17 - Hardware Design

CEG 4330/6330 - Microprocessor-Based Embedded Systems

Max Gilson

Hardware Design

- Often times you may want to design your own hardware
  - More affordable
  - More specialized
  - Different form factor
- Hardware design can get very complicated depending on requirements
- It is rare that design happens from scratch, most design is based on some previous design or other works
  - Very similar to software design, where most software is not brand new and often uses examples from others

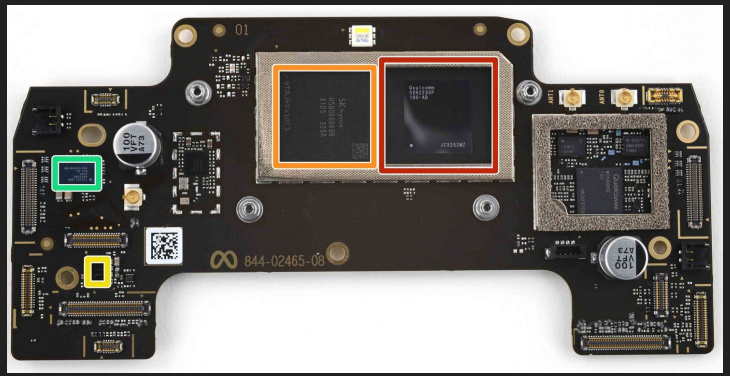
Closed Source Design

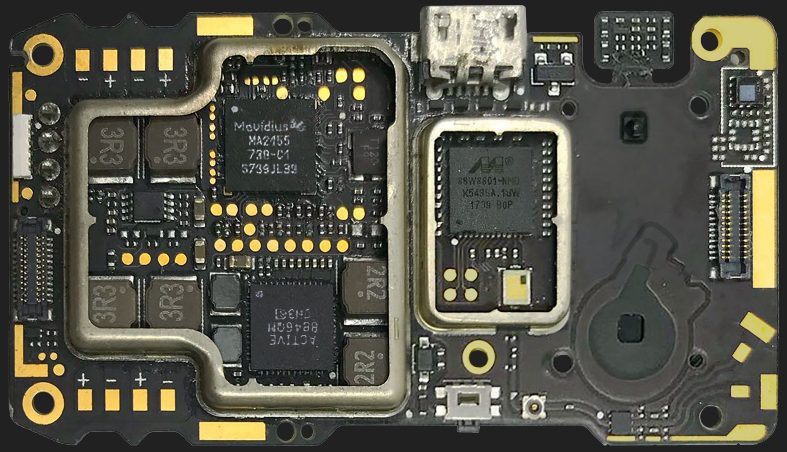
- Closed source designs use components that are not publicly available
  - Can be closed hardware, software, or both
  - Usually, both are closed source
- In order to develop a system that uses closed source components, you have to work with the company that owns the hardware and/or software
- If you want to develop hardware on your own or while working for a small company, you will have to rely on open source hardware

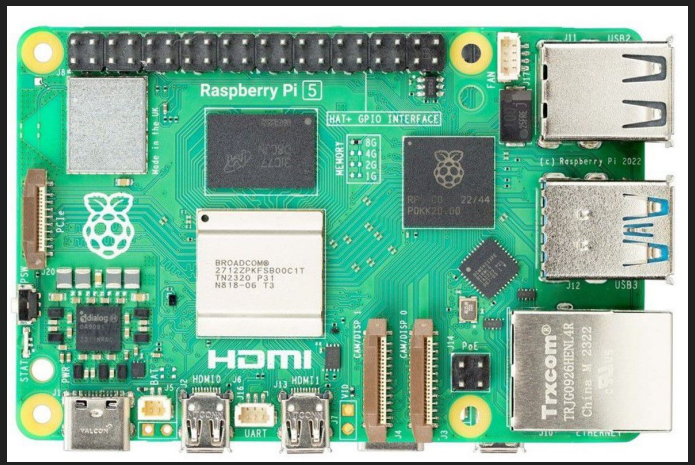
Closed Source Examples

- Some examples of closed source hardware:
  - Broadcom BCM2712 (Raspberry Pi)
  - Intel Movidius MA2155 (DJI Tello)
  - Qualcomm Snapdragon XR2 Gen 2 (Meta Quest 3)
  - Qualcomm Snapdragon 8 Gen 3 (Samsung S24)
- If you ever find yourself wanting to use any of these, you should work at a large company with a lot of money
- Most closed source hardware companies will not work with you unless they know you will buy 100,000’s of units

Closed Source Examples







Open Source Design

- Open source designs use components that are publicly available
  - Can be open hardware, software, or both
- Using open source designs is your only option if you do not work at a large reputable company with lots of money
- Open source designs often benefit from large online communities that offer technical support through forums, discussion boards, or Reddit

Open Source Examples

- Some examples of open source hardware:
  - STMicroelectronics STM32
  - Atmel ATMega328
  - Espressif Systems ESP32
  - Allwinner SoCs
  - RockChip SoCs
- All of the above have publicly available schematics, CAD projects, and source code
- Researching schematics and board designs is a great place to start

Open Source Licensing

- Just because hardware or software is publicly available does not mean you can legally use it however you want
- Common licenses:
  - MIT License
    - Very open and forbids very little
    - A no attribution version is also available
  - GNU General Public License (GPL)
    - You are required to open source your code
  - Apache
    - Carries some patent protections
  - BSD
    - Various clauses
- When in doubt, ask a lawyer, not an engineer

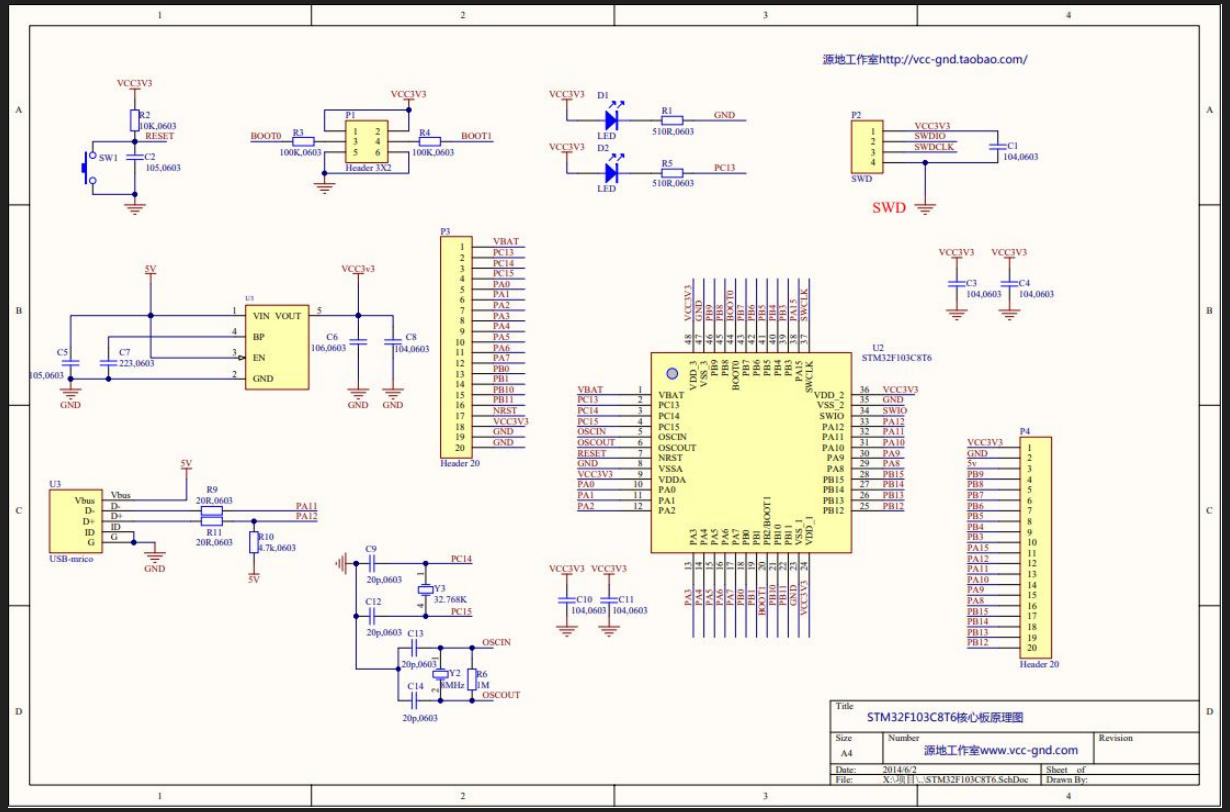
Development Kits

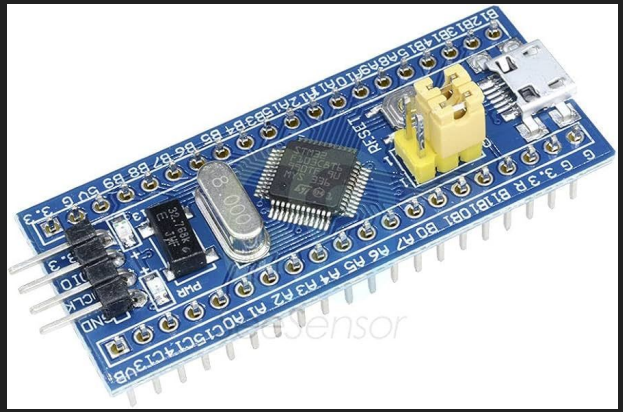
- Different microcontrollers and microprocessors have development kits available
- Using these development kits, you can test software before diving into hardware development
  - Helps validate ideas and test performance, power consumption, etc.
  - Allows you to write code before designing any hardware
- Once software is verified, you can start on PCB design if necessary

STM32 Development Kits

- The STM32 is a very popular microcontroller using an ARM processor
  - Used in many commercial products
- Schematics and code is publicly available
- Can run RTOS
- Development kits and components are available and cheap
- Some variants can be programmed with Arduino IDE
- Plenty of community support
  - YouTube, Reddit, forums, etc.

STM32 Development Kits

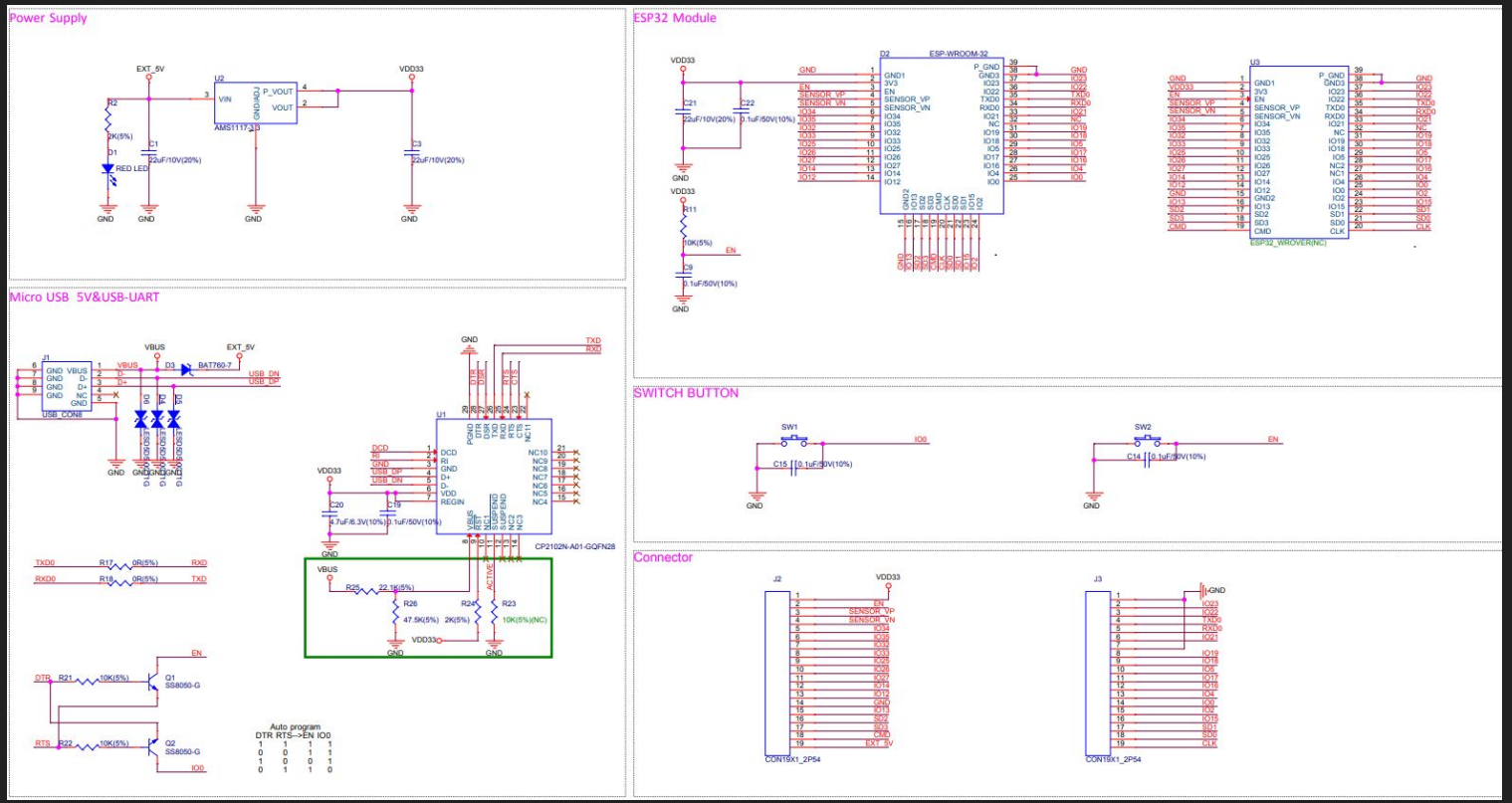


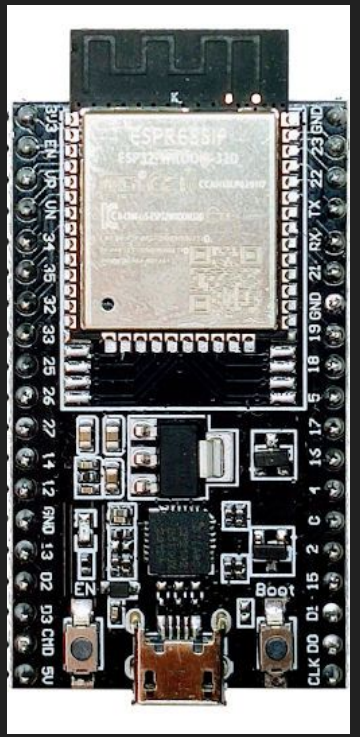


ESP32 Development Kits

- The ESP32 is also very popular
  - Used in IoT products (smart devices)
- Schematics and code is publicly available
- Development kits and components are available and cheap
- Built in support for camera, WiFi, and Bluetooth
- Plenty of community support
  - YouTube, Reddit, forums, etc.

ESP32 Development Kits

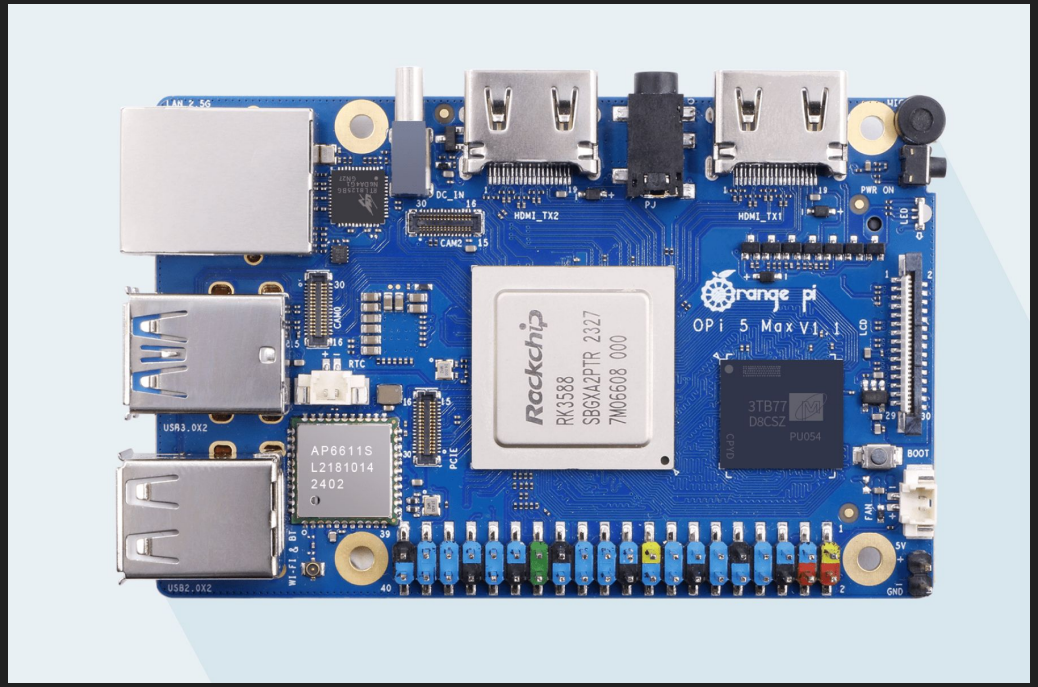


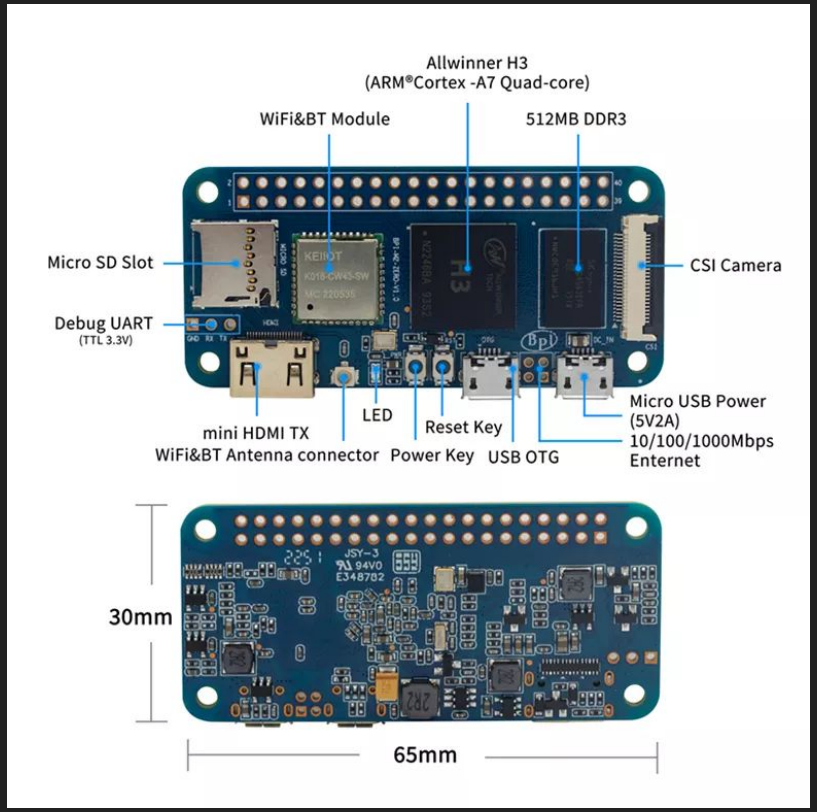


Allwinner and RockChip Development Kits

- Allwinner and Rockchip are popular microprocessors
  - Used in many commercial products
  - Very similar to Raspberry Pi
- Schematics and code is publicly available
  - https://wiki.banana-pi.org/Banana_Pi_BPI-M2_ZERO
- Runs Linux variant OS
  - Armbian (Debian for ARM processors)
- Development kits can be hard to find and can be expensive
- Setup can be complicated and/or poorly documented
- Plenty of community support
  - https://wiki.banana-pi.org/Main_Page
  - http://www.orangepi.org/orangepiwiki/index.php/Main_Page
  - https://linux-sunxi.org/Main_Page

Allwinner and RockChip Development Kits





Getting Started

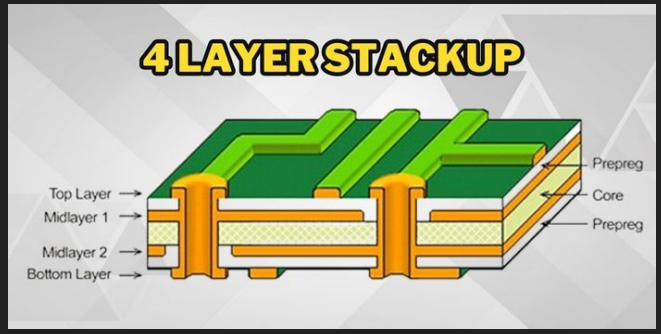
- If you decide to design your own embedded system using any of the open source designs mentioned you should do the following
  - Test on a development board first
  - Find any and all reference schematics
    - Example: ESP32 schematic and any sensor schematics needed
  - Design your schematic
  - Use CAD software for PCB layout and creating your own schematic
    - Eagle (free), Altium (many features), KiKad, EasyEDA (simple designs like Arduino)

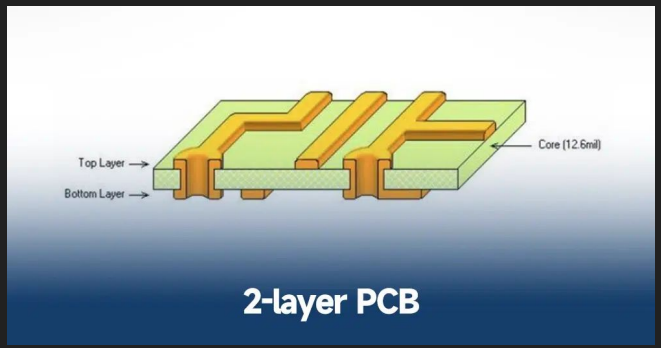
Designing a Schematic

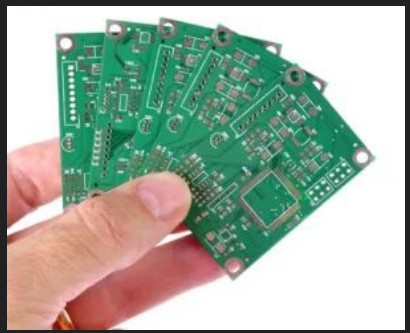
- Designing a schematic is usually done using references
  - Reference data sheets for some parts like voltage regulators
  - Reference other designs and remove what you don’t need
- Design a schematic that implements only what you need from your project
  - For example, if you need a microcontroller with a specific sensor, find a reference, and remove everything that does not use that sensor
- Add other components and voltage regulation as necessary

Printed Circuit Board

- A printed circuit board (PCB) holds and connects all of our components together
- This board is made up of various layers
- Components are soldered to pads or holes
- Some of the layers act like copper wires called traces, connecting our components
- Connections to other layers are done through vias (plated holes)
- Special considerations have to be made for power, noise, timing, etc.
    - Physics of the board can get complicated



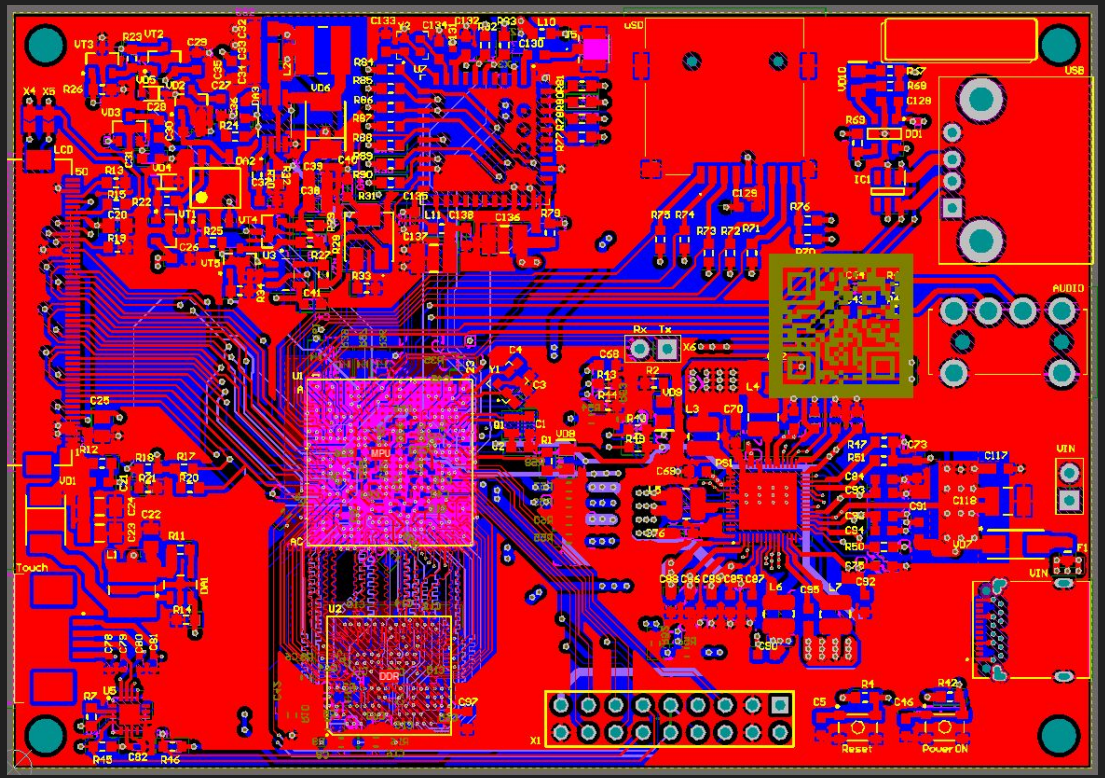


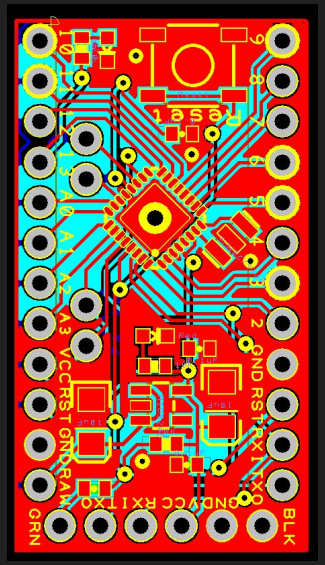


Know Your Complexity

- Developing a microcontroller board may be complex, but it is nothing compared to a microprocessor board
- A microcontroller might have 1 input voltage, 2 layer board, very few pins, and no high speed (GHz range) signals to worry about
- A microprocessor might have 3 or 4 input voltages, 6 layer board, hundreds of pins, and high speed signals (noise and timing requirements)

Know Your Complexity





Examples

- https://oshwlab.com/coold69/arduino-uno
- https://oshwlab.com/lerenler/robot_doly_a64
- https://oshwlab.com/contact63/stm32-balancing-robot

PCB Traces and Vias

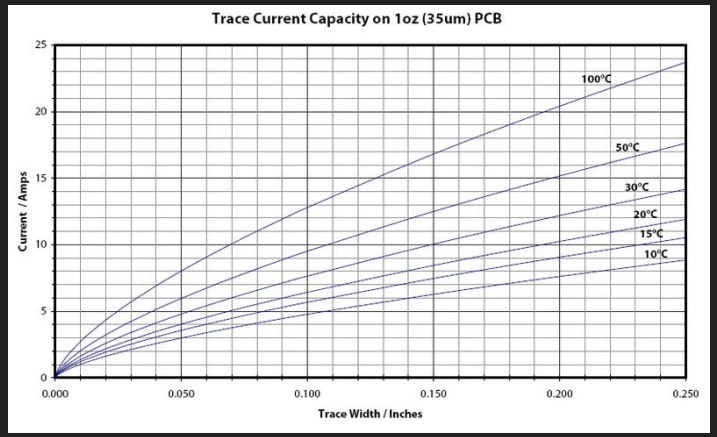
- Traces and vias on the board are wires that connect our components
  - They have capacitance, inductance, resistance, and a maximum current capacity
  - They are influenced by noise, frequency, ambient temperature, and adjacent traces

Trace Current Capacity

- IPC is a set of standards that outline how PCBs should be assembled and manufactured
  - IPC includes a standard for the safe maximum current capacity for traces
  - You wouldn't want a trace to explode or burn when operating your device!
  - https://www.mclpcb.com/blog/pcb-trace-width-vs-current-table/
- Understanding the theory and reasoning is important
- In practice, use a calculator

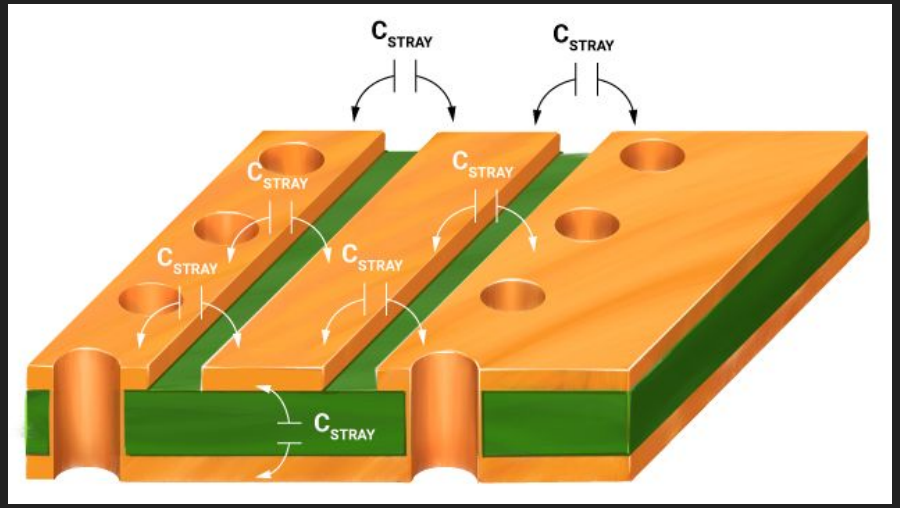
IPC Recommended Track Width For 1 oz cooper PCB and 10 °C Temperature Rise

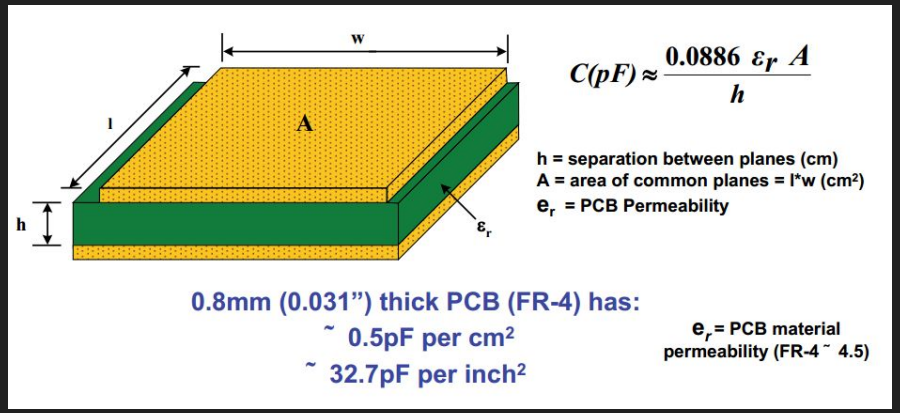
Current/A	Track Width(mil)	Track Width(mm)
1	10	0.25
2	30	0.76
3	50	1.27
4	80	2.03
5	110	2.79
6	150	3.81
7	180	4.57
8	220	5.59
9	260	6.60
10	300	7.62



Trace Parasitics

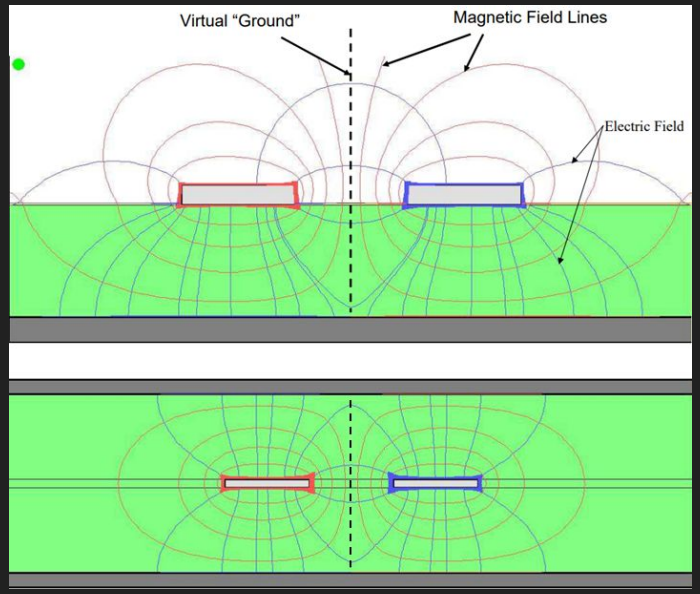
- PCB traces have various electrical characteristics
  - Inductance
  - Capacitance
  - Resistance
- To calculate these values for each trace requires a deep understanding of mathematics and physics
- In practice, use a calculator
  - CAD software often supplies this information

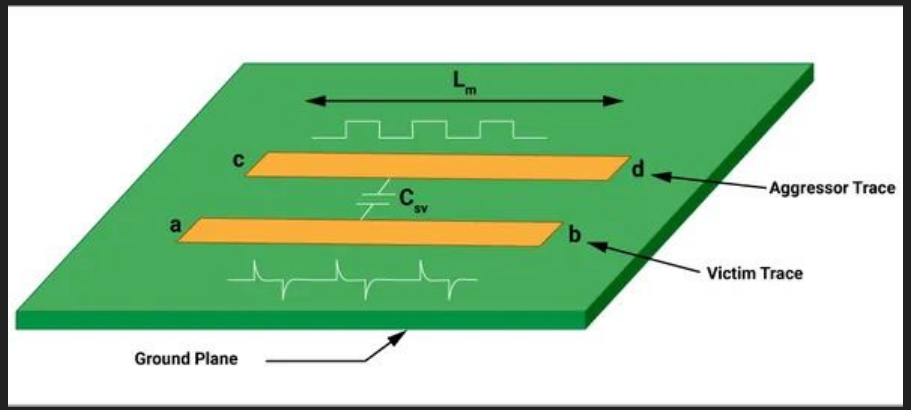




Trace Crosstalk

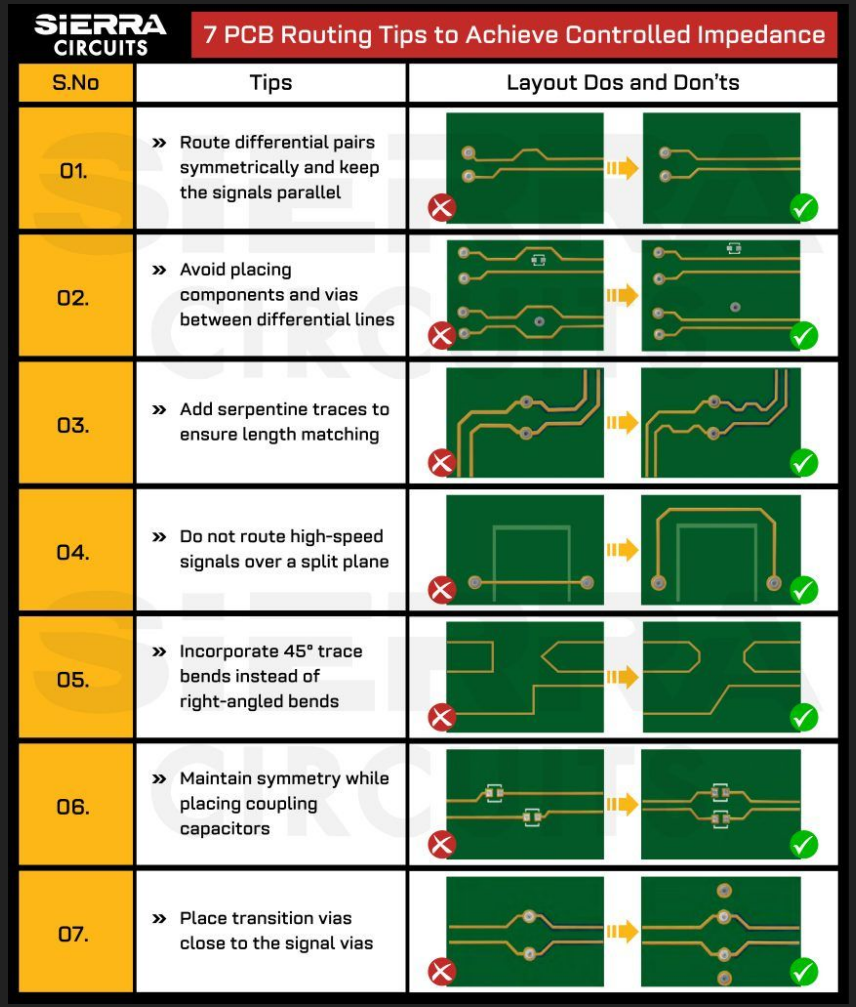
- PCB traces are wires and they produce magnetic and electric fields
- Traces that are adjacent to each other will receive noise from these fields called “crosstalk”
- To avoid this issue, have traces far away from each other if on same layer or perpendicular to each other on different layers





Trace Timing

- Signals do not transmit across traces instantly
  - Signals travel almost at the speed of light
- Some signals require high speeds
  - Example: DDR3 RAM has a clock at 2.133 GHz (period of 0.5 ns!)
- If you are working with high speed signals, you need to ensure they all have similar lengths
  - This allows all signals to “arrive” at the same time, or propagation delay
  - https://www.zuken.com/us/blog/how-to-calculate-trace-length-time-delay-value-high-speed-signals/

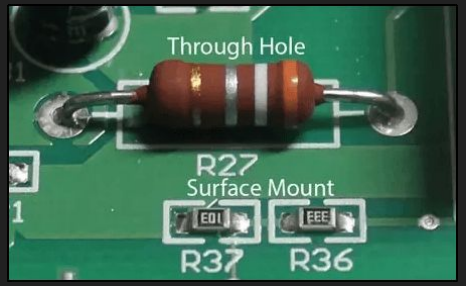


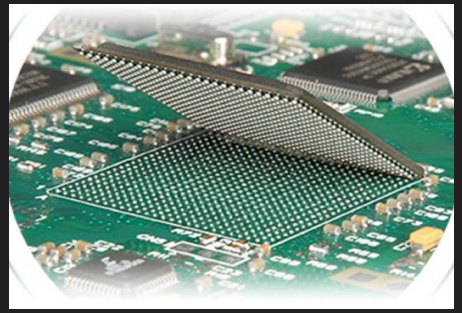
PCB Manufacturing

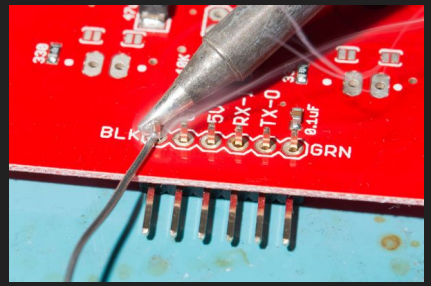
- Before you start designing a board it is important to make sure that whatever you end up designing is actually manufacturable
- Find a manufacturer, and a list of their capabilities
  - Some manufacturing capabilities may be expensive like 0.2mm hole/vias
  - Example: https://www.pcbway.com/capabilities.html
- Many manufacturers have easy ways to receive quotes

PCB Assembly

- Components for your board can come in two styles
  - Through Hole
  - Surface Mount (SMD)
- Surface mount components can easily be assembled by machines and many can fit onto a board
  - Some SMDs cannot easily be soldered by hand
- Through hole components are more expensive to install in the board since they must be installed by hand or by more complicated machines
  - Can be expensive $$$
- Of course you can always try to assemble yourself

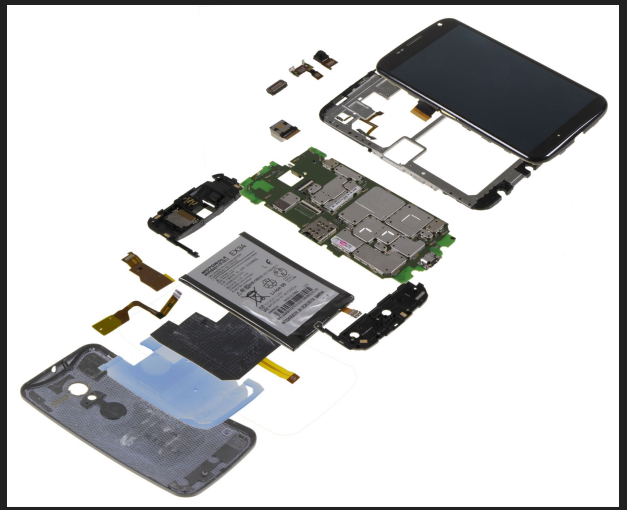






Industrial Design

- Industrial design is a design process for products that are to be mass manufactured
  - Could be an entire course itself
- From a hardware standpoint, industrial design is very important
  - 3D models of your board and components can help



Industrial Design Considerations

- Product life cycle
  - Can we source all the parts needed for manufacture?
  - What do we do for end of life parts?
  - Reliability
- Form factor
  - Does our embedded system need a nice enclosure?
  - Ergonomics, comfort, and human factors
  - Water or dirt resistance
  - Thermal management and thermal dissipation
- User interface
  - How does the user interact with our device?
  - Buttons, LEDs, speakers, displays, etc.
  - Haptic feedback
- Aesthetic appeal
  - How does our design look?
  - Enclosure design and packaging materials

Industrial Design Considerations