FINAL LECTURE

End of Year Wrap Up

SECTION I

Circuits Review

Circuits Reflection

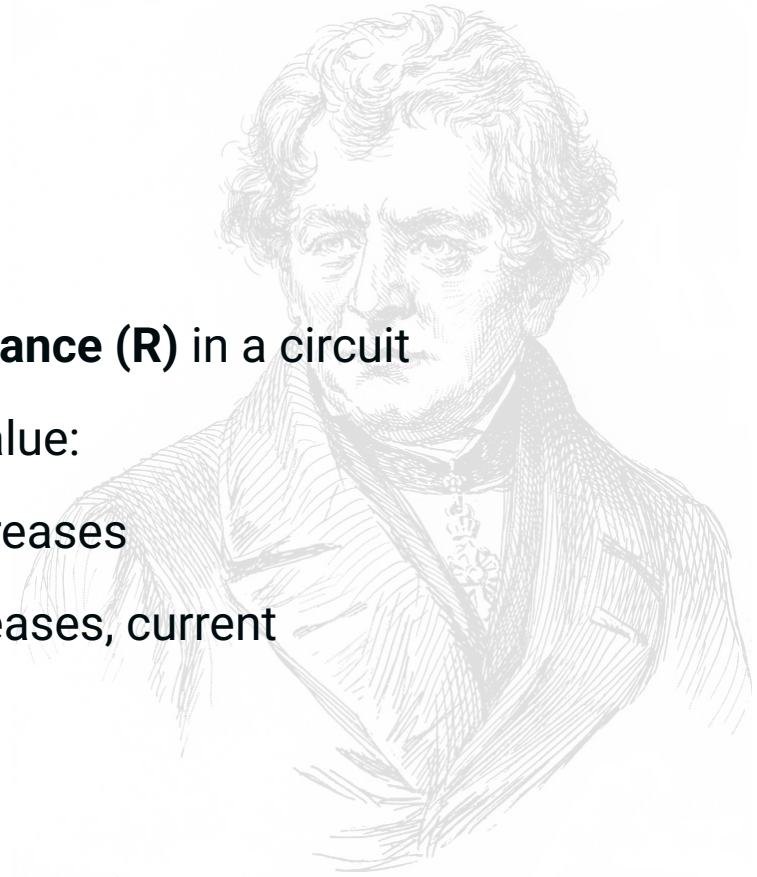
- What are the core concepts of circuits that you should take away from OPS?
- What pitfalls are important to look out for when trying to implement a circuit?

Voltage

- **Voltage** is the **electric potential difference**
  - Current can flow where **voltage** exists
  - Measured in **Volts, V**
  - Ex. **Batteries** are a voltage source
- **Requires a reference point** because it measures the difference in volts between two points

Ohm's Law

$$V = IR$$



- Relates the **voltage (V)**, **current (I)**, and **resistance (R)** in a circuit
- Suppose we hold the voltage at a constant value:
  - As the resistance increases, current decreases
  - In the opposite case, as resistance decreases, current increases

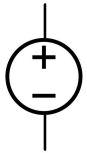
SECTION II

Circuit Analysis Review

Schematic Symbols

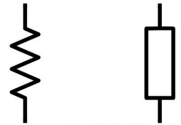
Battery

Voltage Source



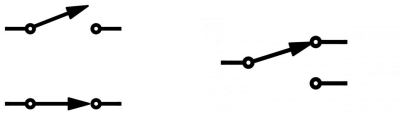
Ground

Resistor

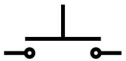


Potentiometer

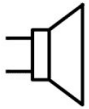
Switches



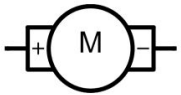
Pushbutton



Speaker



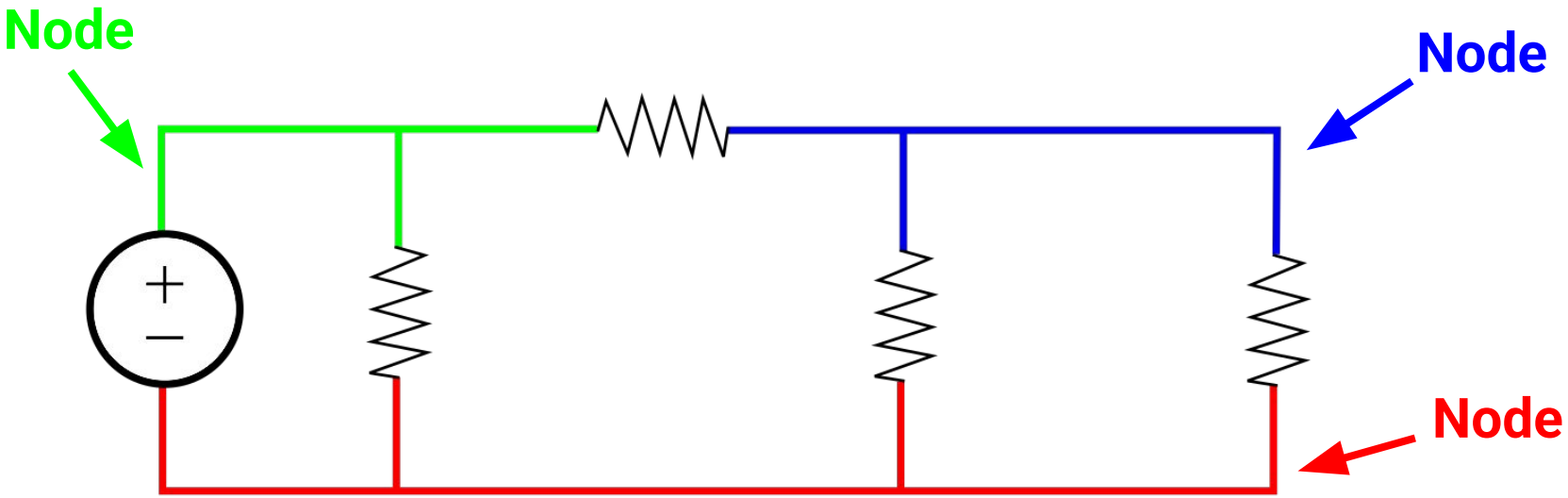
DC Motor



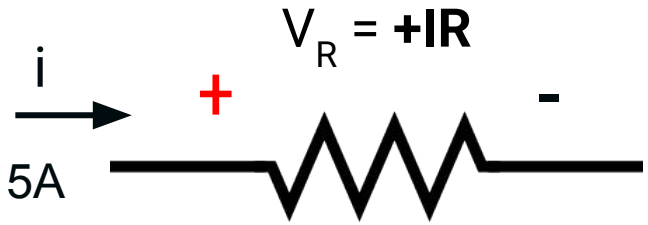
LED

Nodes

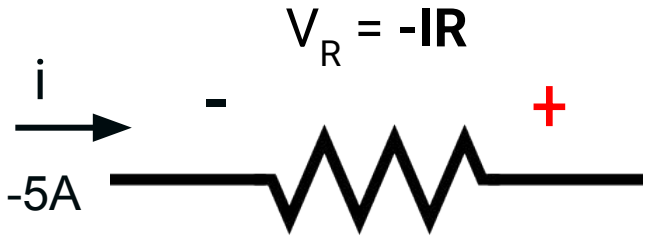
- A **node** is a **connection** between two or more **components**
- All points on the **same node** have the **same voltage**



Voltage Drop Sign Convention (Cont'd)



- The sign of the voltage drop is **positive** if current flows from **+** to **-** terminals
  - Conventional current flows from **+** to **-** terminals

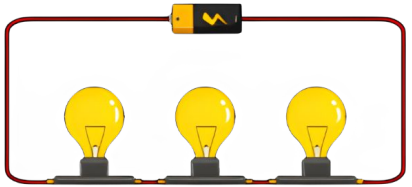


- The sign of the voltage drop is **negative** if current flows **-** to **+** terminals

Series and Parallel Circuits

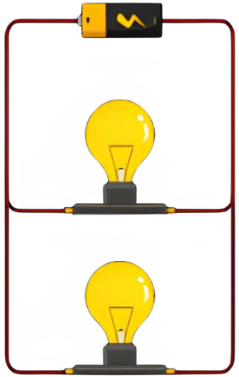
Series

- Two or more elements are in **series** if they exclusively are connected in one line
- Elements in series carry the **same current**



Parallel

- Two or more elements are in **parallel** if they are connected to they share the same two nodes
- Elements in parallel have the **same voltage** across them

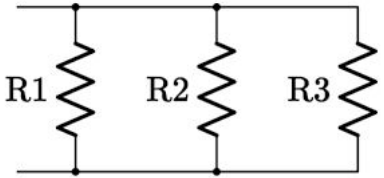


Series and Parallel Circuits

- If **multiple resistors** are arranged **in series or parallel**, we can treat them as having a **single equivalent resistance**

Parallel

$$\frac{1}{R_{eq}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}$$

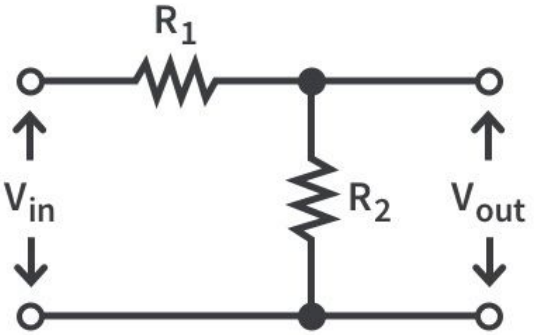


Series

$$R_{eq} = R_1 + R_2 + R_3$$

Resistor Divider

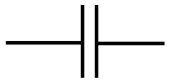
- A circuit configuration with **two resistors** that is used to scale down a voltage (**Vout**).
- Creates a ratio of two resistors to achieve a desired output voltage
  - **R1** and **R2** are connected in series
  - **Vin** is the input voltage
  - **Vout** is the output voltage across **R2**



$$V_{out} = \left(\frac{R_2}{R_1 + R_2}\right) V_{in}$$

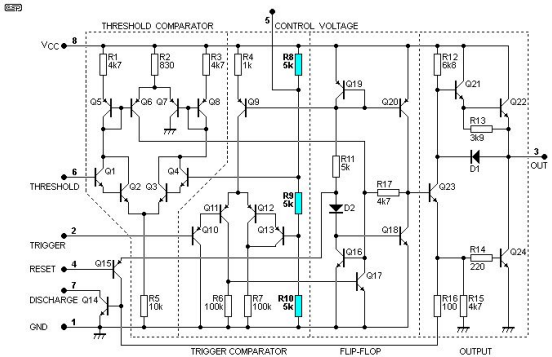
Capacitors

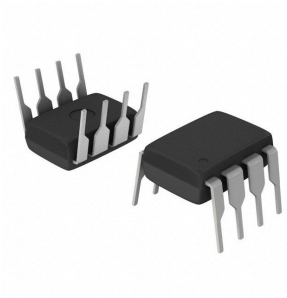
- A **capacitor** stores electrical energy which it charges and discharges
  - The ability of a capacitor to store energy is its **capacitance**, measured in **Farads** (**F**)
- Unlike a battery, a capacitor can only briefly store a small amount of energy
- When a capacitor is connected to a voltage source, it charges until it reaches the **same voltage** as the voltage source
- Used for filtering, storing a small amount of energy, and timing circuits



Integrated Circuits

- An **integrated circuit** (**IC, chip**, **microchip**) is a set of electronic circuits on one small flat piece of semiconductor material
  - Electronic components are **integrated** on the chip
  - Complex circuits can be **scaled down** and **mass-produced**
  - Used for a specific application





SECTION III

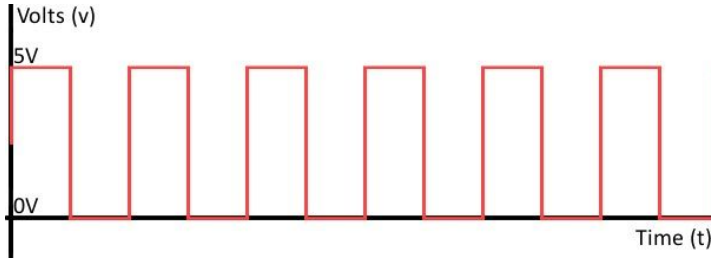
Microcontrollers

Microcontroller Reflection

- What does GPIO stand for?
- What are the main differences between digital and analog signals and when should you use one over the other?
- What is PWM and why is it important?

Digital Signals

- **Computers (and microcontrollers) transfer data** across wires/lines as **digital** (discrete) **voltage signals**
  - These signals are either a **HIGH** or **LOW** voltage
  - Contrasts from **analog signals** which can be values in a continuous range

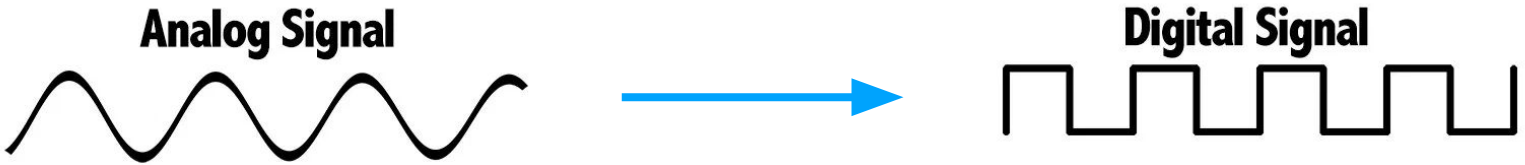


This is a **digital signal** where the waveform is either **5V** or **0V** (two *discrete* values)

- **Digital signals are translated** to the **Binary** number system (1s and 0s)

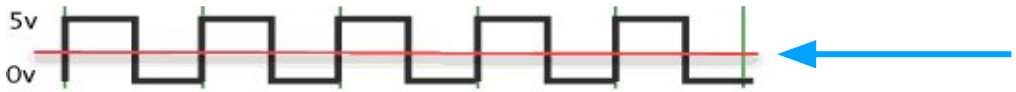
ADC and `analogRead`

- `analogRead` **utilizes the analog-to-digital converter** inside the microcontroller to **measure the real-world, analog signal** and **convert it to a digital signal**
    - The measurement resolution is 10 bits, which is why the function returns values from `0-1023`



PWM

- The ESP32 board (the underlying AVR microcontroller) is a **digital source**, meaning it can only output a `HIGH` (5V) or `LOW` (0V) voltage
  - Then how does `analogWrite` output analog signals?
- **Pulse Width Modulation** (**PWM**) is an oscillating digital waveform that emulates an analog output
  - By oscillating a signal from `HIGH` to `LOW` quickly, the average voltage over time will be *between* `HIGH` and `LOW` - an analog value



The average value, the *analog value*, of this waveform is **2.5V**

SECTION IV

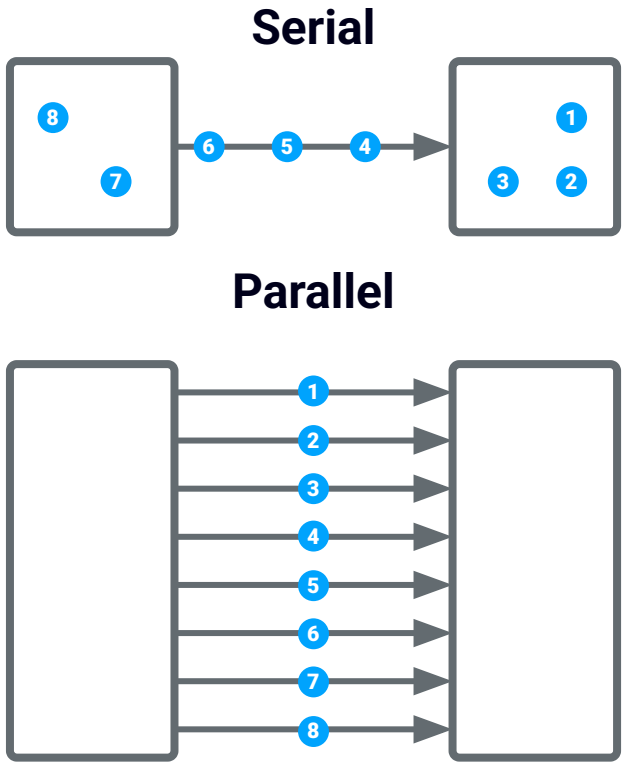
Communication Protocols

Communication Protocols Reflection

- What's the difference between synchronous and asynchronous protocols?
  - What synchronous protocols did we learn?
  - What asynchronous protocols did we learn?
- Why are clock signals important for SPI and I2C?

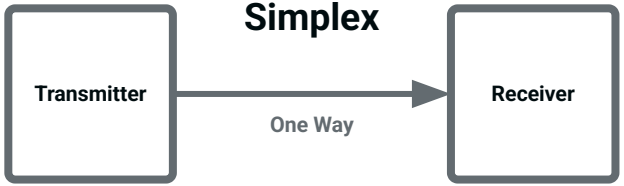
Serial vs Parallel Protocols

- **Serial** - bits are sent over a connection one by one to a device
  - Bits are sent in a specific order
  - The most common protocols are serial
- **Parallel** - multiple bits (often one byte) are sent simultaneously over a connection
  - This connection requires more wires, which takes up more space
  - Higher transmission rate

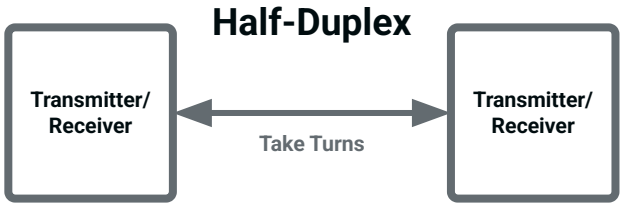


Transmission Modes

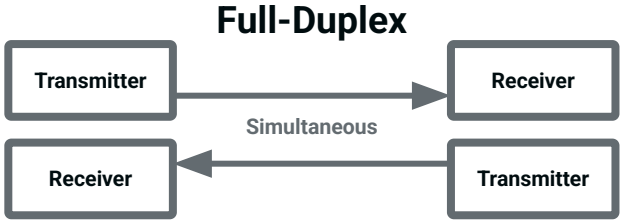
- **Simplex** protocols allow communication in only one direction



- **Half-Duplex** protocols allow communication in both directions but only in one direction at a time



- **Full-Duplex** protocols allow communication in both directions simultaneously

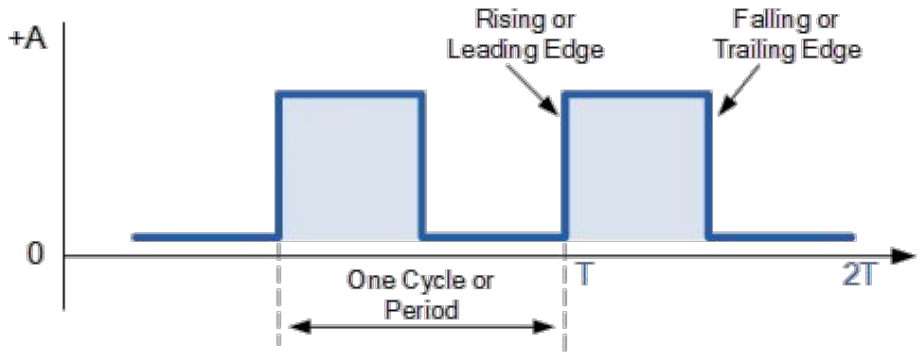


UART Protocol

- UART is a **communication protocol with certain characteristics**. It has rules for data transmission.
  - UART can be configured to **full-duplex, half-duplex,** and **simplex modes**
  - There are only **two lines** (electrical connections)
  - There is **no clock signal**/line hence the “asynchronous” part of the title
  - The data transmission speed is determined by a **baud rate**
    - It is quite **slow compared to other protocols**

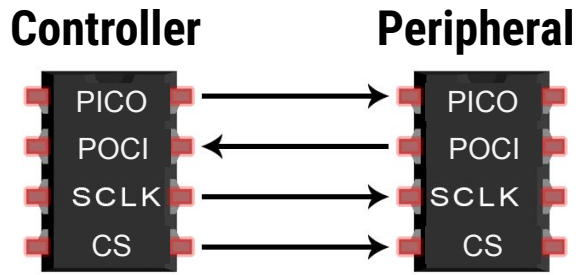
Clock Signals

- **Digital signals** are timed to a **clock signal** - most often, a square wave with a 50% duty cycle
  - For example, in a serial signal, one bit of data is sent per clock cycle
    - The bit might be timed to the rising or falling edge of the clock signal
- **Clock speed** determines the rate at which data is transmitted
  - measured as **frequency (Hz)**
  - inversely proportional to cycle (or period)

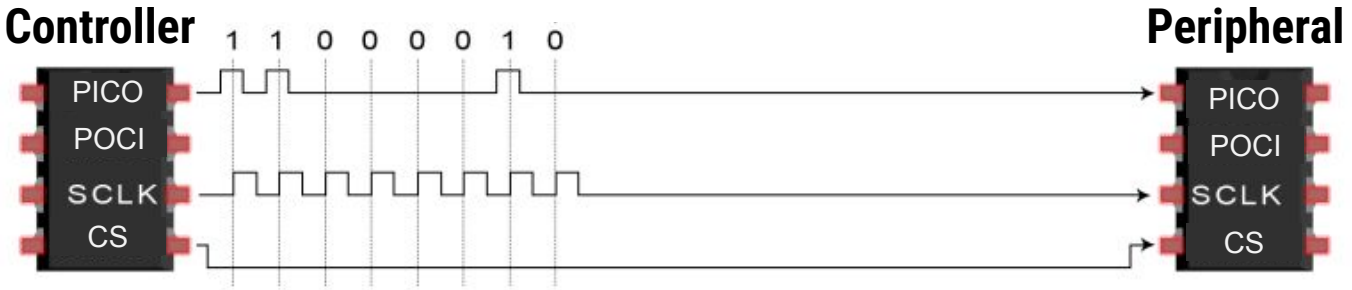


SPI Protocol

- **Serial Peripheral Interface** or **SPI** is a serial communication protocol which is **synchronized with a clock**
- Unlike UART, **data is transferred continuously** in SPI
  - There are no packets, start bits, stop bits, parity bits, or anything else
- One device acts as a **controller** to one or more **peripheral** devices
- Supports **full-duplex** communication
  - Bi-directional communication



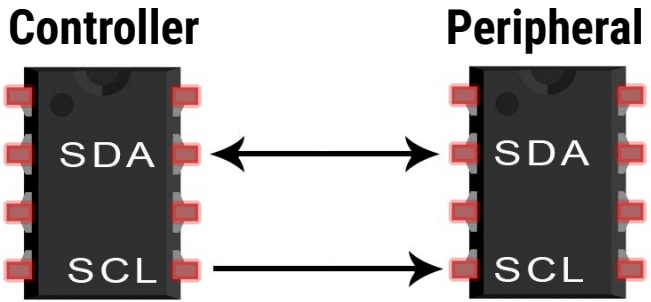
SPI Transmission



1. The controller first **selects a peripheral** by pulling a HIGH signal to LOW along the Chip Select (CS) line
2. The controller **sends data** to the selected peripheral along the Peripheral-In/Controller-Out (PICO) line
    - Bits in the PICO signal are **synchronized with the rising edge** of the oscillating clock signal
    - Alternatively, the peripheral sends data along the Peripheral-Out/Controller-In (POCI) line

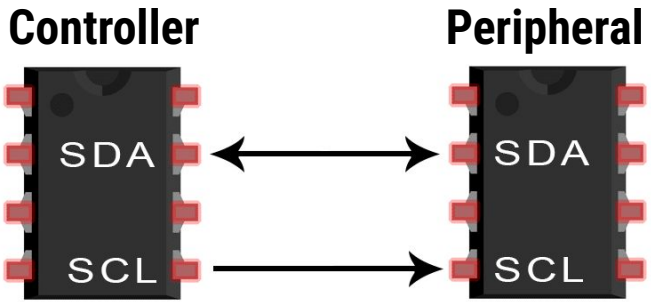
I²C Protocol

- **Inter Integrated Circuit** or **I²C** is another synchronized, serial protocol
- Unlike UART and SPI, I²C supports **multiple controllers**
- I²C features **acknowledgements** to confirm if messages are received and addressing for peripherals
- Supports **half-duplex** communication
    - One way communication



I²C Layout

- SDA (**Serial Data**) - the line for the controller and peripheral to send and receive data
- SCL (**Serial Clock**) - the line that carries the clock signal
- All peripherals can share the same two SDA and SCL lines



UART vs SPI vs I^2C

Protocol	UART	SPI	I^2C
# of Lines	2	4+	2
# of Controllers	1	1	1+
# of Peripherals	1	1+	1+
Transmission Type	Full-Duplex	Full-Duplex	Half-Duplex
Error-Checking	✅	❌	✅
Speed	Slowest	Fastest	Slower

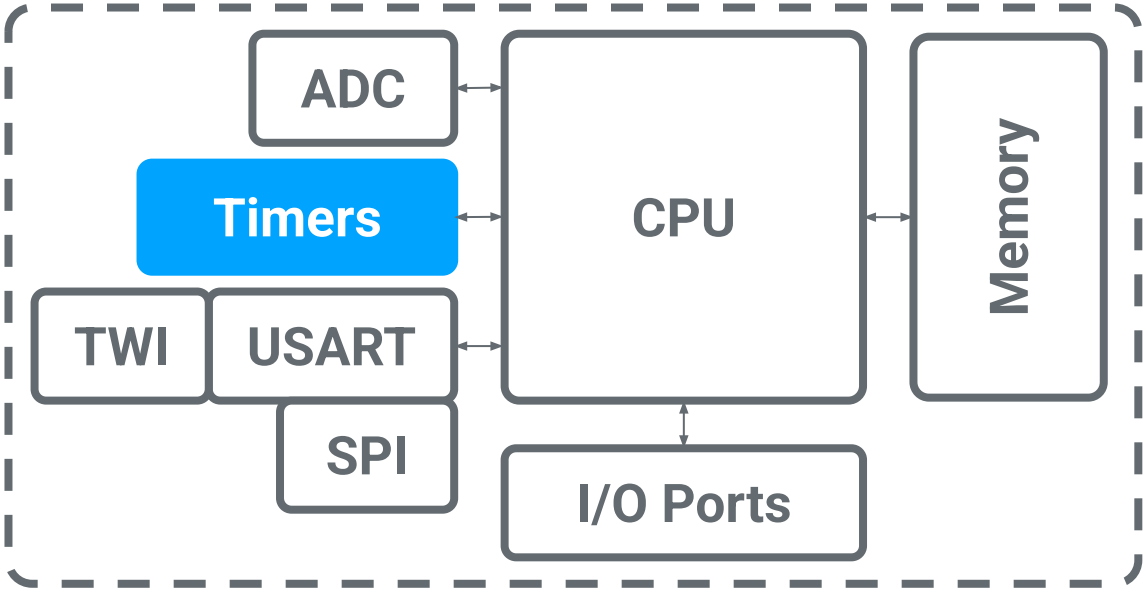
SECTION V

Timers & Interrupts

Timers & Interrupts Reflection

- What do timers do and what applications could they have?
- Why do we use interrupts and why can they be useful?
- What's the difference between hardware and software interrupts?

Timers



Timers are used by the microcontroller to **control the timing** of program execution or output signals and to **measure time**. Based on microcontroller clock cycle

Types of Timers

- Timer types can be distinguished by when the **timeout** triggers
  - A **timeout** is an event that occurs after a preset period of time
- A **one-shot timer** has a single timeout
  - For the timer to be run again, it must be **reloaded manually**
  - Ex) A countdown timer runs just once, stopping when it reaches zero
- A **periodic** (or **auto-reload**) **timer** has *periodic* timeouts that occur at a fixed interval
  - The timer **reloads automatically**
  - Ex) An alarm clock rings once every morning

Interrupts

- An **interrupt** is a request for the CPU to **halt the currently executing code** when an event occurs
  - The CPU suspends the current program to *handle* the event by executing a function called an **interrupt handler** or **interrupt service routine** (**ISR**)
  - When an interrupt halts the current code, a separate program / function can be called and run.
  - When the interrupt handler finishes execution, the CPU returns to the old program

Polling

- **Polling** is a method of periodically checking the status of a device
  - Ex) A person checking their phone for new messages every 10 minutes
    - We have been using polling in the `void loop()`
  - Polling loops cost additional CPU time
- Should we use interrupts instead of polling?
  - Interrupts signal *when* an event occurs whereas polling *checks* to see if an event occured
    - Ex) A person reading their phone upon receiving a push notification
  - **Use interrupts when an event is urgent and/or infrequent**

Types of Interrupts

- A **software interrupt** is triggered by an instruction
  - Also called a **trap** or **exception**
  - Often caused by an illegal operation
    - Ex) Dividing by zero, Accessing memory without permission
- A **hardware interrupt** is triggered by devices external to the CPU
  - Also called an **external interrupt**
  - Caused when a devices sends an **interrupt request** (**IRQ**) signal to the CPU
    - Ex) Timer timeout, button press, power failure

SECTION VI

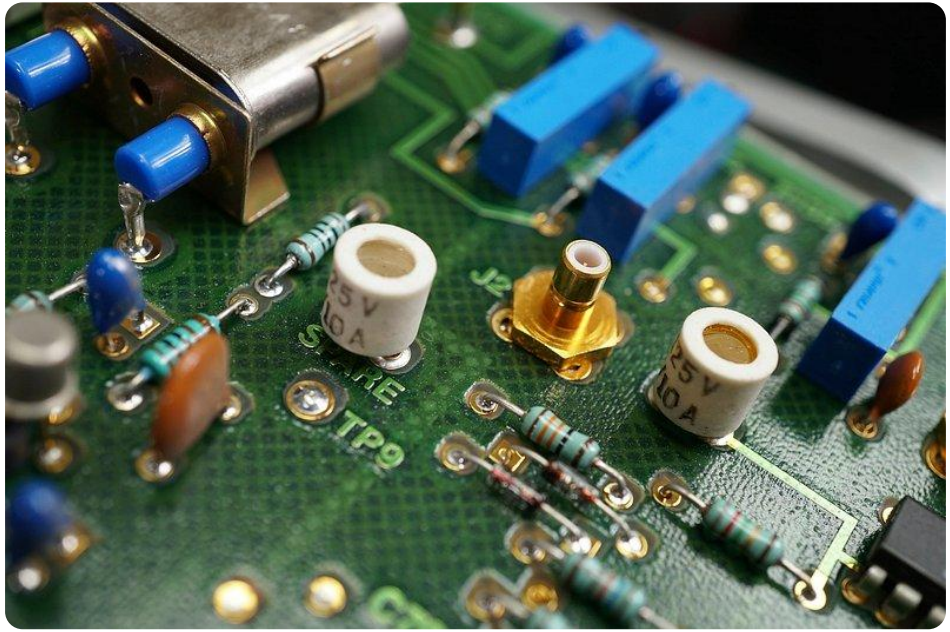
PCB Design

PCB Design Reflection

- Why do we use PCBs instead of breadboards?
- What are common issues that can occur when creating/assembling a PCB?

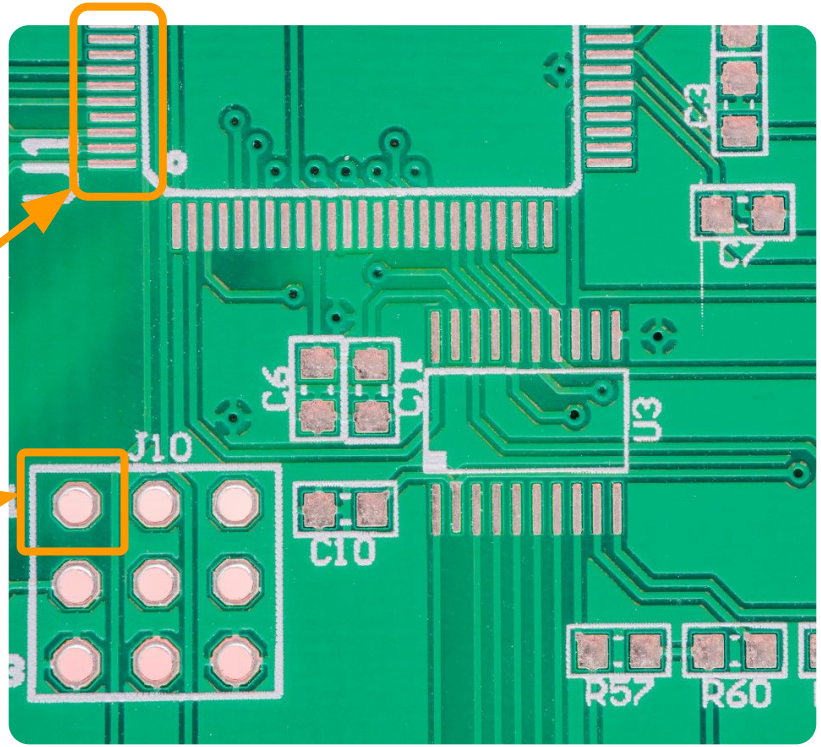
Printed Circuit Boards (PCBs)

- A **printed circuit board** (**PCB**) is an electronic assembly that uses copper conductors to **connect components**
  - It acts as a permanent map in placing and connecting electronic components
  - Made from **multiple**, **alternating layers** of **conductive** (usually copper) and **insulating material**



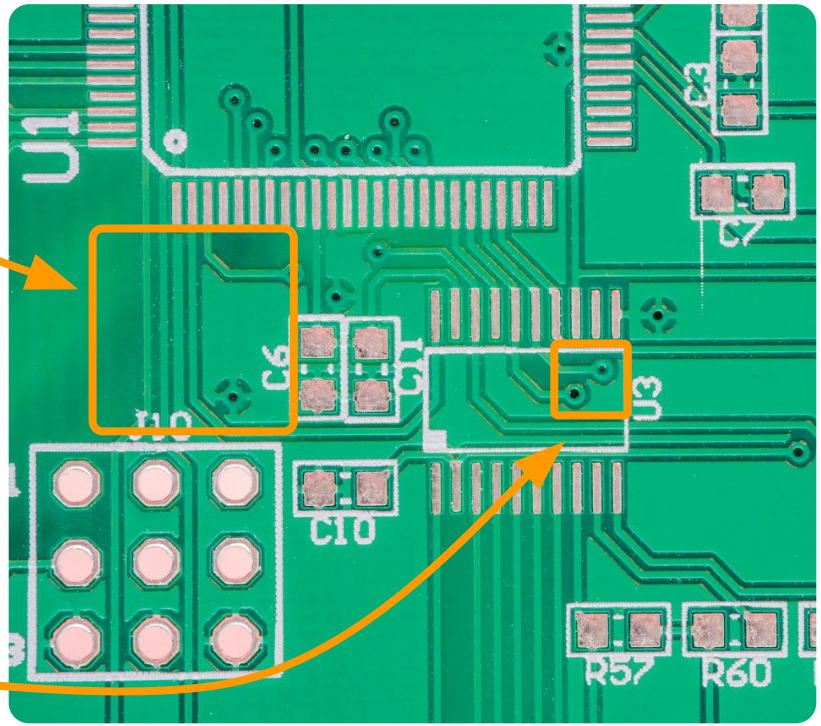
How are Components Connected?

- **Pads** are exposed copper surfaces which connect the leads of the electronic components to the board
  - Some pads are designed for **surface-mounted** (**SMD**) leads while others are for **through-hole** (**THT**) leads
  - Component **leads are soldered to the pads**



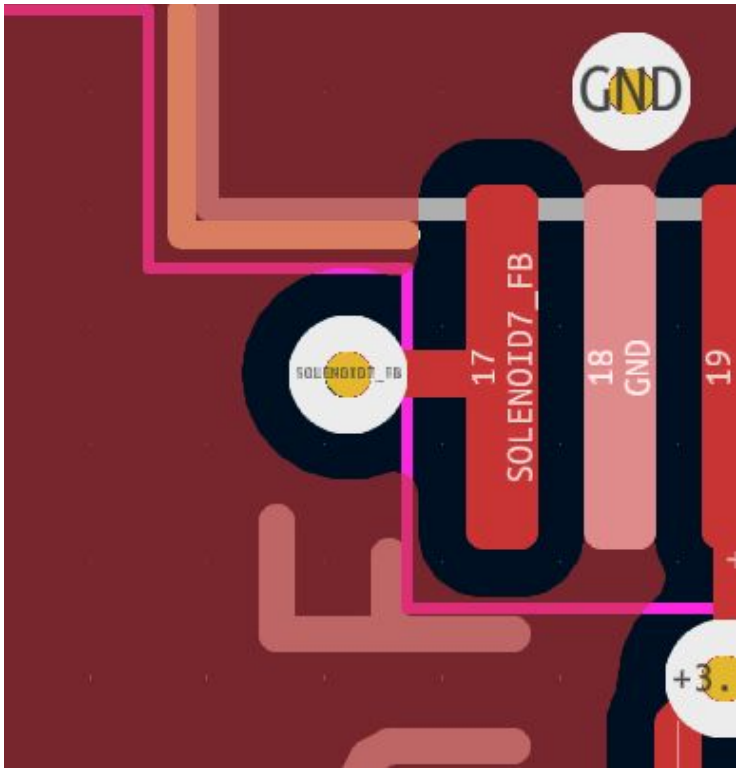
How are Components Connected? (Cont'd)

- **Traces** are **copper tracks** which connect pads
  - They are covered by a layer called the **solder mask** (colored green in the image)
- **Vias** are **conductive holes** drilled into the board to **connect different copper layers**



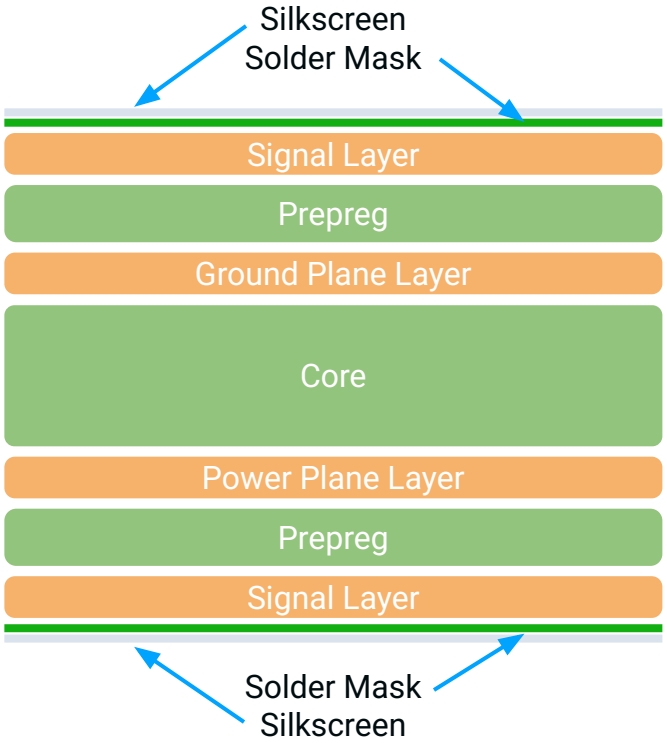
How are Components Connected? (Cont'd)

- A **plane** is an **inner conductive layer**
  - Used to create a ground point
- **Fills** are **large areas of copper** used for the same purpose as planes but can be integrated into the **same layer as traces**
  - The transparent shape surrounding the vias and the pads on the right is a Fill, that is used to ground the ground pad



PCB Stack-Up

- The **PCB Stack-Up** is the **arrangement** of PCB layers
- Most PCBs have multiple of the following layers:
  - **Copper layer**
    - Signal or routing layer
    - Ground/Power plane layer
  - **Insulation layer**
    - Core
    - Prepreg
  - **Solder mask layer**
  - **Silkscreen layer**



FAIR USE DISCLAIMER

Copyright Disclaimer under section 107 of the Copyright Act 1976, allowance is made for “fair use” for purposes such as criticism, comment, news reporting, teaching, scholarship, education and research.

Fair use is a use permitted by copyright statute that might otherwise be infringing.

Non-profit, educational or personal use tips the balance in favor of fair use.

CC BY-NC-SA 4.0