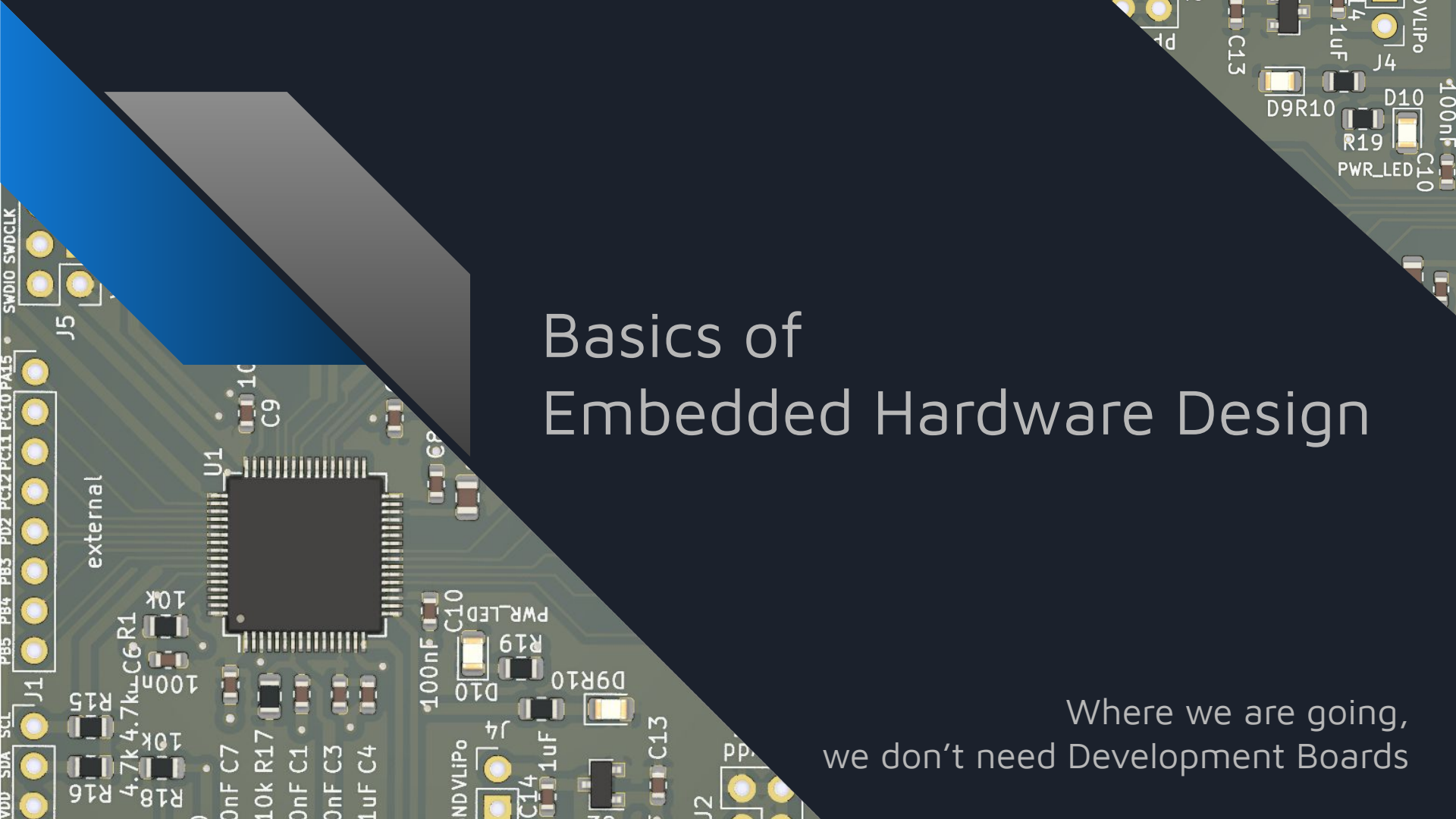


Basics of Embedded Hardware Design

Where we are going,
we don’t need Development Boards

Arduino UNO anyone?

Fueled the maker movement

Abstraction in software and hardware

No fiddling with datasheets or SMD soldering

BUT: cost of abstraction!

Sounds like a lot of work.

Dev boards are awesome, but:

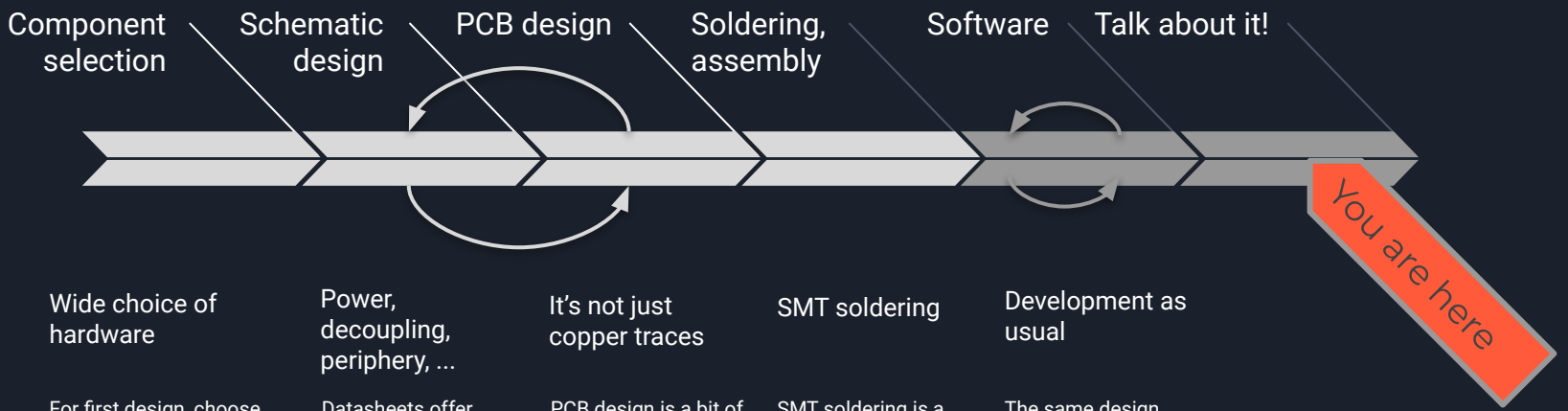
- Sometimes large, bloated
- May not offer desired functionality
- May support unneeded functionality
- Some processors don’t have dev boards

Why even do this?

Custom solutions:

- Small and light
- Can be very specialized
- Seamless integration with other systems
- Custom footprint
- Fun and challenging
- Career aspects

Waterfall Model (it is Hardware after all)



Wide choice of hardware

For first design, choose processor where schematic is publicly available (for example, from ST NUCLEO board)

Power, decoupling, periphery, ...

Datasheets offer valuable information. Mistakes here can be expensive

It's not just copper traces

PCB design is a bit of an art

SMT soldering

SMT soldering is a bit of a craft

Development as usual

The same design criteria as for normal software apply

Processor Choice

Some criteria:

- Which communication protocols are needed?
- Will there be many floating point operations? (FPU)
- Are there limits to energy consumption?
- Price, ease of soldering and flashing
- Programming environment, community/support, Toolchain

For first design: Choose processor where schematic is available (Like one from a Nucleo or Discovery board)

Processor support circuitry

General components many processors require to operate

- Regulated, protected power supply
- Bypass/decoupling capacitors for power
- Pull-up/Pull-down resistors for reset/boot/debug lines
- Debouncing capacitors for reset button etc.
- Crystal oscillator (may be optional)
- ADC reference voltage/ferrite bead for analog decoupling
- Debug/Flash port (SWD, SPI, JTAG, ...)

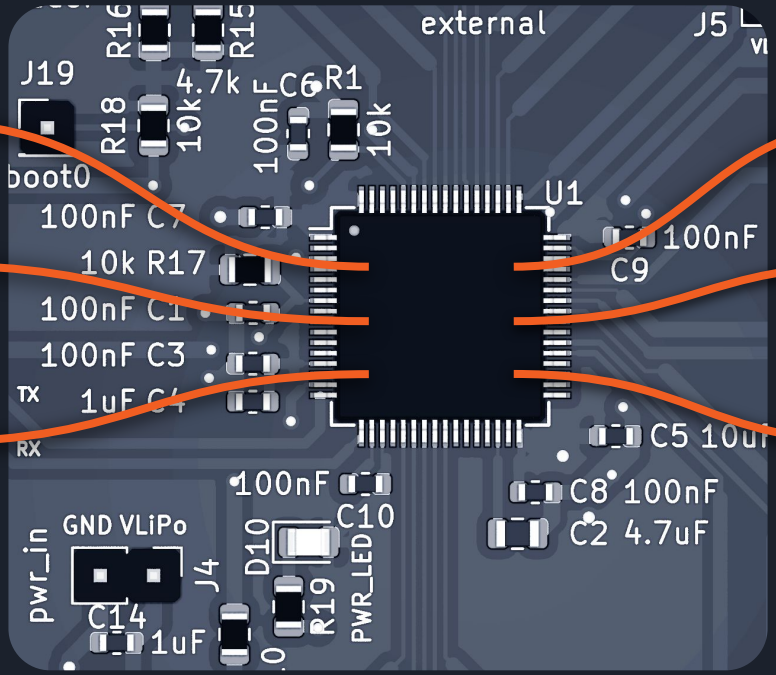
Specific Processor Overview:

STM32F401RDT6

84 MHz ARM Cortex-M4

Single Precision Floating Point Unit

96kB SRAM
384kB Flash



3.3V Voltage Level

“High Speed” internal oscillator

Down to 10uA sleep current

ST overview page with datasheet etc: https://www.st.com/en/microcontrollers/stm32f401rd.html

Processor support circuitry

STM32 specifics

ST Microelectronics Application Note 2586:
Getting started with STM32F4xxxx MCU hardware development

- Only 50 pages!
- Notes about power supply, PCB routing, reference design schematic, oscillator options, reset/boot, debugging, ...
- Different for each specific processor! No way around it.

STM32F401RD Datasheet:

- 135 Pages!
- Mostly less hardware-related information
- Valuable specific information (pin combinations, power supply, ...)

STM32F401 Reference Manual:

- 832 Pages

Specific Processor Pins
NRST and BOOT0/1

- NRST Pin: Issue a RESET interrupt when line is LOW.
- Interrupt (in ARM) is a form of Exception. Exceptions have handlers:
  - `Map<ExceptionID, Handler>` (actually just an array at beginning of flash memory)

- Reset Exception can come from various sources indicated in the RCC_CSR register:
  - `LPWRRSTF`: Low Power Reset Flag
  - `IWDGRSTF`: Independent watchdog reset flag
  - `SFTRSTF`: Software reset flag
  - `PINRSTF`: PIN reset flag - the only **external** one!

ST overview page with datasheet etc: https://www.st.com/en/microcontrollers/stm32f401rd.html

Specific Processor Pins
NRST and **BOOT0/1**

RESET exception handler: look at `BOOT0/1` pins and decide where to boot from

- If BOOT0 LOW: execute firmware
  If BOOT0 HIGH, BOOT1 LOW (if applicable): System memory
- Booting from external memory is also supported

- When booting from system memory:
  - Boot loader checks peripherals for new firmware and flashes it
  - Various protocols (SPI, UART, SWD, etc.) supported
  - SWD: just JTAG, but with only 2 pins

ST overview page with datasheet etc: https://www.st.com/en/microcontrollers/stm32f401rd.html

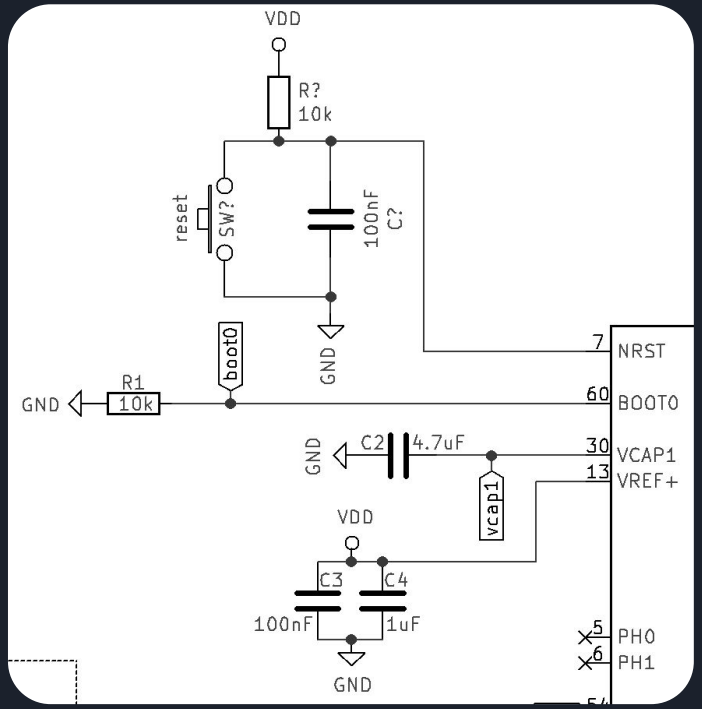
Specific Processor Pins
Schematic detail

NRST pulled high (“yes, no reset”)
- Cap prevents any spurious resets
- Also debounces button

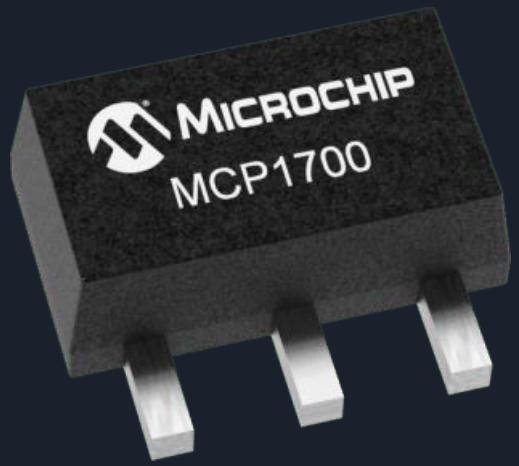
Boot pulled low
- But can be pulled high
  by other boot0 circuitry for flashing

VCAP1, VREF+, and others: according to datasheet

Bonus question: why use 2 capacitors for VREF+?



3.3V Power Supply: MCP1700



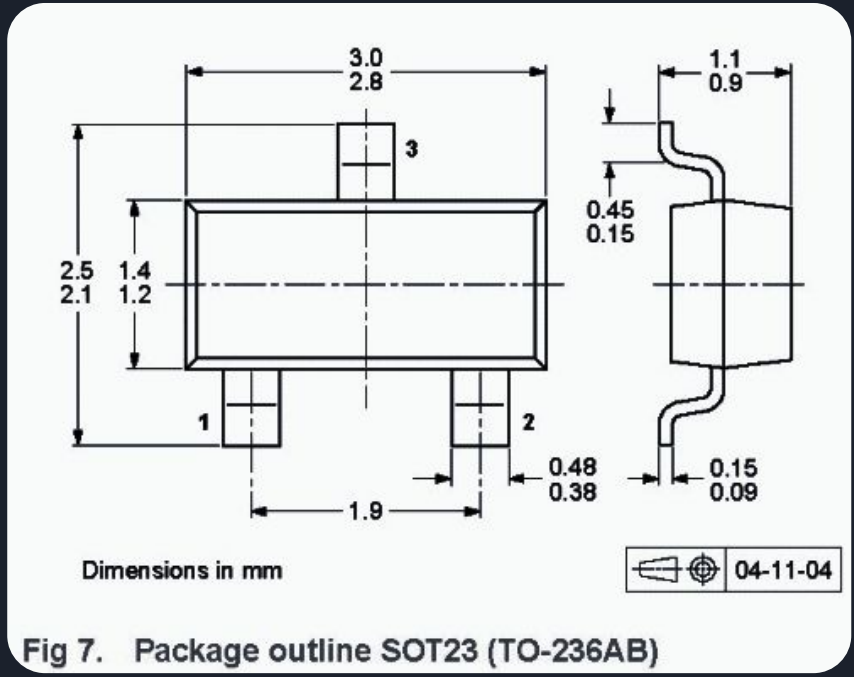
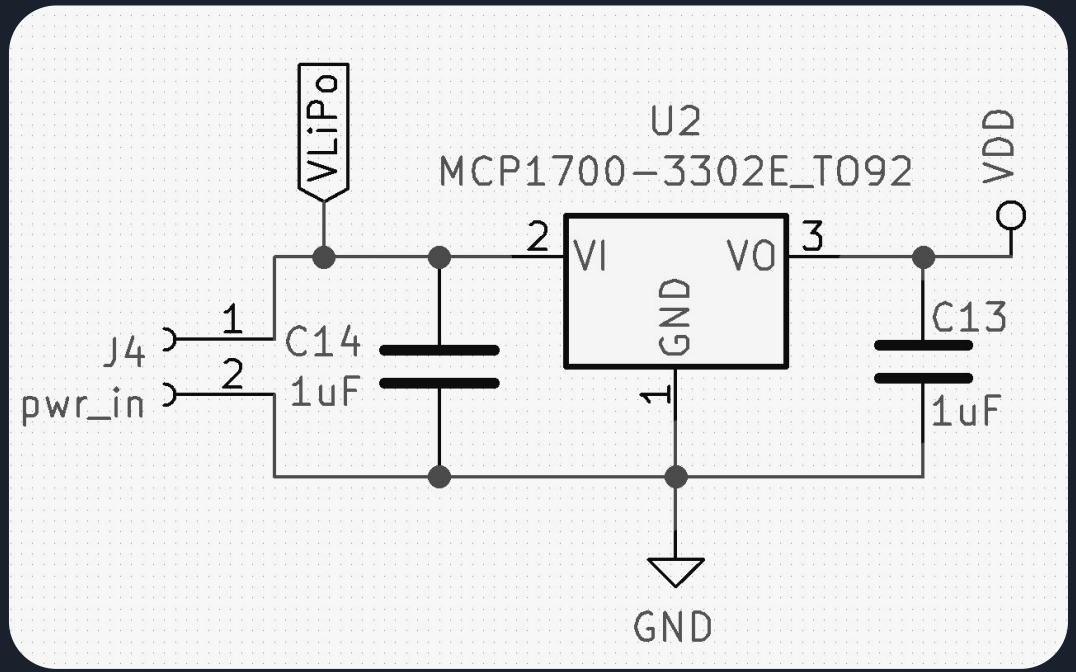


Fig 7. Package outline SOT23 (TO-236AB)

3.3V Power Supply: MCP1700

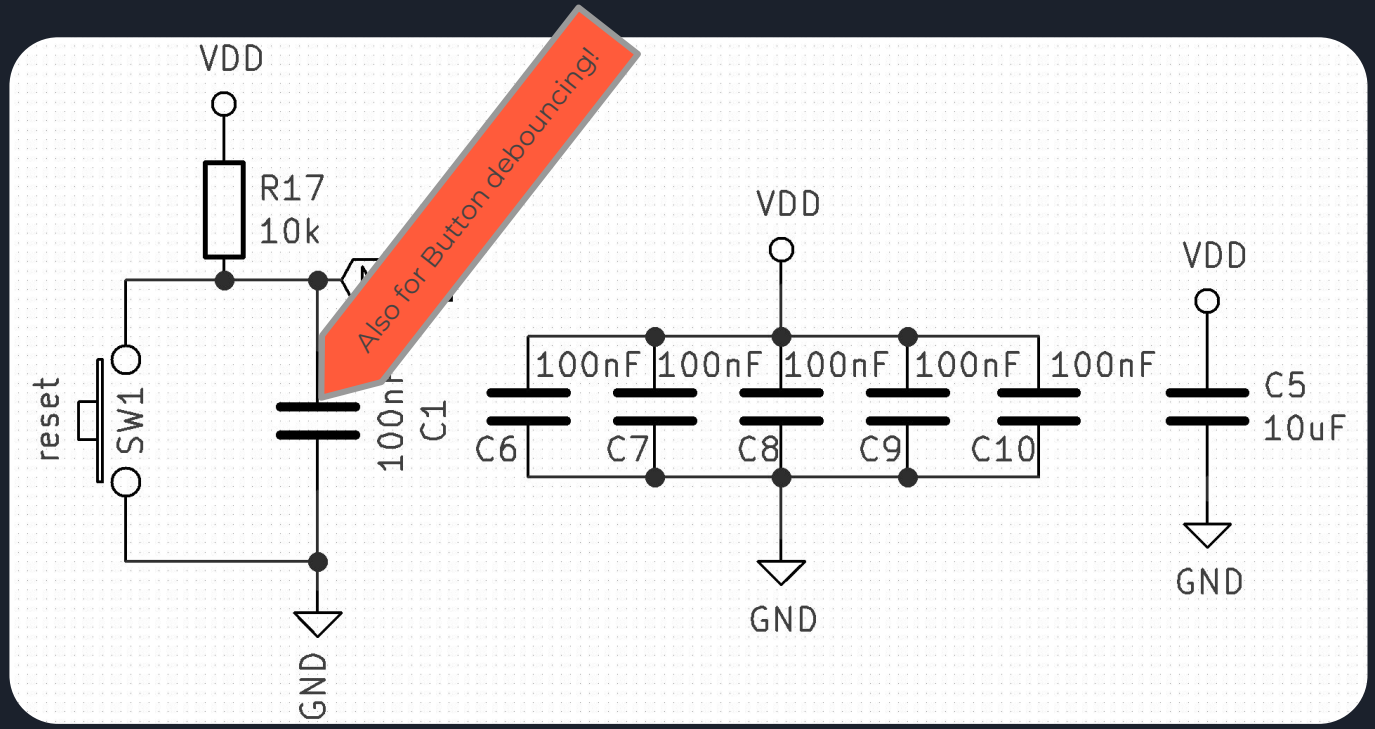
Capacitors for:
Brownout protection
Ripple filtering



Bypass/Decoupling Capacitors

100nF is enough for everyone?

One of them is not like the others!



Bypass/Decoupling Capacitors

100nF is enough for everyone?

Why decouple VDD from GND?

- Decoupling capacitors **oppose change in voltage!**
- (Parasitic) inductances **oppose change in current!**

- Capacitors compensate for parasitic resistance and inductance of traces and components
- Capacitors act as low pass filters by blocking high frequency ripple

- Those things matter more as frequency goes up!

"Fundamentals Friday" on Bypass Capacitors by Dave Jones of EEVBlog fame
Shiftedphase on bypass caps

Not just any capacitor

- Usually, ceramic (MLCC) or polycarbonate caps are fine
- The smaller, the better! (But harder to solder)
- Other capacitor types:
  - Electrolytic (large capacitance, polarized, large ESR/ESI)
  - Tantalum (medium capacitances, expensive, conflict mineral)
- The datasheet is your bible
- At really high frequencies, some capacitor types become inductive

”Mr. Carlson's Lab” on different capacitor types

Speaking of small

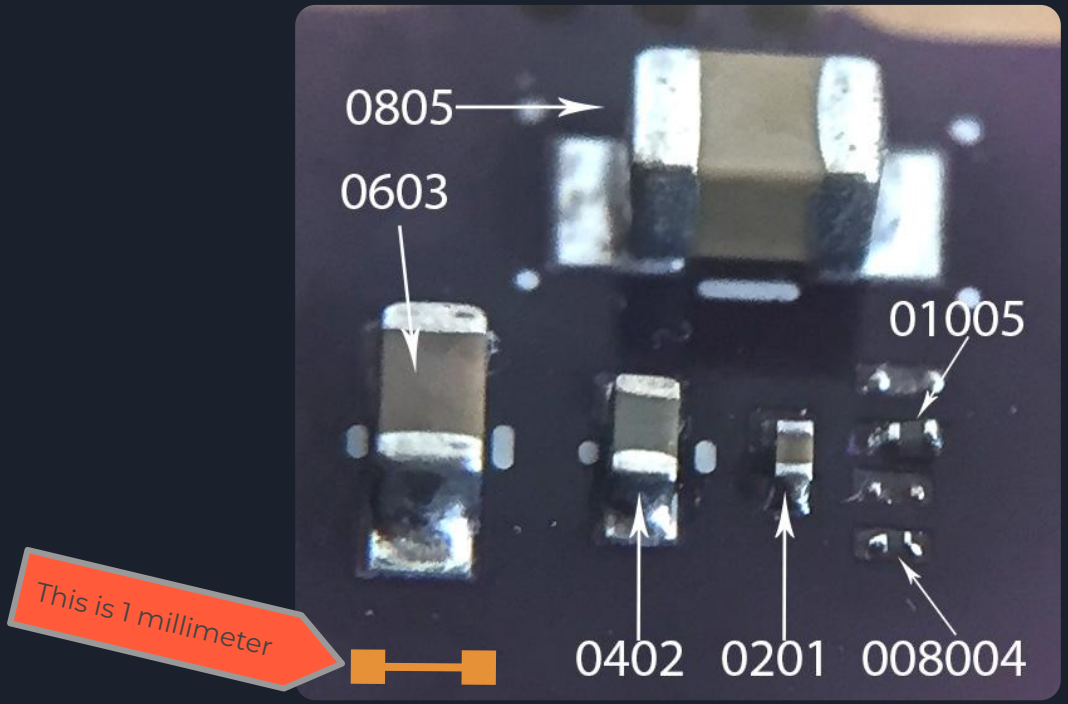
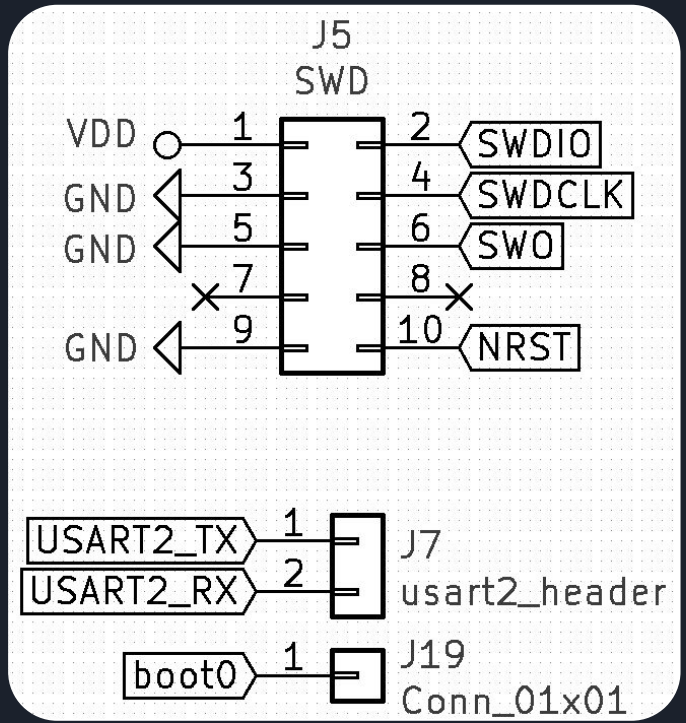


Image: Someone on the eevBlog forums soldered really really small caps

Standard SWD Debug/Flash Connector

Allows to:

- Flash programs onto the chip
- Debug them in real time
- Modify memory locations
- Standard: 1.27mm (0.05”) pin spacing
- Usually, used with SEGGER j-link or similar, ST-Link for STM32

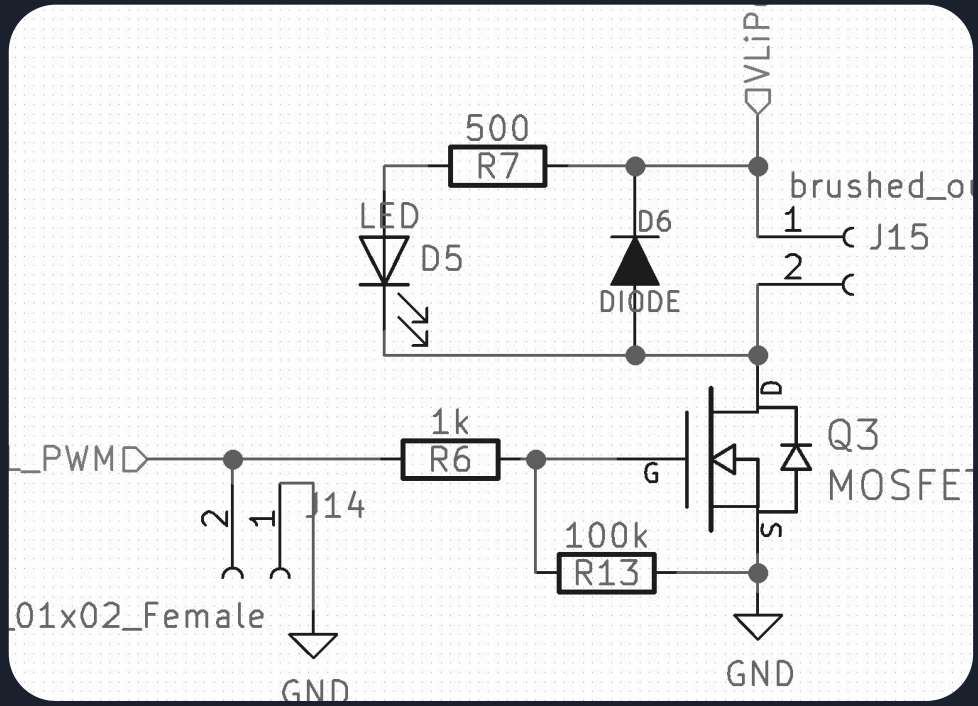


External Circuitry: MOSFET driver

IRLML6244TRPBF: fabulous SOT-23 MOSFET

IRLML6244TRPBF MOSFET:

- Drive larger loads around 5A
- Logic Level MOSFET
- REALLY cheap!
  50 cents europe, 6 cents asia
- Lead/Halogen free! (F variant)



IRLML6244TRPBF Datasheet

Schematic Intermezzo

KiCad `eeschema` subprogram

Only responsible for schematic - not for physical layout or component footprints

You may have to create custom symbols for specific components

`eeschema` produces a netlist file: text file that specifies which components are connected to which nets,
abstract graph representation of your schematic

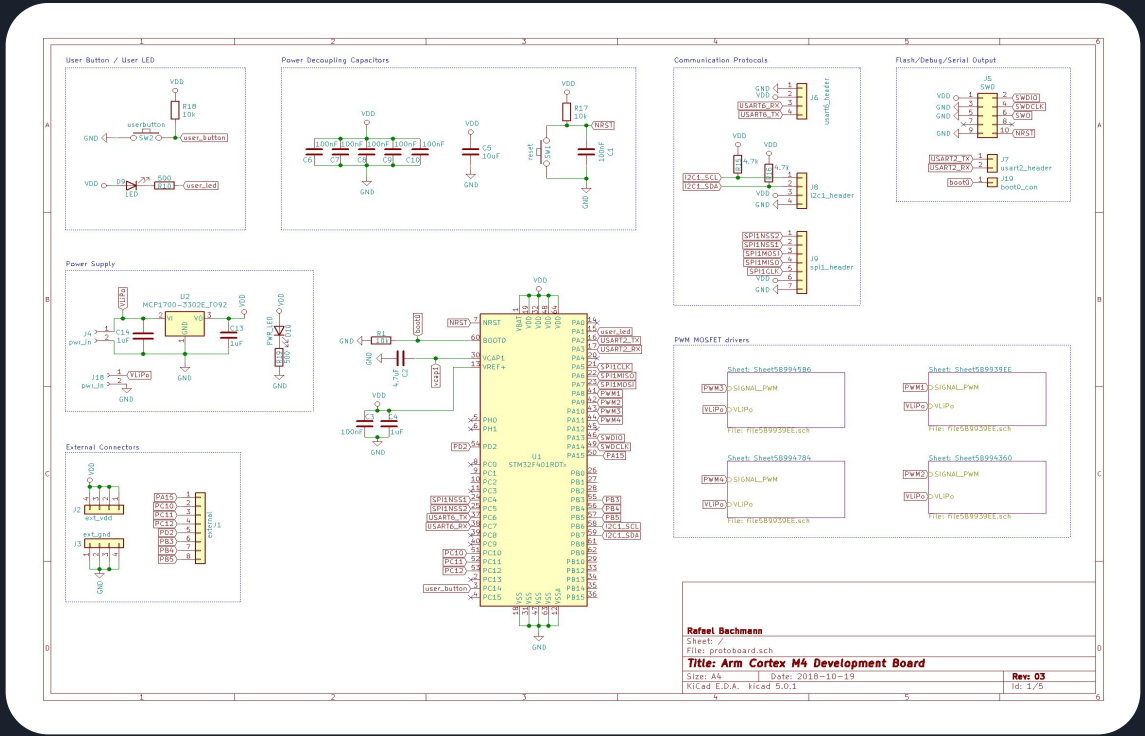
Treat schematic like code!

- Clear signal flow from left to right, if possible
- Localize modular functionality
- Naming things is important (netlist names)
- Use hierarchical schematics
- Double check all connections

  (get this right the first time to save money)

Treat schematic like code!

Not the best example, but at least it's realistic!



PCB Layouting

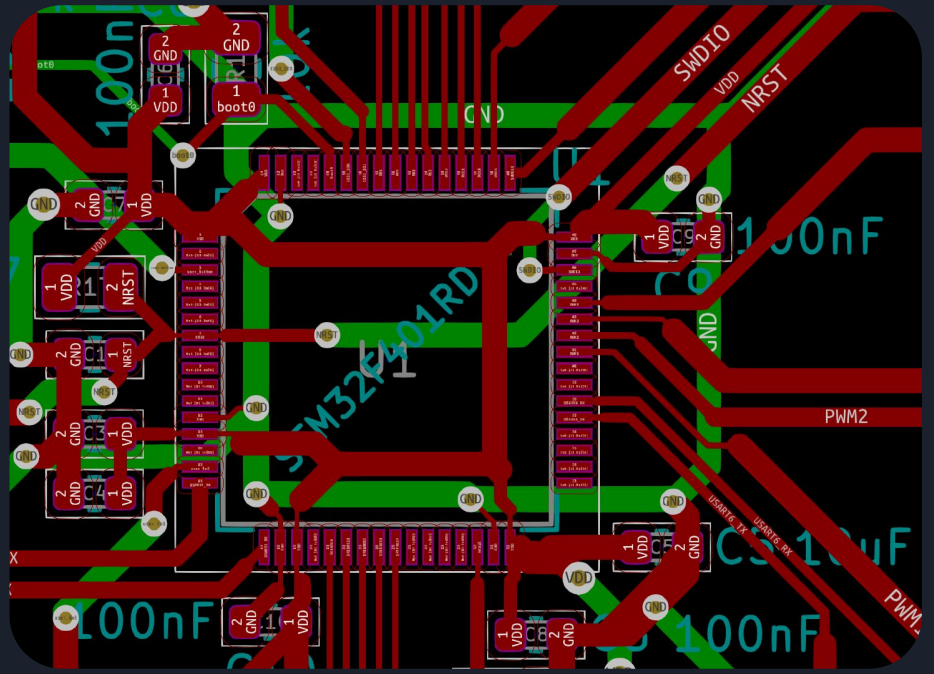
Use component footprints and netlist file to create a physical layout of the components on the board

Output: Files which describe the desired circuit board to a manufacturer

It's just a board with copper traces on both sides which may be connected by vias, holes for components, and some printed text.

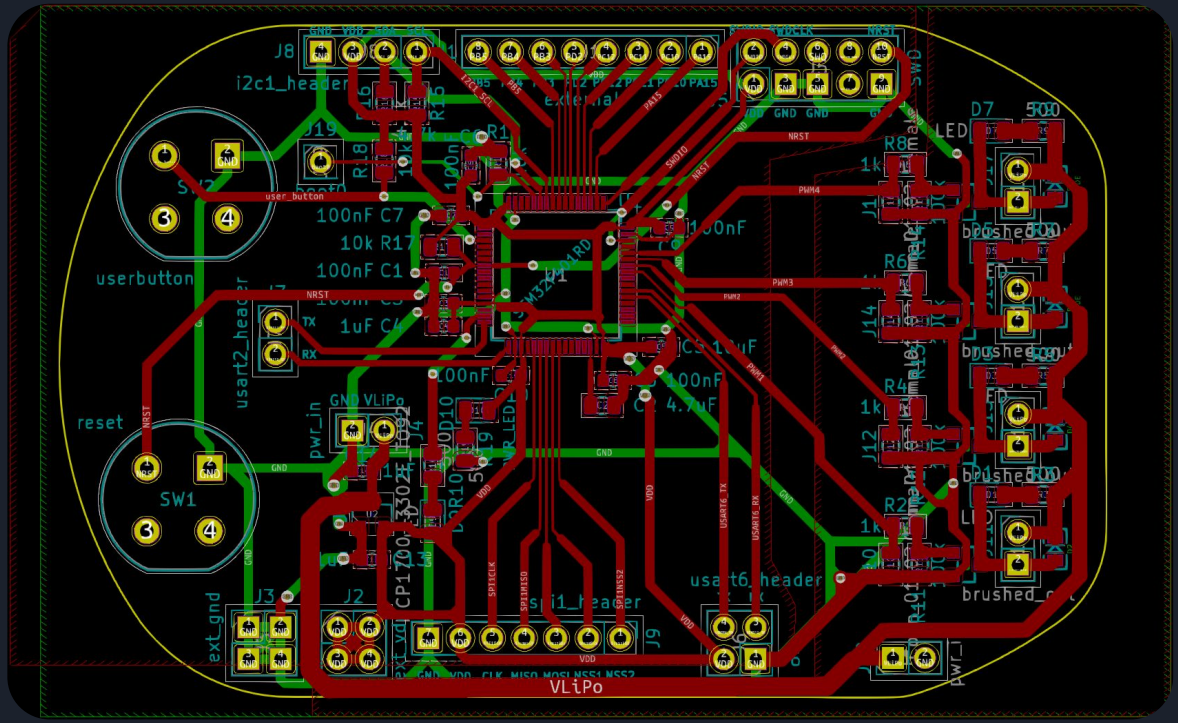
PCB Layout

Green: trace on bottom layer
Red: trace on top layer
Gray dot (Via): connection between top and bottom layer

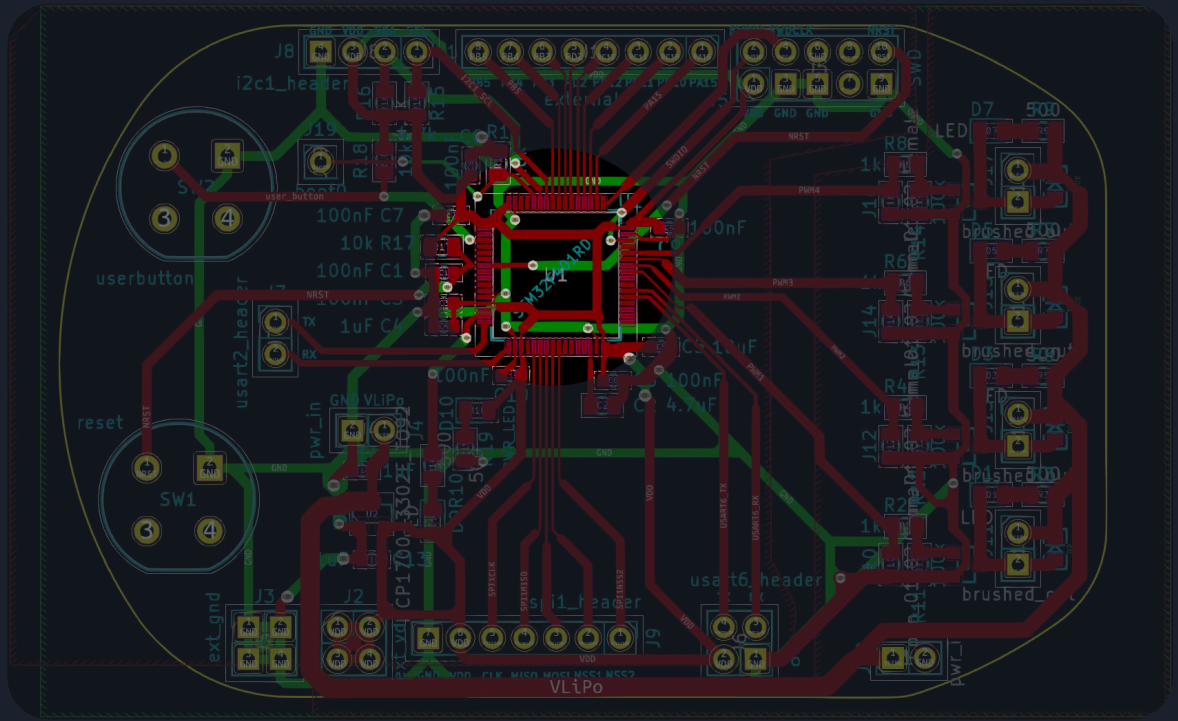


"Fundamentals Friday" on Bypass Capacitors by Dave Jones from EEVBlog fame

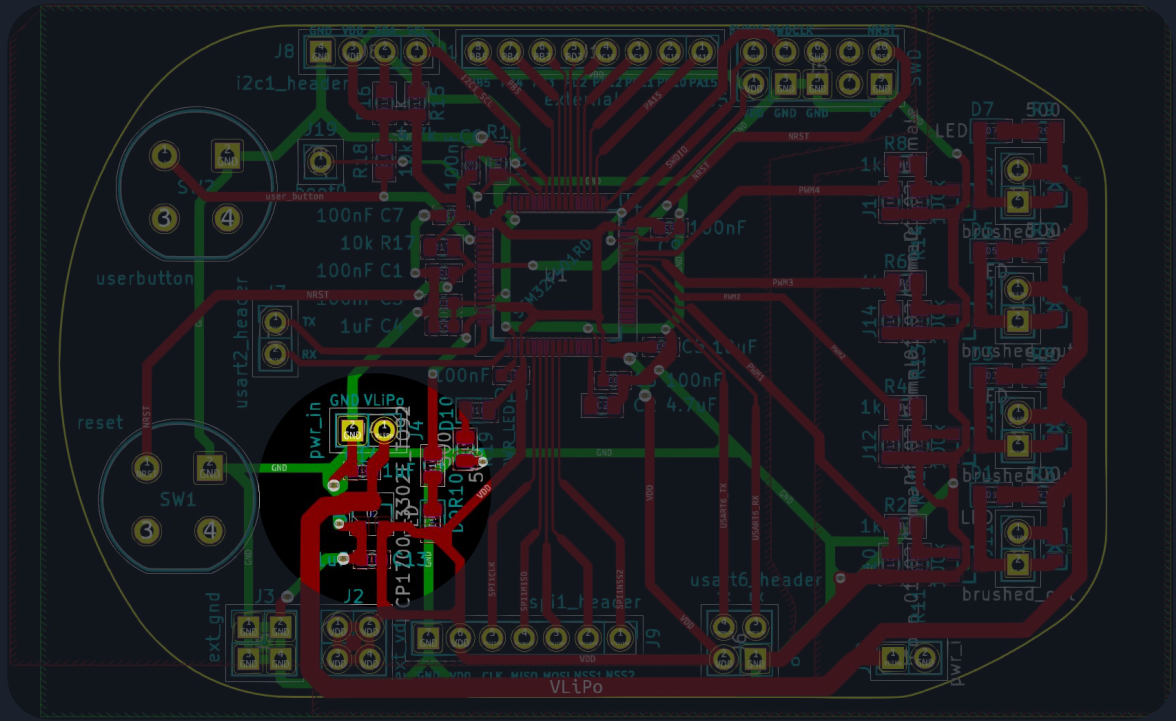
Complete Layout



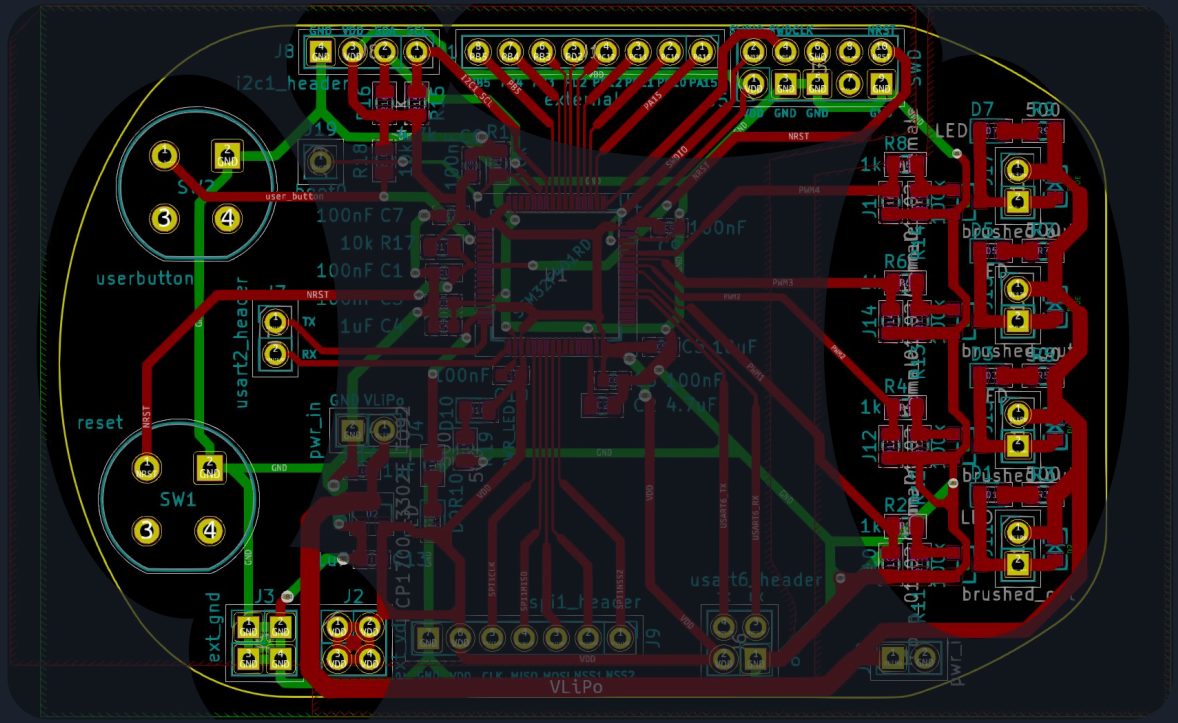
Central: CPU


i2c1_header
userbutton
reset
SW1
usart2_header
pwr_in
spi1_header
usart6_header
ext_gnd
LED
brushed
VLiPo

Power Circuitry



External Circuitry



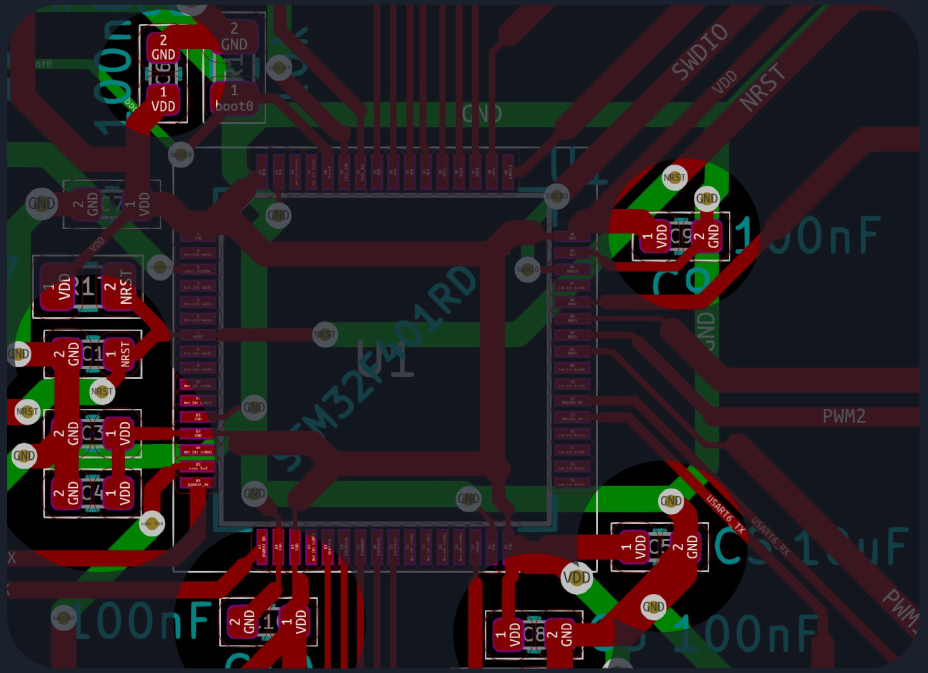
Bypass/Decoupling Capacitors

100nF is enough for everyone?

100nF ceramic
between GND, VDD pin pairs

Tantalum (high capacitance)

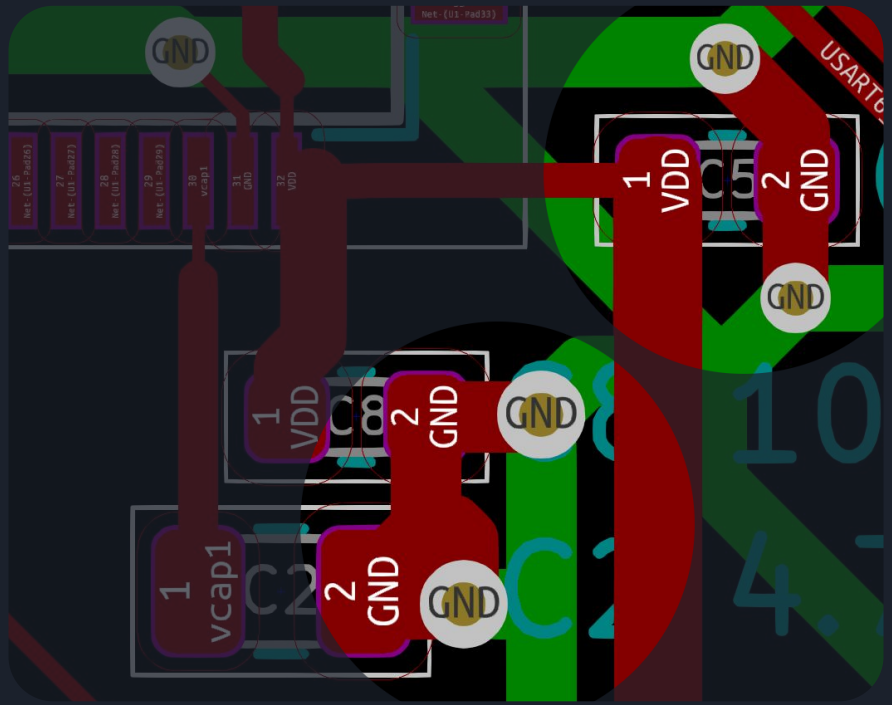
As close to the processor as possible!



"Fundamentals Friday" on Bypass Capacitors by Dave Jones from EEVBlog fame

Bypass/Decoupling Capacitors

100nF is enough for everyone?



Vias to GND bottom layer
Recommended in datasheets!

"Fundamentals Friday" on Bypass Capacitors by Dave Jones from EEVBlog fame

PCB Layouting Concerns

- All traces have resistance, inductance, capacitance
  - Traces should be as short as possible, especially if they carry high frequency signals or high current
  - MYTH: sharp corners and right angles should be avoided
  - VDD traces preferably on upper side of PCB, GND on lower

- Decoupling capacitors should be as physically close as possible to the thing they are supposed to protect

Who's Afraid of the Big Bad Bend?

PCB Layout Intermezzo

KiCad `pcbnew` subprogram

Only responsible for physical layout, ensures that implementation (layout) adheres to specification (schematic)

You may have to create custom footprints for specific components

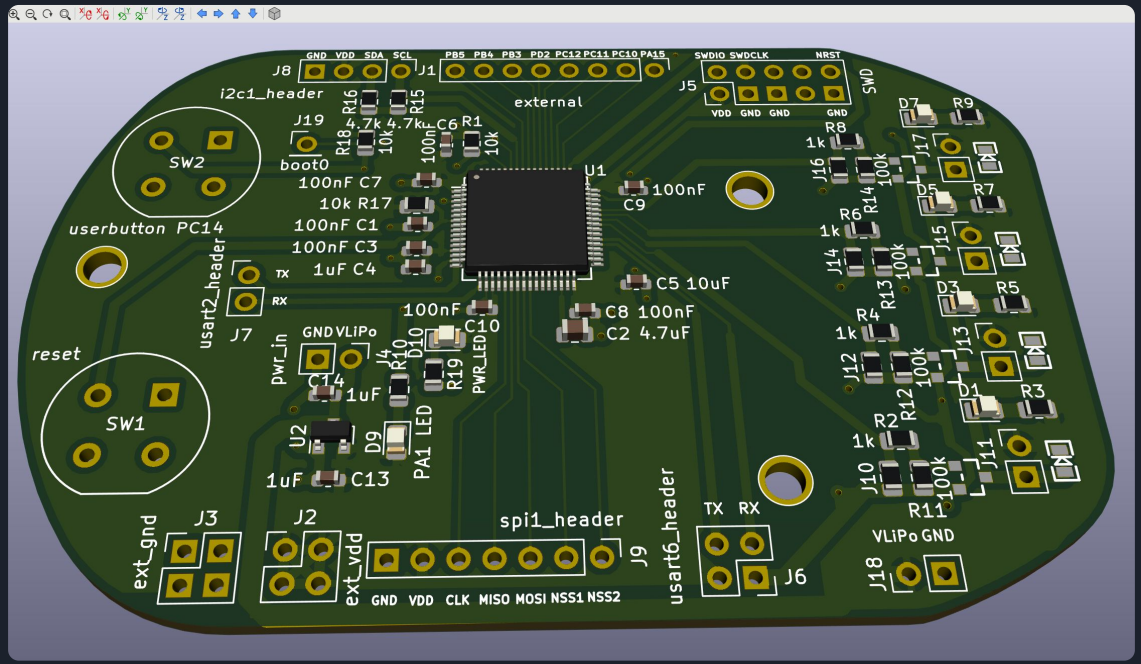
`pcbnew` produces Gerber files: text files that specify where the manufacturer should drill holes, place traces and vias, print solder mask and silkscreen, ...

PCB Layout: use the 3D View!

Available for each component separately, too

Extremely handy to quickly check results

You can create 3D models of your components with external tools like FreeCAD

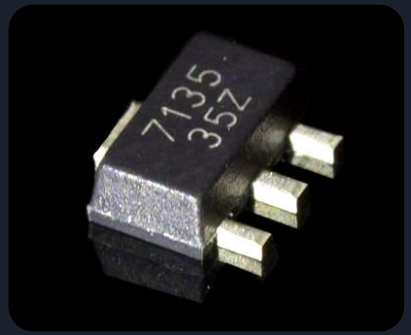


Example Circuit

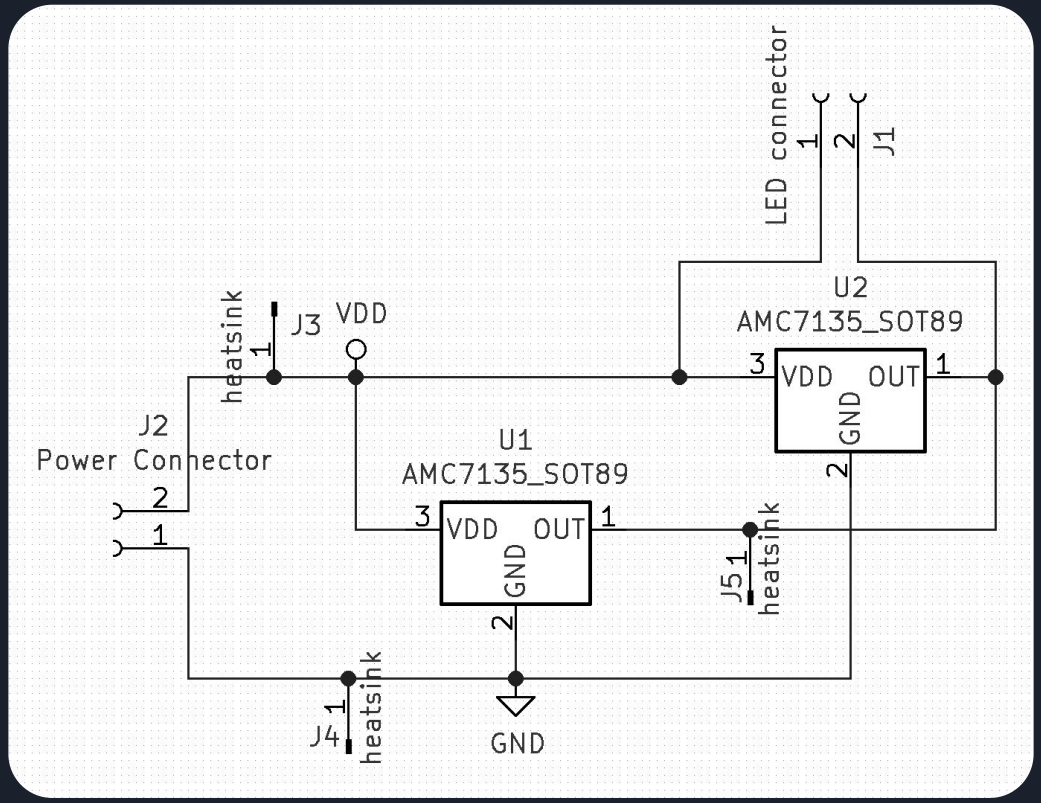
2-Cascade of AMC7135 350mA constant current LED driver

Goal: Battery-driven 700mA constant current source breakout board to drive 3W power LED

AMC7135 supply voltage range: 2.7 - 6V (perfect for 1S-LiPo battery)



Example Circuit

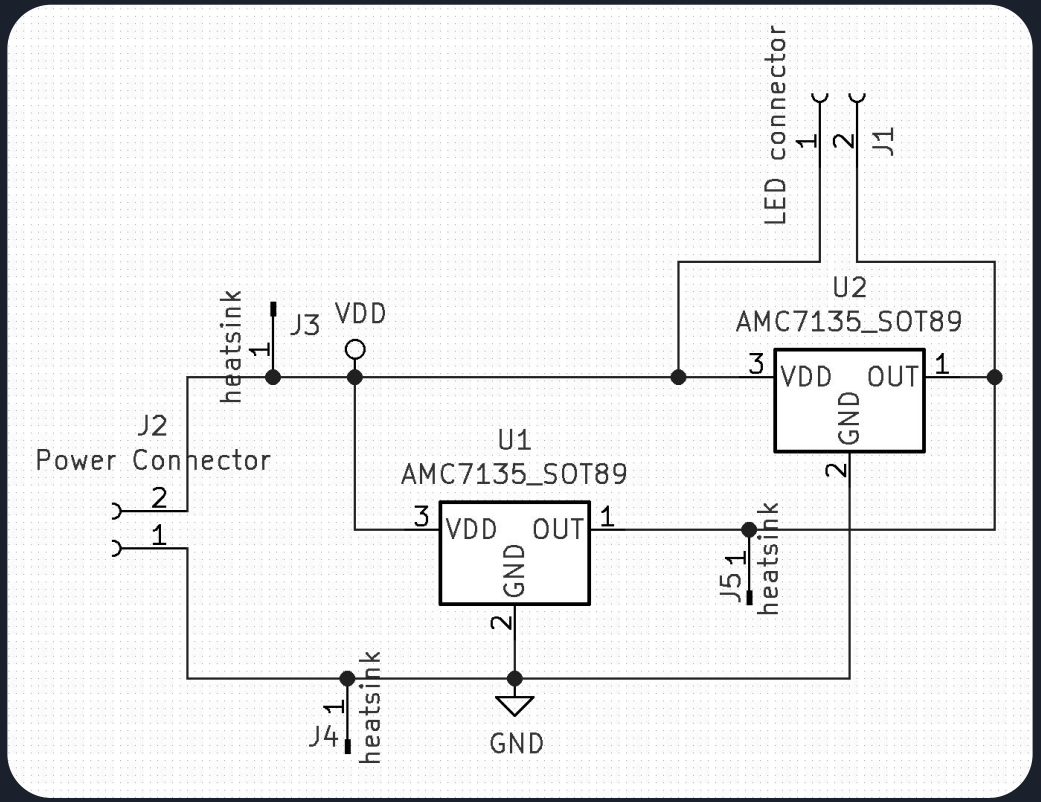


Example Circuit

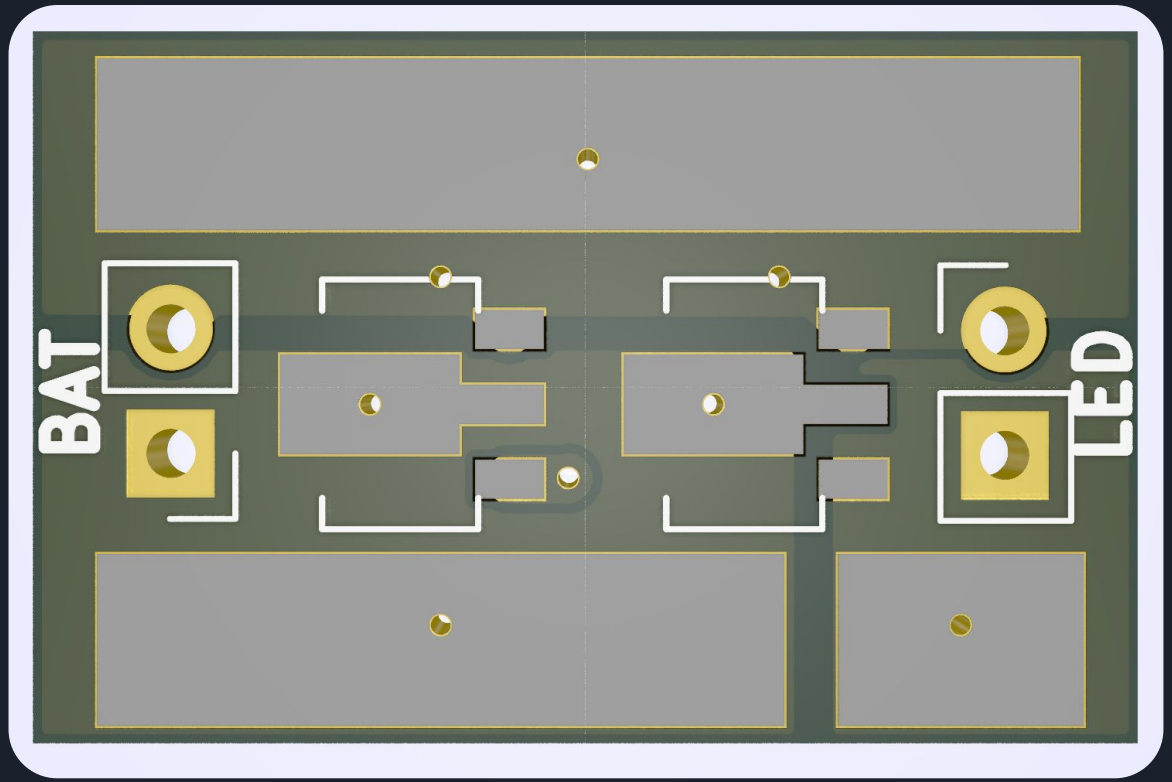
Let's add a bypass capacitor!

The datasheet recommends one

Necessary?
Depends on power supply



Example PCB



KiCad Demo time!

The KiCad workflow

KiCad has subprograms with separate file formats for:

- Symbol editing
- Schematic editing
- Footprint editing
- PCB layout editing
- Mapping of Symbols -> Footprints
- Gerber file Viewer
- Several smaller tools

EAGLE and other tools have one file format and one program to edit it.
KiCad is frequently ridiculed for it’s unusual workflow.

The KiCad workflow

- Multiple perspectives on the same data
- All files are text based! (version control)
- Enables cooperative work on different aspects of the same circuit (“Schematic designer + PCB monkey”)
- Enables the generation of several different PCB variants from the same schematic, with different footprints and board shape/layout

SMD Soldering

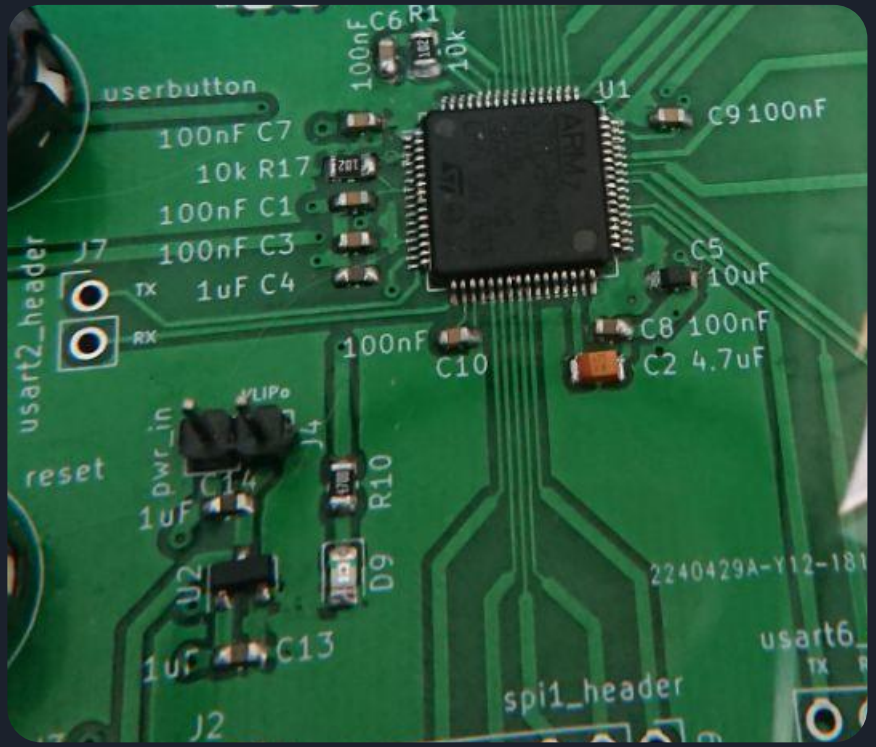
Can be pretty messy ->

Always check for short circuits and solder bridges

Don’t overheat!

Use solvent to wipe off flux

A cheap smartphone microscope can help!



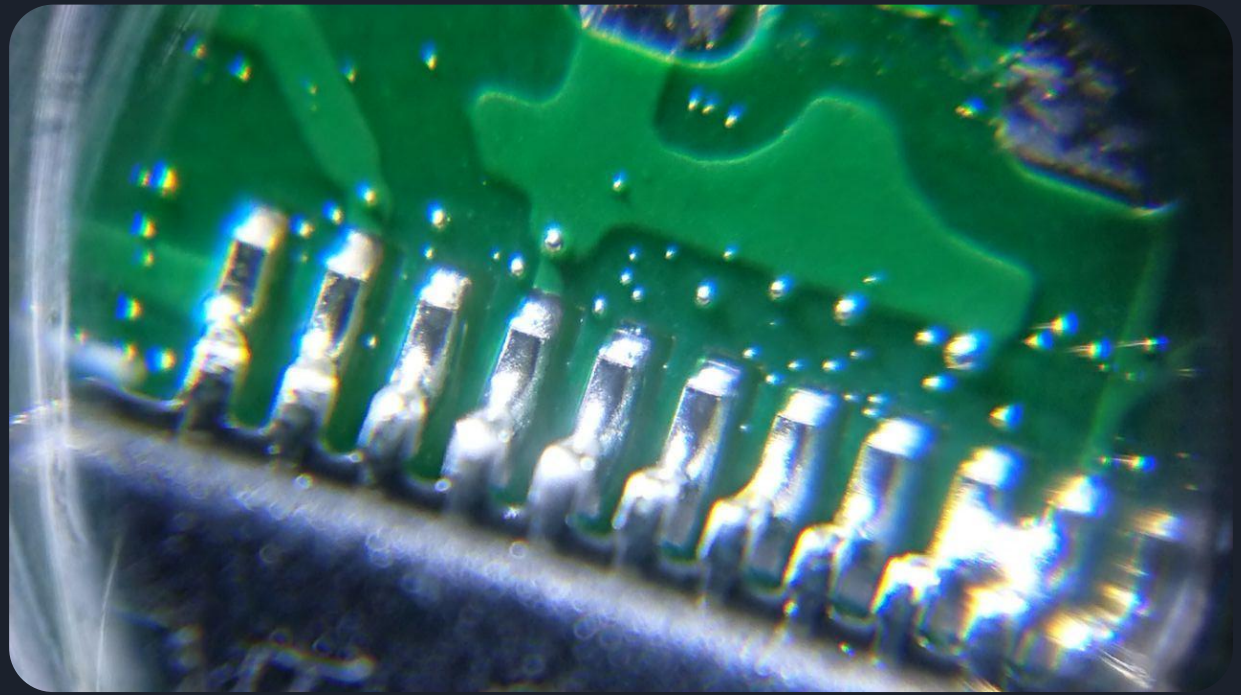
Solder Paste

Looks like mud from the Inn river

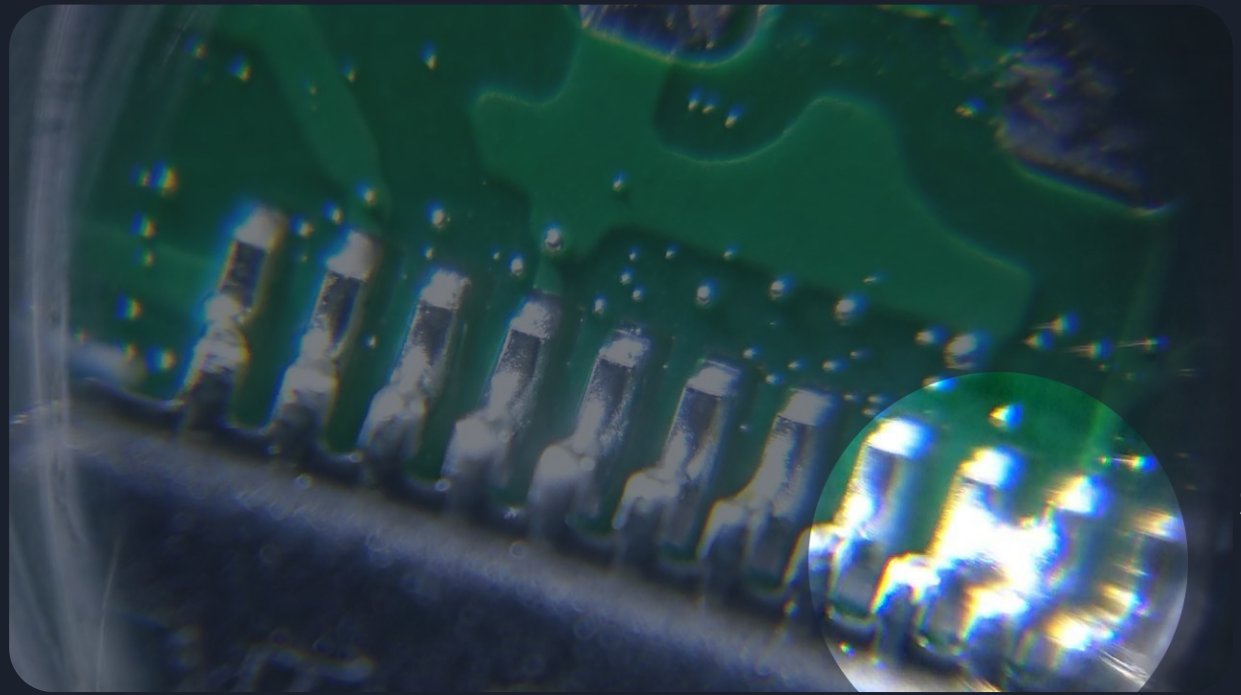
Turns into liquid tin when heated up!

Makes soldering with a heat gun very simple

SMD Soldering

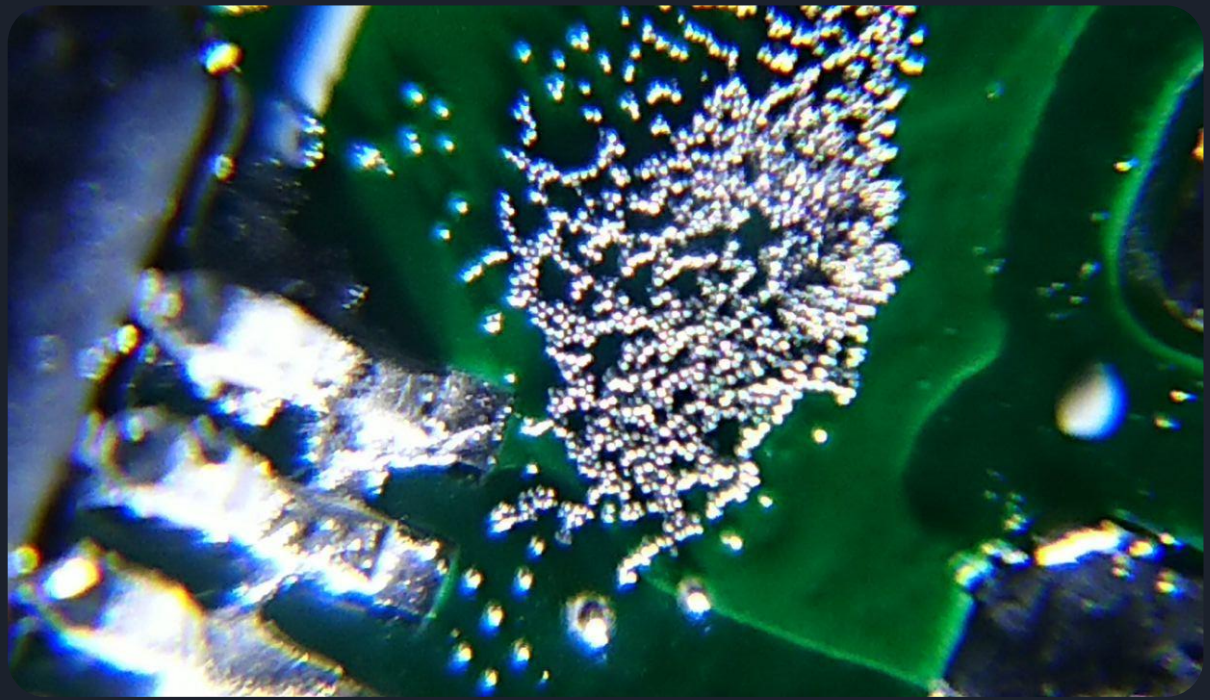


SMD Soldering

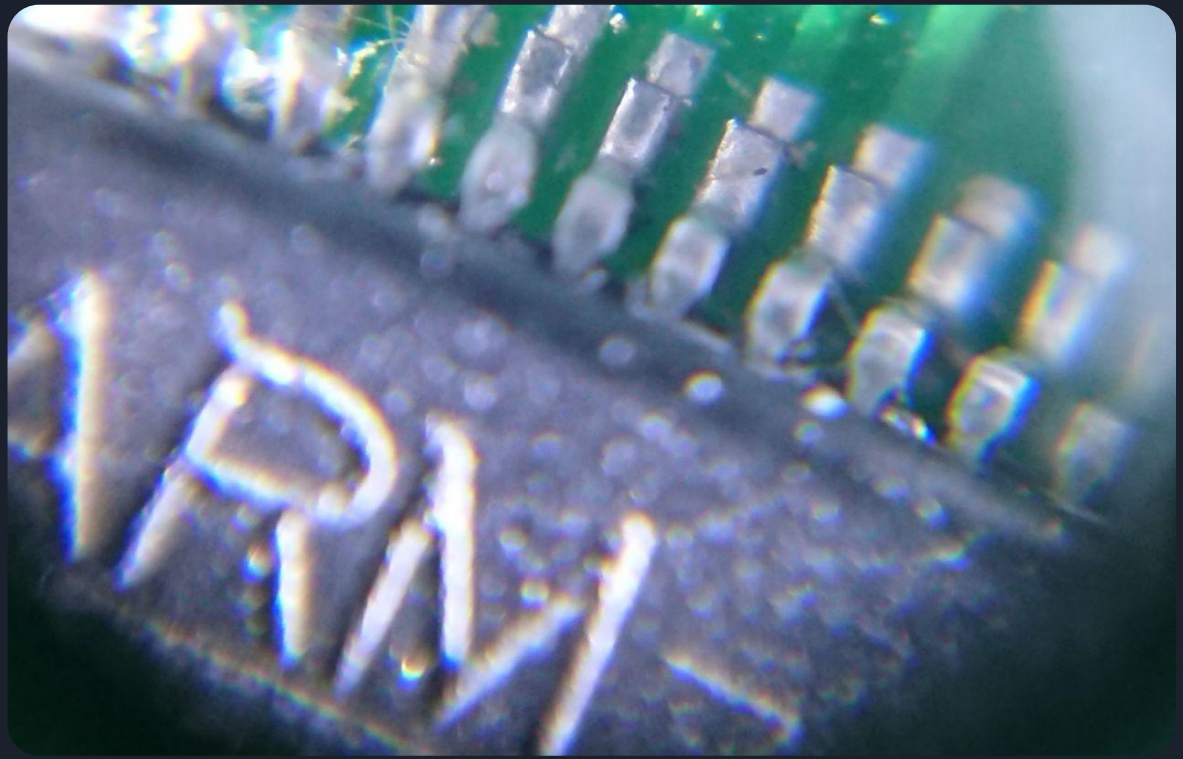


Solder bridges!
Evil!

SMD Soldering



SMD Soldering



Contains product placement

Opportunities for Error so far

- Mistake in part choice
- Mistake in schematic (wrong connection)
- Suboptimal routing in the PCB
  (hard to solder, high-frequency problems)
- Destruction of components while soldering

No testing possible so far!

No testing possible so far!

This is insane. That is why dev boards exist.

Possible conclusion from this talk: doing this is stupid.

Last step!

- it’s “just software”
- Complex toolchain and CPU-specific code
- But tools like STM32CubeMX can help get started
- Moving target: Compilers, Hardware Abstraction Layer libraries, board/processor support, IDE configuration/plugin APIs (eclipse), upload tools, …