



Lecture 11
March 11, 2025

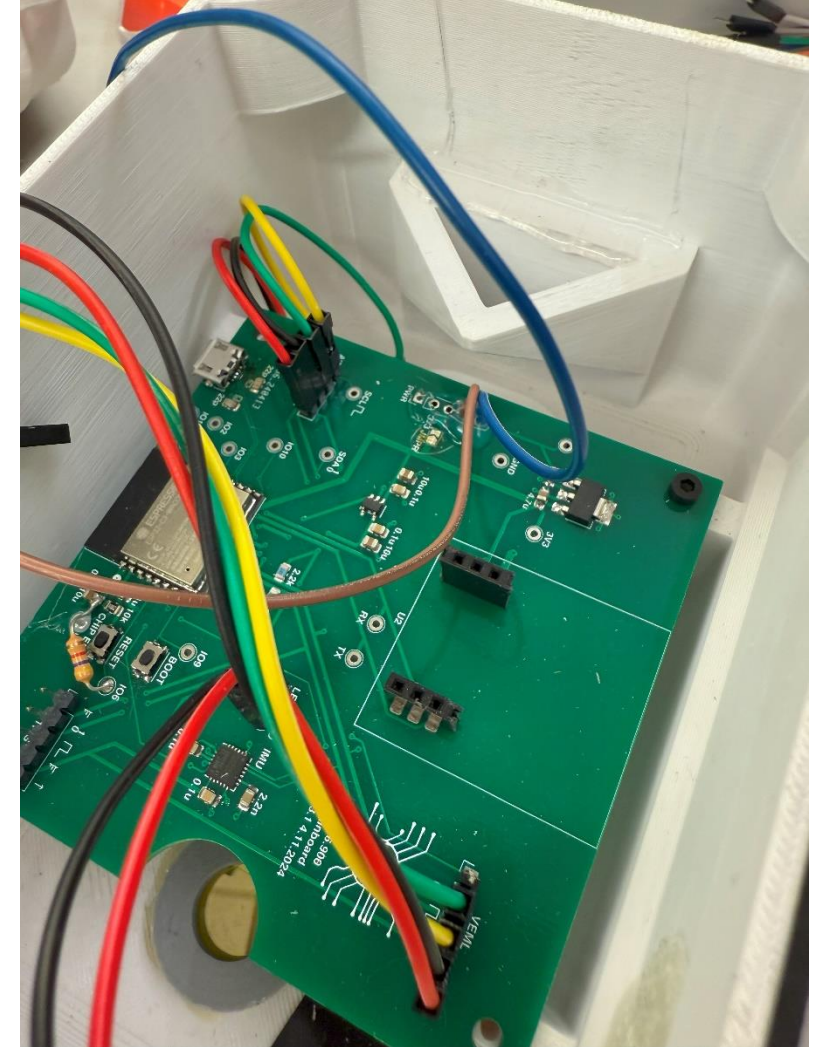
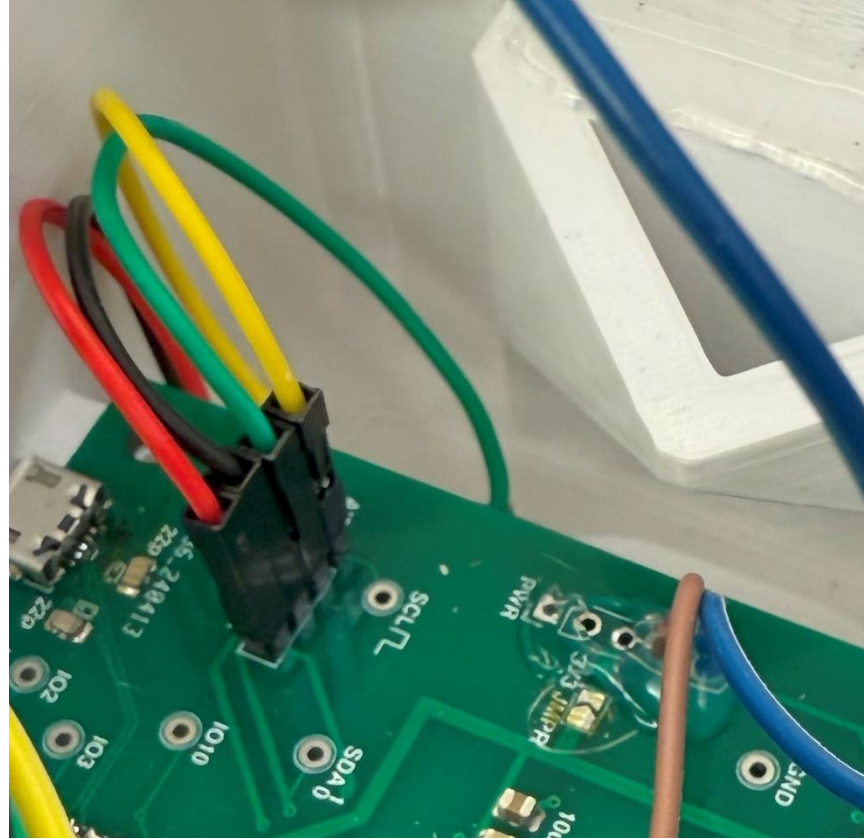
TODAY

- A variety of things to look out for to increase chance of success
- Across industrial design, electronics design, power, FW

Disclaimer: I'll be showing some "issues" with 2024 project. This isn't meant to criticize them. They did great! But we can continue to do better!

Electrical connections

Don't use hookup wires + header pins



Electrical connections

Use cables + connectors instead

More robust

Easier/faster/more reliable assembly/disassembly

Electrical connections

Common cabling/connector specs

- Number of conductors
- Voltage & current ratings
- Frequency
- Temperature
- Pitch
- Plating
- Insertion force
- SMT, thru-hole, panel mount, etc.
- Locking, polarized, keyed

**Some cable approaches need
special tooling to create
Beware!**

Nothing in our projects is too crazy. Take the win!

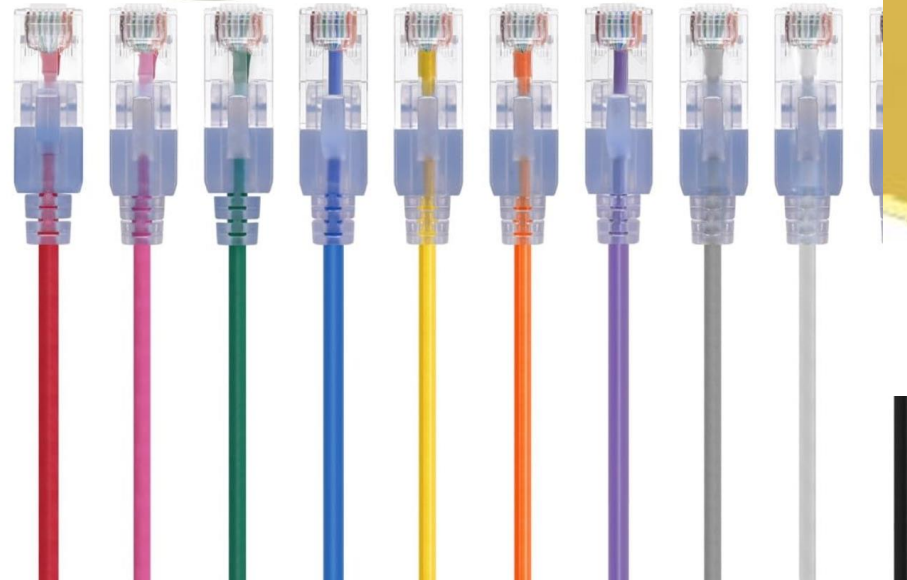
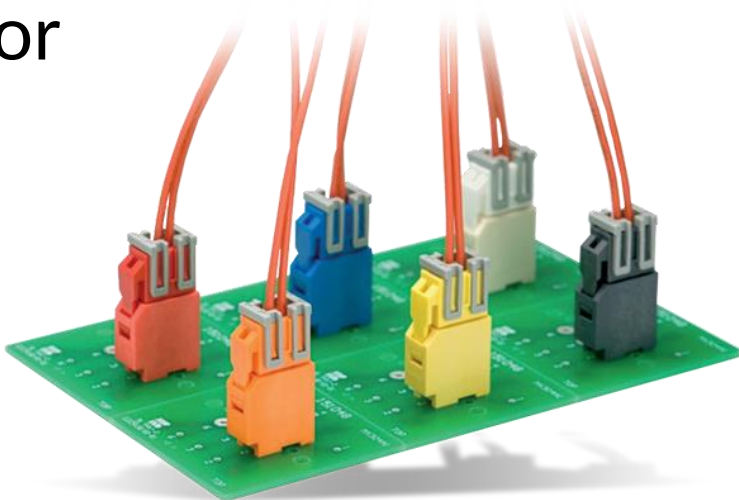
Disambiguating connectors

If you have 12 2x2 connectors in your system

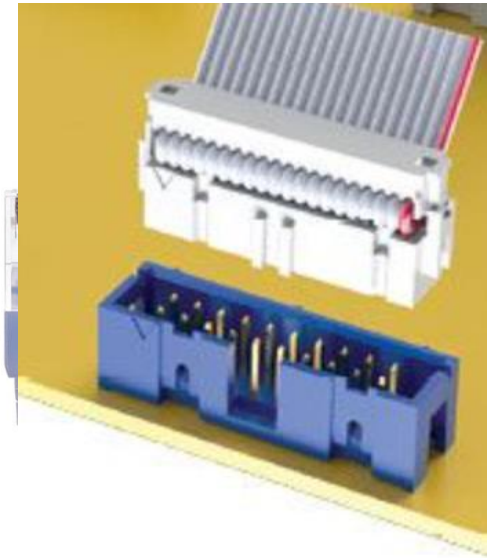
- What cable goes to what?

You can use

- Keying
- Polarization
- Color

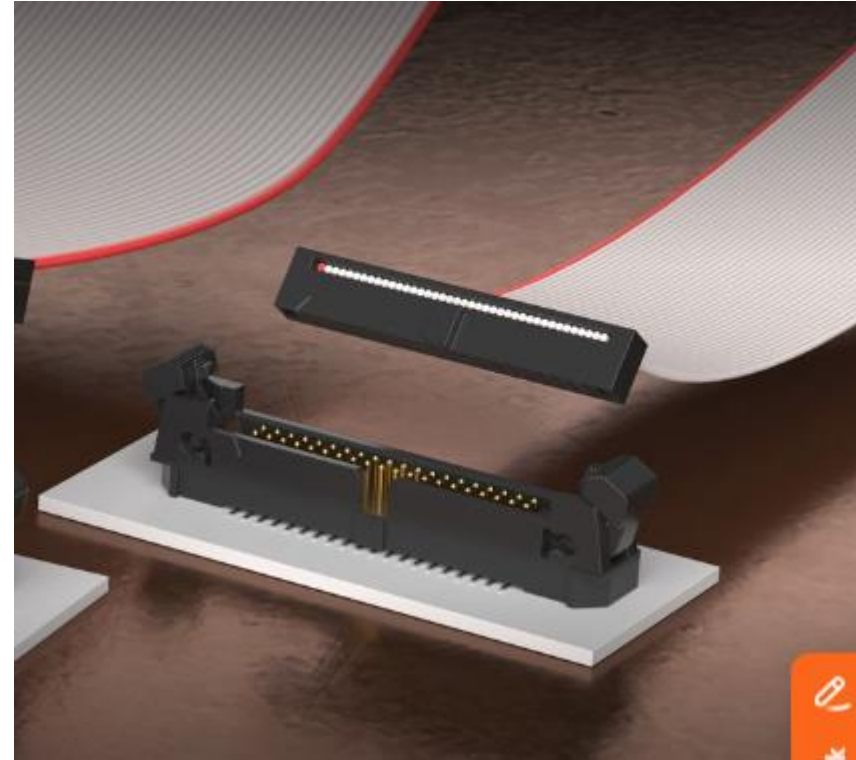


Keyed connectors



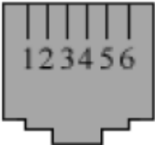
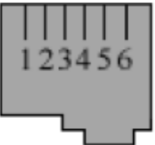
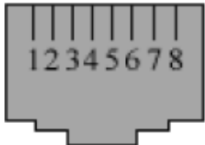
Locking connectors

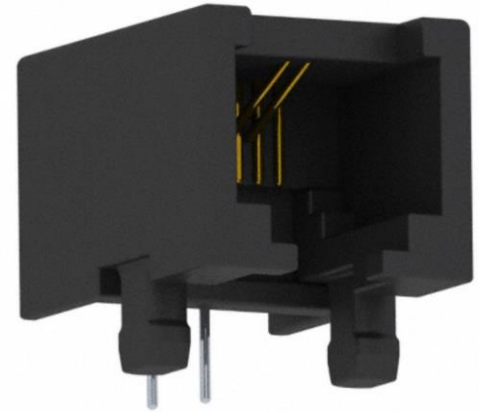
- Vibration happens
- Not avail in 2x2 (4-pin) configuration (I think)



Locking connectors

- There are plenty of other types of locking connectors

	<u>4 Position Modular Jack</u> (Often called an RJ11 jack or plug.)
	<u>6 Position Modular Jack</u> (Often called an RJ11 or RJ12 jack or plug.)
	<u>6 Position Modified Modular Jack</u> (Often called an MMJ jack or plug.)
	<u>8 Position Modular Jack</u> (Often called an RJ45 jack or plug.)

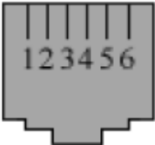
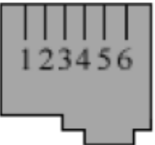
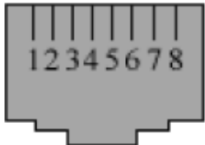


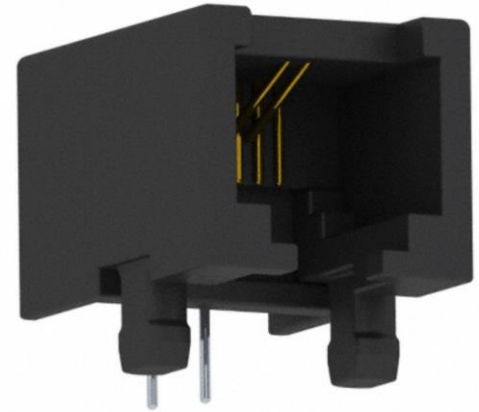
RJ11 jack
\$0.109@150



Locking connectors

- There are plenty of other types of locking connectors

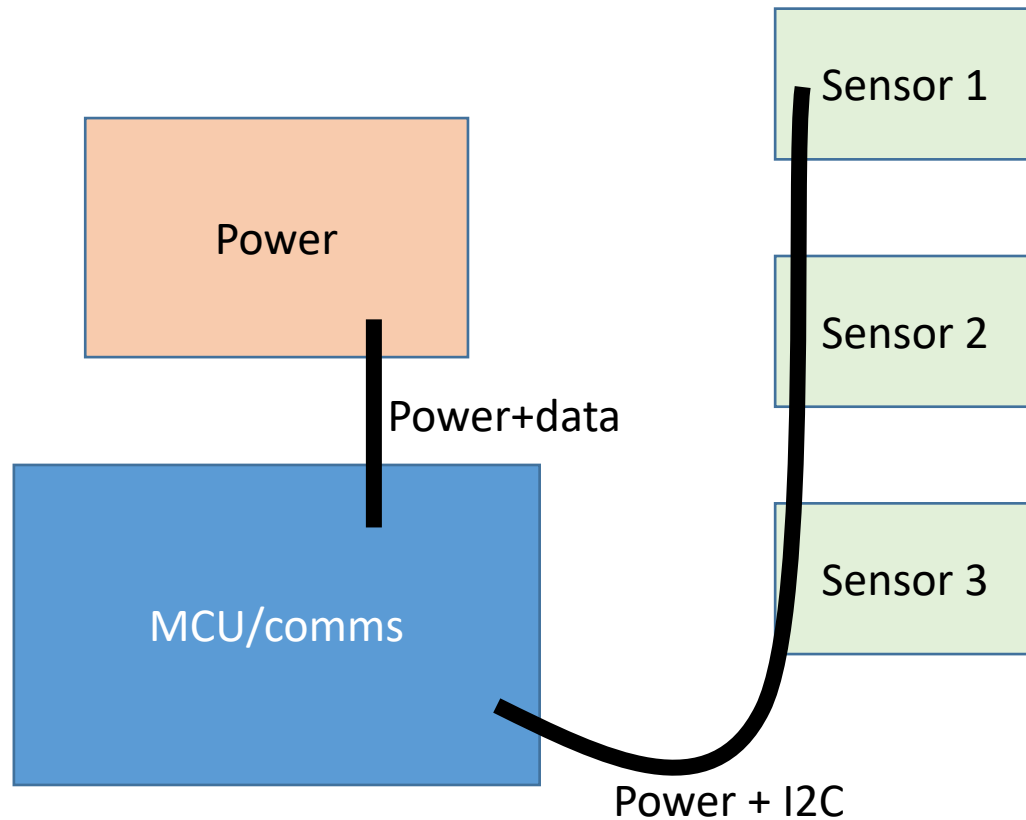
	<u>4 Position Modular Jack</u> (Often called an RJ11 jack or plug.)
	<u>6 Position Modular Jack</u> (Often called an RJ11 or RJ12 jack or plug.)
	<u>6 Position Modified Modular Jack</u> (Often called an MMJ jack or plug.)
	<u>8 Position Modular Jack</u> (Often called an RJ45 jack or plug.)



RJ11 jack
\$0.109@150

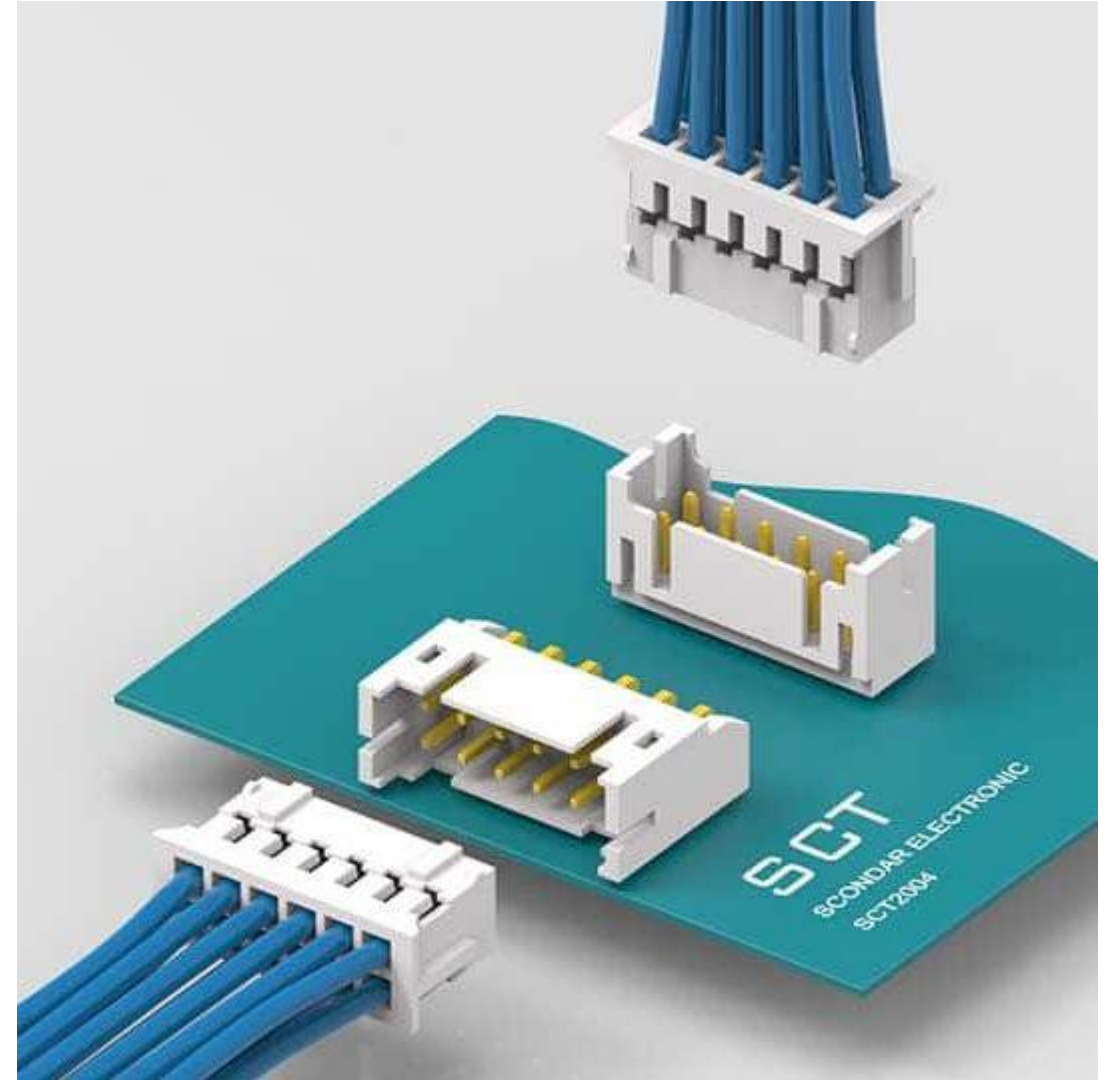
One cable many connectors

- IDC can do multiple connectors with one cable
- Why have a bunch of I2C cables when they all share same SDA, SCL, PWR, GND

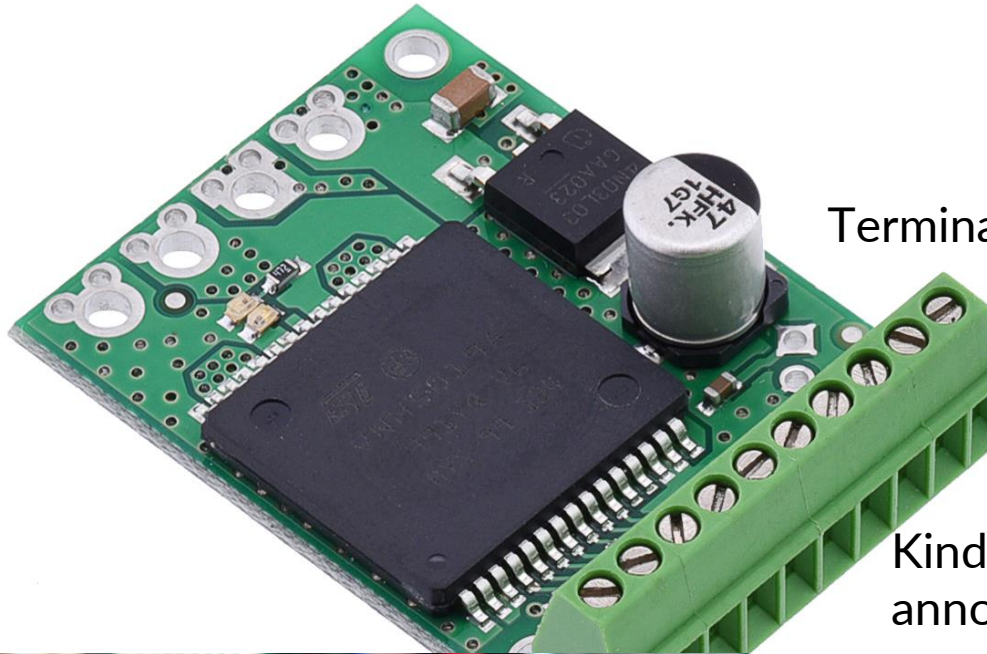


Connectors up or side

- Connectors can “vertical” (point up) or be “right angle”
- Almost any connector



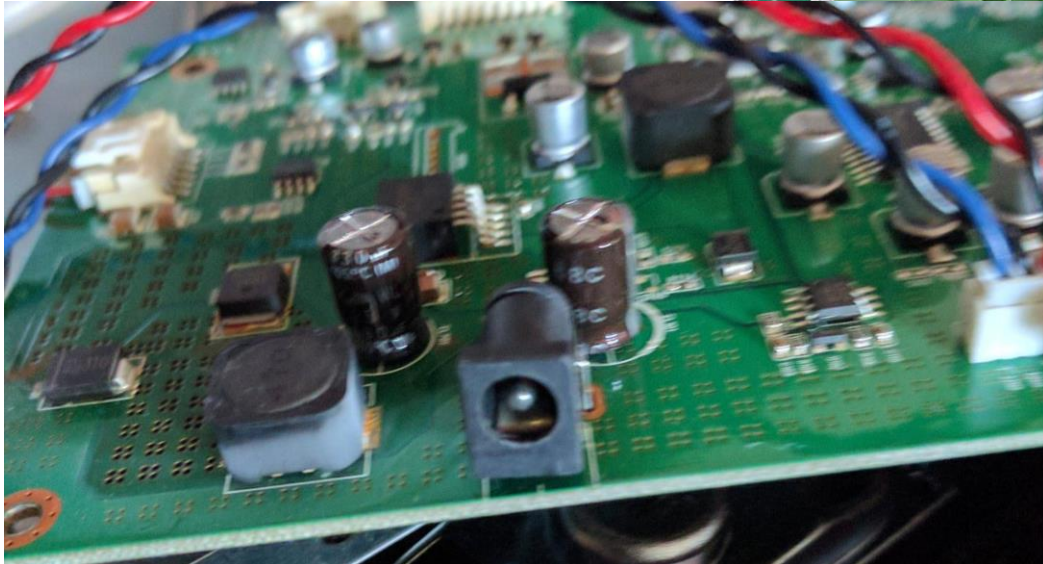
More connectors



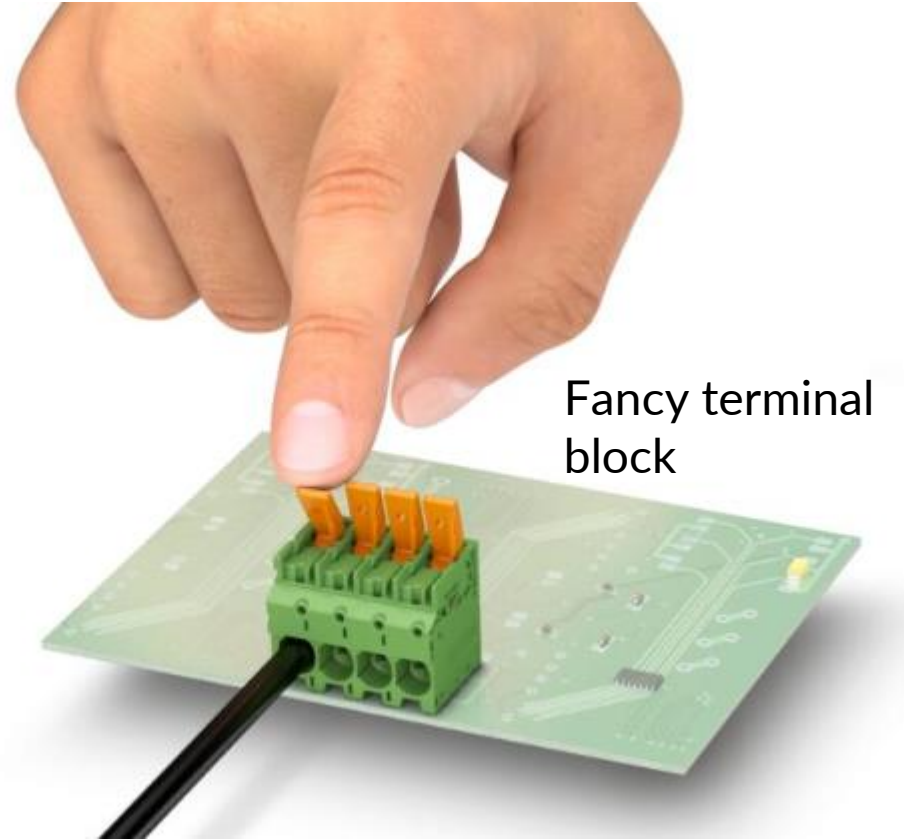
Terminal block

Kinda
annoying

Barrel jack



Common for
power, a bit bulky

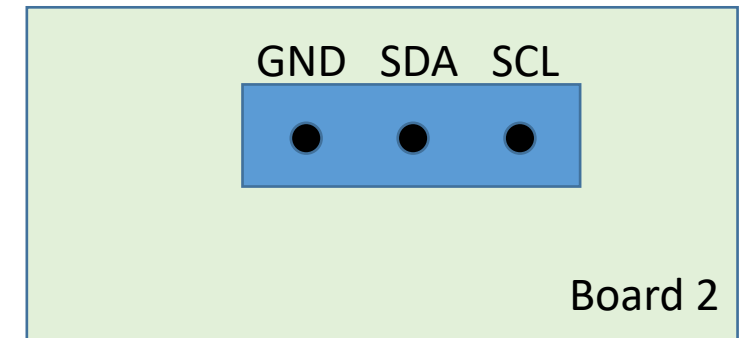
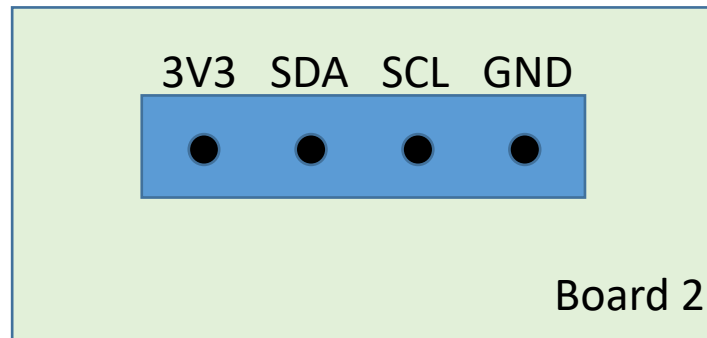
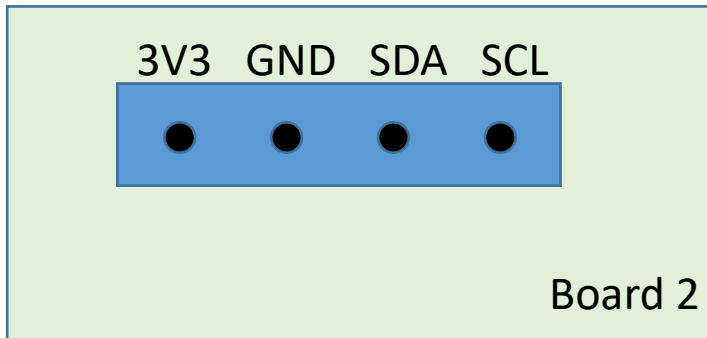
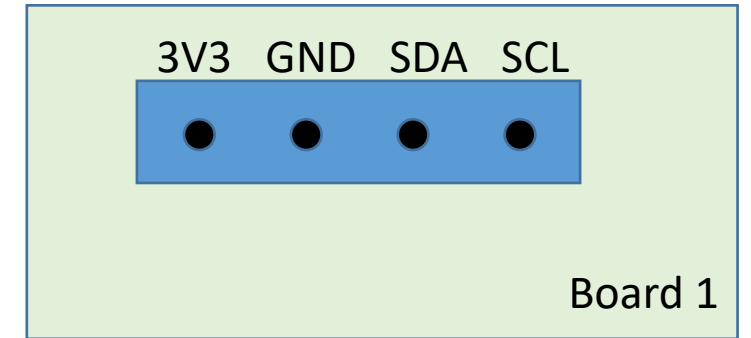
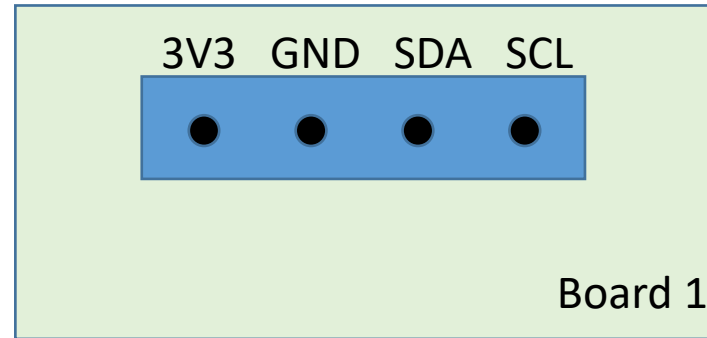
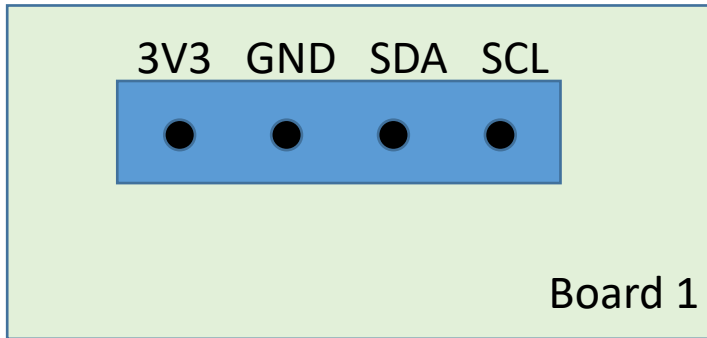


Fancy terminal
block

Cute, a bit pricey

Define the physical/electrical interface

- Cabling will ONLY work if connectors on different boards correspond



Where does happiness lie?

Make your own boards

- You get exactly what you want
- And nothing else (to eat your power)

It's fine (preferable, actually) to use breakout boards for prototyping

But final product should use your own boards*

*Unless there's some crazy part that is too hard to deal with

Make your own boards

- Be careful in choosing the ESP32-C3 chip

NO



ESP32-C3 chip

NO



ESP32-C3 MINI-1 module

YES



ESP32-C3-WROOM-02 module

Make your own boards

- Be careful in choosing the ESP32-C3 pins

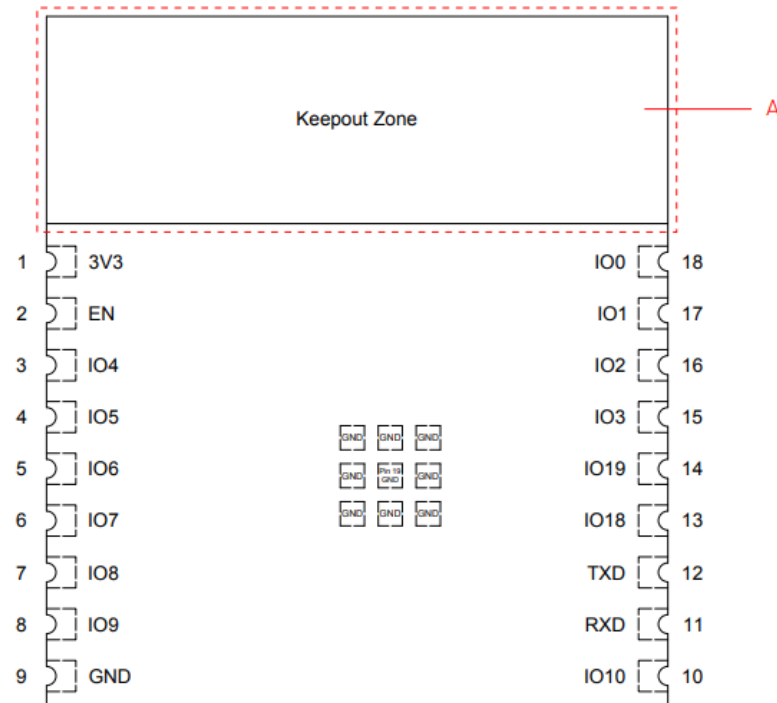


Table 3: Pin Definitions

Name	No.	Type ¹	Function
3V3	1	P	Power supply
EN	2	I	High: on, enables the chip. Low: off, the chip powers off. Note: Do not leave the EN pin floating.
IO4	3	I/O/T	GPIO4, ADC1_CH4, FSPIHD, MTMS
IO5	4	I/O/T	GPIO5, ADC2_CH0, FSPIWP, MTDI
IO6	5	I/O/T	GPIO6, FSPICLK, MTCK
IO7	6	I/O/T	GPIO7, FSPID, MTDO
IO8	7	I/O/T	GPIO8
IO9	8	I/O/T	GPIO9
GND	9,19	P	Ground
IO10	10	I/O/T	GPIO10, FSPICSO
RXD	11	I/O/T	GPIO20, U0RXD
TXD	12	I/O/T	GPIO21, U0TXD
IO18	13	I/O/T	GPIO18, USB_D-
IO19	14	I/O/T	GPIO19, USB_D+
IO3	15	I/O/T	GPIO3, ADC1_CH3
IO2	16	I/O/T	GPIO2, ADC1_CH2, FSPIQ
IO1	17	I/O/T	GPIO1, ADC1_CH1, XTAL_32K_N
IO0	18	I/O/T	GPIO0, ADC1_CH0, XTAL_32K_P

¹ P: power supply; I: input; O: output; T: high impedance.

Make your own boards

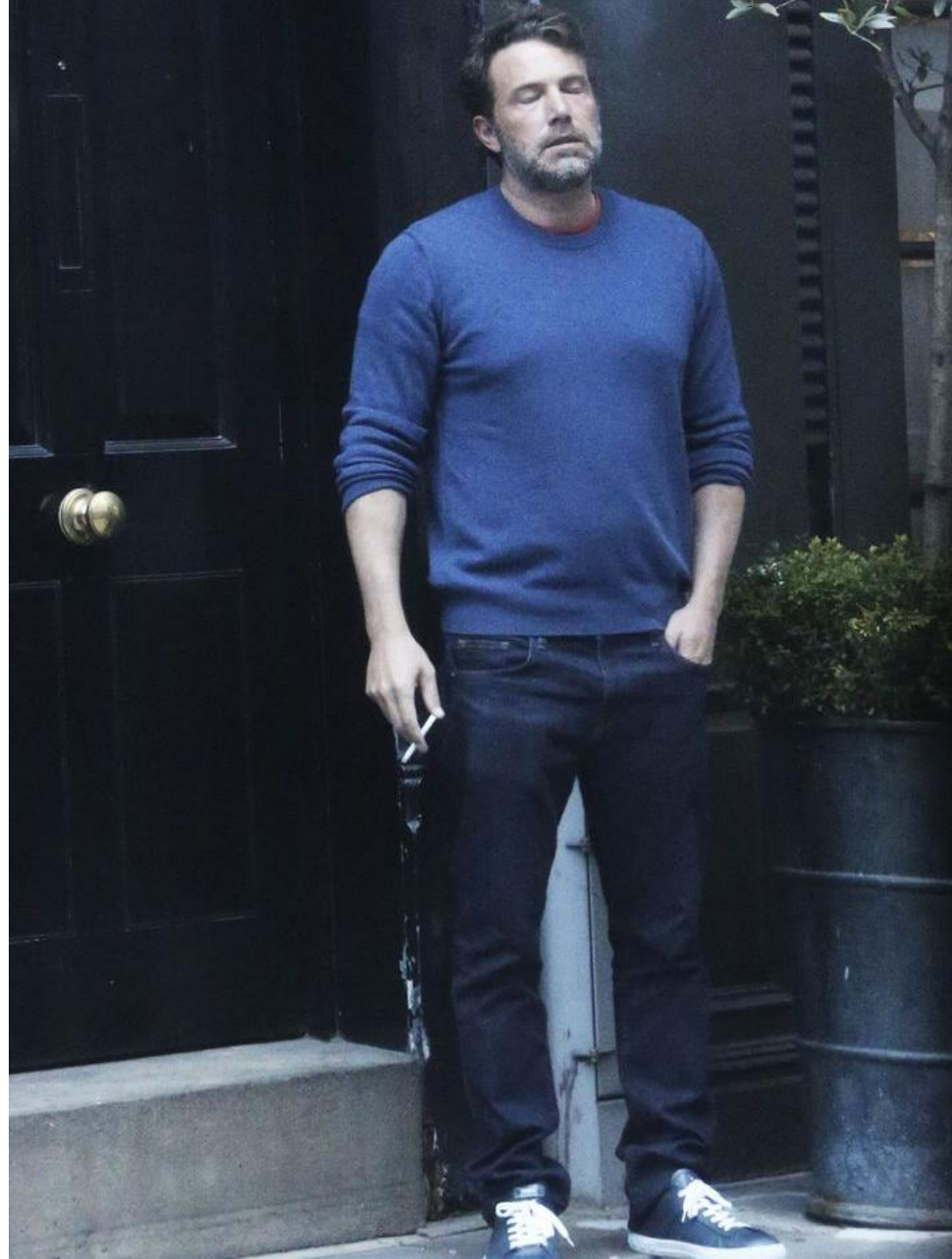
- Be careful in choosing the ESP32-C3 pins
- Figure out what pins/peripherals you need
 - USB, I2C, SPI, UART, ADCs, GPIOs, etc.
- Watch out for strapping pins
- Map those early!
- Make sure you have enough pins!

Table 3: Pin Definitions

Name	No.	Type ¹	Function
3V3	1	P	Power supply
EN	2	I	High: on, enables the chip. Low: off, the chip powers off. Note: Do not leave the EN pin floating.
IO4	3	I/O/T	GPIO4, ADC1_CH4, FSPIHD, MTMS
IO5	4	I/O/T	GPIO5, ADC2_CH0, FSPIWP, MTDI
IO6	5	I/O/T	GPIO6, FSPICLK, MTCK
IO7	6	I/O/T	GPIO7, FSPID, MTDO
IO8	7	I/O/T	GPIO8
IO9	8	I/O/T	GPIO9
GND	9,19	P	Ground
IO10	10	I/O/T	GPIO10, FSPICSO
RXD	11	I/O/T	GPIO20, U0RXD
TXD	12	I/O/T	GPIO21, U0TXD
IO18	13	I/O/T	GPIO18, USB_D-
IO19	14	I/O/T	GPIO19, USB_D+
IO3	15	I/O/T	GPIO3, ADC1_CH3
IO2	16	I/O/T	GPIO2, ADC1_CH2, FSPIQ
IO1	17	I/O/T	GPIO1, ADC1_CH1, XTAL_32K_N
IO0	18	I/O/T	GPIO0, ADC1_CH0, XTAL_32K_P

¹ P: power supply; I: input; O: output; T: high impedance.

Realizing on Apr 15 that
you don't have enough pins



Make your own boards

- Build in time for two revisions
- It takes ~1 week to go from Gerbers to boards in hand
 - Plus time needed to do schematic design, layout, reviews, assembly, test
- One suggested schedule
 - Rev 1: board files ready Fri Mar 21
 - Rev 2: board files ready Fri Apr 11
- **It is possible** to have JLC do assembly in addition to fab
 - They will do better soldering than you...
 - Will save you assembly + test time

BUT

 - Takes ~1 extra week [they quote 24-48h...]
 - You must design to the parts they have on hand
 - Or will need to assemble those parts

Make your own boards

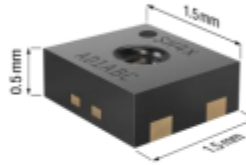
- Assembly + test
 - Make sure to build in time for this
 - Create test jigs + FW to make it easy to test boards
 - Test all the bits and pieces of each board

Make your own boards

- Manufacturers have many resources to help you succeed

Design Guide for Humidity and Temperature Sensors

A comment on how to properly design-in an SHTxx or STSxx.



SHTxx are humidity and temperature sensors of highest quality with a broad range of different features. To take full advantage of their outstanding performance, several housing and PCB design rules need to be considered. This document aims to provide help during the design-in phase of your product and fosters a deeper understanding of the sensor's functionality. Please note that unbeneficial housing and/or PCB designs may cause significant temperature and humidity deviations as well as an increased response times.

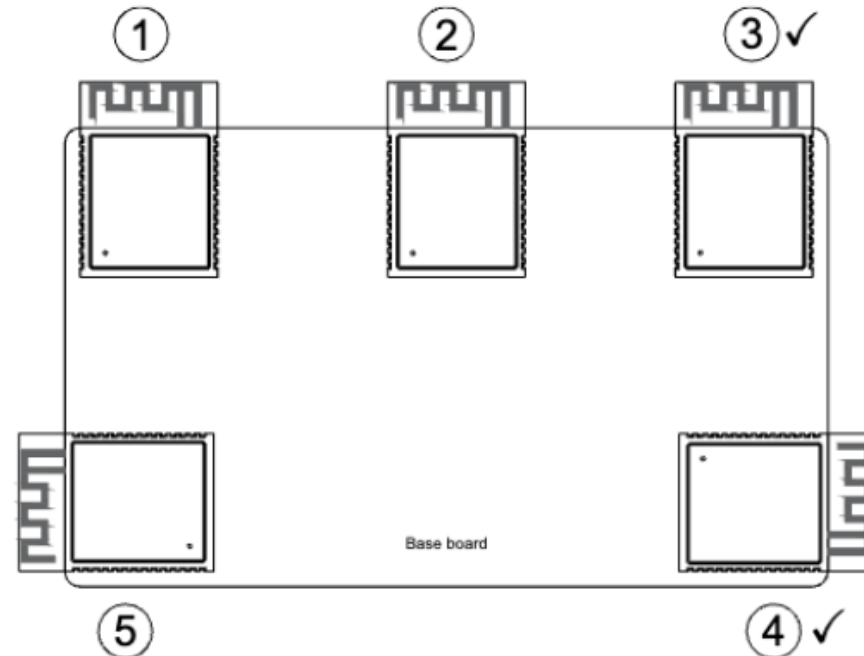
Make your own boards

- Manufacturers have many resources to help you succeed

PCB Layout Design

[中文]

This chapter introduces the key points of how to design an ESP32-C3 PCB layout using an ESP32-C3 module (see Figure [ESP32-C3 Reference PCB Layout](#)) as an example.



Placement of ESP32-C3 Modules on Base Board (antenna feed point on the right)

Make your own boards

- Manufacturers have many resources to help you succeed

The DNA of tech:

Designing the VEML7700 Into an Application

MECHANICAL CONSIDERATIONS AND WINDOW CALCULATION FOR THE VEML7700

The ambient light sensor will be placed behind a window or cover. The window material should be completely transmissive to visible light (400 nm to 700 nm). For optimal performance the window size should be large enough to maximize the light irradiating the sensor. In calculating the window size, the only dimensions that the design engineer needs to consider are the distance from the top surface of the sensor to the outside surface of the window and the size of the window. These dimensions will determine the size of the detection zone.

First, the center of the sensor and center of the window should be aligned. The VEML7700 has an angle of half sensitivity of about $\pm 55^\circ$, as shown in the figure below.

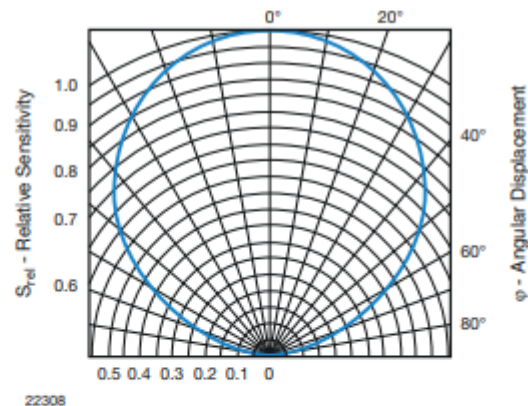


Fig. 17 - Relative Radiant Sensitivity vs. Angular Displacement

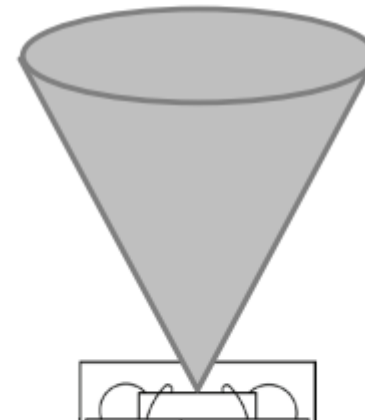


Fig. 18 - Angle of Half Sensitivity: Cone

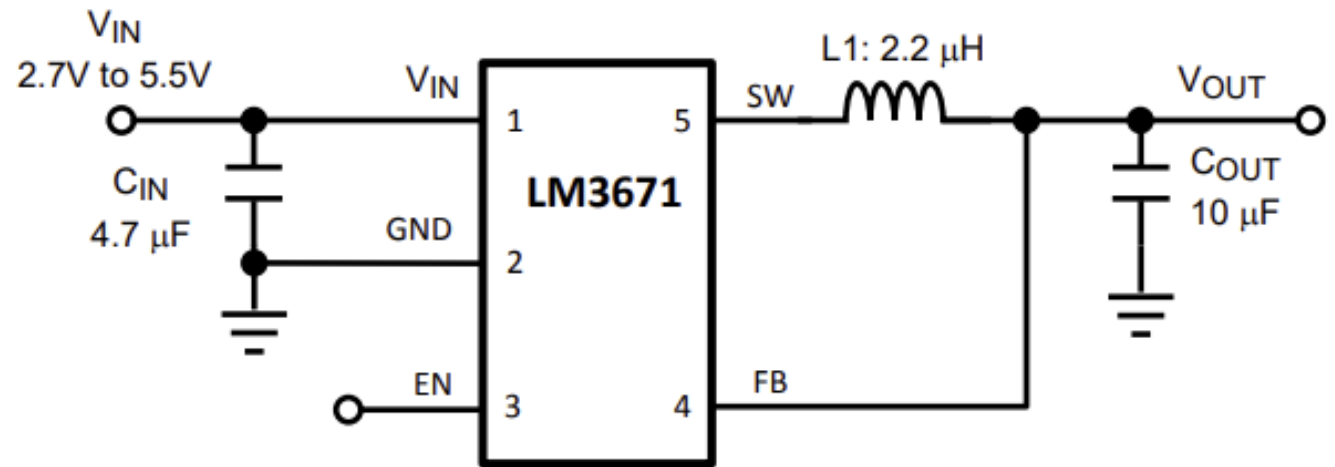
Remark:

This wide angle and the placement of the sensor as close as possible to the cover is needed, if it should show comparable results to an optometer, which also detects light reflections from the complete surroundings.

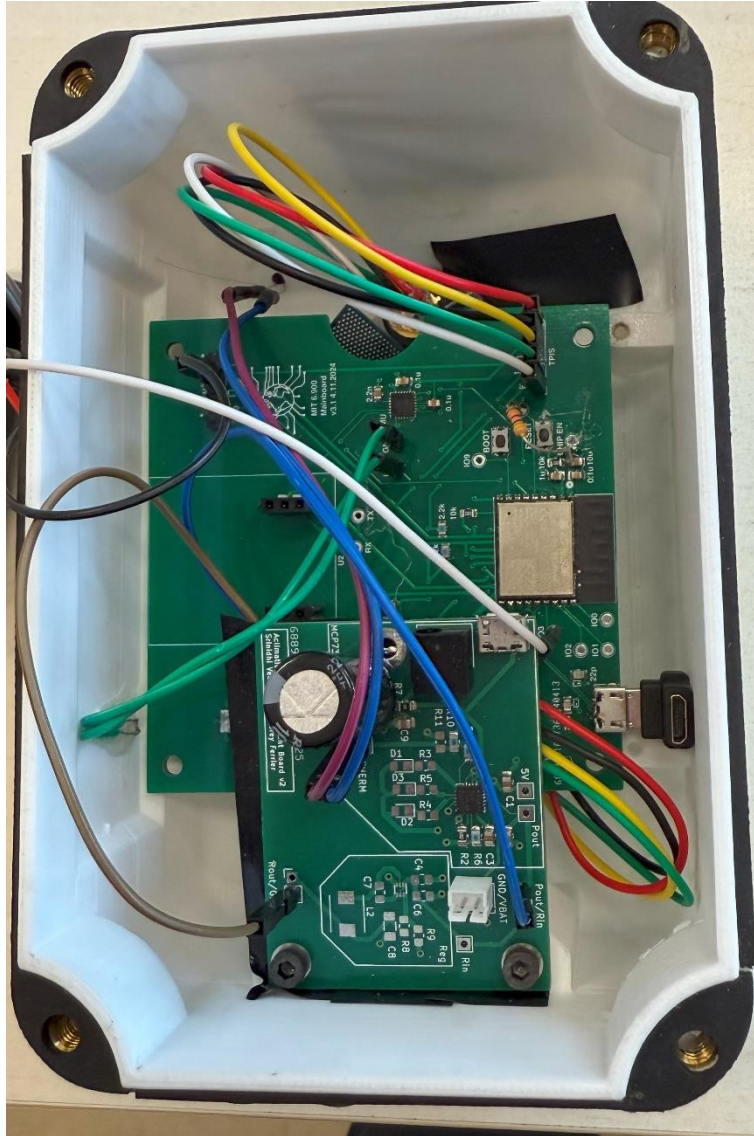
Design for test & assembly & debug

Put testpoints everywhere

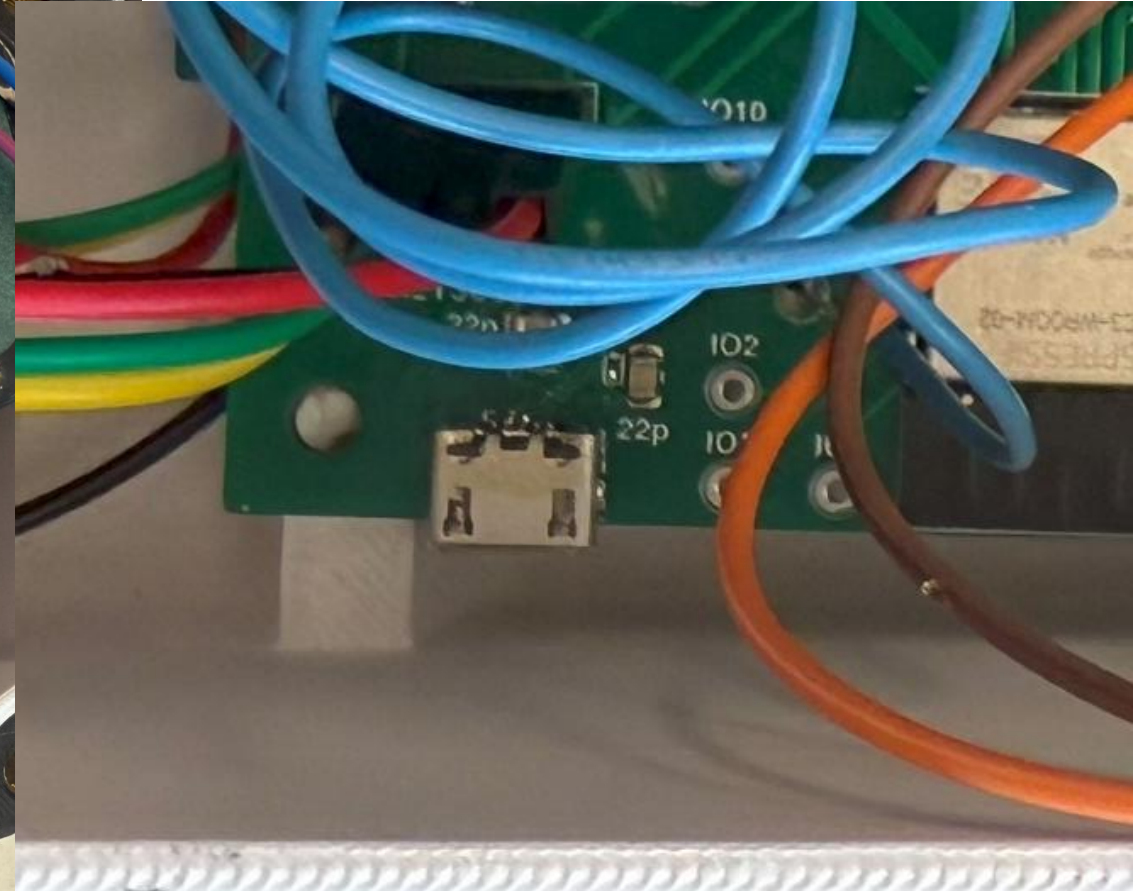
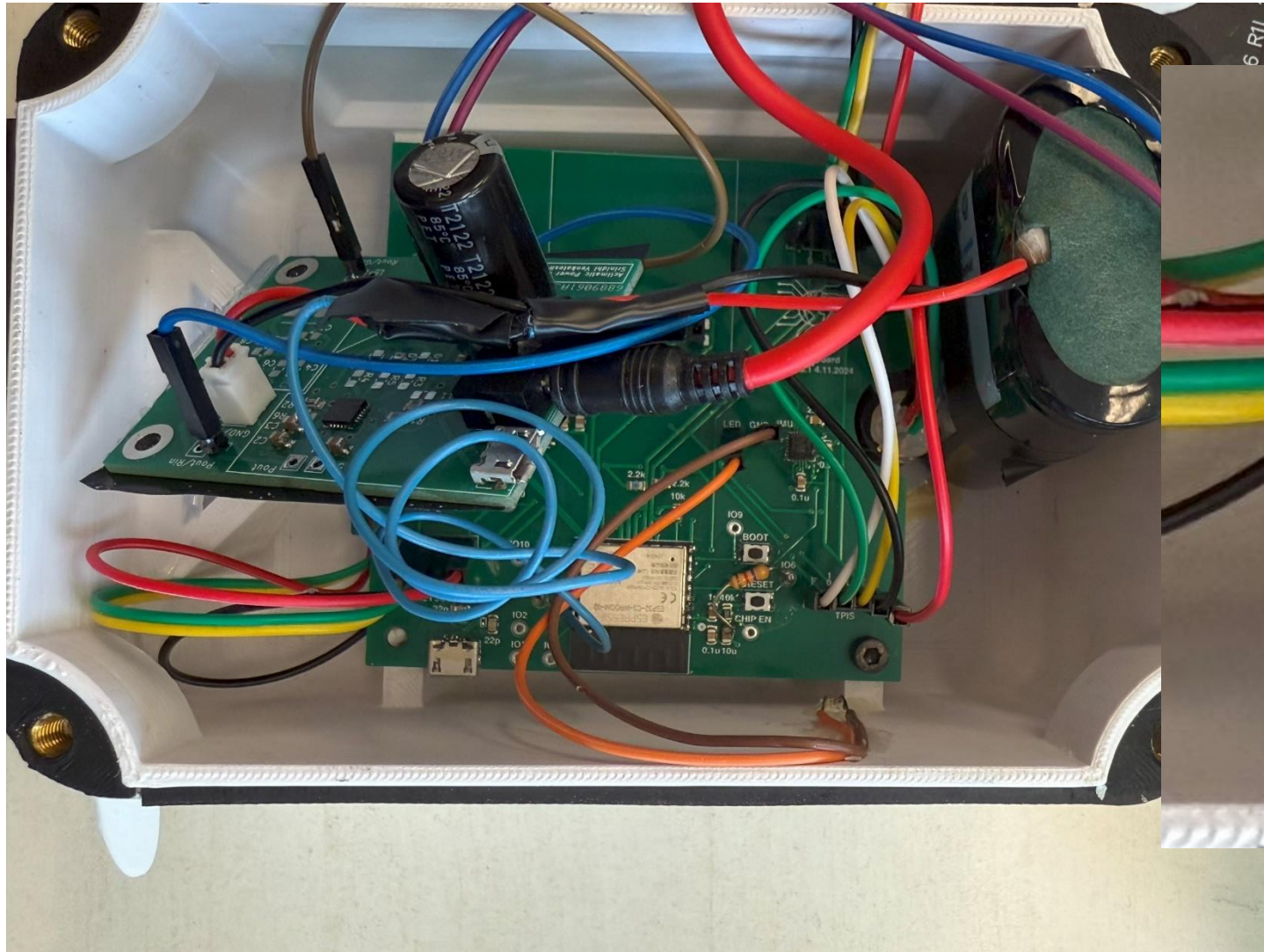
- Especially on
 - Unused GPIOs
 - Pins you have >1% chance of wanting to connect differently
 - EN pins, for example



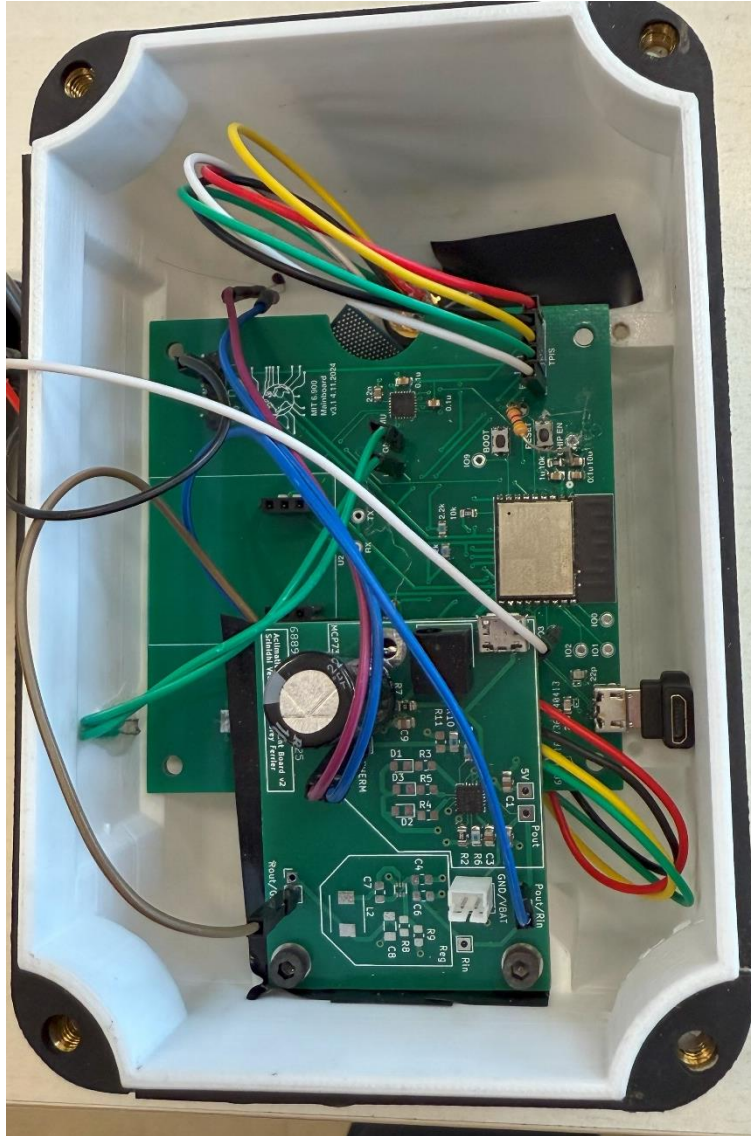
Design for test & assembly & debug



Design for test & assembly & debug

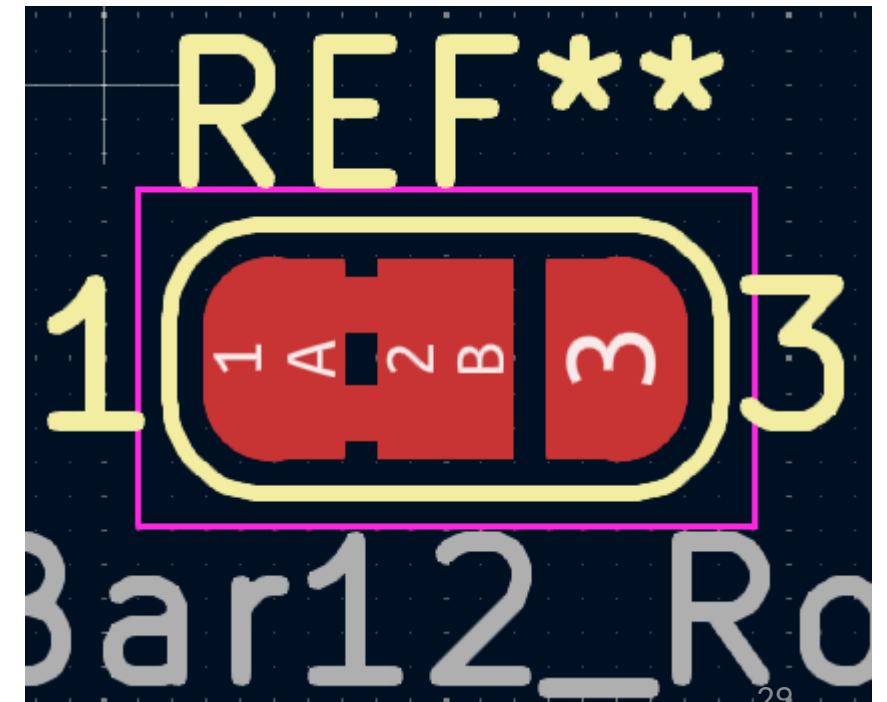
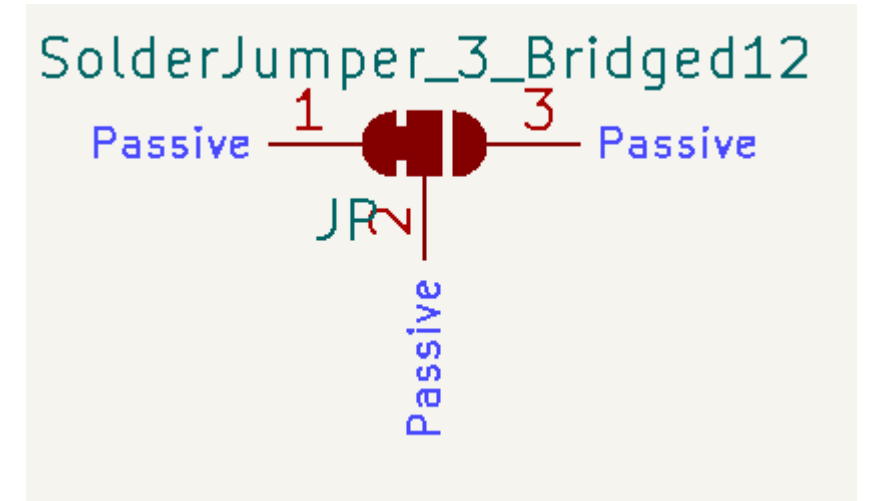
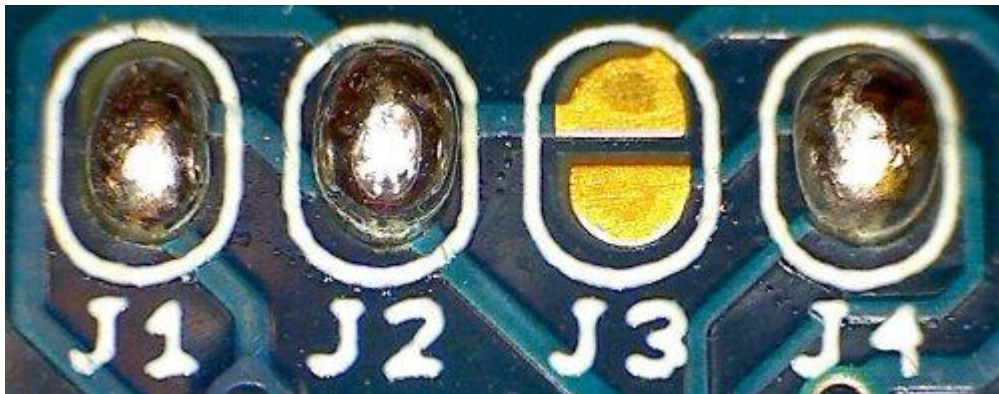
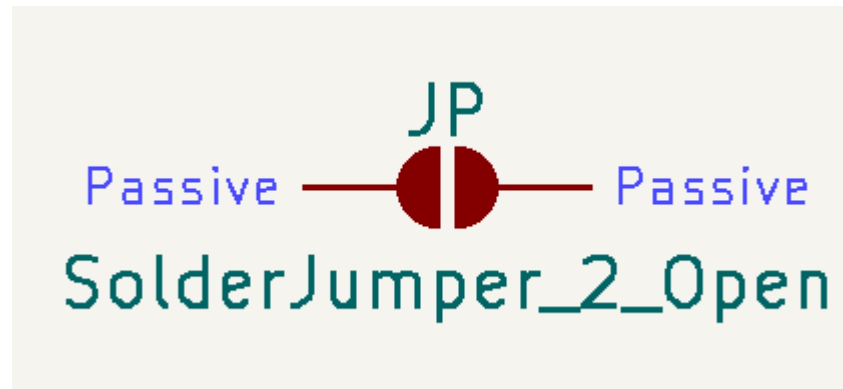


Design for test & assembly & debug



Design for test & assembly & debug

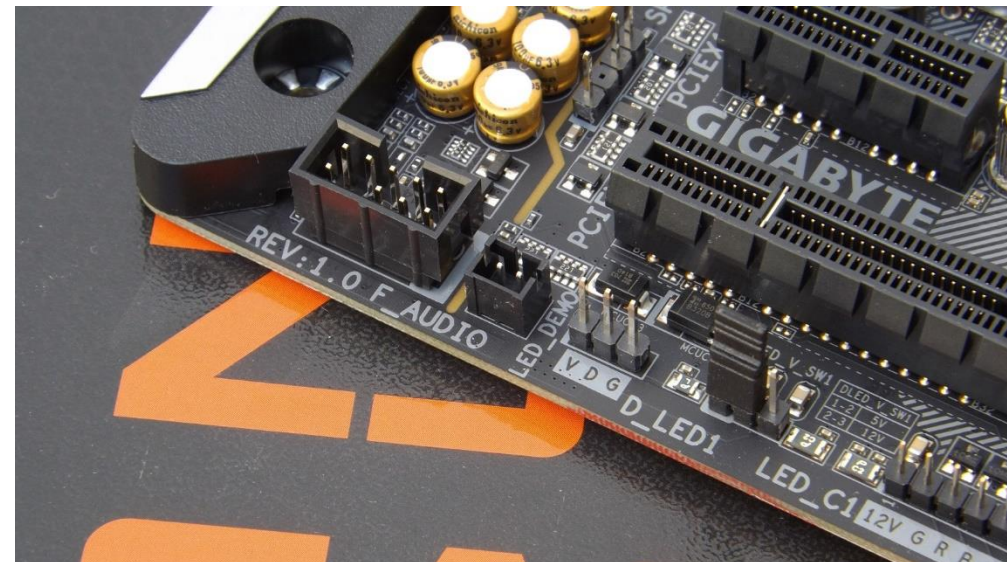
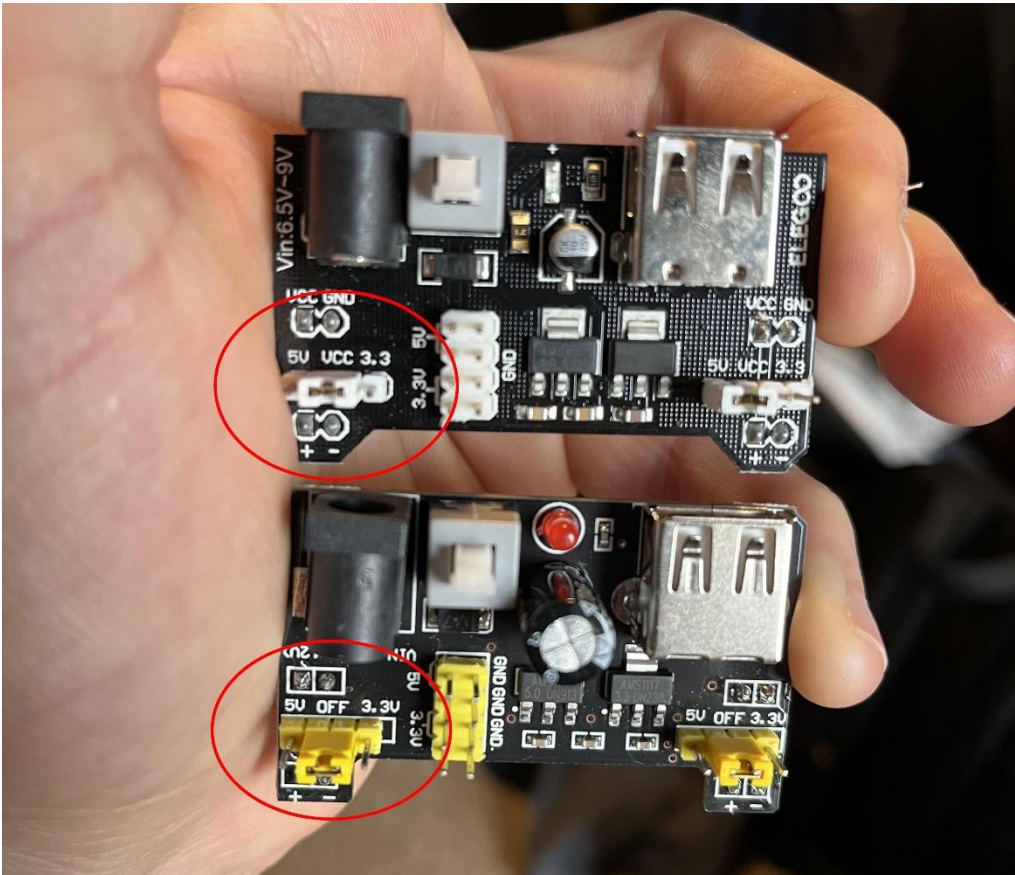
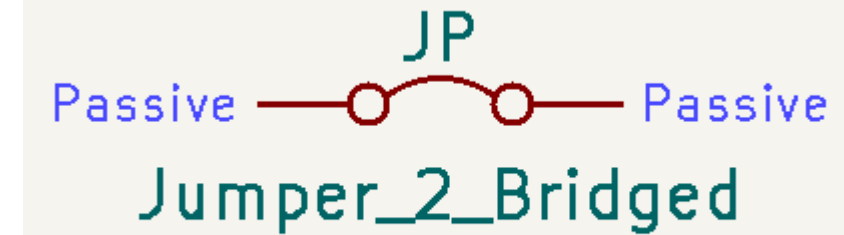
- Jumpers for connections that you
 - Want to specify later on
 - Might want to change



Design for test & assembly & debug

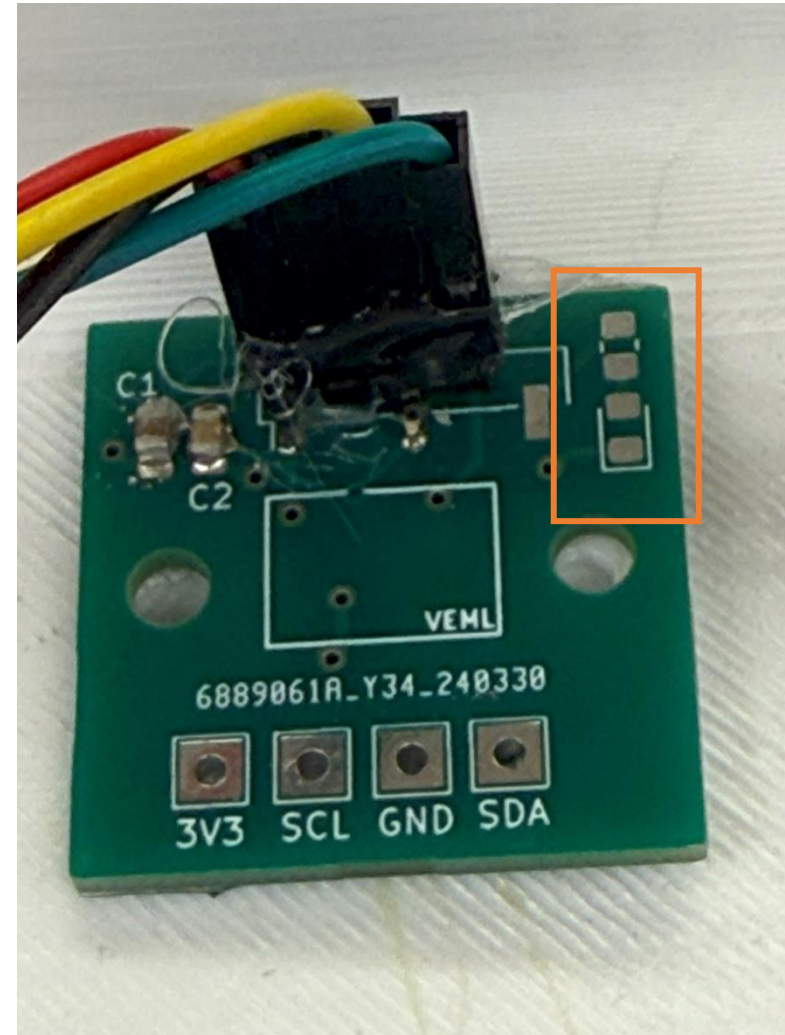
- Jumpers for connections that you
 - Want to specify later on
 - Might want to change

It's a good idea to put in a header wherever you want to use Joulescope to measure current



Unassembled parts

- Components in parallel can be placed without being assembled
- Put in a pull-down and pull-up resistor, choose which one to use later on
- Put in I2C pull-ups on each sensor board, only assemble on one



pull-ups

When things break

- When something goes wrong...what do you do?
- LEDs
- Displays
- SD cards
- Connectors

No data on dashboard

Is battery dead?
Sensor disconnected?
Not getting on WiFi?
MCU crash?
Some other issue?

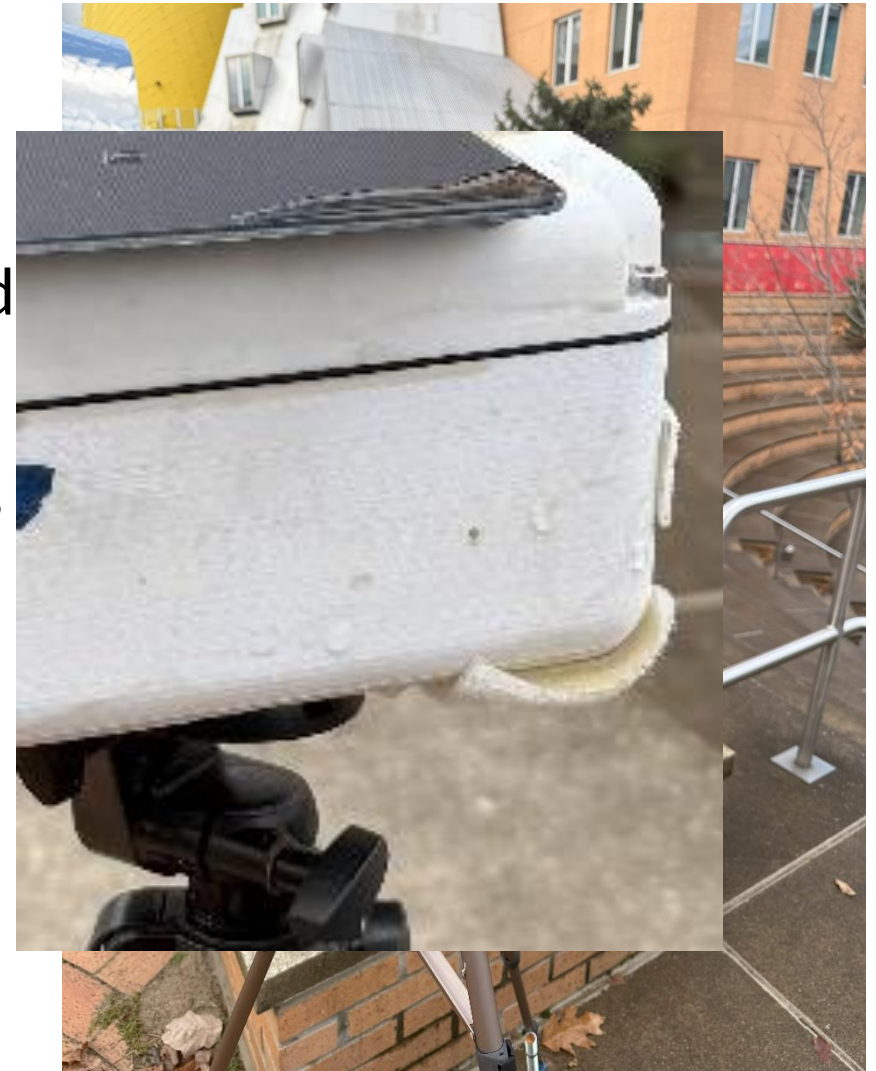


When things break

- When something goes wrong...what do you do?
- LEDs
- Displays
- SD cards
- Connectors

No data on dashboard

Is battery dead?
Sensor disconnected?
Not getting on WiFi?
MCU crash?
Some other issue?



When things break

- When something goes wrong...what do you do?
- LEDs
- Displays
- SD cards
- Connectors

When things break

- When something goes wrong...what do you do?

- LEDs



Status Light

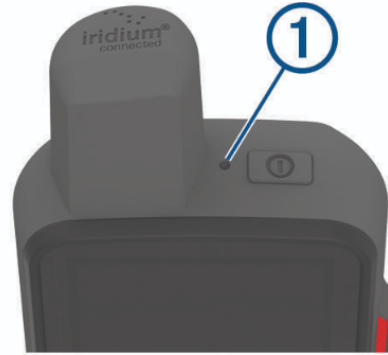
Light	Status
Off	Sonos product is not receiving power. • Ensure the power cable is fully inserted into the speaker and a working outlet and then check the Sonos app again. Sonos product is functioning properly and the light is disabled. • From the Settings tab, tap System and select your product's room name. Locate Status Light to enable or disable the LED.
Solid White	Sonos product is powered up and functioning properly. • If your product is not visible in the Sonos app, click here to add it back.
Flashing White	Sonos product is booting up after being plugged into power. Sonos product is waiting to receive an IP address from the router.
Solid Green	Sonos product is muted. • Use the Sonos app or volume buttons on the product to increase volume.
Flashing Green	Sonos product is powered on and ready to be set up.
Solid Orange	Sonos product is powered on and was unable to complete setup. • Reboot your Sonos product. Sonos product is overheating. • Turn off your Sonos product and allow it to cool down before attempting to power it back on. • If light remains solid orange, contact Sonos Customer Care.

One RGB LED with color+blinking

When things break

- When something goes wrong...what do you do?

- LEDs



LED Activity	Status
Double flashing green	You have an unread message.
Flashing green	The device is in expedition mode.
Flashing red	The device does not have a clear view of the sky.
	The device is below 10% battery power.
Alternating red and green	The device is in SOS mode.

When things break

- When something goes wrong...what do you do?

- LEDs

To conserve power, do not have an LED on all the time

Blinking is ok (great, actually)

Blink frequently enough that it's not annoying (<30 sec?
<15 sec?)

Use colors that are easy to see outdoors

Each LED requires a hole in your enclosure...

Or have a way of triggering debug mode

When things break

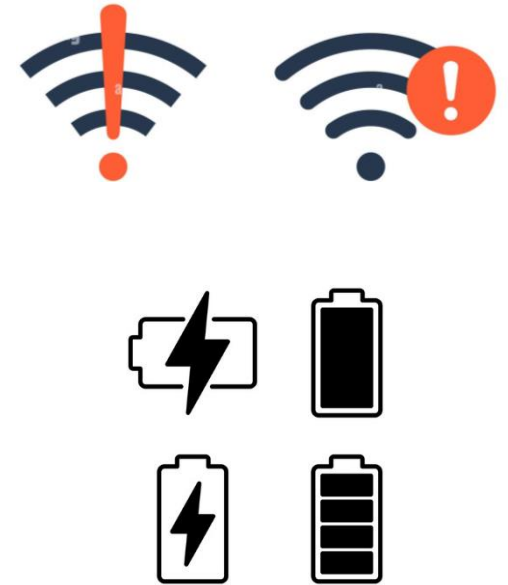
- When something goes wrong...what do you do?

- Displays

E-paper uses no static power

OK for MITOS

Probably not worth it for Bike lane



When things break

- When something goes wrong...what do you do?
- SD cards
 - Simplest ones are basically just SPI interface to ESP32
 - SD_MMC also exists (faster, more pins) but not avail via Arduino on ESP32-C3
 - Easy tutorials online



When things break

- When something goes wrong...what do you do?
- SD cards
 - How will you log?
 - Write a function to write custom messages to SD card

```
// This calls a function that appends  
// char* msg to the file denoted in char* path  
log_SD(path, msg);  
  
void log_SD(char * path, char * message) {  
    open file on SD card  
    print message to file  
    close file on SD card  
}
```

When things break

- When something goes wrong...what do you do?
- SD cards
 - How will you log?
 - Write a function to write custom messages to SD card

```
void setup() {  
→ Serial.begin(9600); //begin serial  
  // while (!Serial) {  
  //   delay(100);  
  // }  
  
  Wire.begin(I2C_SDA, I2C_SCL);  
  if (! sht4.begin()) {  
→   Serial.println("Couldn't find SHT4x");  
    while (1) delay(1);  
  }  
  sht4.setPrecision(SHT4X_HIGH_PRECISION);  
  sht4.setHeater(SHT4X_NO_HEATER);  
→ Serial.println("Found SHT4x sensor");  
→ Serial.print("Serial number 0x");  
→ Serial.println(sht4.readSerial(), HEX);  
  
  WiFi.begin(network, password);  
  //if using channel/mac specification for crowded bands use the  
  //WiFi.begin(network, password, channel, bssid);  
  uint8_t count = 0; //count used for Wifi check times  
→ Serial.print("Attempting to connect to ");  
→ Serial.println(network);  
  while (WiFi.status() != WL_CONNECTED && count < 6) { //can chan  
    delay(500);  
→   Serial.print(".");  
    count++;  
  }
```

When things break

- When something goes wrong...what do you do?
- SD cards
 - How will you log?
 - Write a function to replace all of your logging

```
mylog(path, msg);

// Could also include Serial.println in
// that function
void mylog(char * path, char * message) {
    open file on SD card
    print message to file
    print message to Serial
    close file on SD card
}
```

When things break

- When something goes wrong...what do you do?

- SD cards
 - Still missing plenty of info

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
confsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1184
load:0x40078000,len:13260
load:0x40080400,len:3028
entry 0x400805e4
```

When things break

- When something goes wrong...w
- SD cards
 - Still missing plenty of info

```
// Fix mount issue - sdWait fail ignored before command GO_IDLE_STATE
digitalWrite(card->ssPin, LOW);
if (!sdWait(pdrv, 500)) {
    log_w("sdWait fail ignored, card initialize continues");
}
if (sdCommand(pdrv, GO_IDLE_STATE, 0, NULL) != 1) {
    sdDeselectCard(pdrv);
    log_w("GO_IDLE_STATE failed");
    goto unknown_card;
}
sdDeselectCard(pdrv);

token = sdTransaction(pdrv, CRC_ON_OFF, 1, NULL);
if (token == 0x5) {
    //old card maybe
    card->supports_crc = false;
} else if (token != 1) {
    log_w("CRC_ON_OFF failed: %u", token);
    goto unknown_card;
}

if (sdTransaction(pdrv, SEND_IF_COND, 0x1AA, &resp) == 1) {
    if ((resp & 0xFFFF) != 0x1AA) {
        log_w("SEND_IF_COND failed: %03X", resp & 0xFFFF);
        goto unknown_card;
    }
}

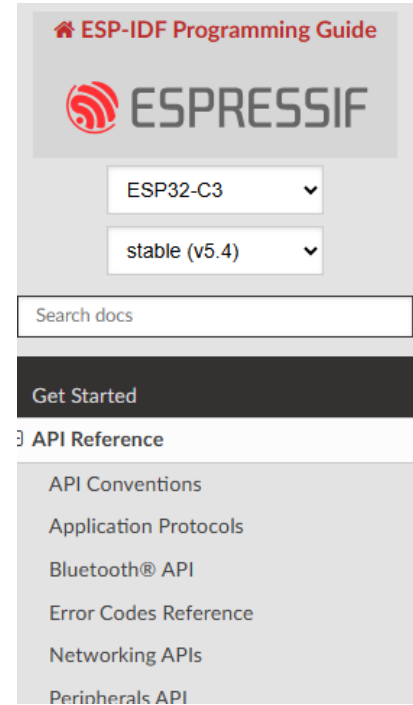
if (sdTransaction(pdrv, READ_OCR, 0, &resp) != 1 || !(resp & (1 << 20))) {
    log_w("READ_OCR failed: %X", resp);
    goto unknown_card;
}
```

When things break

- When something goes wrong...what do you do?

- SD cards

- How will you log?
- Utilize native ESP32 logging library, direct to Serial or SD
- If you could get this to work, would get *all* log messages



» API Reference » System API » Logging library

Logging library

[\[中文\]](#)

Overview

The logging library provides three ways for setting log verbosity:

- **At compile time:** in menuconfig, set the verbosity level using the option `CONFIG_LOG_DEFAULT_LEVEL`.
- Optionally, also in menuconfig, set the maximum verbosity level using the option `CONFIG_LOG_MAXIMUM_LEVEL`. By default, this is the same as the default level. It can be set higher in order to compile more optional logs into the firmware.
- **At runtime:** all logs for verbosity levels lower than `CONFIG_LOG_DEFAULT_LEVEL` are enabled by default. The function `esp_log_level_set()` can be used to set a log level.

[Redirect ESP32 log messages to SD card](https://community.platformio.org/t/redirect-esp32-log-messages-to-sd-card/33734/8)

<https://community.platformio.org/t/redirect-esp32-log-messages-to-sd-card/33734/8>

When things break

- When something goes wrong...what do you do?
- SD cards
 - You can adjust the SPI bus frequency
 - Consider write size
 - Measure power consumption
 - May vary with different cards, write buffer sizes
 - May choose different amounts of logging for testing vs. deployment

When things break

- When something goes wrong...what do you do?
- Connectors
 - Can bring USB (or other interface) to outside of box without needing to access interior
 - Or consider an access compartment in enclosure



\$2.35@100

When things really go wrong

- Sometimes even with careful design, code fails
- How to reset device in the field?



When things really go wrong

- Sometimes even with careful design, code fails
- How to reset device in the field?







EXCLUSIVE

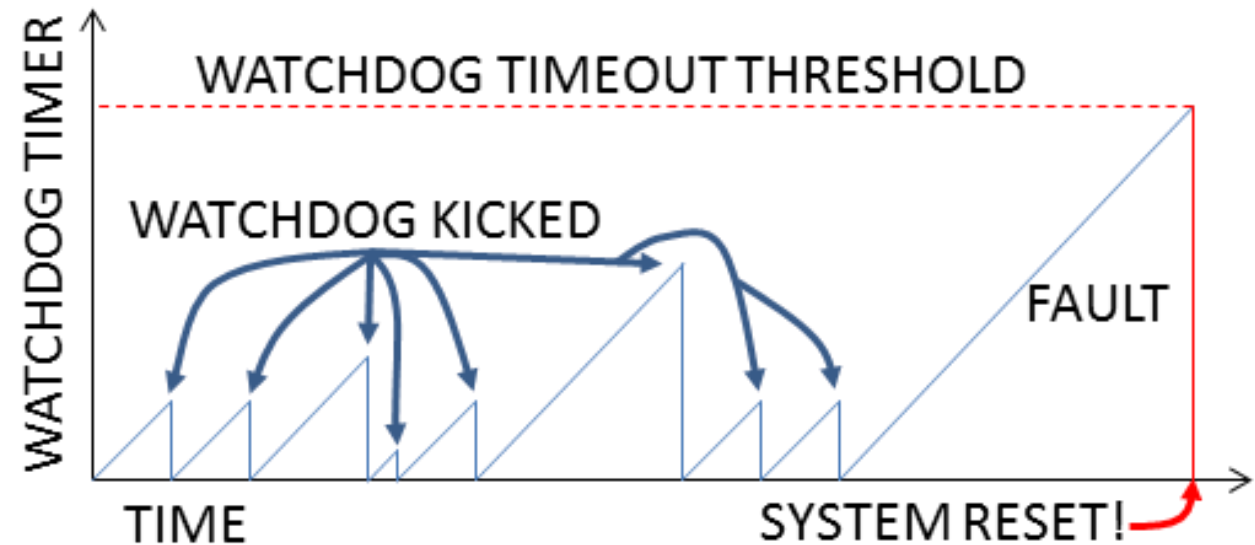
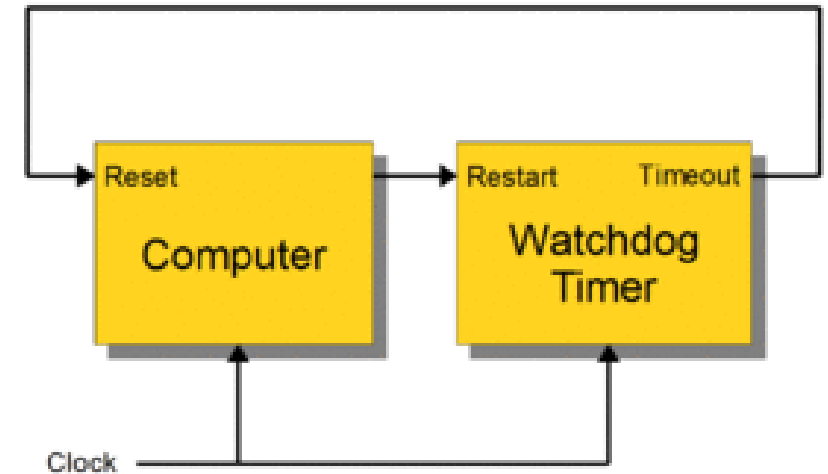
POP CULTURE

EXCLUSIVE: Fyre Festival 2: Dates, location and how to get tickets

“My dream is finally becoming a reality,” Billy McFarland, the founder of the controversial festival, told TODAY.

Watchdog timers

- A sentinel that monitors your MCU
- A timer that counts up
- If the timer reaches threshold, it reboots the MCU*
- The MCU code periodically resets the watchdog when it is sure that everything is OK



*or does something else specified



Watchdog timers

- Software watchdog
 - A function with ESP.restart() inside it
 - Called when a timer variable goes off
 - Basically useless
 - Don't do this
- Hardware watchdog
 - A distinct internal HW block
 - This is what we'll use
- A separate HW watchdog chip
 - If you **really** want to be safe

```
Serial.print("Attempting to connect to ");
Serial.println(network);
while (WiFi.status() != WL_CONNECTED && count < 6) { //can change this
  delay(500);
  Serial.print(".");
  count++;
}
delay(2000); //acceptable since it is in the setup function.
if (WiFi.isConnected()) { //if we connected then print our IP, Mac, a
  Serial.println("CONNECTED!");
  delay(500);
} else { //if we failed to connect just Try again.
  Serial.println("Failed to Connect :/ Going to restart");
  Serial.println(WiFi.status());
  ESP.restart(); // restart the ESP (proper way)
}
```

- Software watchdog
 - A function with `esp_restart()` inside it
 - Basically useless
 - Don't do this
- Hardware watchdog
 - A distinct internal HW block
 - This is what we'll use
- A separate HW watchdog chip
 - If you **really** want to be safe

µP Reset Circuits with Capacitor-Adjustable Reset/Watchdog Timeout Delay

Selector Guide and Ordering Information appear at end of

ESP32 watchdog timers

- Four on-board watchdog timers
 - Two main system watchdog timers
 - This is what you'll be using, as the “task watchdog timer” monitoring (on MWDT0)
 - When it fires, will cause core reset
 - There is also an interrupt watchdog timer (on MWDT1) to make sure interrupts don't get blocked
 - If something disables interrupts for too long
 - RTC Watchdog
 - Used during boot to make sure boot occurs quickly enough
 - One low-power “super watchdog” timer
 - Operates independently, looking for feed from CPU every second

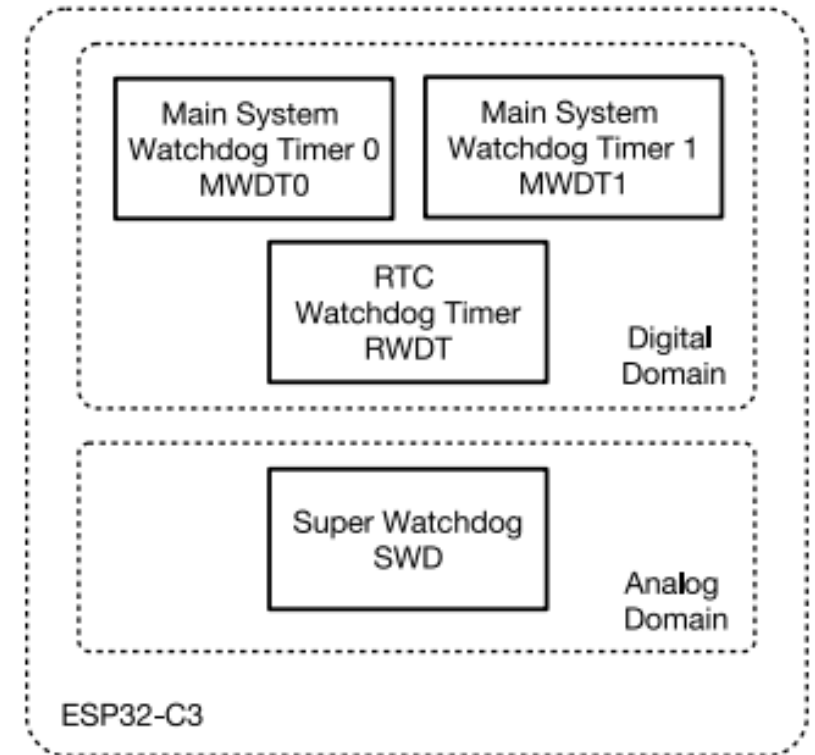


Figure 12.1-1. Watchdog Timers Overview

Watchdog timers

- Not for general error handling
- For catching unanticipated errors
 - ESD, electrical noise, etc.
 - Memory leak causing overflow
- Or things that really shouldn't happen
 - Up to you to decide!

Watchdog timers

- Accessible via Arduino framework

```
1  #include <Arduino.h>
2  #include "esp_task_wdt.h"
3
4  //3 seconds WDT
5  #define WDT_TIMEOUT 3
6
7  #define I2C_SDA 4
8  #define I2C_SCL 5
9
10 Adafruit_SHT4x sht4 = Adafruit_SHT4x();
11 int RHT_MEASUREMENT_INTERVAL = 1000;
12
13 void setup() {
14     Serial.begin(115200);
15     Serial.println("Configuring WDT...");
16     esp_task_wdt_init(WDT_TIMEOUT, true); //initialize WDT
17     esp_task_wdt_add(NULL); //add current thread to WDT watch
18 }
```

```
36
37 void loop() {
38     // do some stuff
39     esp_task_wdt_reset(); // feed the WDT
40 }
```

*Note that this format has apparently changed in more recent version of arduino-esp
But at least my platformio is using this version*

Watchdog timers

- Setting the WDT timeout
- **Longer than the longest you expect to go thru your loop**
- This can get tricky with
 - Comms: WiFi, cellular, etc.
 - SD card writes
 - etc.

Your projects are not life-threatening or safety critical...
...err on side of too long

```
1  #include <Arduino.h>
2  #include "esp_task_wdt.h"
3
4  //3 seconds WDT
5  #define WDT_TIMEOUT 3
6
7  #define I2C_SDA 4
8  #define I2C_SCL 5
9
10 Adafruit_SHT4x sht4 = Adafruit_SHT4x();
11 int RHT_MEASUREMENT_INTERVAL = 1000;
12
13 void setup() {
14     Serial.begin(115200);
15     Serial.println("Configuring WDT...");
16     esp_task_wdt_init(WDT_TIMEOUT, true); //initialize WDT
17     esp_task_wdt_add(NULL); //add current thread to WDT watch
18 }
```

Watchdog timers

- Simplest architecture
- Execute each time you go thru your loop
- Simpler, a bit less robust
- Will this catch all the faults you might expect to occur?

```
28
29  loop {
30      read_sensor1();
31      read_sensor2();
32      connect_to_wifi();
33      esp_task_wdt_reset();
34  }
35
```

Watchdog timers

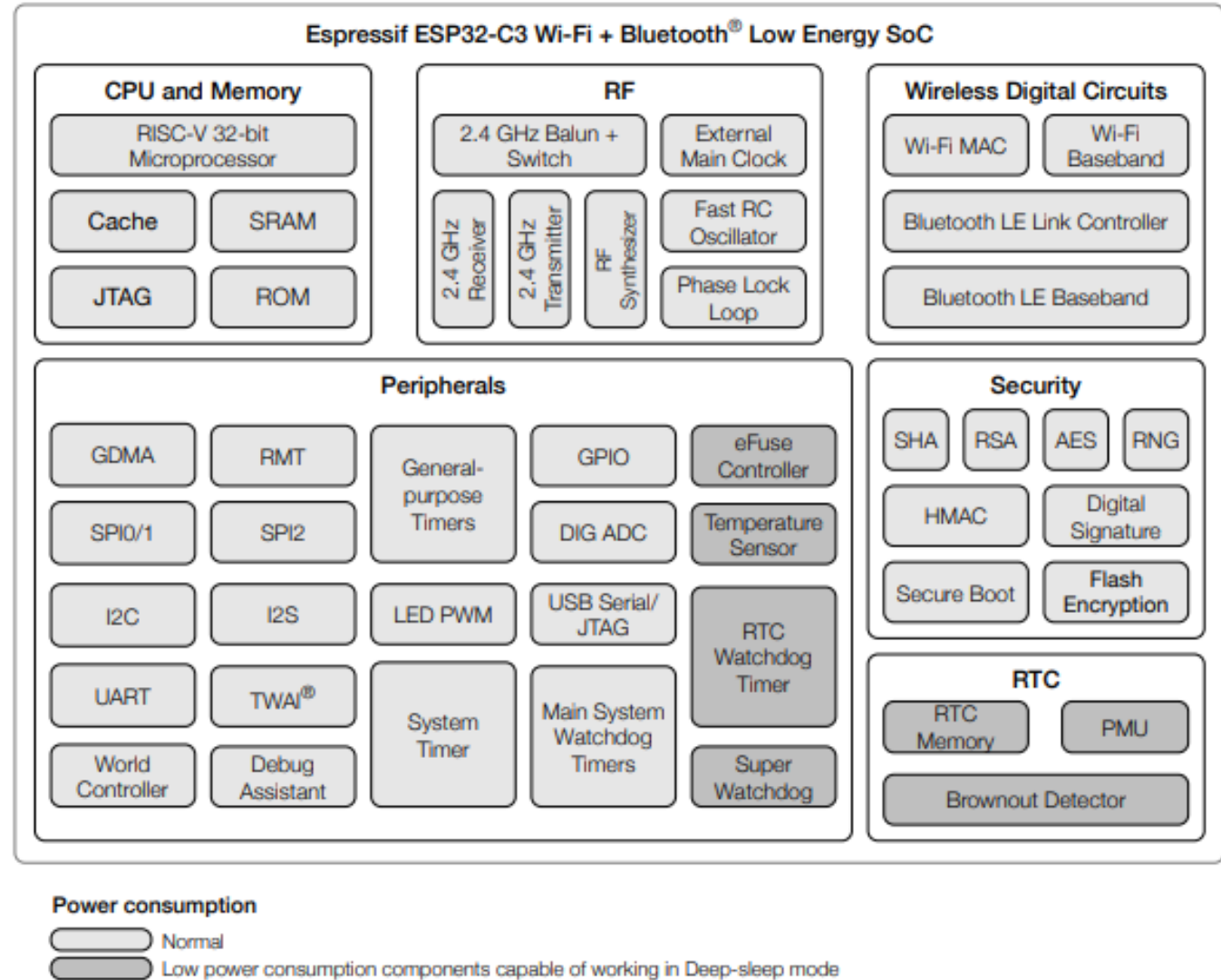
- A bit more complex
- Run various checks during your loop
- Only feed WTD if all checks are good
- Make sure this is what you want!
 - Do you really want to reboot if any of these checks are not ok?

```
1  RTC_DATA_ATTR int wtd_state = 0;
2
3
4  loop {
5      wtd_state = 0;
6      status = read_sensor1();    // returns 1 if OK
7      wtd_state |= status << 0;
8
9      status = read_sensor2();
10     wtd_state |= status << 1;
11
12     status = connect_to_wifi();
13     wtd_state |= status << 2;
14
15     ...
16
17     feed_wtd(wtd_state);
18 }
19
20
21 void feed_wtd(int &s) {
22     if (s == 0b111) {
23         esp_task_wdt_reset();
24         s = 0;
25     }
26 }
```

Watchdog timers

- What happens during sleep?
- According to block diagram, WDT should be active during light sleep
- But in my testing, it's not!
- Definitely not active during deep sleep

Please test!



ESP32-C3 Functional Block Diagram

Watchdog timers

- What to do after WTD timeout?



Watchdog timers

- What to do after WTD timeout?



Startup Settings

Press a number to choose from the options below:

Use number keys or functions keys F1-F9.

- 1) Enable debugging
- 2) Enable boot logging
- 3) Enable low-resolution video
- 4) Enable Safe Mode
- 5) Enable Safe Mode with Networking
- 6) Enable Safe Mode with Command Prompt
- 7) Disable driver signature enforcement
- 8) Disable early launch anti-malware protection
- 9) Disable automatic restart after failure

Watchdog timers

- What to do after WTD timeout?
- Definitely capture your reboot reason
 - `esp_reset_reason()`
 - This is an enum that will tell you whether reboot was due to WTD, deepsleep, brownout, power-on, etc.
- Then what?
- Some options
 - Act as if nothing bad happened
 - Go into “safe mode”
 - E.g., wait for firmware update

Bad FW design

- Beware of infinite loops!
 - Unless you are sure that this is what you want to do
 - For example, trigger WTD reset
- Avoid using delays (blocking code)
- Use timers and check

```
wire.begin(I2C_SDA, I2C_SCL),  
if (! sht4.begin()) {  
    Serial.println("Couldn't find SHT4x");  
    while (1) delay(1);  
}  
sht4.setPrecision(SHT4X HIGH PRECISION);
```

Bad FW design

- Beware of infinite loops!
 - Unless you are sure that this is what you want to do
 - For example, trigger WTD reset
- Avoid using delays (blocking code)
- Use timers and check

2+ sec delay each time you boot!

```
void setup() {
  Serial.begin(9600); //begin serial
  while (!Serial) {
    delay(100);
  }

  WiFi.begin(network, password);
  //if using channel/mac specification for crowded bands use the following:
  //WiFi.begin(network, password, channel, bssid);
  uint8_t count = 0; //count used for Wifi check times
  Serial.print("Attempting to connect to ");
  Serial.println(network);
  while (WiFi.status() != WL_CONNECTED && count < 6) { //can change this to
    delay(500);
    Serial.print(".");
    count++;
  }
  delay(2000); //acceptable since it is in the setup function.
  if (WiFi.isConnected()) { //if we connected then print our IP, Mac, and S
    Serial.println("CONNECTED!");
    delay(500);
  } else { //if we failed to connect just Try again.
    Serial.println("Failed to Connect :/  Going to restart");
    Serial.println(WiFi.status());
    ESP.restart(); // restart the ESP (proper way)
  }
  randomSeed(analogRead(A0)); //"seed" random number generator
}
```

Bad FW design

- Beware of infinite loops!
 - Unless you are sure that this is what you want to do
 - For example, trigger WTD reset
- Avoid using delays (blocking code)
- Use timers and check

```
if ((millis() - last_time) > GETTING_PERIOD) { // GETTING_PERIOD since
    sht4.getEvent(&humidity, &temp); // populate temp and humidity object
    float bat = read_analog_voltage(1, 10);
    bat = bat * (47+22.0)/22.0;
    Serial.println(bat);
    //formulate GET request...first line:
    sprintf(request_buffer, "POST http://efpi-10.mit.edu/tunnel_voldman/
    // sprintf(request_buffer, "GET http://numbersapi.com/%d/trivia HTTP
    strcat(request_buffer, "Host: efpi-10.mit.edu\r\n"); //add more to t
    strcat(request_buffer, "\r\n"); //add blank line!
    //submit to function that performs GET. It will return output using
    do_http_GET("efpi-10.mit.edu", request_buffer, response_buffer, OUT_
    last_time = millis(); //remember when this happened so we perform nex
}
}
```

Go to sleep!

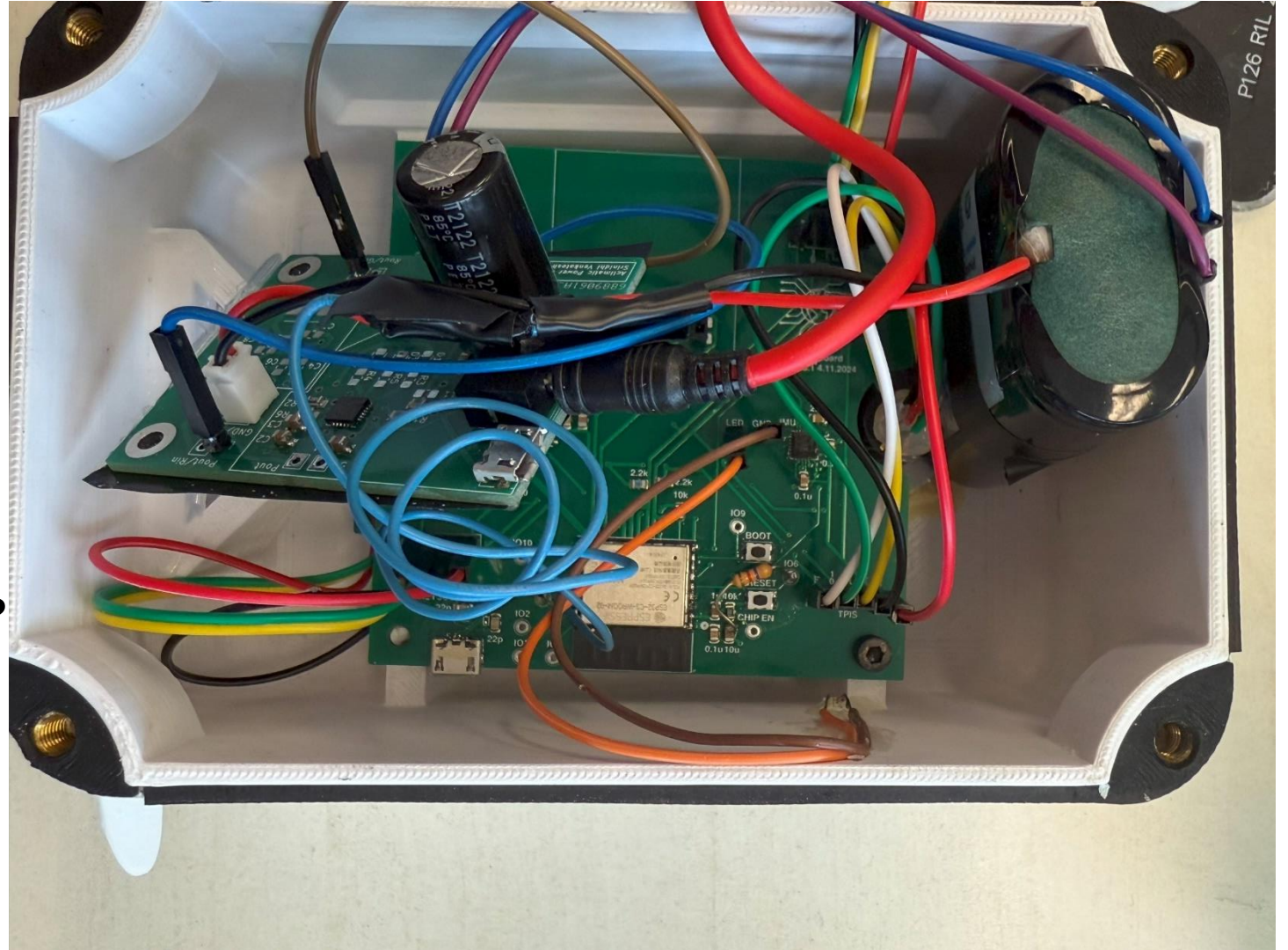
- ESP clock is 160 MHz → 6 ns/clock tic
- Modem-sleep: ??? to turn off/on radios
- Light sleep: ~320 us to enter, ~500 us to wake
- Deep sleep: ??? to enter, 200+ ms to wake
 - Apparently can be made faster via optimization, down to 10s of ms
- If you don't need radios, turn them off
- If you are doing nothing for more than a few ms, go to light sleep
- If you are doing nothing for more than a few seconds, go to deep sleep

Caveat: these values are from internet (not Espressif) so YMMV

Rebooting

- Sometimes you need to really power cycle

How to power cycle this?



Rebooting

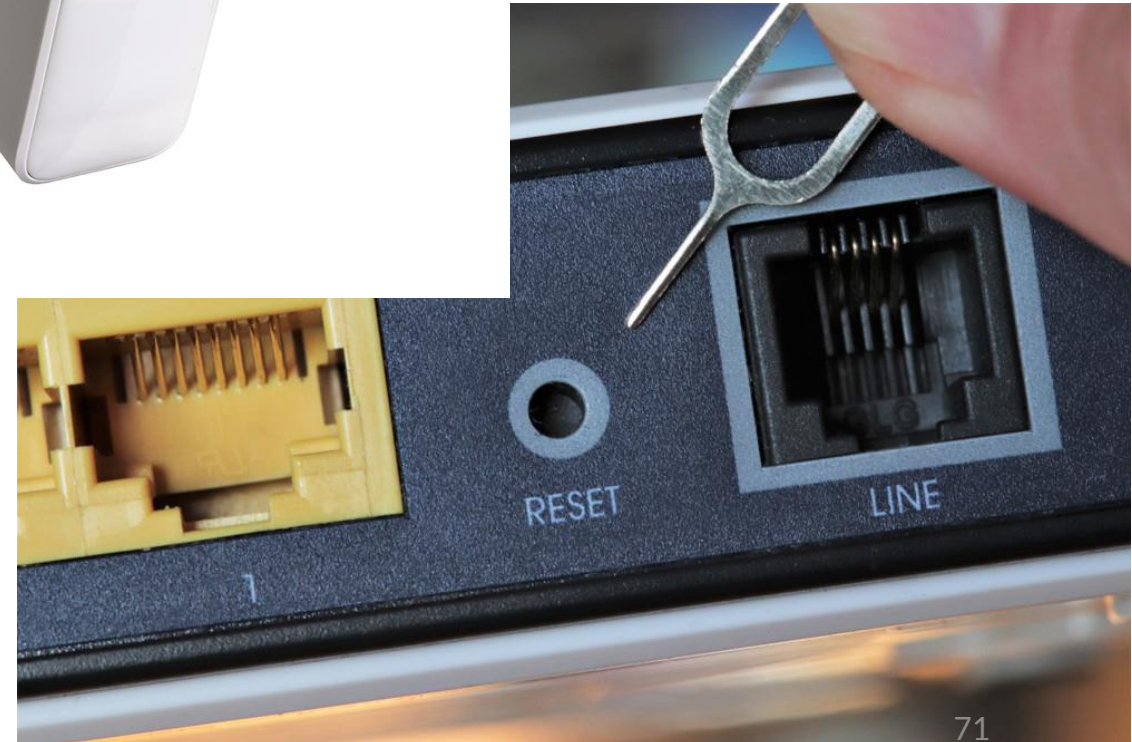
- Sometimes you need to really power cycle
- Put in a reset switch connected to MCU
- Put in a switch connected to battery
- Consider an external option



Maybe not this?



GPB527D202BB
\$3.09@100



System variables you may want to track

- Raw and processed data
 - RH, T, wind
 - Object presence/absence over time
 - Timestamped tripwire
 - Images
- Communications parameters
 - SSID, RSSI, etc.
 - Number of failed connection attempts
 - Number of failed POSTs, GETs
- Time How will you know this?

Early on in testing/validation, keep raw-er data

Beware of privacy issues!

System variables you may want to track

- Raw and processed data
 - RH, T, wind
 - Object presence/absence over time
 - Timestamped tripwire
 - Images
- Communications parameters
 - SSID, RSSI, etc.
 - Number of failed connection attempts
 - Number of failed POSTs, GETs
- Time How will you know this?

Early on in testing/validation, keep raw-er data
Beware of privacy issues!



System variables you may want to track

- Raw and processed data

- RH, T, wind
- Presence/absence over time
- Timestamped tripwire
- Images

Early on in testing/validation, keep raw-er data
Beware of privacy issues!

- Communications parameters

- SSID, RSSI, etc.
- Number of failed connection attempts

- Time

- Can get time from internet
 - From your server
 - SNTP: Simple Network Time Protocol
- Can keep time on ESP32
 - RTC timer works thru deep sleep
 - 150 kHz RC oscillator

Anywhere from 1 sec/min to 1 sec/day

System variables you may want to track

- System data
 - System temperature & RH
 - ESP32C3 has an internal temperature sensor (but I don't think its accessible via Arduino framework)
 - Last reset reason
 - System memory utilization
 - Memory leak?
`heap_caps_get_free_size(MALLOC_CAP_DEFAULT) ;`

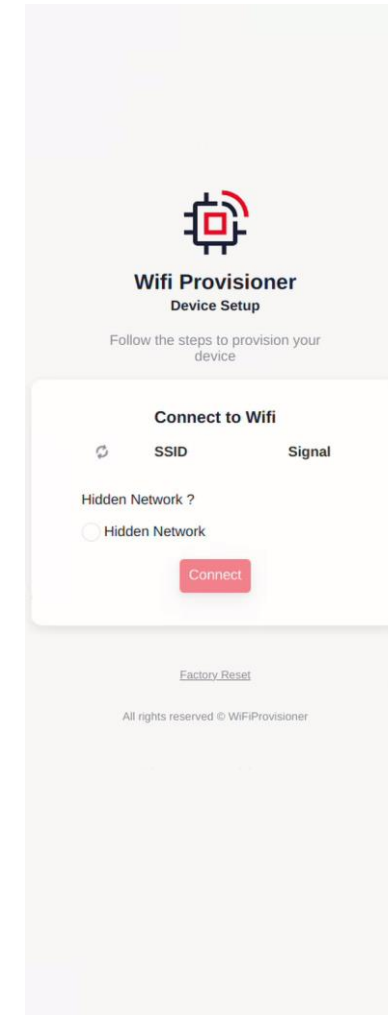
System variables you may want to track

- System data
 - System temperature & RH
 - ESP32C3 has an internal temperature sensor (but I don't think its accessible via Arduino framework)
 - System memory utilization
 - Memory leak?
`heap_caps_get_free_size(MALLOC_CAP_DEFAULT) ;`
- Battery state
 - In lab01 we measured battery voltage
 - Even better is to measure SOC
 - MAX17048 is a good part for this
 - I2C, one external passive
 - If you are not using LiPo cells, you might need a different fuel gauge!

WiFi provisioning

- If using WiFi, can hardcode credentials
 - This can get annoying if you are deploying in locations with different SSIDs
- Or can put in a set of SSIDs and connect to strongest one that is available
- Or can add code to send credentials to ESP32 over BLE or WiFi
 - Various tutorials exist online
 - <https://randomnerdtutorials.com/esp32-wi-fi-provisioning-ble-arduino/>

```
//wifi network credentials for 6.08 Lab (this is a decent 2.4 GHz network)  
// char network[] = "608_24G";  
// char password[] = "608g2020";
```

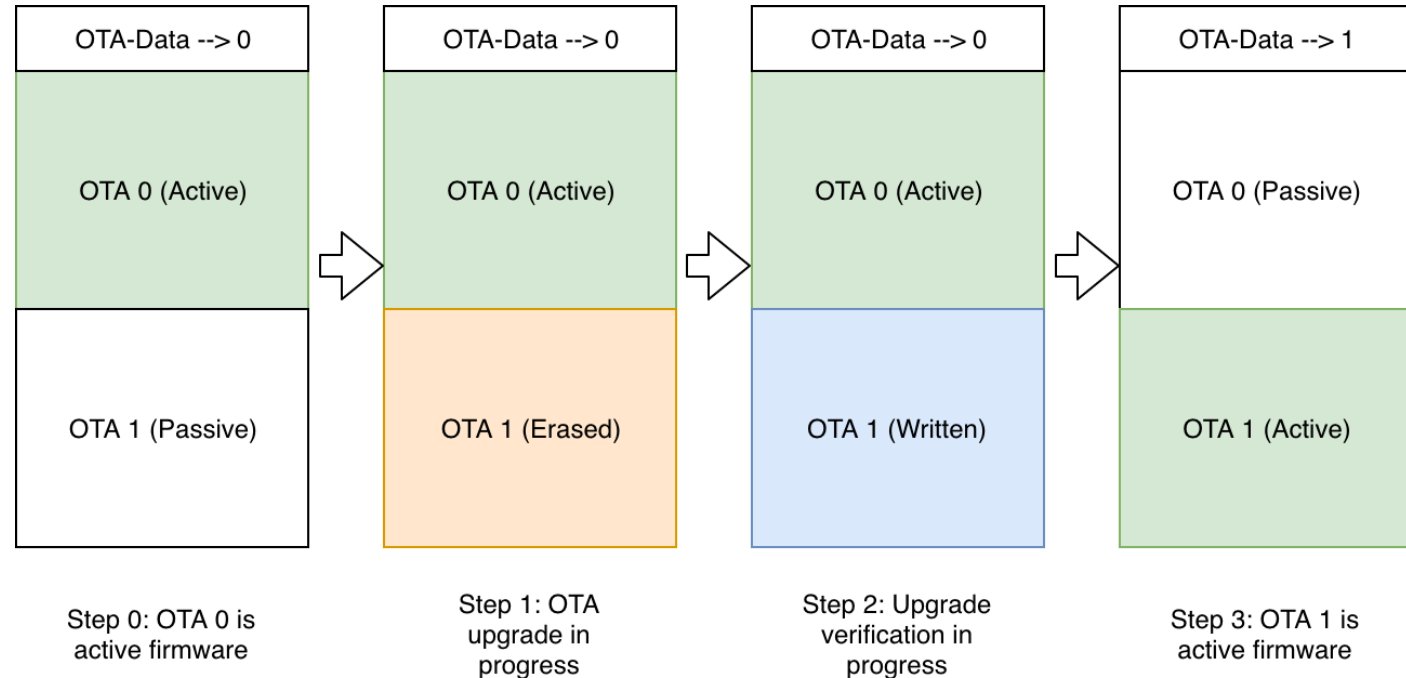


OTA updates

- It's *possible* that your FW will not be perfect
 - It might have bugs
 - It might need new features
 - It might have a testing variant and a release variant
- ESP32 can do over-the-air (OTA) FW updates

OTA updates

- It's *possible* that your FW will not be perfect
 - It might have bugs
 - It might need new features
 - It might have a testing variant and a release variant
- ESP32 can do over-the-air (OTA) FW updates



OTA updates

- It's *possible* that your FW will not be perfect
 - It might have bugs
 - It might need new features
 - It might have a testing variant and a release variant
- ESP32 can do over-the-air (OTA) FW updates
- Various libraries exist to do this
 - Last year used: esp32FOTA
 - Others exist
- Store program bin on server
 - With associated JSON telling FW version
- On ESP32
 - Go to server and get JSON
 - Check version against current
 - If update needed, pull new bin
 - Place into new OTA partition
 - Reboot and run new partition FW

OTA updates

- It's *possible* that your FW will not be perfect
 - It might have bugs
 - It might need new features
 - It might have a testing variant and a release variant
- ESP32 can do over-the-air (OTA) FW updates
- Various libraries exist to do this
 - Last year used: esp32FOTA
 - Others exist
- They also got it to work with cellular

LoRa? Doubt it...