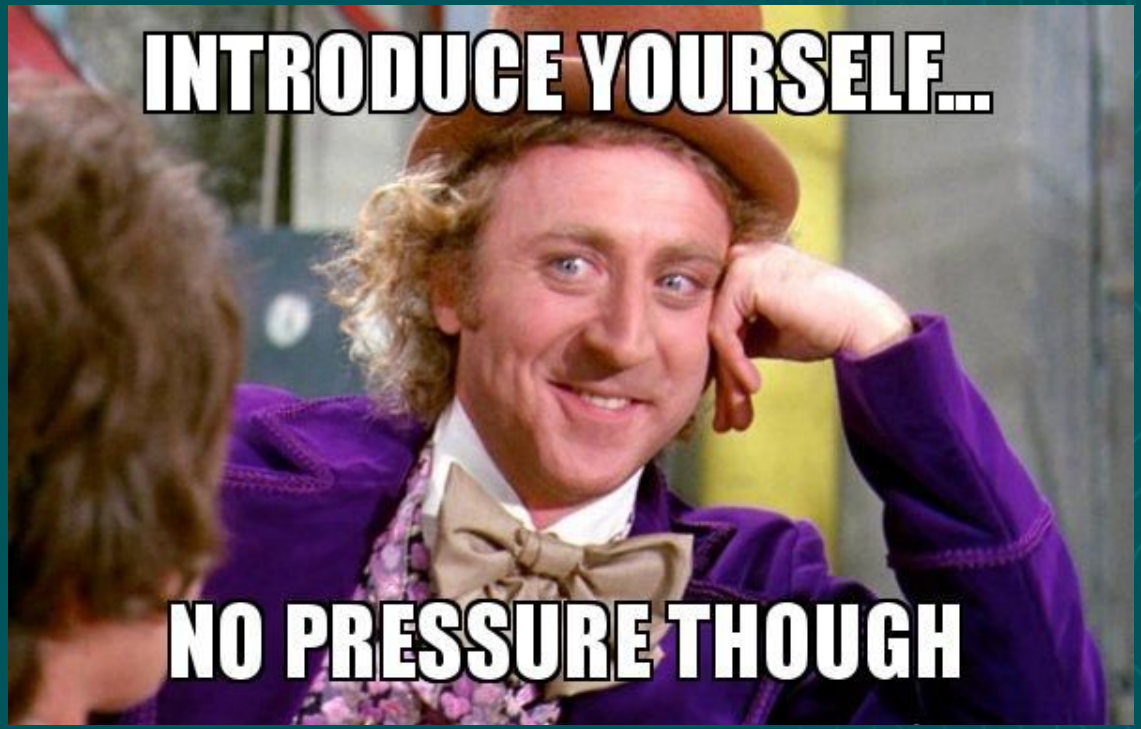
Microcontroller
(Session 1)

Introductions



What all do we plan to cover today?

Introduction to Microcontrollers

- Tiny computers that can be programmed to control electronic devices
- A self contained system with its own processor, memory, IO peripherals, all integrated on one chip

Microprocessor vs Microcontroller

- Microprocessor - It is only a processor, so memory and I/O components need to be connected externally
- Microcontroller - Micro Controller has a processor along with internal memory and I/O components

ESP32 + Arduino IDE

- ESP32 is a microcontroller that combines Wi-Fi and Bluetooth capabilities.
- Arduino IDE is special software running on your system that allows you to write program for different Arduino boards

Microcontroller vs Microprocessor

What is a microcontroller?

A small computer on a single integrated circuit that contains a processor, memory, and input/output peripherals, designed to control a specific task or set of tasks within an embedded system.

Features and applications of microcontrollers

- **Low power consumption**
  Microcontrollers are designed to operate on minimal power, making them ideal for battery-powered devices.

- **Integrated peripherals**
  Microcontrollers often come with a variety of built-in peripherals such as timers, communication interfaces, and analog-to-digital converters.

- **Real-time processing**
  Microcontrollers can be programmed to respond to events in real-time, making them ideal for control applications.

- **Cost-effective**
  Microcontrollers are generally less expensive than microprocessors, making them a popular choice for cost-sensitive applications.

- **Examples of microcontroller applications**
  Microcontrollers are commonly used in embedded systems, consumer electronics, automotive systems, and industrial control applications.

Examples of microcontroller-based projects

Using microcontrollers to control home appliances and lighting systems

A project that involves programming and interfacing a microcontroller with various sensors and devices to automate home functions.

Building robots with microcontrollers

A project that involves designing and programming a microcontroller-based robot that can perform specific tasks such as obstacle avoidance or line following.

Using microcontrollers to measure and track environmental parameters

A project that involves using a microcontroller to collect data from various sensors that measure environmental factors such as temperature, humidity, and air quality.

Creating wearable devices with microcontrollers

A project that involves designing and programming a microcontroller-based wearable device such as a fitness tracker or a smartwatch.

A microprocessor is the brain of a computer system, responsible for executing instructions and performing calculations.

Features and applications of microprocessors

- **Embedded systems**
  Microprocessors are commonly used as the main processing unit in embedded systems such as home appliances, automotive electronics, and industrial control systems.

- **Personal computers**
  Microprocessors are the main component in personal computers, responsible for performing the basic arithmetic and logic operations.

- **Mobile devices**
  Microprocessors are used in mobile devices such as smartphones and tablets to handle various tasks including processing user input, running applications, and managing power consumption.

- **Gaming consoles**
  Microprocessors are used in gaming consoles to render graphics, run game engines, and handle user input.

- **Digital signal processing**
  Microprocessors are used in digital signal processing applications such as image and audio processing to perform complex mathematical calculations.

Examples of microprocessor-based projects

To showcase practical examples of projects that use microprocessors

Autonomous robots

Use microprocessors to make decisions and control movements in real-time.

Smart home automation

Microprocessors are used to control and manage connected devices in homes and offices.

Automotive electronics

Microprocessors are used to control engine management systems, safety systems, navigation, and multimedia systems.

Video game consoles

Microprocessors handle graphics, sound, and gameplay logic in modern video game consoles.

Mobile devices

Microprocessors power all aspects of modern mobile devices, from processing power to wireless connectivity.

Main differences between microcontrollers and microprocessors

- **Architecture**
  Microcontrollers have a built-in memory and input/output peripherals in a single chip, while microprocessors require external memory and peripherals.
- **Cost**
  Microcontrollers are usually cheaper than microprocessors due to their integrated design.
- **Power consumption**
  Microcontrollers typically consume less power than microprocessors, making them suitable for battery-powered devices.
- **Complexity**
  Microprocessors are more complex and versatile, allowing for a wider range of applications and higher computational power.
- **Programming**
  Microcontrollers are often programmed using a specialized Integrated Development Environment (IDE), while microprocessors can be programmed using a variety of languages and tools.

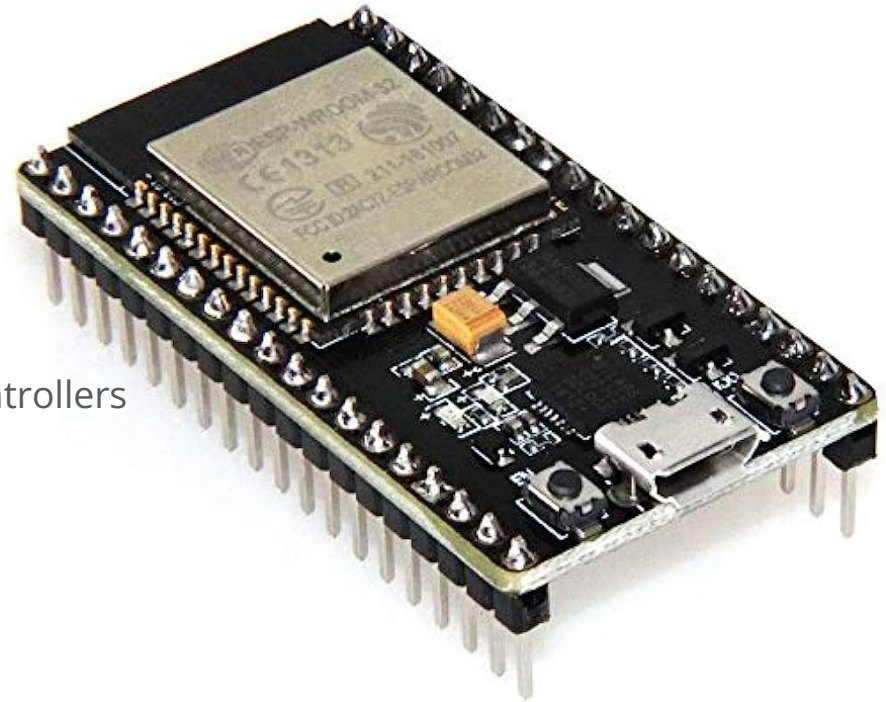
ESP32
The Microcontroller

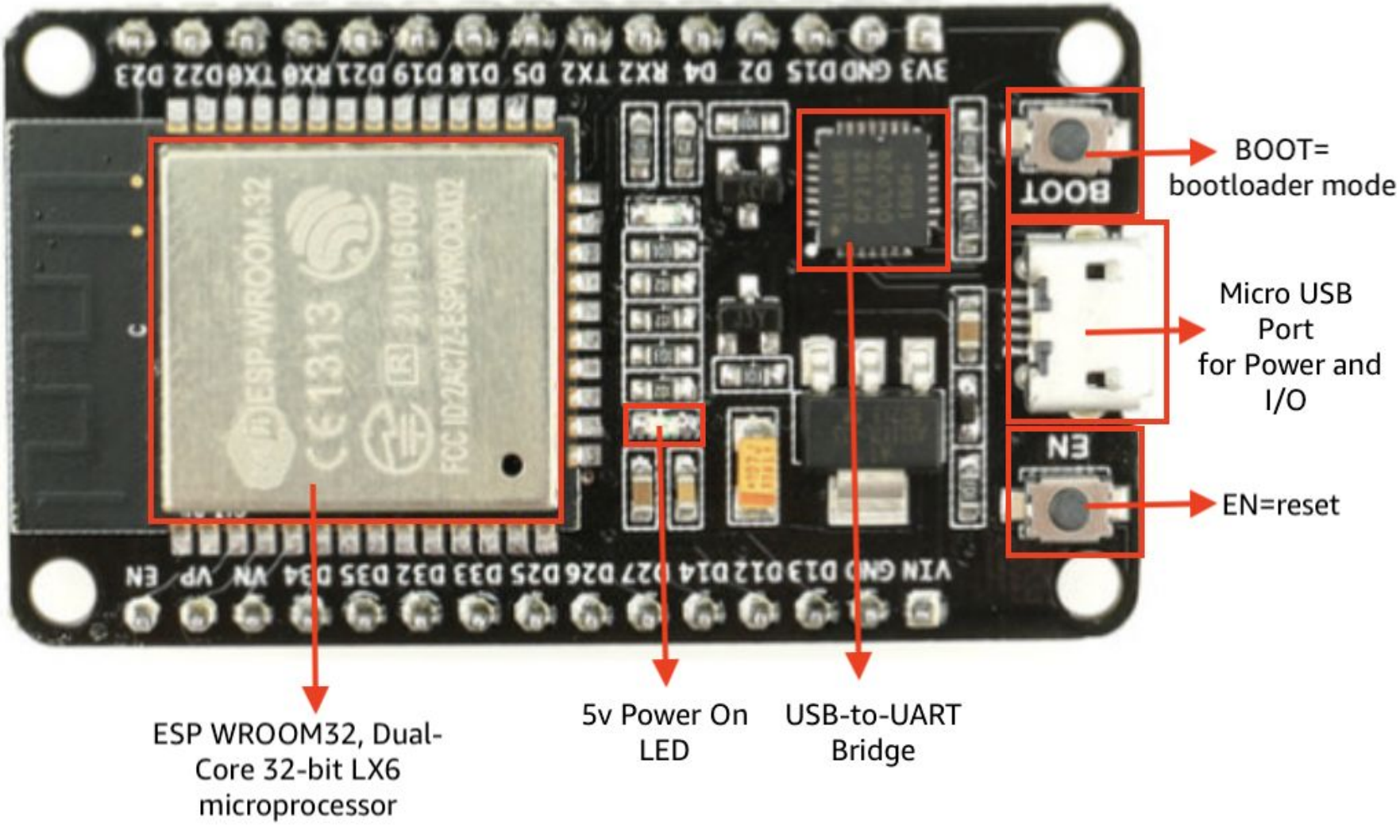
What is ESP32?

- A low-cost, low-power Microcontroller with an integrated Wi-Fi and Bluetooth

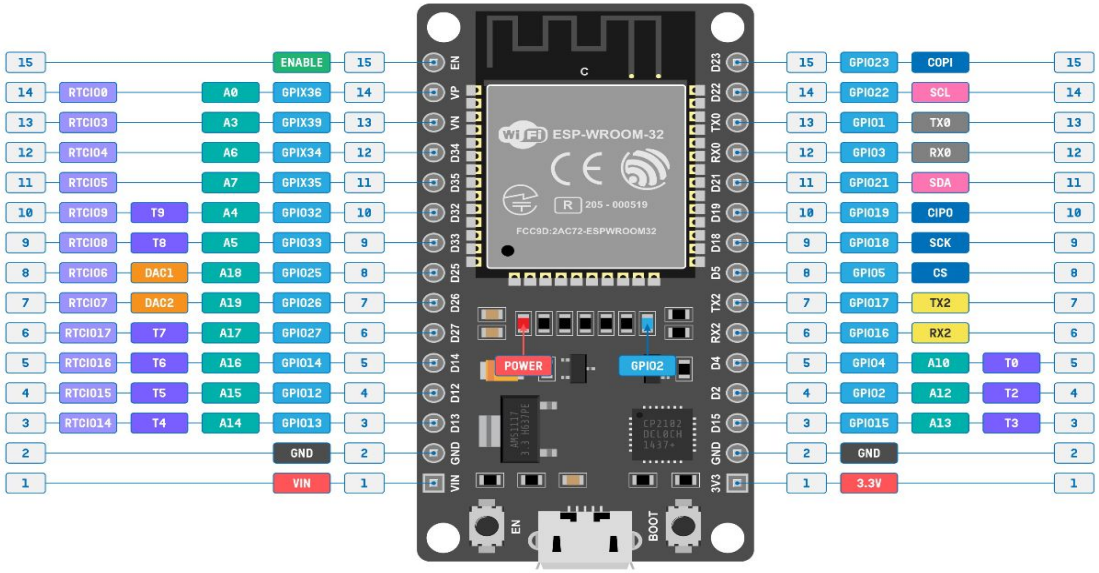
GPIO

- General Purpose Input Output
- Signal pin on an integrated circuit or board that can be used to perform digital input or output functions
- Standard interface used to connect microcontrollers to other electronic devices




BOOT=
bootloader mode
Micro USB
Port
for Power and
I/O
EN=reset
ESP WROOM32, Dual-
Core 32-bit LX6
microprocessor
5v Power On
LED
USB-to-UART
Bridge

DOIT ESP32 DEVKIT V1 PINOUT



PHYSICAL PIN	POSITIVE SUPPLY	DAC OUTPUTS	SPI PINS
CONTROL PINS	GROUND SUPPLY	TOUCH INPUTS	UART PINS
GPIO PINS	ADC INPUTS	I2C PINS	EXCLUDED PINS

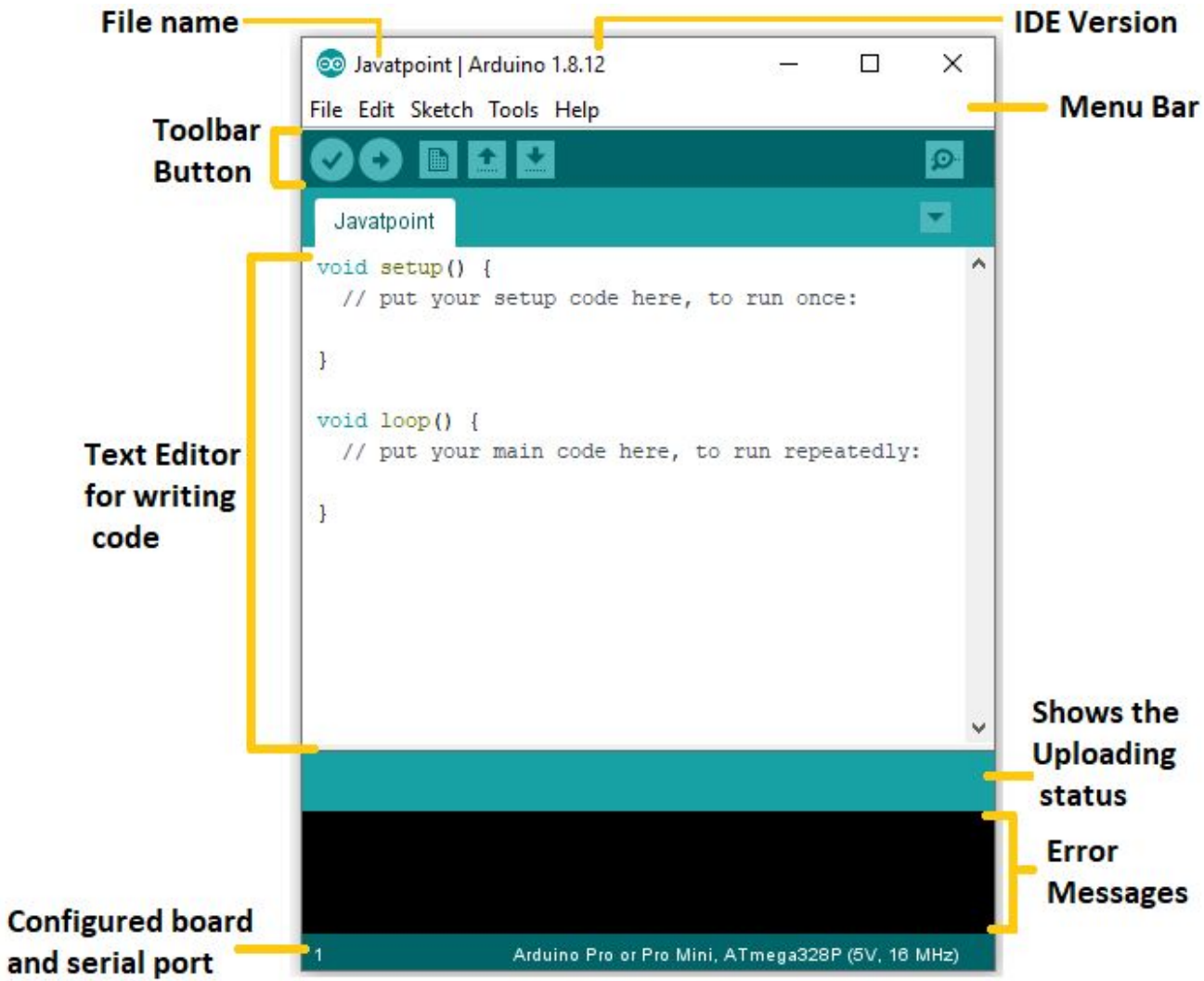
- GPIO pins **34**, **35**, **36** and **39** are input only.
- **TX0** and **RX0** (**Serial0**) are used for serial programming.
- **TX2** and **RX2** can be accessed as **Serial2**.
- Default SPI is VSPI. Both VSPI and HSPI pins can be set to any GPIO pins.
- All GPIO pins support PWM and interrupts.
- Builtin LED is connected to **GPIO2**.
- Some GPIO pins are used for interfacing flash memory and thus are not shown.

Rev. 0.1, 17-12-2022
Design: Vishnu Mohanan

Arduino IDE

Why IDE and What is an IDE?

- To program your boards, you need an IDE (Integrated Development Environment) to write your code
- Software tool that provides a comprehensive environment for writing, compiling, and debugging code in a single interface
- C and C++ like syntax
- The program or code written in the Arduino IDE is often called as sketching


File name
IDE Version
Javatpoint | Arduino 1.8.12
File Edit Sketch Tools Help
Menu Bar
Toolbar Button
Javatpoint
void setup() {
// put your setup code here, to run once:
}
void loop() {
// put your main code here, to run repeatedly:
}
Text Editor for writing code
Shows the Uploading status
Error Messages
Configured board and serial port
1
Arduino Pro or Pro Mini, ATmega328P (5V, 16 MHz)

Serial Printer

- A separate window that acts as a terminal that communicates by receiving and sending Serial Data
- Can be used as a debugging tool, testing out concepts or to communicate directly with the board

Serial Plotter

- Takes incoming serial data and displays them in a plot, giving us the ability to visualize the data in a plot that is updated in real time

Baud Rate

- Number of signal changes to the signal occurs per second when it passes through a transmission medium
- The higher a baud rate is the faster the data is sent/received

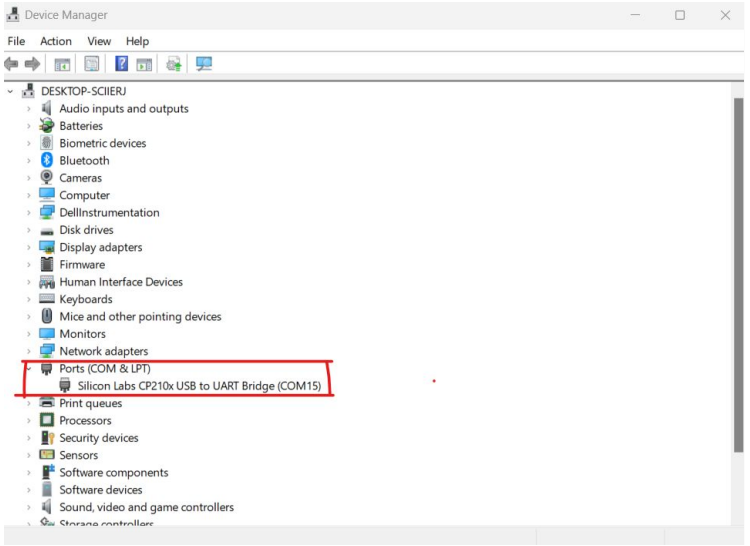
Microcontroller
(Session 2)

Connecting ESP32 & Testing

Installing Driver

- If you are unable to connect to the port, then follow the below steps:

1. Go to https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads
2. Download **CP210x Universal Windows Driver** and extract the folder
3. Go to Device Manager (hint - use search bar)
4. Right click and select update driver
5. Choose manual Update. Locate the folder
6. Click on Update



Test Code

```
1  void setup() {
2    // put your setup code here, to run once:
3    pinMode(2, OUTPUT);
4  }
5
6  void loop() {
7    // put your main code here, to run repeatedly:
8    delay(500);
9    digitalWrite(2,HIGH);
10   delay(500);
11   digitalWrite(2,LOW);
12
13 }
```

Umm, Thank You, I Guess?

See you in the next session!

Break Time

Microcontroller
(Session 2)

Test Code

```
1  void setup() {
2    // put your setup code here, to run once:
3    pinMode(2, OUTPUT);
4  }
5
6  void loop() {
7    // put your main code here, to run repeatedly:
8    delay(500);
9    digitalWrite(2,HIGH);
10   delay(500);
11   digitalWrite(2,LOW);
12
13 }
```

pinMode()

- Configures the specified pin to behave either as an input, or an output

delay()

- This number represents the time (measured in milliseconds).
- The program should wait until moving on to the next line of code when it encounters this function

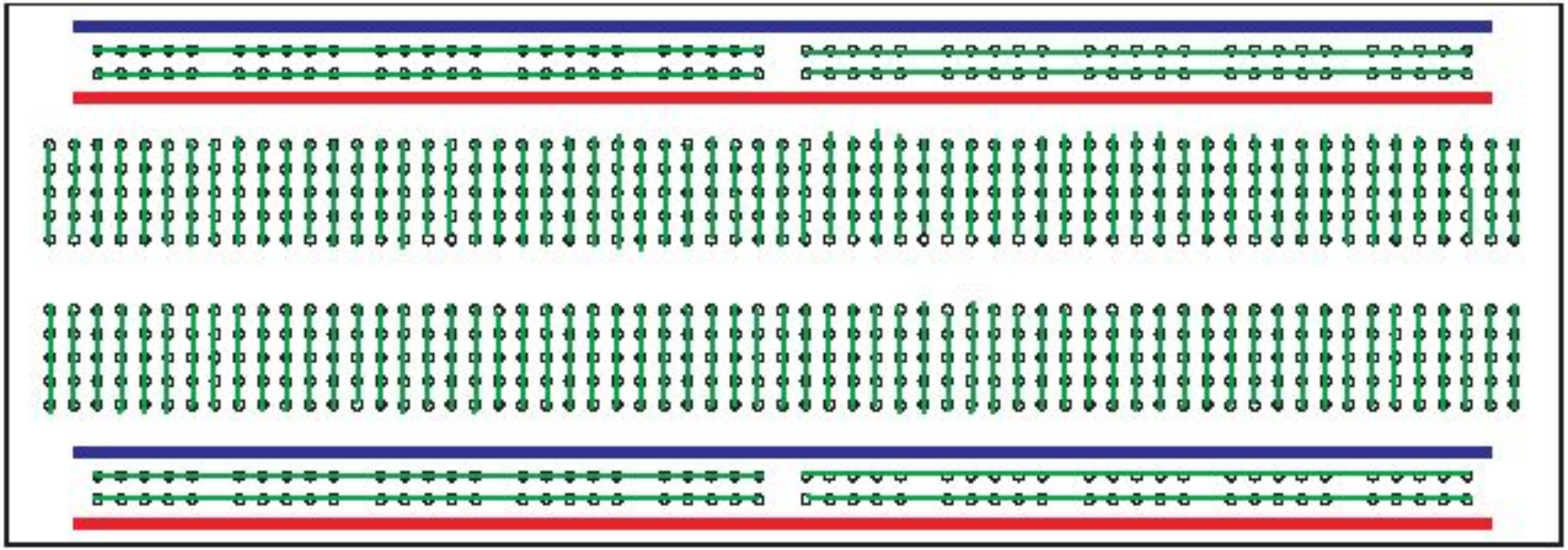
digitalWrite()

- Sets the pin to an high or low value that remains at exactly that value until digitalWrite is called for that pin again.

Quick Recap - Breadboard

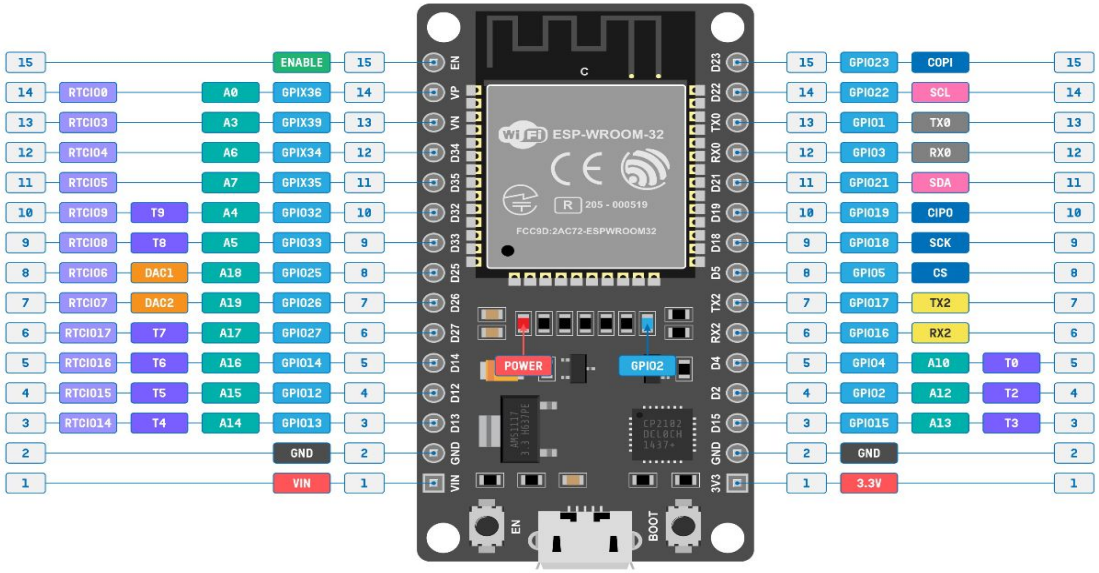
Breadboard

- A breadboard is a rectangular plastic board with a bunch of tiny holes in it.
- These holes let you easily insert electronic components to prototype, without having to solder each component



External LED Blink

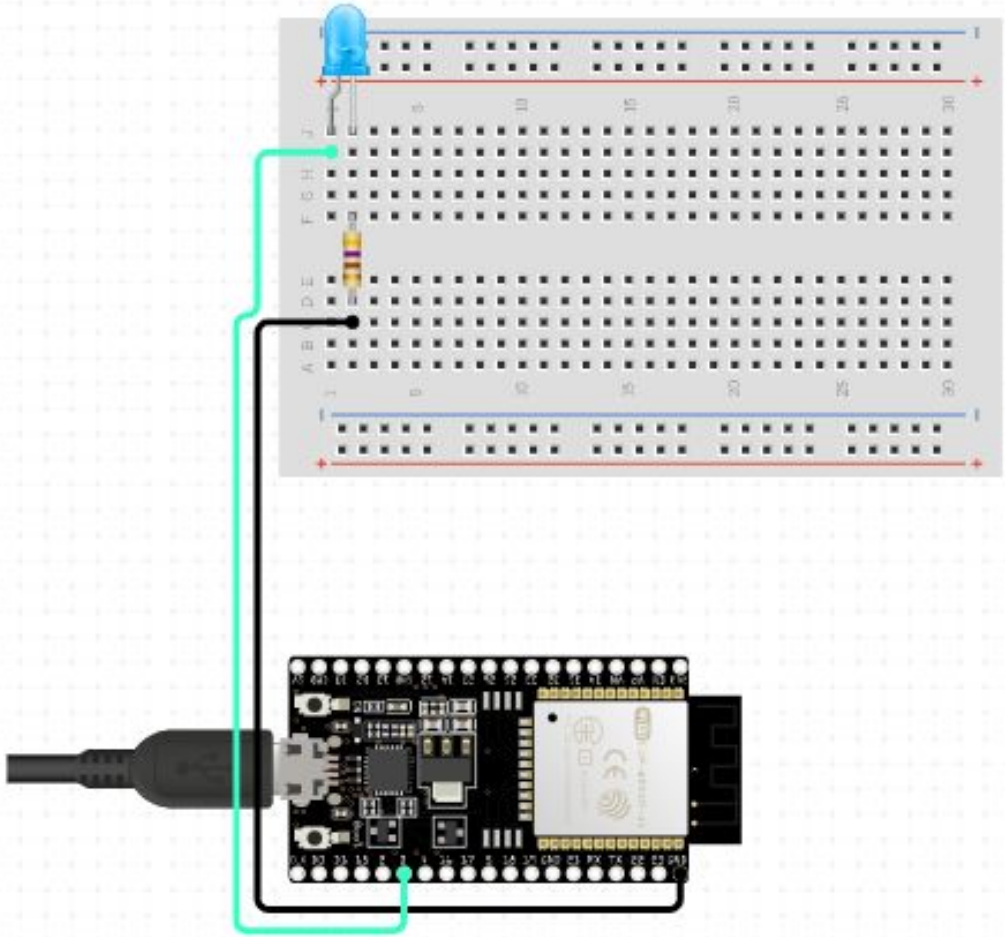
DOIT ESP32 DEVKIT V1 PINOUT



PHYSICAL PIN	POSITIVE SUPPLY	DAC OUTPUTS	SPI PINS
CONTROL PINS	GROUND SUPPLY	TOUCH INPUTS	UART PINS
GPIO PINS	ADC INPUTS	I2C PINS	EXCLUDED PINS

- GPIO pins **34**, **35**, **36** and **39** are input only.
- **TX0** and **RX0** (**Serial0**) are used for serial programming.
- **TX2** and **RX2** can be accessed as **Serial2**.
- Default SPI is VSPI. Both VSPI and HSPI pins can be set to any GPIO pins.
- All GPIO pins support PWM and interrupts.
- Builtin LED is connected to **GPIO2**.
- Some GPIO pins are used for interfacing flash memory and thus are not shown.

Rev. 0.1, 17-12-2022
Design: Vishnu Mohanan



```
#define LED 15
#define time 250
void setup() {
  // put your setup code here, to run once:
  pinMode(LED, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(LED, HIGH);
  delay(time);
  digitalWrite(LED, LOW);
  delay(time);
}

```

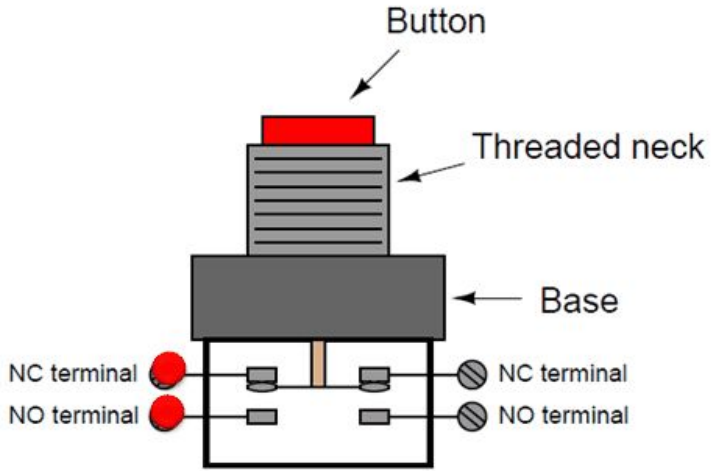
Push button

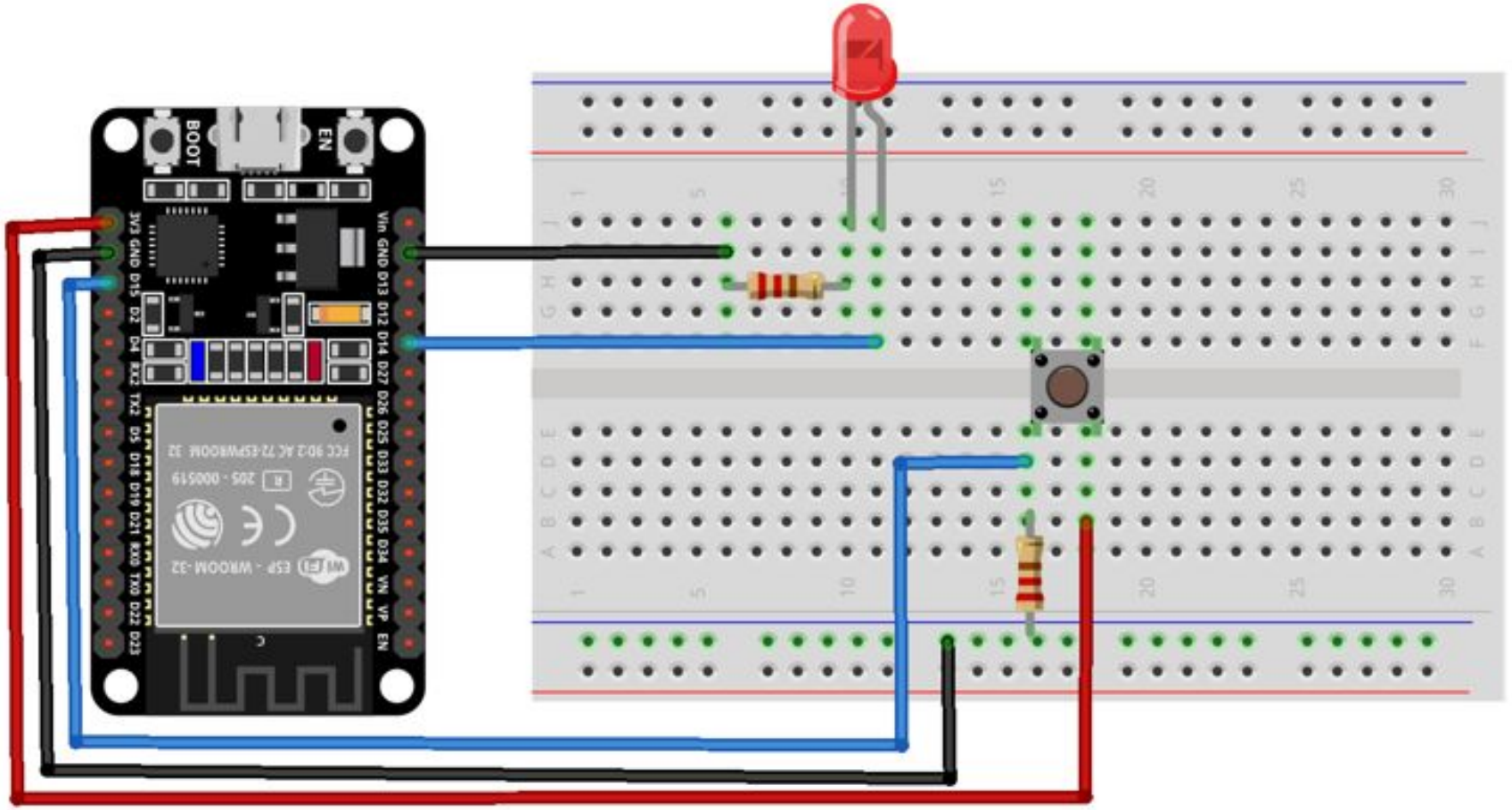
Push button

- A mechanical device used to control an electrical circuit in which the operator manually presses a button to activate an internal switching mechanism

Working Principle

- There is an electromagnet adsorption device inside the button.
- When the button is pressed down, the electromagnet is energized to generate magnetism, and the circuit is connected or disconnected by the adsorption device to realize functions such as remote control circuit.




BOOT
EN
3V3 GND D15 D2 D4 RX2 TX2 D5 D18 D19 D21 RX0 TX0 D22 D23
Vin GND D13 D12 D14 D27 D26 D25 D33 D32 D35 D34 VN VP EN
ESP - WROOM-32

```
#define LED 15
#define time 250
#define button 23

int buttonvalue = 0;
void setup() {
  // put your setup code here, to run once:
  pinMode(button, INPUT);
  pinMode(LED, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  buttonvalue = digitalRead(button);
  if (buttonvalue == HIGH){
    digitalWrite(LED, HIGH);
  }
  else{
    digitalWrite(LED, LOW);
  }
}
```

Toggle LED

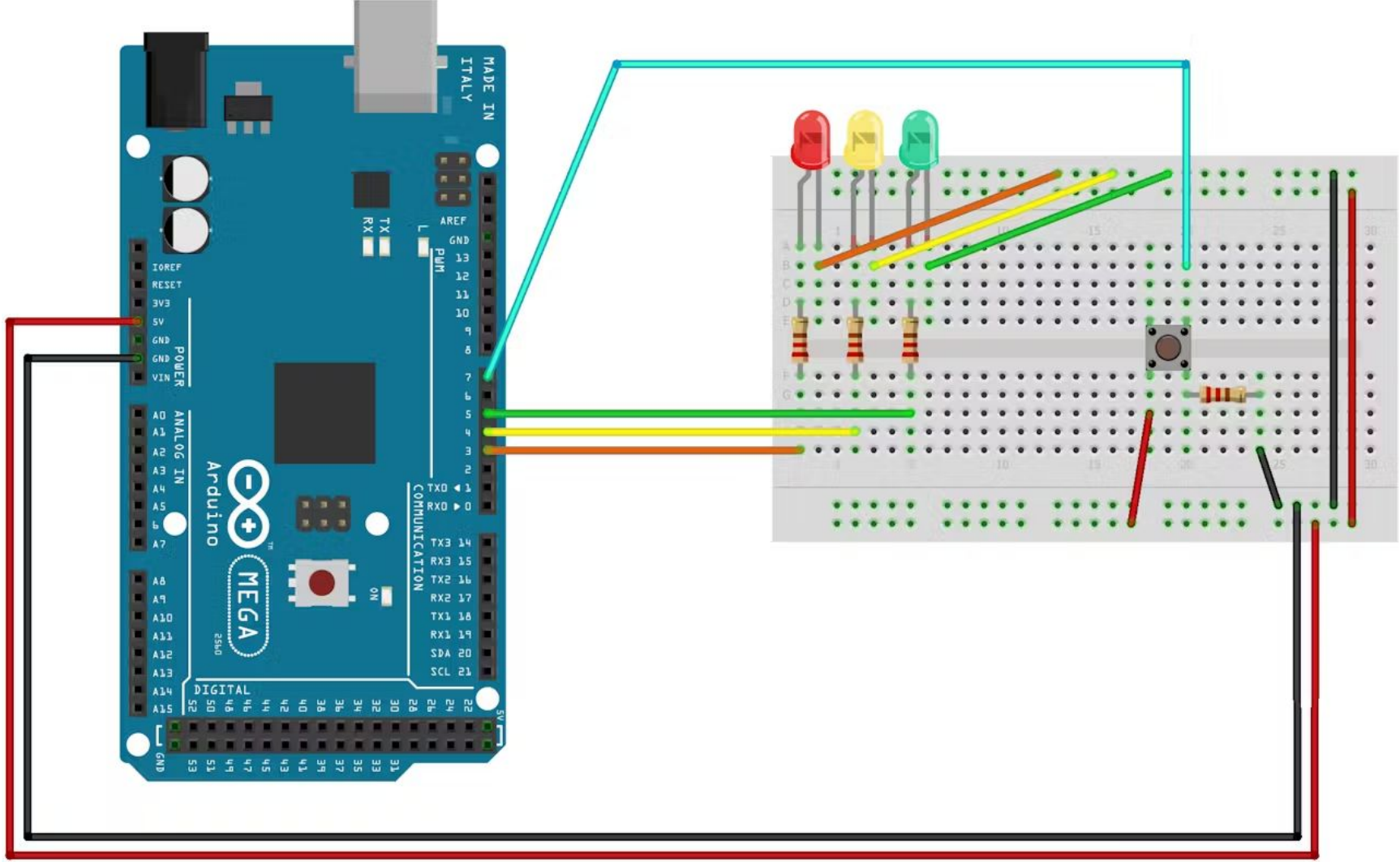
```
1   #define LED 2
2
3   #define time 250
4   #define button 23
5   int buttonValue = 0;
6
7   int lastButtonValue = 0;
8   int ledValue = LOW;
9
10  void setup() {
11    // put your setup code here, to run once:
12    pinMode(button, INPUT);
13    pinMode(LED, OUTPUT);
14    digitalWrite(LED, ledValue);
15  }
```

```
17  void loop() {
18    // put your main code here, to run repeatedly:
19    buttonValue = digitalRead(button);
20    if (lastButtonValue != buttonValue){
21      if (buttonValue == LOW){
22        ledValue = !ledValue;
23        digitalWrite(LED, ledValue);
24        delay(time);
25      }
26      lastButtonValue = buttonValue;
27    }
28  }
29
```

Traffic Light

Problem Statement

Given 3 LED and 1 push button, make a traffic light system which changes the color everytime a push button is pressed.


MADE IN ITALY
Arduino
MEGA
2560
POWER
ANALOG IN
DIGITAL
COMMUNICATION
PWM
IOREF
RESET
3V3
5V
GND
GND
VIN
AREF
TX
RX
ON
TXD
RXD
TX3
RX3
TX2
RX2
TX1
RX1
SDA
SCL

```
#define LEDR 15
#define LEDG 18
#define LEDY 19

#define time 250
#define button 23
int buttonValue = 0;

int lastButtonValue = 0;
int ledRValue = HIGH;
int ledGValue = LOW;
int ledYValue = LOW;

void setup() {
  // put your setup code here, to run once:
  pinMode(button, INPUT);
  pinMode(LEDR, OUTPUT);
  pinMode(LEDG, OUTPUT);
  pinMode(LEDY, OUTPUT);
  digitalWrite(LEDR, ledRValue);
  digitalWrite(LEDG, ledGValue);
  digitalWrite(LEDY, ledYValue);
}

void loop() {
  // put your main code here, to run repeatedly:
  buttonValue = digitalRead(button);
  if (lastButtonValue != buttonValue){
    if (buttonValue == LOW){
      ledRValue = !ledRValue;
      ledYValue = !ledYValue;
      ledGValue = !ledGValue;
      digitalWrite(LEDR, ledRValue);
      digitalWrite(LEDY, ledYValue);
      digitalWrite(LEDG, ledGValue);
      delay(time);
    }
    lastButtonValue = buttonValue;
  }
}
```

Morse Code

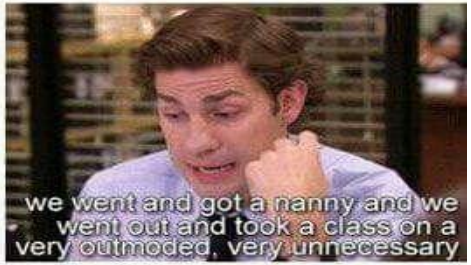

[clicking]


[clicking]


Stop it.
JIM: Stop what?

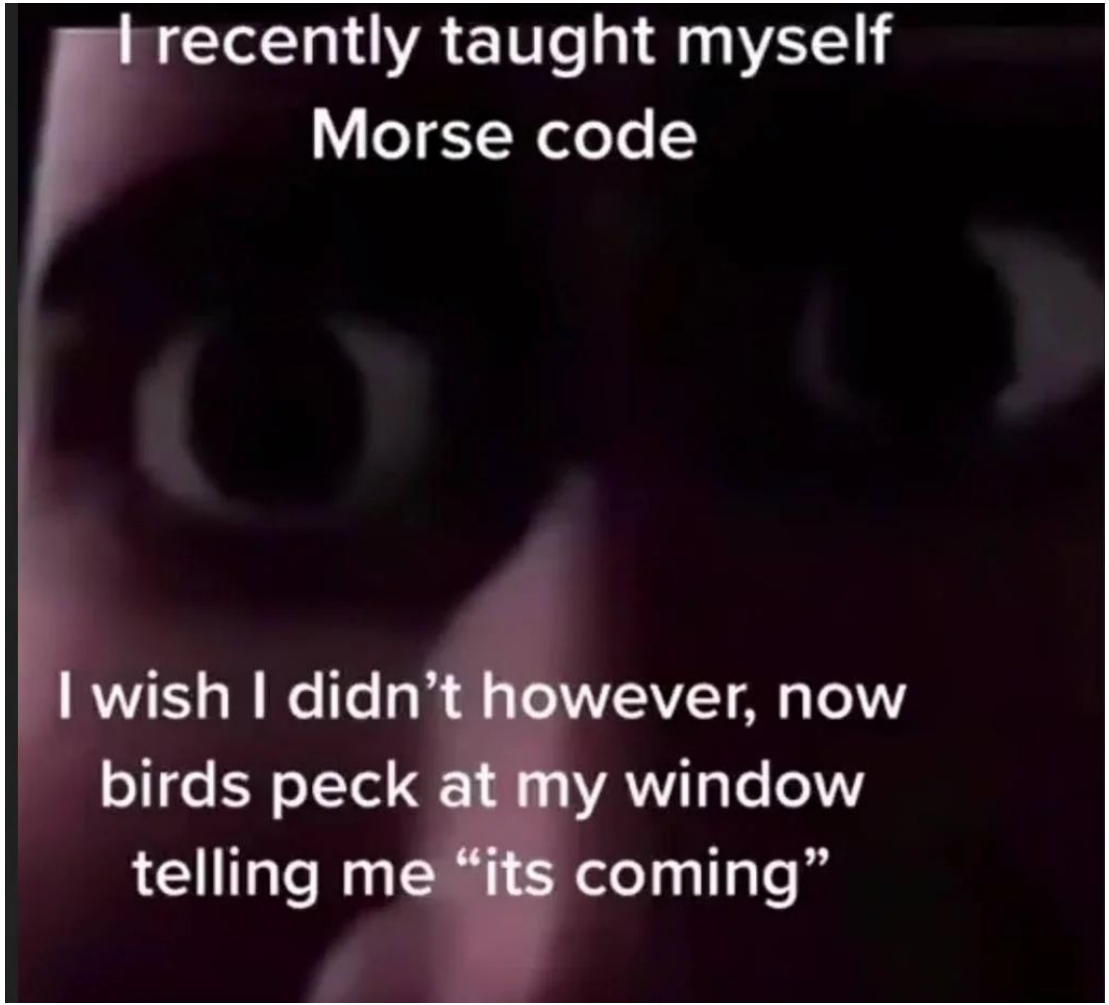

You're talking about me in Morse Code. Joke's on you because
know Morse Code. Ha!

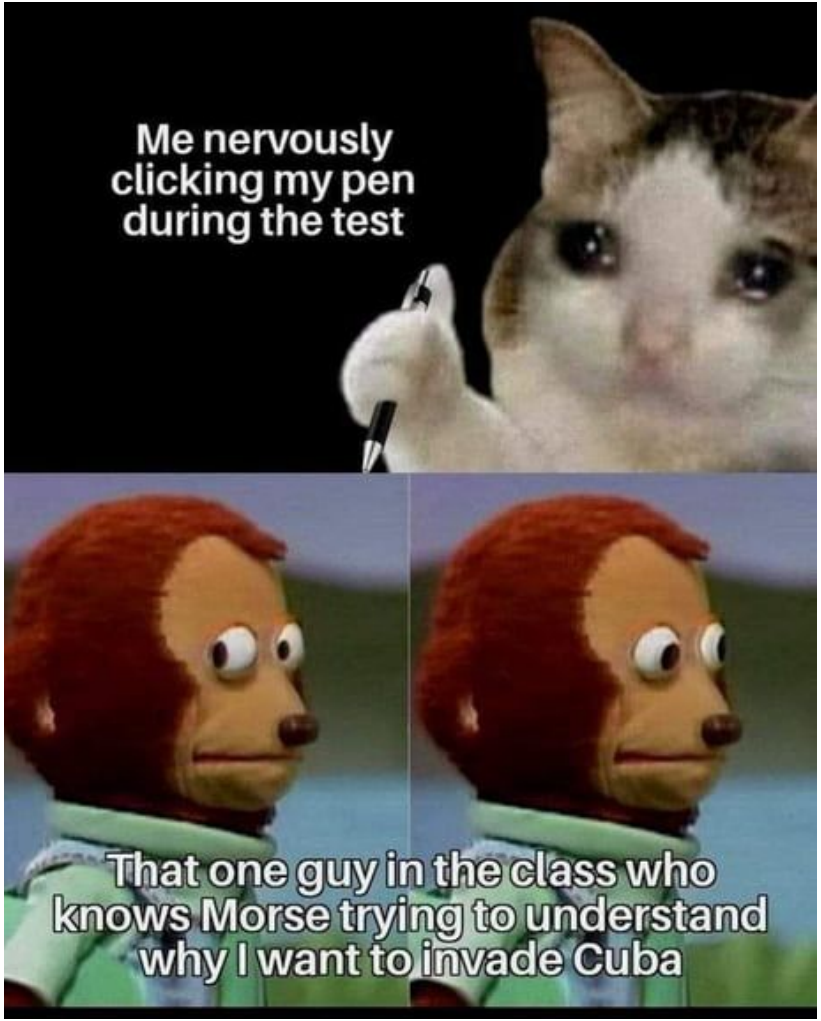

Yeah, that's what we're doing. In our very limited free time and with our very limited budget.


we went and got a nanny and we went out and took a class on a very outmoded, very unnecessary


form of communication just so we could talk about you in front of you.


Yep. That's exactly what we did


I recently taught myself
Morse code
I wish I didn't however, now
birds peck at my window
telling me "its coming"


Me nervously clicking my pen during the test
That one guy in the class who knows Morse trying to understand why I want to invade Cuba

What is morse code?

- Morse code is a method of communication that uses a series of dots and dashes to represent letters, numbers, and other characters.

- Each combination is unique.

What is morse code?

- Morse code is a method of communication that uses a series of dots and dashes to represent letters, numbers, and other characters.
- Each combination is unique.

Morse Code and LED?

- To transmit Morse code, you can use various methods, such as a telegraph key or a flashlight.
- Morse code is a binary code – ON or OFF – like an LED

What is morse code?

- Morse code is a method of communication that uses a series of dots and dashes to represent letters, numbers, and other characters.
- Each combination is unique.

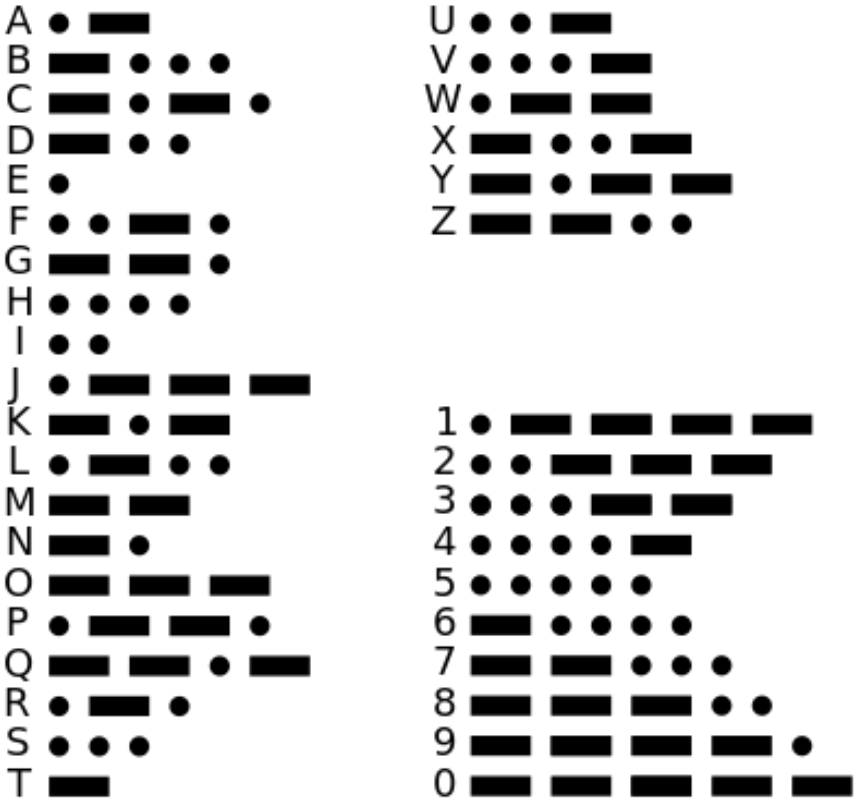
Morse Code and LED?

- To transmit Morse code, you can use various methods, such as a telegraph key or a flashlight.
- Morse code is a binary code – ON or OFF – like an LED

Activity

- **Given an input letter (A-Z, a-z) or number (0-9), blink the LED in its morse code pattern**

International Morse Code



- The length of dot is one unit
- The length of dash is three units
- The space between parts of same letter is one unit
- The space between two letters is three units
- The space between two words is seven units

```
1   const int ledPin = 13;  // Pin number for the LED
2   const int dotDuration = 200;  // Duration of a dot in milliseconds
3
4   void setup() {
5     Serial.begin(9600);  // Initialize serial communication at 9600 bps
6     pinMode(ledPin, OUTPUT);  // Set the LED pin as output
7   }
8
```

```
String getMorseCode(char letter) {
  // Define the Morse code patterns for each letter
  // You can add more letters and symbols as needed
  switch (letter) {
    case 'A':
    case 'a':
      return ".-";
    case 'B':
    case 'b':
      return "-...";
    case 'C':
    case 'c':
      return "-.-.";
    case 'D':
    case 'd':
      return "-..";
    case 'E':
    case 'e':
      return ".";
```

```
132         case '8':
133             return "---..";
134         case '9':
135             return "----.";
136         // Add more cases for other letters or symbols
137         default:
138             return "";  // Return an empty string for unsupported characters
139     }
140 }
```

```
void blinkMorseCode(char letter) {
  // Convert the letter to Morse code pattern
  String morseCode = getMorseCode(letter);

  // Blink the LED based on the Morse code pattern
  for (int i = 0; i < morseCode.length(); i++) {
    if (morseCode[i] == '.') {
      digitalWrite(ledPin, HIGH);  // Turn on the LED
      delay(dotDuration);  // Duration for a dot
    } else if (morseCode[i] == '-') {
      digitalWrite(ledPin, HIGH);  // Turn on the LED
      delay(dotDuration * 3);  // Duration for a dash (3 times a dot)
    }
    digitalWrite(ledPin, LOW);  // Turn off the LED
    delay(dotDuration);  // Pause between dots and dashes
  }
}
```

```
void loop() {
  if (Serial.available() > 0) {  // Check if data is available to read
    char input = Serial.read();  // Read the incoming data
    blinkMorseCode(input);  // Call the function to blink Morse code
  }
}
```

Umm, Thank You, I Guess?

See you in the next session!

Break Time

Microcontroller
(Session 3)

Potentiometer

What are potentiometers?

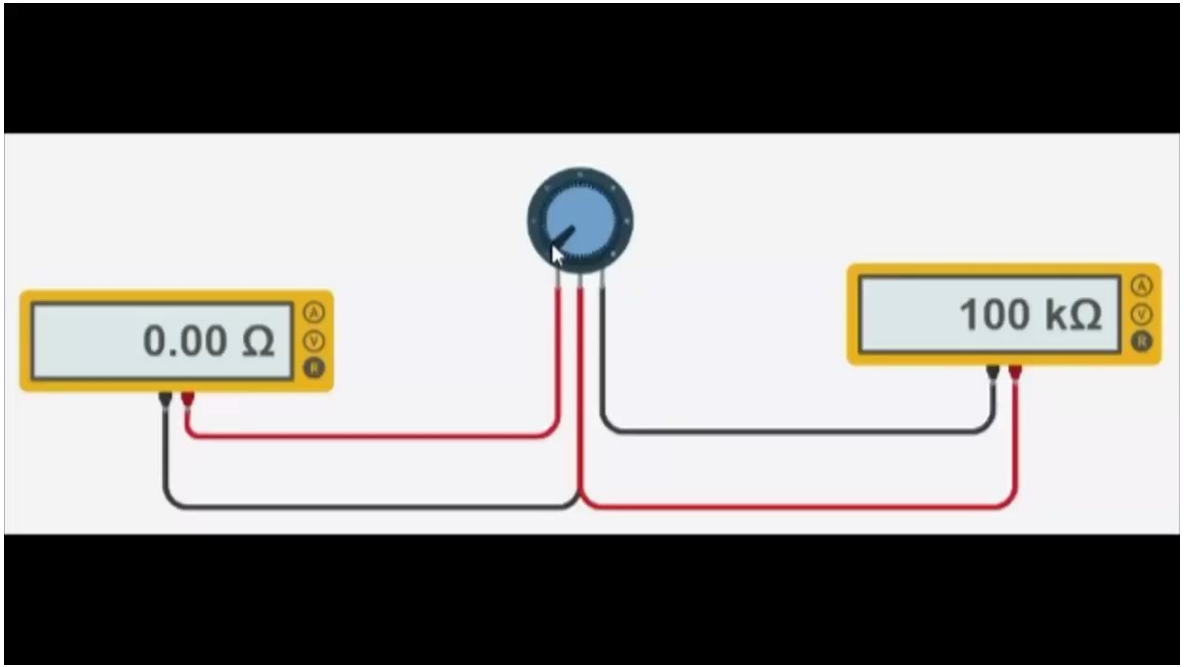
- *"A potentiometer is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider."* - Wikipedia

The Virgin Standard Pot

The Chad Fender Pure Vintage 250K Split Shaft Potentiometer

But what are potentiometers actually !?
(I am bored of this theory)

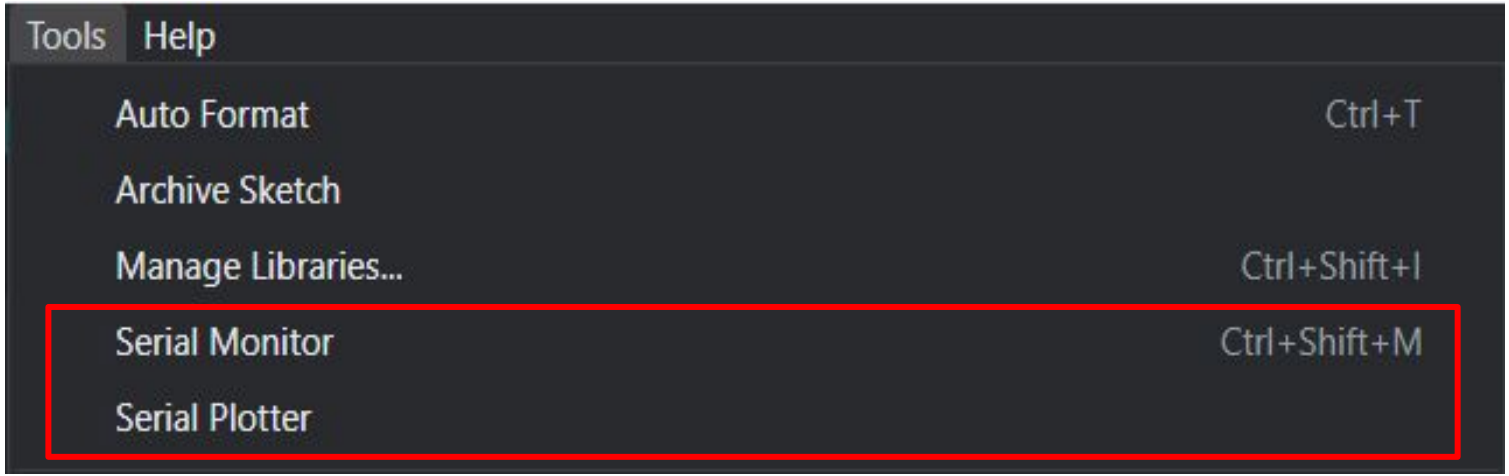
- In short: Most Common Variable Resistor (more in electronics class)
- The three terminals: The end two terminals have fixed resistance
    - The terminal in between (in relation to other) provide variable resistance

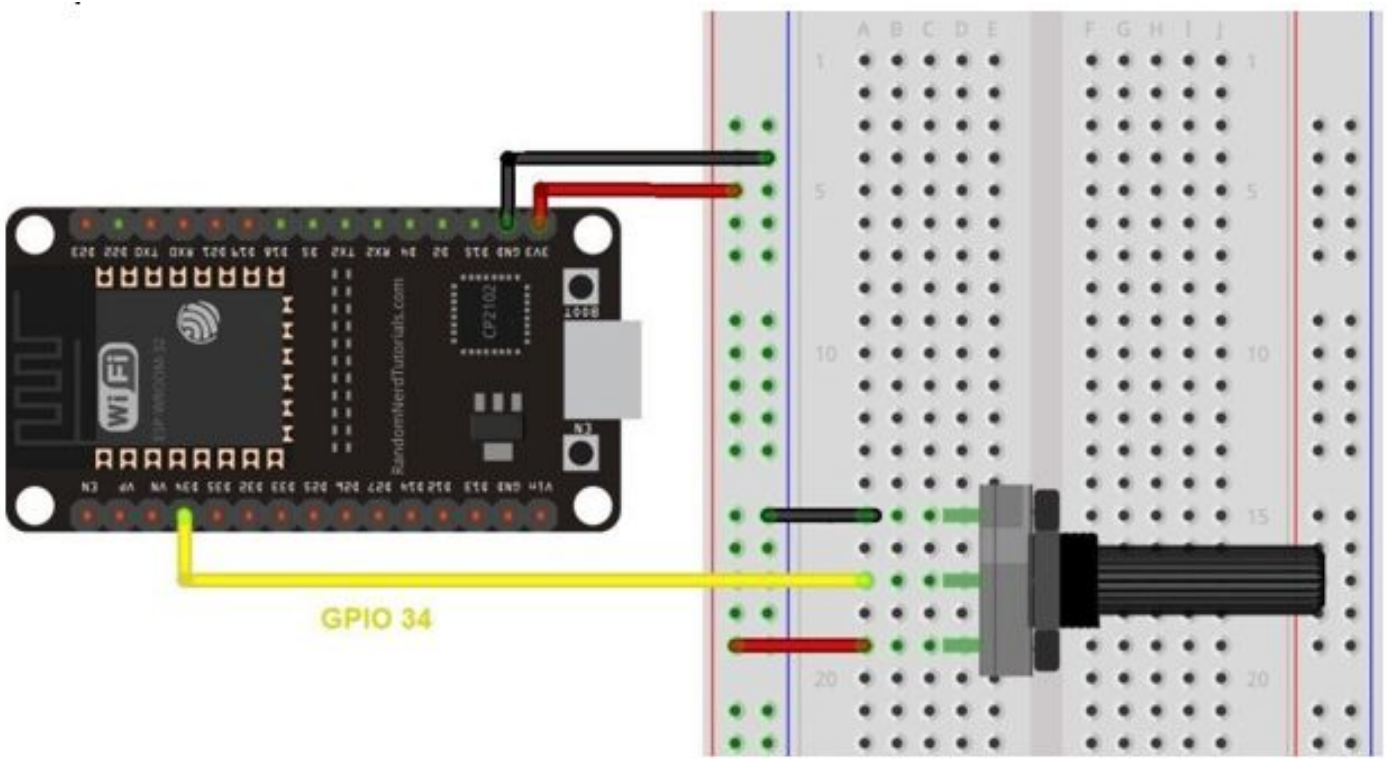


Potentiometer & ESP32

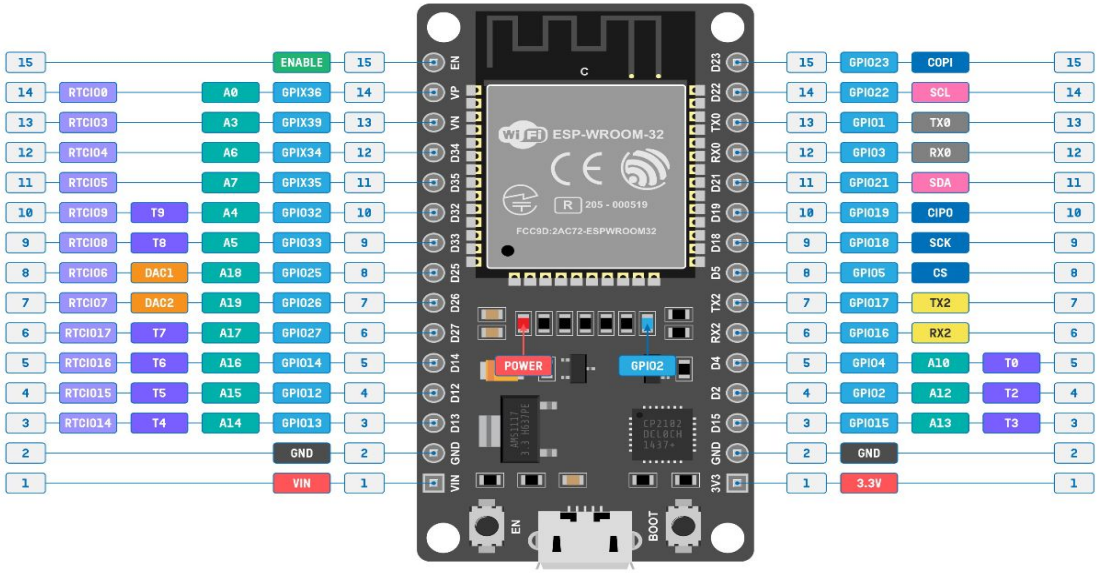
Potentiometers and ESP32

- The ESP32 will take the input from Potentiometer
    - Print the discrete values on Serial Monitor
    - Plot a graph using the Serial Plotter




GPIO 34

DOIT ESP32 DEVKIT V1 PINOUT



PHYSICAL PIN	POSITIVE SUPPLY	DAC OUTPUTS	SPI PINS
CONTROL PINS	GROUND SUPPLY	TOUCH INPUTS	UART PINS
GPIO PINS	ADC INPUTS	I2C PINS	EXCLUDED PINS

- GPIO pins **34**, **35**, **36** and **39** are input only.
- **TX0** and **RX0** (**Serial0**) are used for serial programming.
- **TX2** and **RX2** can be accessed as **Serial2**.
- Default SPI is VSPI. Both VSPI and HSPI pins can be set to any GPIO pins.
- All GPIO pins support PWM and interrupts.
- Builtin LED is connected to **GPIO2**.
- Some GPIO pins are used for interfacing flash memory and thus are not shown.

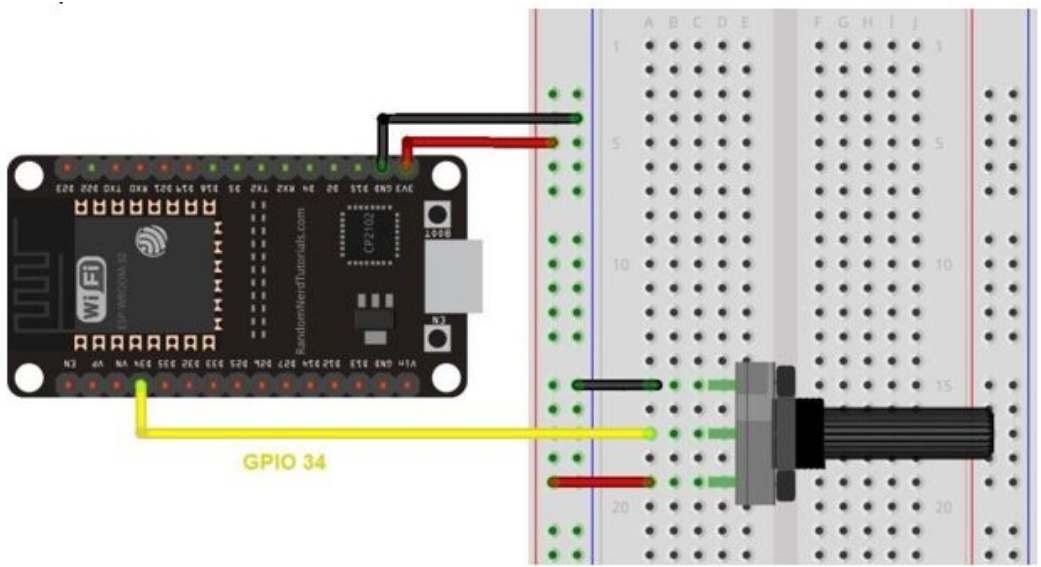
Rev. 0.1, 17-12-2022
Design: Vishnu Mohanan

```
#define PIN 4
int ADC = 0;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);

}

void loop() {
  // put your main code here, to run repeatedly:
  ADC = analogRead(PIN);
  Serial.print("ADC VALUE = ");
  Serial.println(ADC);

}
```

Potentiometer & LEDs

Potentiometers and ESP32

- Remember this circuit?
  - Now add a LED in series to it.
  - You can use it to control the brightness of the LED.
  - Build Your Own Circuit!



```
#define PIN 4
#define LED 22
int ADC = 0;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(LED, OUTPUT);

}

void loop() {
  // put your main code here, to run repeatedly:
  ADC = analogRead(PIN);
  Serial.print("ADC VALUE = ");
  Serial.println(ADC);

  // scales it to brightness (value between 0 and 255)
  int brightness = map(ADC, 0, 4095, 0, 255);

  analogWrite(LED, brightness);
  delay(100);

}
```

LCDs

LCDs

But Before that - Theory

LCD (JHD162A) with ESP32 Dev Module

There are two ways to do this:

1. Directly with ESP32
2. Using I2C Adapter*

* we will talk about I2C before we move into the second method

LCD with ESP32 (Directly)

Circuit Diagram

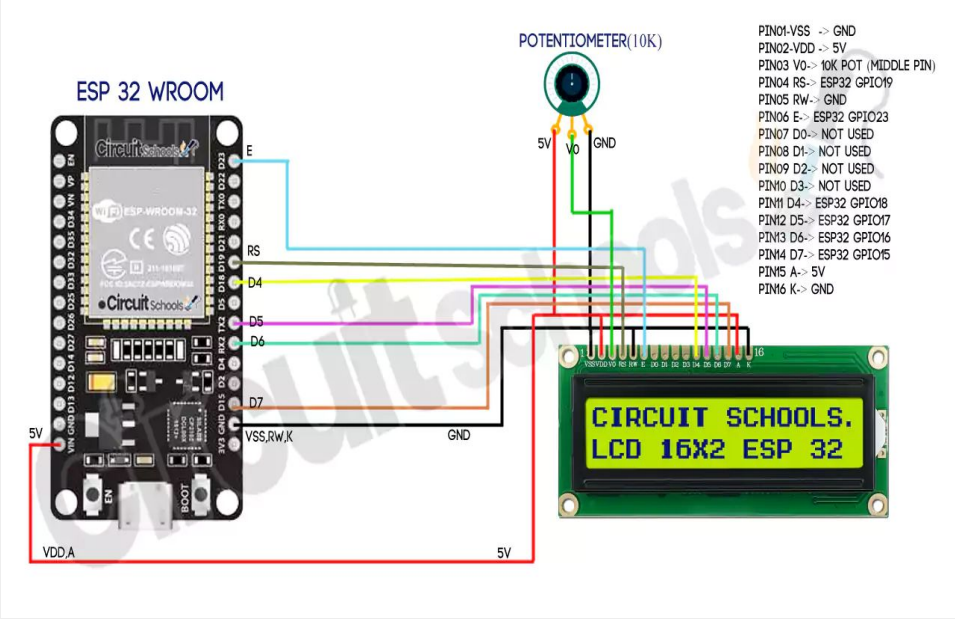
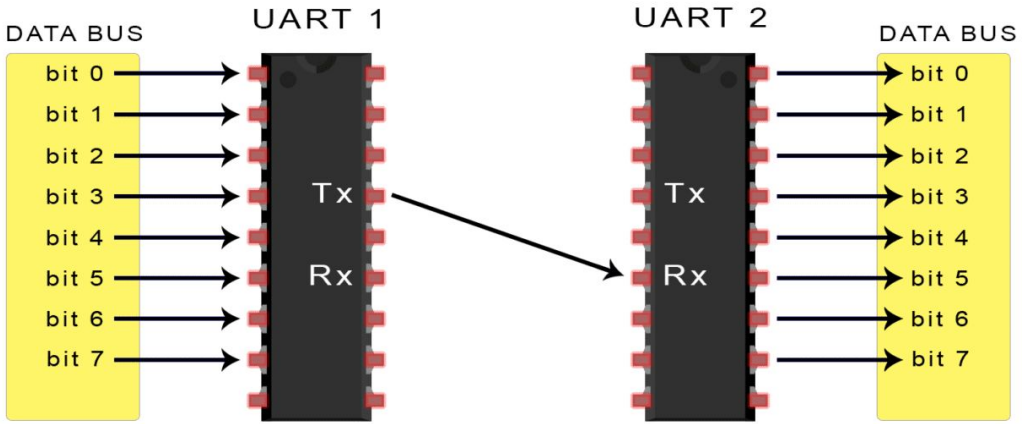


Diagram Ref: https://rb.gy/3cyrt

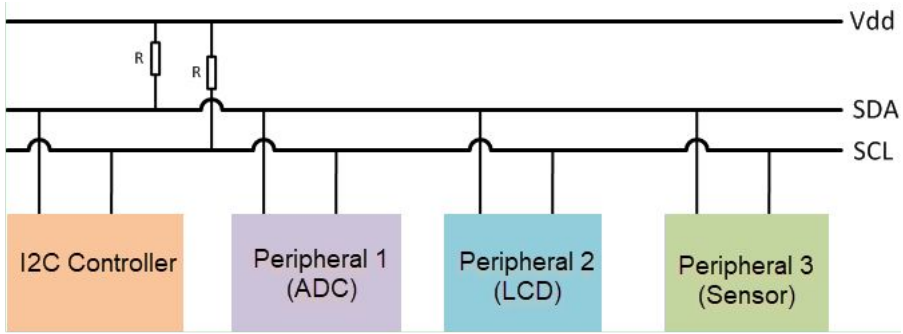
UART

- UART stands for Universal Asynchronous Receiver/Transmitter.
- It’s not a communication protocol like SPI and I2C, but a physical circuit in a microcontroller, or a stand-alone IC.
- **UART’s main purpose is to transmit and receive serial data.**



About I2C (Inter-Integrated Circuit)

A communication protocol used to communicate with each other using just two wires: a serial data line (**SDA**) and a serial clock line (**SCL**), connected in a bus configuration. Design involves a master device, which initiates and controls the communication, and slave devices, which respond to commands and provide data.



About I2C Adapter

Using I2C Adapter, we can use this protocol in our circuit.

Due to the Master-Slave relationship and the bus configuration, the process of connecting and communicating with multiple devices is simplified by using a minimal number of wires.

It also makes it easier if in future we want to add more devices to the circuit.

Hence, in the case of the LCD, we will use this more often.

I2C ADAPTER

LCD with ESP32 (using I2C adapter)

Circuit Diagram

SDA -> 21
SCL -> 22

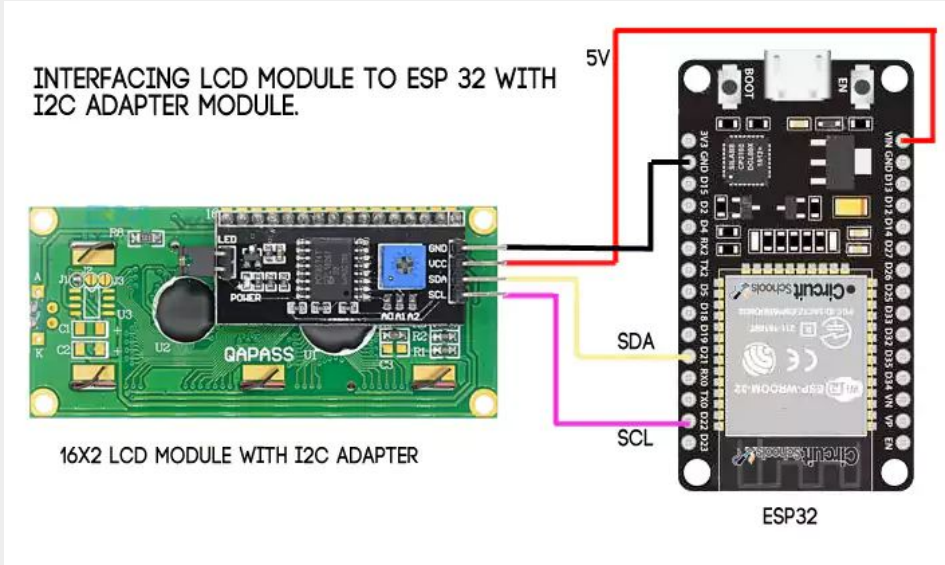


Diagram Ref: https://rb.gy/3cyrt

https://rb.gy/41oeg

```ino
#include <LiquidCrystal_I2C.h>

// Set up LCD JDH 162A 16x2 display
// Create the lcd object
LiquidCrystal_I2C lcd(0x27, 16, 2); // set the LCD address to 0x27 for a 16 chars and 2 line display
// 0x27 is a default address. If it does not work, try 0x3F


void setup() {
  lcd.begin(); // initialize the lcd
  // Turn on the backlight on LCD.
  lcd.backlight();
  //print something .....

  lcd.setCursor(0,1);
}


void loop() {

}
```

```ino
#include <LiquidCrystal_I2C.h>

// Set up LCD JDH 162A 16x2 display
// Create the lcd object
LiquidCrystal_I2C lcd(0x27, 16, 2); // set the LCD address to 0x27 for a 16 chars and 2 line display
// 0x27 is a default address. If it does not work, try 0x3F

int count = 0;

void setup() {
  lcd.begin(); // initialize the lcd
  // Turn on the backlight on LCD.
  lcd.backlight();
  lcd.print("Hello YTS!");
  lcd.setCursor(0,1);
  lcd.print("Demo");
  delay(2000);
}


void loop() {
 // ESP32 LCD-1602-I2C
  lcd.clear();// clear previous values from screen
  lcd.print("How are you?");
  lcd.setCursor(0,1);
  lcd.print("Counting:");
  lcd.setCursor(11,1);
  lcd.print(count);
  count++;
  delay(200);
  delay (100);
}
```

Umm, Thank You, I Guess?

See you in the next session!

Break Time

Microcontroller
(Session 4)

Distance & Sound

Some Facts

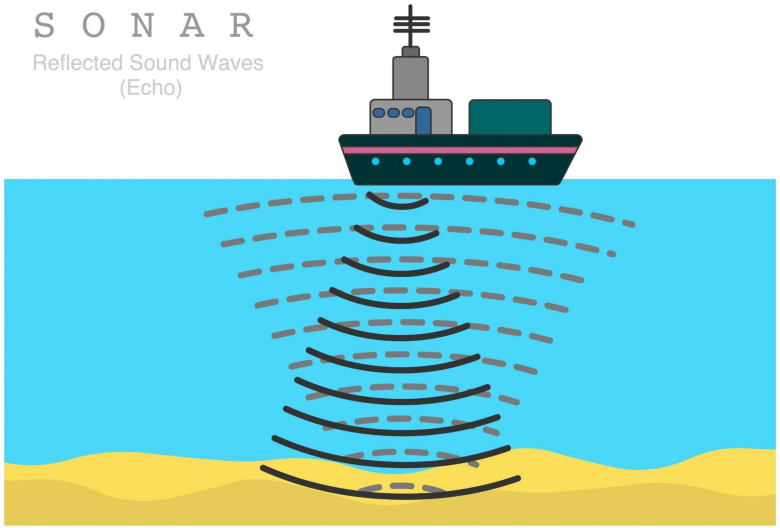
- Speed of sound in air is 340 m/s
- Audible range of human hearing is at 20 kHz. Sounds in the range higher than this are called ultrasonic
- Distance = Time * Speed

Some Facts

- Speed of sound in air is 340 m/s
- Audible range of human hearing is at 20 kHz. Sounds in the range higher than this are called ultrasonic
- Distance = Time * Speed

Some Facts

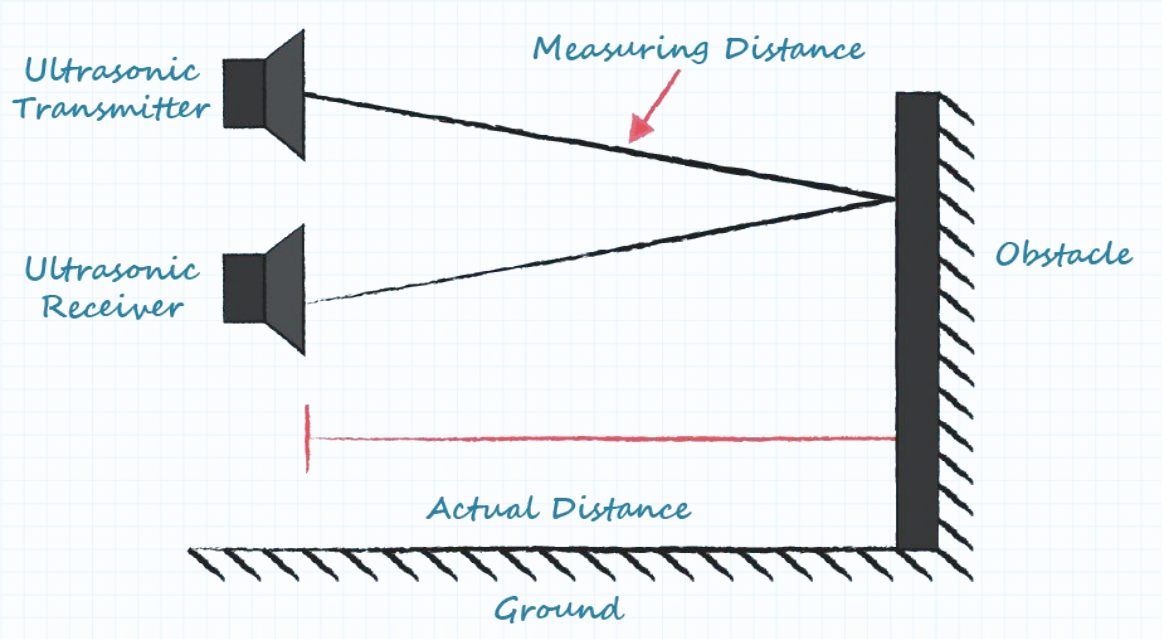
- Speed of sound in air is 340 m/s
- Audible range of human hearing is at 20 kHz. Sounds in the range higher than this are called ultrasonic
- Distance = Time * Speed

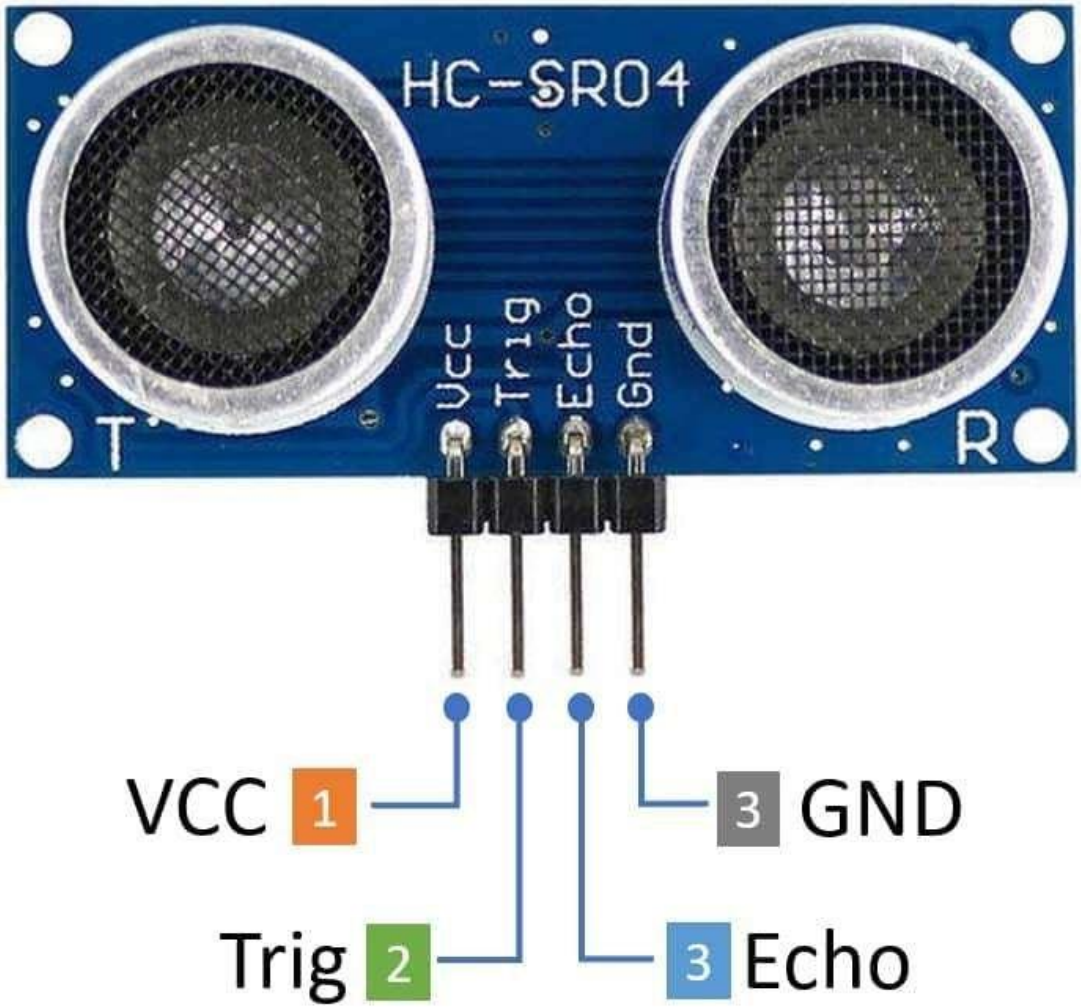


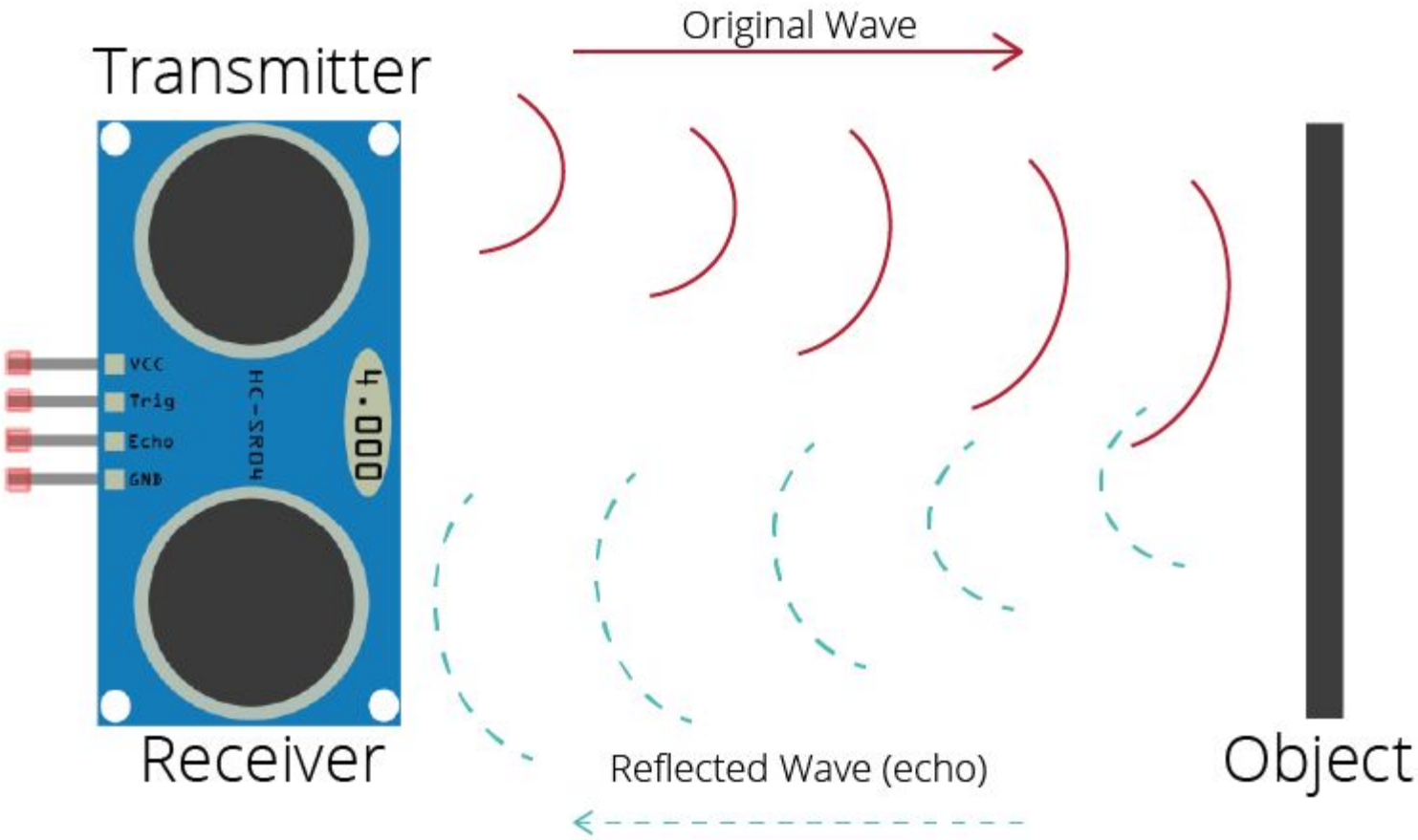
Ultrasonic Sensor

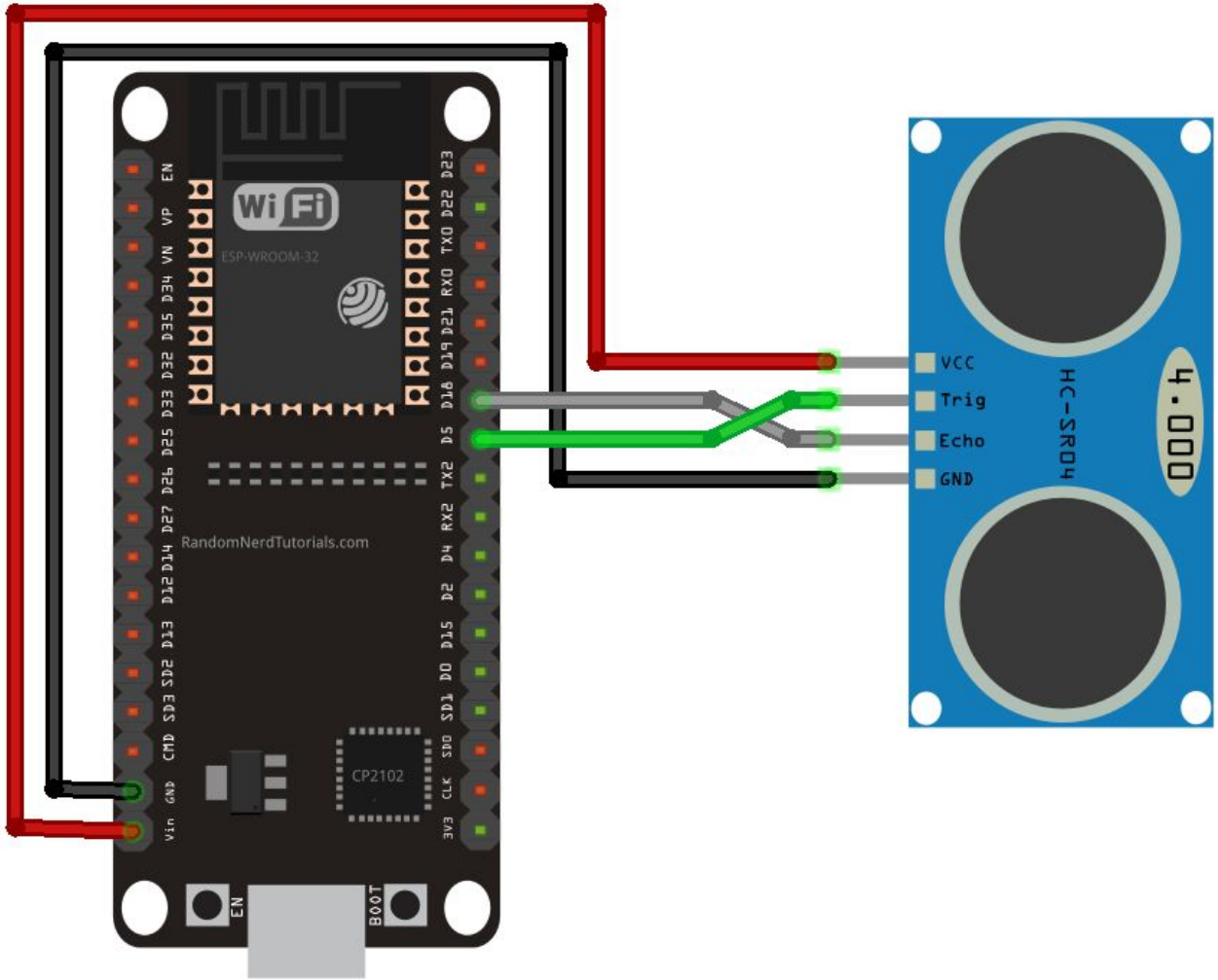
Ultrasonic

- Ultrasonic sensors can measure distance and detect the presence of an object without making physical contact.
- HC-SR04 can read from 2cm to 400cm with an accuracy of 0.3cm




HC-SR04
Vcc
Trig
Echo
Gnd
T
R
VCC 1
Trig 2
3 Echo
3 GND


Transmitter
Original Wave
VCC
Trig
Echo
GND
HC-SR04
Receiver
Reflected Wave (echo)
Object


ESP-WROOM-32
RandomNerdTutorials.com
CP2102
EN
BOOT
VCC
Trig
Echo
GND
HC-SR04

```
#define trigPin 5
#define echoPin 18

//define sound speed in cm
#define SOUND_SPEED 0.034

long duration;
float distanceCm;

void setup() {
  Serial.begin(115200); // Starts the serial communication
  //define trigpin and echopin - which is input, which is output?

}
```

```
16  void loop() {
17    // Clears the trigPin - to unsure we get rid of noise
18    digitalWrite(trigPin, LOW);
19    delay(0.002);
20    // Sets the trigPin on HIGH state for 10 micro seconds
21
22
23
24    // Reads the echoPin, returns the sound wave travel time in microseconds
25    // Hint - use pulseIn() - which returns the length of the pulse in microseconds.
26    // The pulse length corresponds to the time it took to travel to the object plus the time traveled on the way back.
27
28
29    // Calculate the distance
30
31
32
33    // Prints the distance in the Serial Monitor
34
35
36    delay(1000);
37  }
```

```
#define trigPin 5
#define echoPin 18

//define sound speed in cm
#define SOUND_SPEED 0.034

long duration;
float distanceCm;

void setup() {
  Serial.begin(115200); // Starts the serial communication
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
}

void loop() {
  // Clears the trigPin - to unsure we get rid of noise
  digitalWrite(trigPin, LOW);
  delay(0.002);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delay(0.01);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance
  distanceCm = duration * SOUND_SPEED/2;


  // Prints the distance in the Serial Monitor
  Serial.print("Distance (cm): ");
  Serial.println(distanceCm);

  delay(1000);
}
```

ESP32 + WiFi + Server

“Tell Me Why?”

- I Want It That Way, Backstreet Boys (1999)

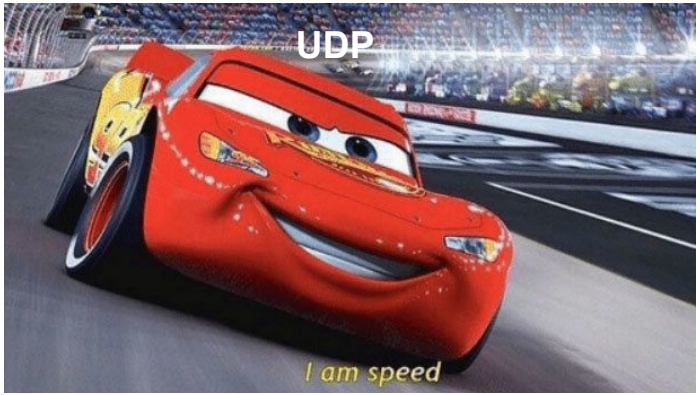
- In many IoT applications, the sensors send their data to a localized server.
  - For example, a sensor monitoring forest fires sending location and alerts to server.

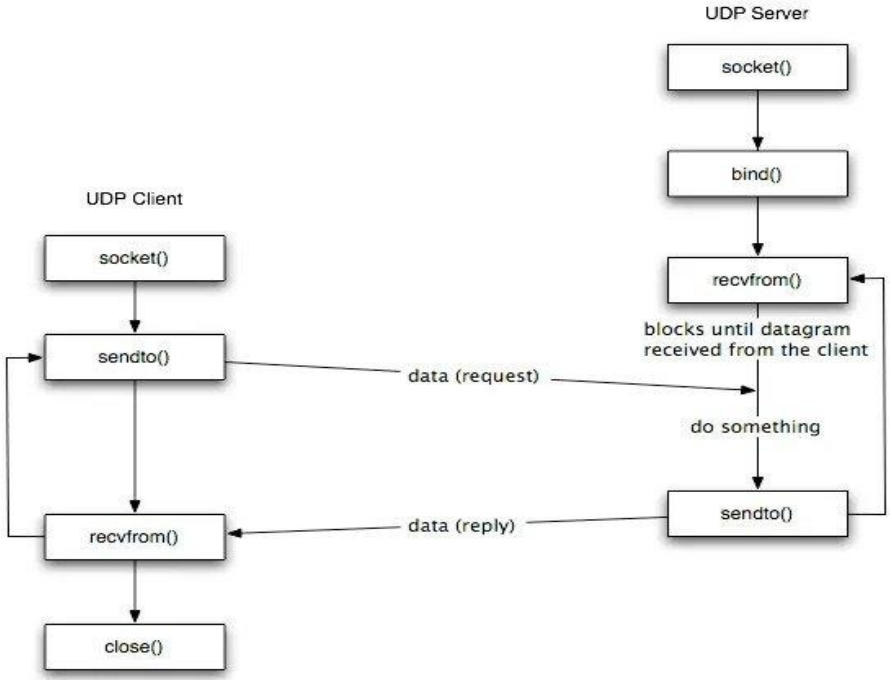
- It is important to understand how this data is sent over a network
  - This includes various data transmission protocols:
    - HTTP, TCP, UDP, etc.

- We will go briefly go over UDP and Sockets.

UDP: User Datagram Protocol

- Sends packets of data - good for unidirectional dataflow
- UDP does not check for errors in the packets, bidirectional communication is eliminated as compared to TCP (transmission control protocol)
- UDP is all about speed



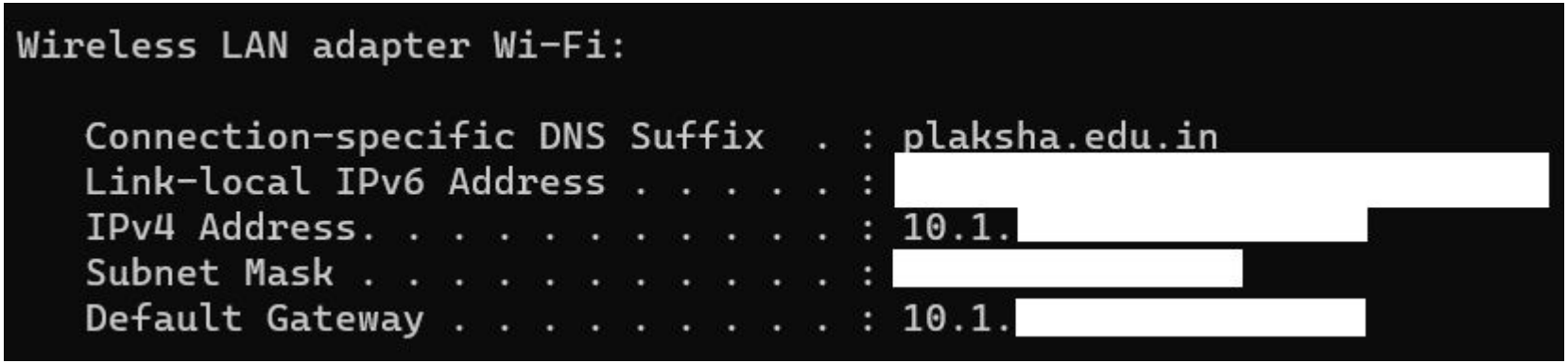


Connecting your ESP32 to WiFi

```
1   #include <WiFi.h>
2   #include <WiFiUdp.h>
3   #include <random>
4
5   const char* ssid = "Name";
6   const char* password = "Password";
7   const char* host = "127.0.0.1";         // IP address of the receiving server
8   const int port = 8888;                 // Port number of the receiving server
9
10  WiFiUDP udp;
```

Checking your IP Address

- Open Command Prompt
- Type: `ipconfig`
- Under ‘`Wireless LAN adapter Wi-Fi:`’
  - Copy ‘`IPv4 Address`’



```
void setup() {
  Serial.begin(115200);
  delay(1000);

  // Connect to Wi-Fi
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  udp.begin(port);
}
```

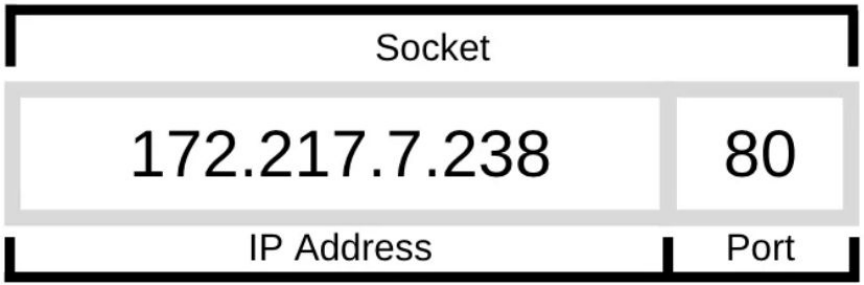
```
void loop() {

  // Generate and send 5 random numbers
  for (int i = 0; i < 5; i++) {

    //std::random_device rd; creates a random number generator
    //object rd from the implementation-defined entropy source
    std::random_device rd;
    //std::mt19937 gen(rd()); creates a Mersenne Twister pseudo-random number
    //generator object gen and seeds it with a random value generated by rd().
    std::mt19937 gen(rd());
    //std::uniform_int_distribution<int> dist(1, 100); creates a uniform integer distribution
    // object dist that generates random integers in the range from 1 to 100 (inclusive).
    std::uniform_int_distribution<int> dist(1, 100);
    //int randomNum = dist(gen); generates a random number using the distribution dist
    //and the generator gen and assigns it to the variable randomNum.
    int randomNum = dist(gen);

    // Convert the random number to a string
    String data = String(randomNum);

    // Send the data over UDP
    udp.beginPacket(host, port);
    udp.print(data);
    udp.endPacket();

    Serial.print("Sent: ");
    Serial.println(data);

    delay(1000);  // Wait for 1 second before sending the next random number
  }
}
```

Server

Sockets

- An interface that allows an application to transmit information through a network.

- A socket is implemented along with a protocol. UDP in our case



But what is a Socket!!??

- The Internet is all “cables”
- Sockets are a way to tap into the “cables”
- More technically: A network is a “cable” where information flows and socket is how you connect to it.

This is Kevin.
Kevin believes sockets belong only on walls.
Don’t be Kevin.

```python
import socket
import csv

UDP_IP = "0.0.0.0"  # Listen on all available network interfaces
UDP_PORT = 8888

csv_file = "random_numbers.csv"

# Create a UDP socket
#These lines create a UDP socket (sock) using the socket.socket function.
# The AF_INET parameter specifies the address family as IPv4,
# and SOCK_DGRAM specifies that it is a UDP socket.
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

#The bind method is then used to bind the socket to the specified IP address and port number.
sock.bind((UDP_IP, UDP_PORT))

connected = False

print("Waiting for device connection...")
```

```
while True:
    # Receive UDP packet
    data, addr = sock.recvfrom(1024)
    random_num = data.decode().strip()

    if connected:
        print("Device connected")
        connected = True

    print("Received:", random_num)

    # Write to CSV file
    with open(csv_file, "a") as file:
        writer = csv.writer(file)
        writer.writerow([random_num])
```

Umm, Thank You, I Guess?

See you in the projects!

Done with microcontrollers?