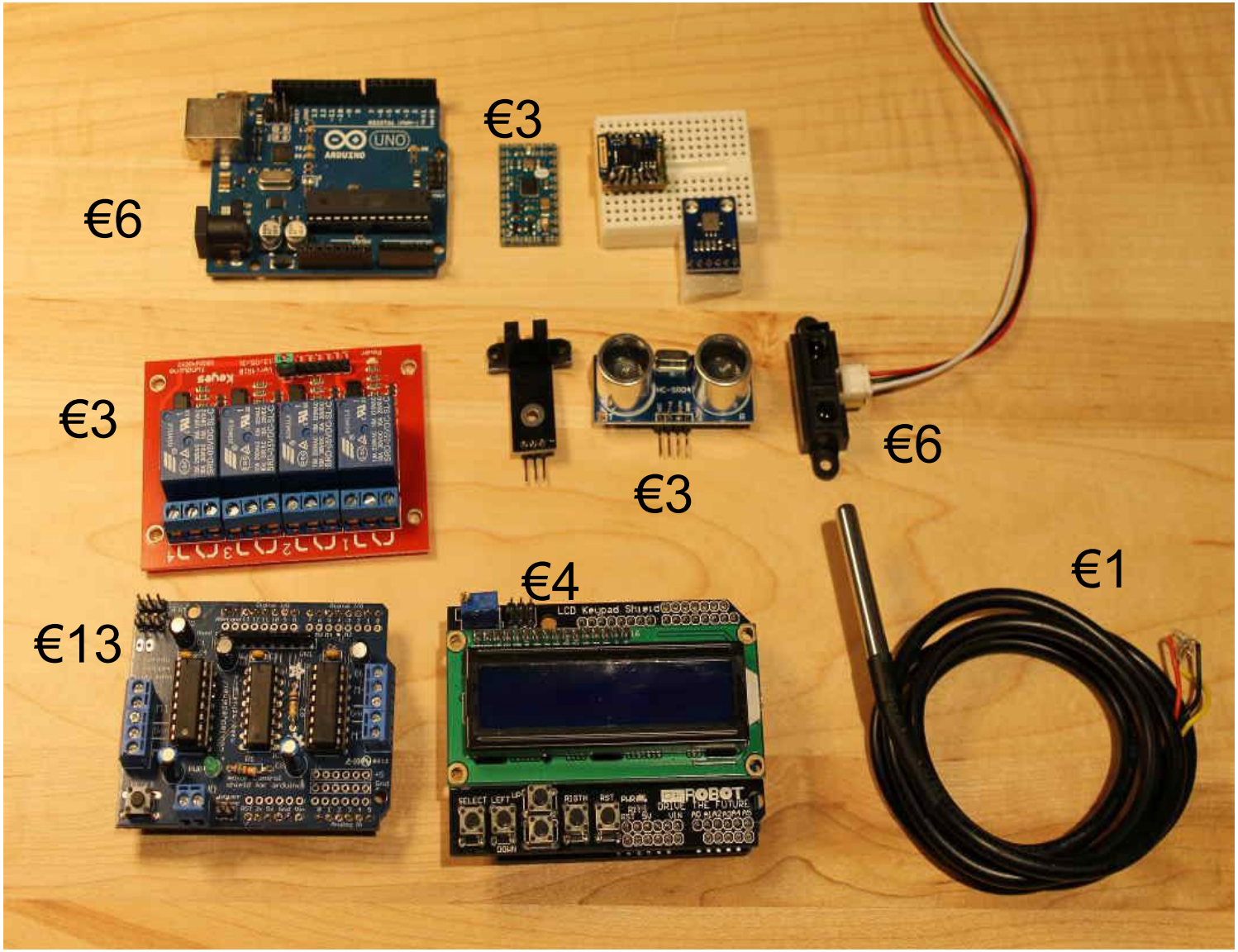
This is the complementary tool you're looking for...

- Capable
- Cheap
- Easy to re-configure
- Huge community online, many libs that are easy to use.

Part of my tool-box

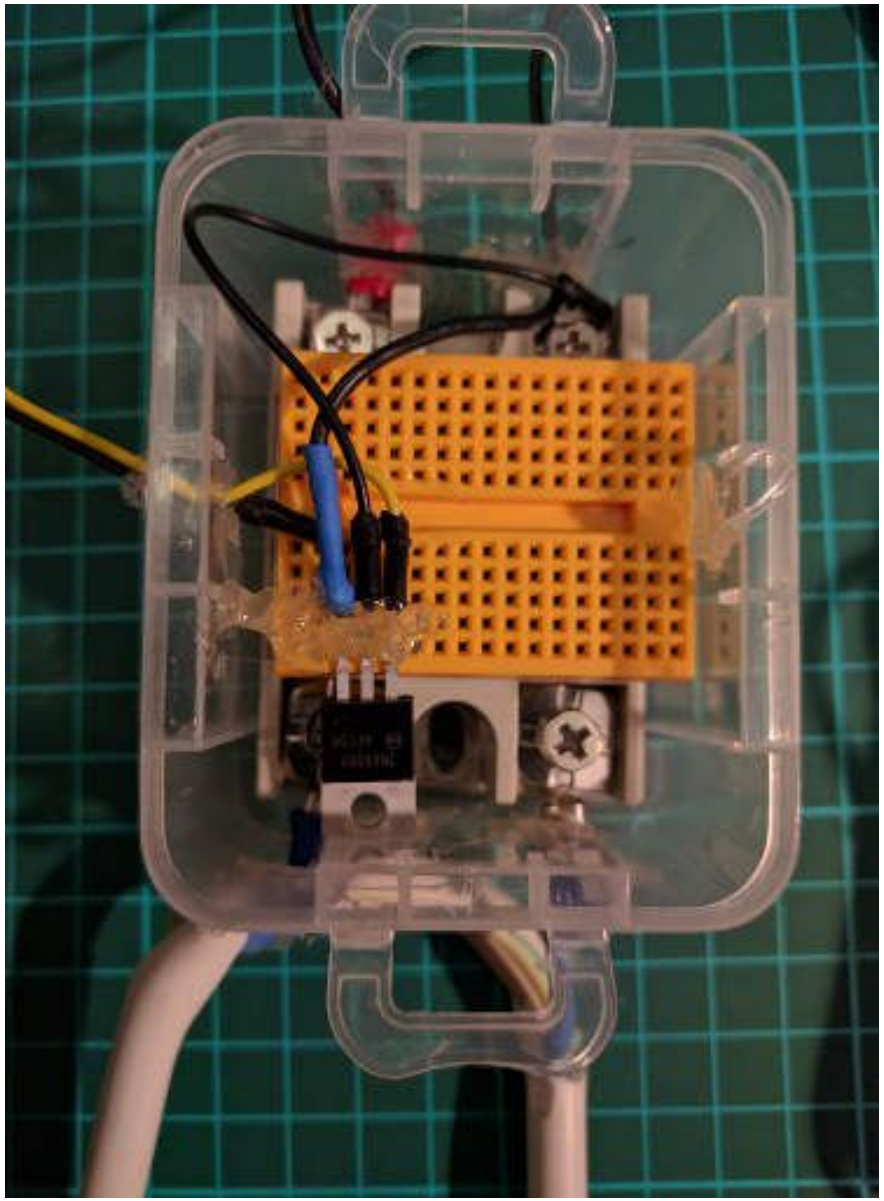


Part of my tool-box

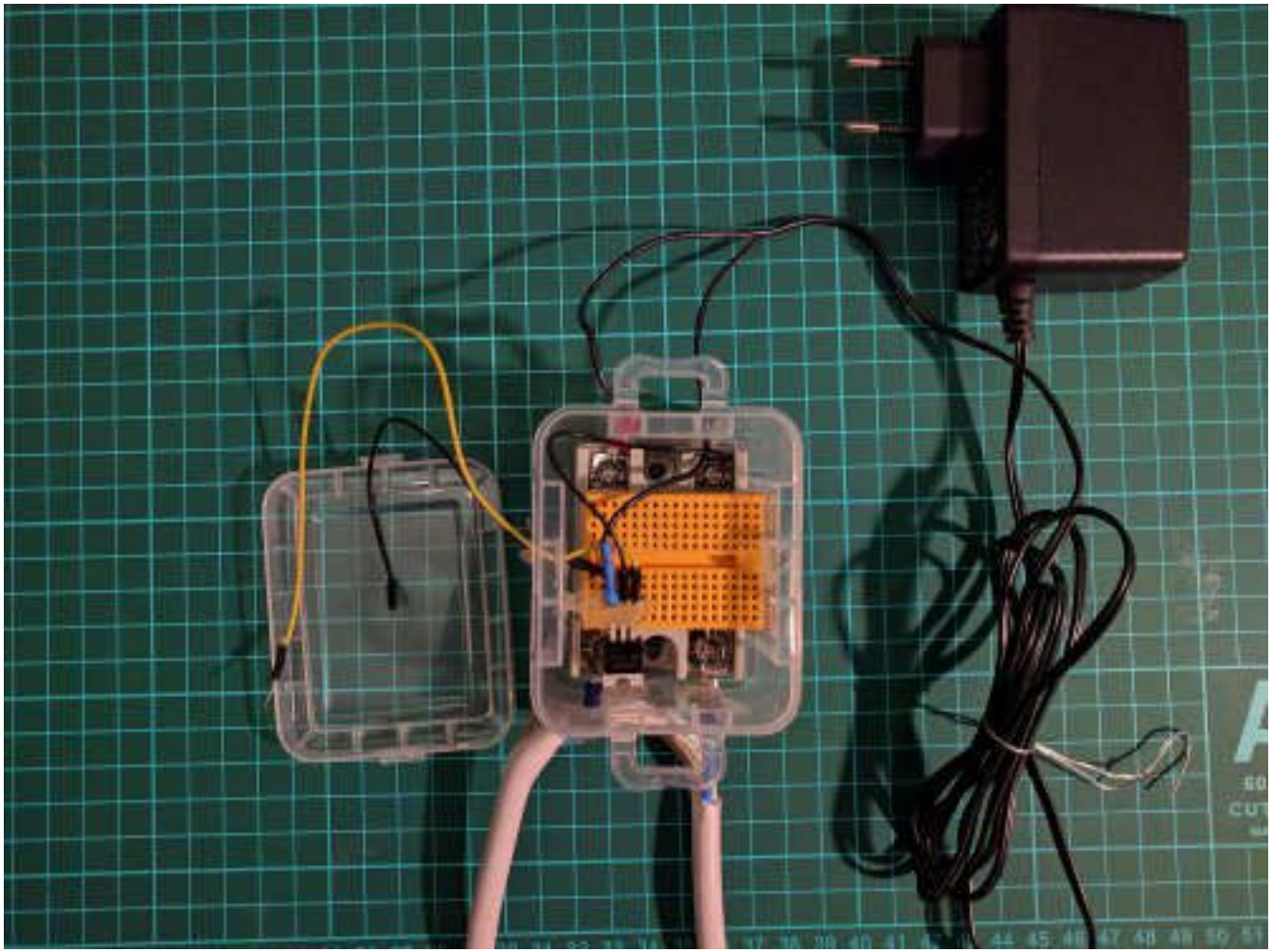


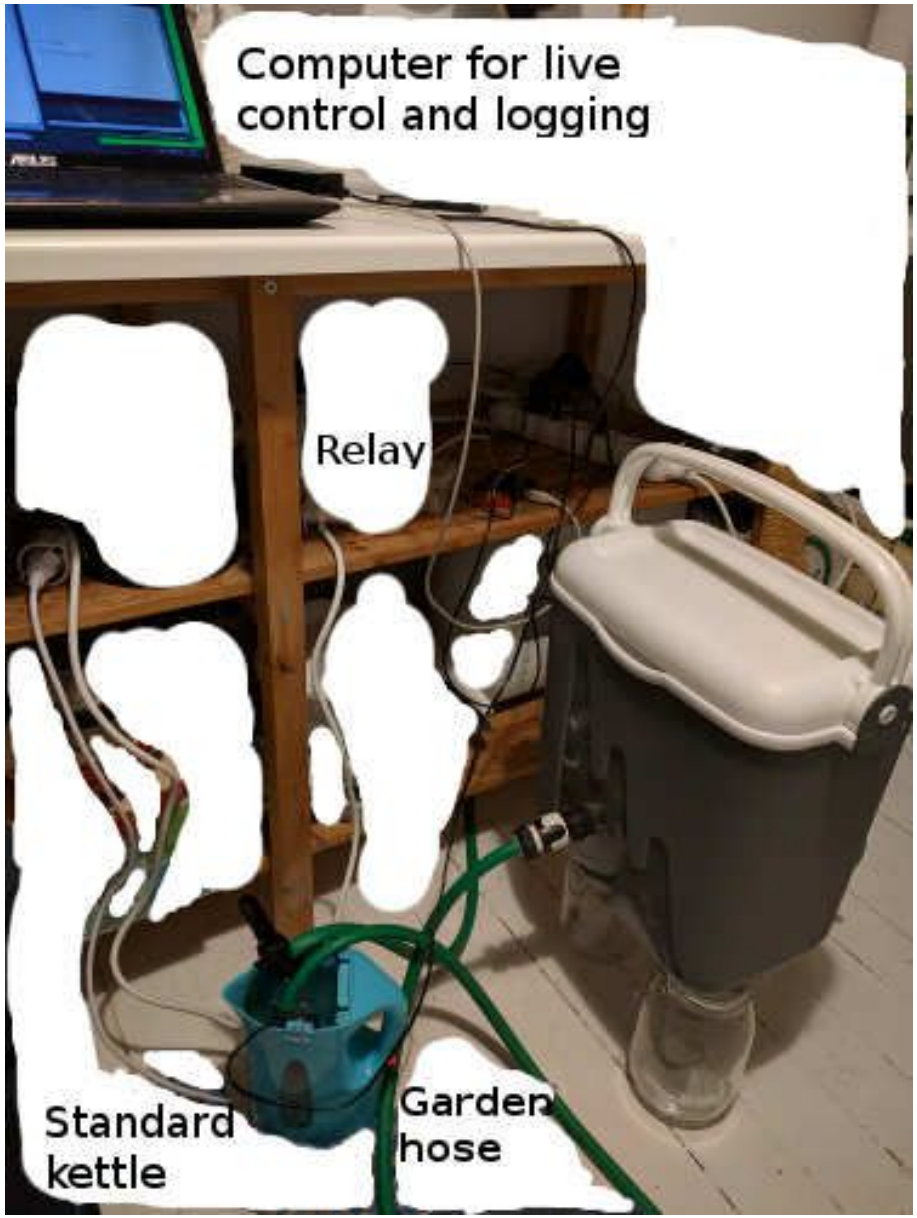
Putting the “power” in power tools

We don't need no stinkin' PCB etching



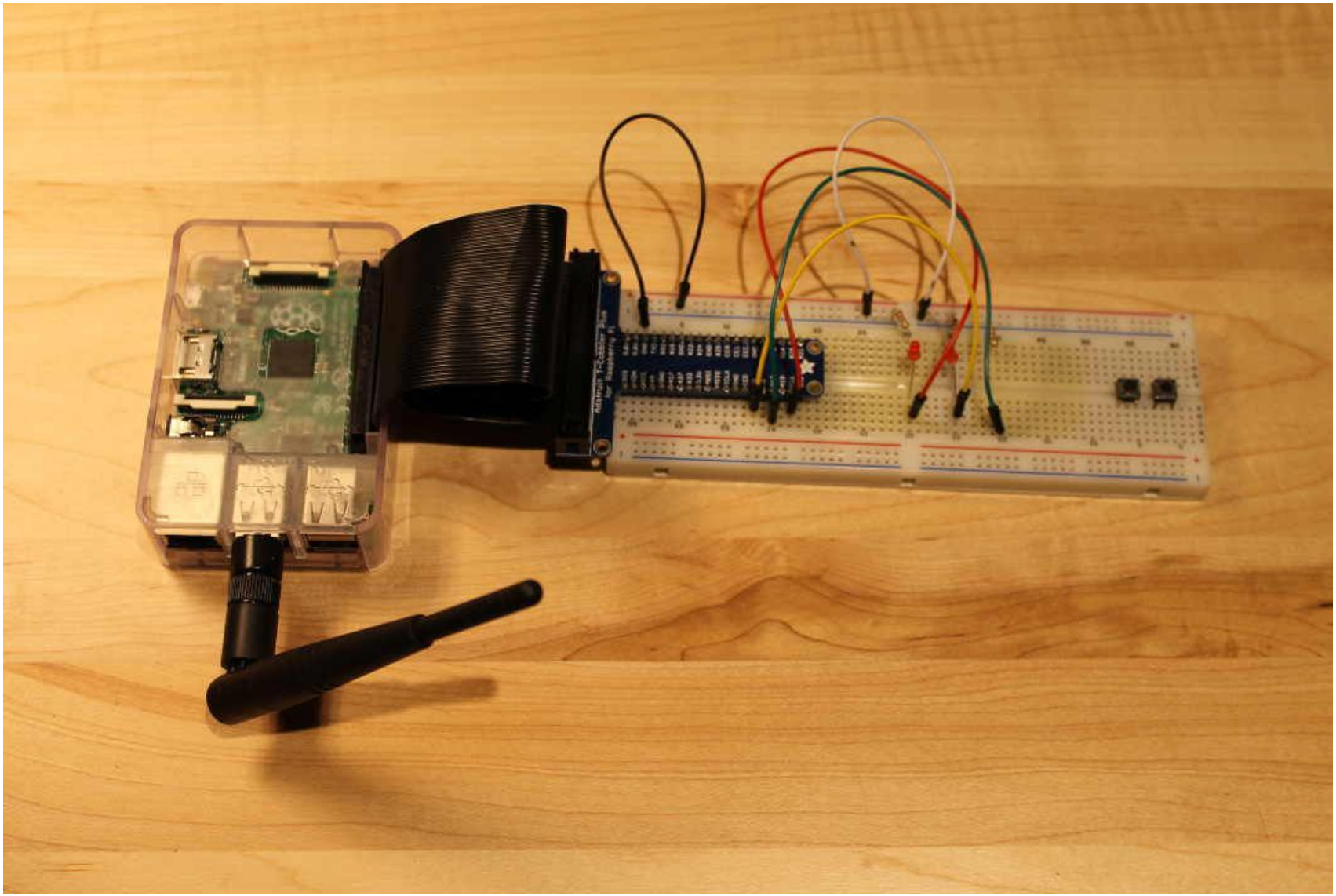
Package, but don't over-do it.



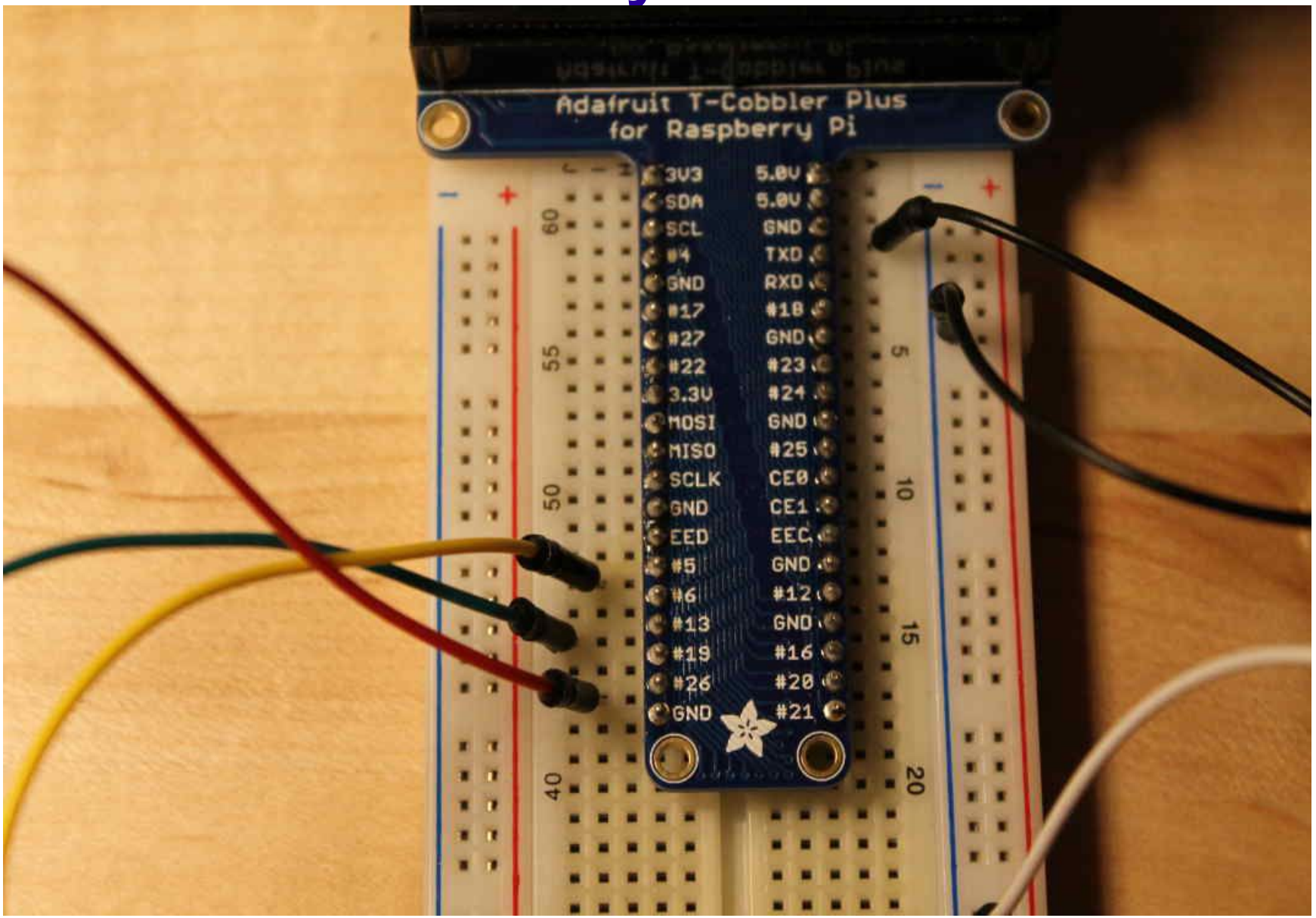


Poor tester's temp chamber

Raspberry Pi



Part of my tool-box



What I hope you take with you:

- If you keep “having to” do manual steps with little thoughts to it, there might be an easy way to automate much of that.
- Add alarms for events or value limits to supervise set-ups for you.
- To get around interference between functionality and test code, consider putting the test code in an external hardware.
- It's easy and cheap to start “trying something out”

Thank you for your time!

Lars.Sjodahl@HouseOfTest.se

@LarsSjodahl