



SHAASTRA2020

presents

Introduction to Arduino and NodeMCU

Workshop Plan

Morning Session :

Arduino, Microcontroller and ESP32 Basics (~ 1 Hour)

Digital and Analog Circuits with Communication Protocols (~ 1 Hour)

Simulate Arduino Circuit using TinkerCAD (~ 2 hours) :

- Projects Demo with Code.
- Projects as Questions.

Afternoon Session :

Projects using ESP8266 and I2C LCD :

- Hands-on Projects

First things first ...

- This Material contains some memes to keep the workshop attendees from feeling bored during the theory part of the workshop - Attendees are to take it light-heartedly
- Morning session hands on will be run in a simulation environment in the website tinkercad.com where we will be programming Arduino with multiple input / output devices to get used to field of Arduino.
- Afternoon session will involve the usage of ESP32 and IoT based hands-on session.
- This workshop will cover the Introduction to these concept , so at any point of time if you have any previous experience and feel bored , let me know , I have tough problems for you guys...

Why are we doing

SPECS/BOARD
Number of Cores
Architecture
CPU Frequency
WiFi
BLUETOOTH
RAM
FLASH
GPIO PINS
Busses
ADC Pins
DAC Pins
Cost*

Introduction to Arduino workshop

Trainer : We will be doing Arduino in Simulation

Workshop Attendees :



imgflip.com

o Uno or NodeMCU?

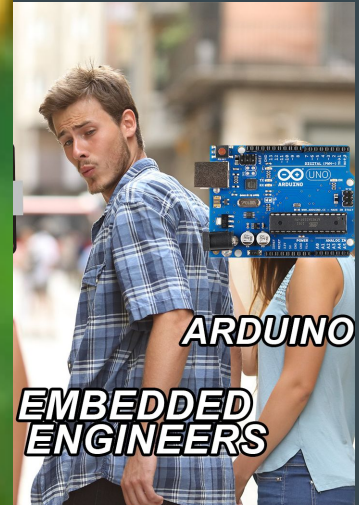
quest



Ive Semiconductors



UNO because it is having inbuilt WIFI, you can do much effort compare to connecting UNO to MCU. We can also program NodeMCU in 'C'



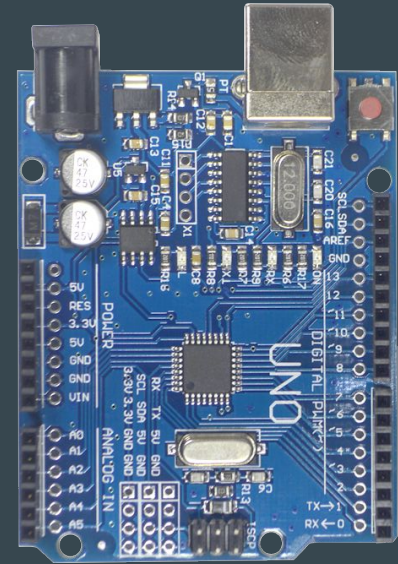
*Amazon.in

What is Arduino ?

Arduino is an open-source hardware and software company, project and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices.

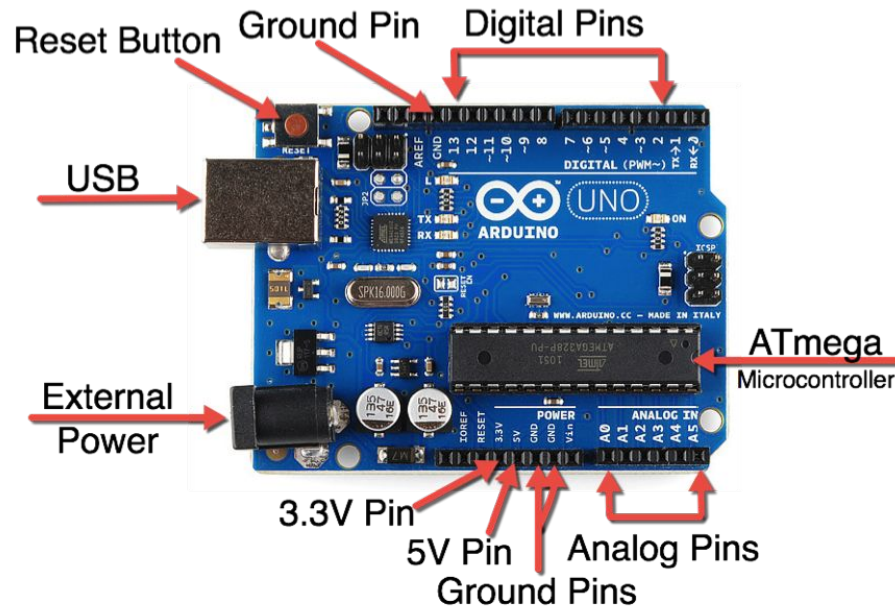
These Single-board microcontroller are intended to make the application of interactive objects or environments more accessible.

These are Designed to make the process of using electronics multi-disciplinary projects more accessible.



An Arduino UNO

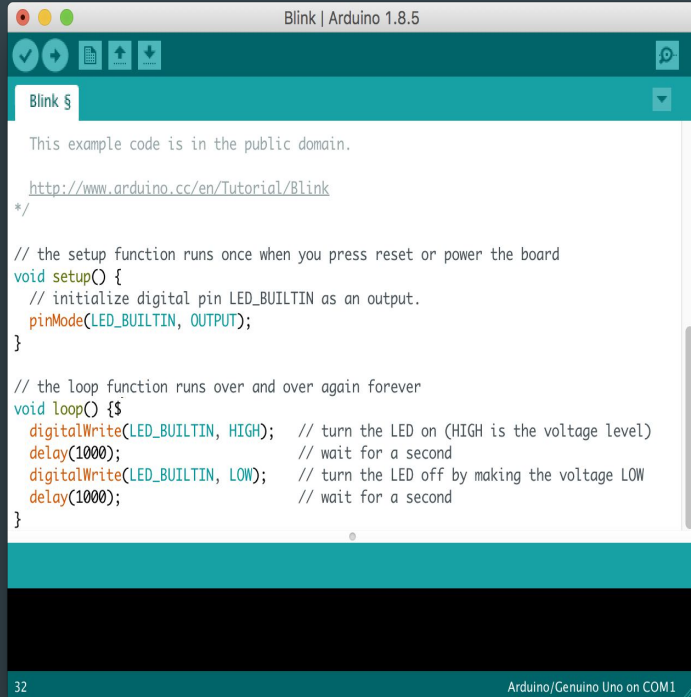
The Basic Arduino Board & Arduino IDE



Core Features of the Arduino UNO Board :
(ATmega 328)

- 14 Digital I/O Pins - 6 PWM
- 32 Kb Flash Memory
- 2 Kb SRAM - 1KB EEPROM
- 16 Mhz Clock Speed
- Input voltage (Recommended) (7 - 12V)
- Input Voltage (Limits) (6 - 20 V)
- USB Power : 5V
- DC Current per I/O : 40mA
- DC Current output for 3.3V Pin : 50mA

Arduino IDE

A screenshot of the Arduino IDE window titled "Blink | Arduino 1.8.5". The interface has a dark teal header bar with icons for checking, running, saving, and uploading. Below the header is a tab labeled "Blink" with a small icon. The main text area contains the following code:

```
This example code is in the public domain.

http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000); // wait for a second
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
  delay(1000); // wait for a second
}
```

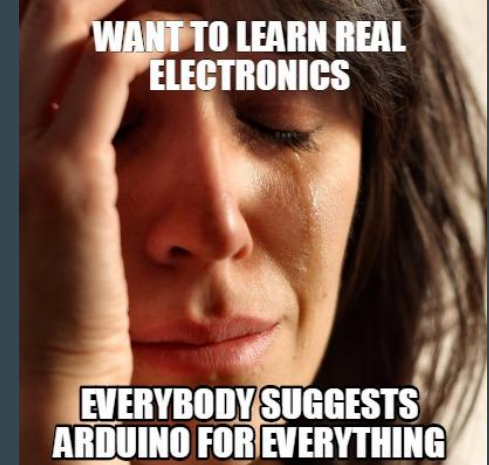
The status bar at the bottom shows "32" on the left and "Arduino/Genuino Uno on COM1" on the right.

- Arduino IDE is an open source software that is mainly used for writing and compiling the code into the Arduino Module.
- It is an official Arduino software, making code compilation too easy that even a common person with no prior technical knowledge can get their feet wet with the learning process.
- It is easily available for operating systems like MAC, Windows, Linux and runs on the Java Platform that comes with inbuilt functions and commands that play a vital role for debugging, editing and compiling the code in the environment..
- The main code, also known as a sketch, created on the IDE platform will ultimately generate a Hex File which is then transferred and uploaded in the controller on the board.
- This environment supports both C and C++ languages.

And Yes , it does have Dark Theme in it :)

But why should you learn Arduino ? :

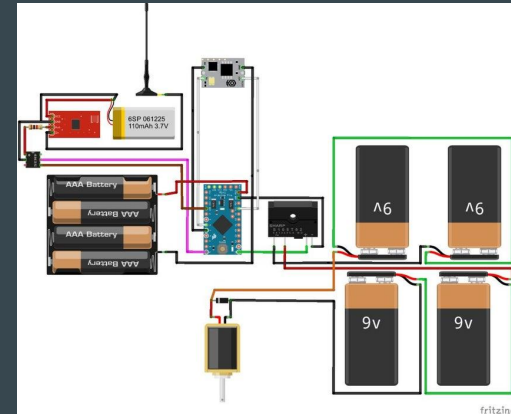
- Easy to learn
- Open-Source
- Has an very active online community
- Multidisciplinary applications
- Simplified and User-friendly Programming Language
- No Additional programmer/burner hardware required for programming board.
- Portable
- Low power consumption



Applications for Arduino :

- Home Automations
- Sensor prototyping Robotics
- ISP programming
- Easy WiFi ,Gsm ,Ethernet , Bluetooth ,Zigbee Connectivity
- Thor Hammer Replica
- Universal Remote Control
- Music Reactive LED Strip

etc ...



Before jumping into Arduino ,let us get ourselves comfortable with some basic concepts

These Single-board microcontroller are intended to make the application of interactive objects or environments more accessible.

But What is a Microcontroller ? and how is it different from the processor in our phones ?

Analog vs Digital Circuits

All physical quantities are analog.

Analog means that the quantity can take any value between its minimum value and maximum value.

Digital means that the quantity can take specific levels of values with specific offset between each other.

Ex: 1 Digital:

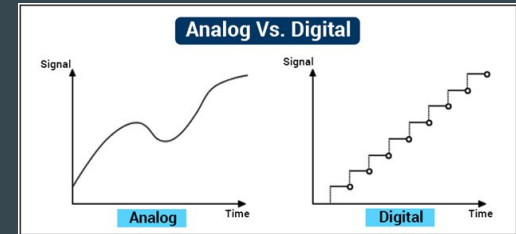
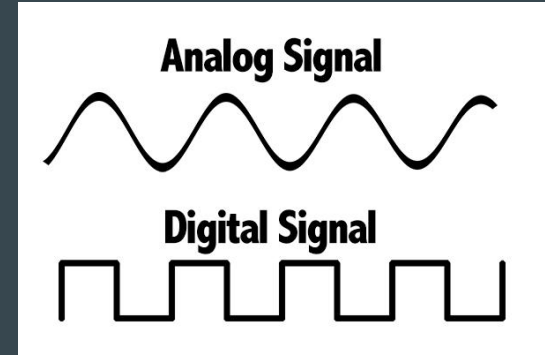
English alpha consists of 26 letter, there is no letter between A and B.

-Square waves are Digital.

Ex.: 2 Analog:

Temperature, can take any value[-1,12.8,25.002,... etc.].

-Sine waves are analog.



Microcontroller unit - MCU & Microprocessing unit - MPU :

Microcontroller	Microprocessor
Microcontroller is a small computer present on a Integrated Circuit.	Microprocessor is the Central Processing unit of the System.
A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip.	External RAM , ROM is required , along with External I/O peripherals.
Low Power Consumption	High Power Consumption
Low Processing Power	High Processing Power
Single-Purpose	General-purpose
Present in Microwave oven , Washing machines etc..	Present in Edge Computing devices such as Phones etc..

So If Microprocessors feel more powerful ,why should we even care about Microcontrollers ?

Let us demystify the question by understanding how our phones work :

Most of our Phones have a ARM (Advanced RISC Machine) Processor as the central processor and they also have different microcontrollers for different actions such as display control , speaker control , touch control etc.. where the microprocessor sends commands to the microcontrollers through various buses present and gets back information to process it and communicate back with the microcontrollers.

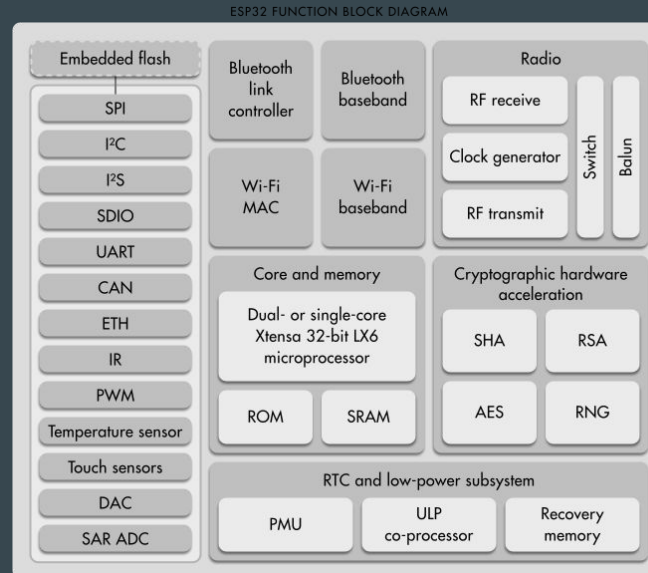
So , most of the Projects are usually a mix of microcontrollers and microprocessors working together.

But OnePlus brags about Snapdragon 8xx or something , what the hell is that then ?

NodeMCU - ESP32

- NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266, ESP32 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module.

Features of ESP32 :



ELECTRONIC COMMUNICATION PROTOCOLS

ESP32'S GPIO PINS SUPPORT I2C, SPI, UART - LET'S SEE WHAT THEY
ARE NOW !

TYPES OF ELECTRONICS COMMUNICATION PROTOCOLS

1. Inter System Protocol

- UART Protocol
- USART Protocol
- USB Protocol

2. Intra System Protocol

- I2C Protocol
- SPI Protocol
- CAN Protocol

BAUD RATE

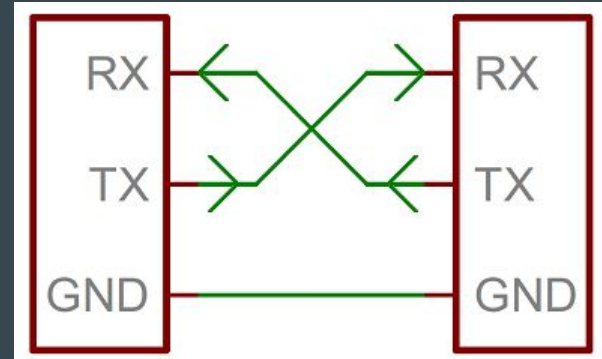
- The baud rate specifies **how fast** data is sent over a serial line. It's usually expressed in units of bits-per-second (bps).
- The only requirement in serial communication is that both devices operate at the same baud rate. One of the more common baud rates, especially for simple stuff where speed isn't critical, is **9600 bps**.
- Other "standard" baud are 1200, 2400, 4800, 19200, 38400, 57600, and 115200.

Inter System Protocol

- Inter System Protocol is used to communicate between two devices.
- Communication Occurs through a Inter Bus System.
- Ex: CPU with Mobile Phone, CPU with microcontroller.

UART Protocol

- UART stands for universal asynchronous transmitter and receiver.
- UART Protocols is a serial communication with two wired protocol (Rx and Tx).
- *Asynchronous* means that data is transferred **without support from an external clock signal**.
- The UART take bytes of data and send the individual bits in sequential manner.



- UART is a half duplex protocol. Half duplex means transferring and receiving the data but not at a same time.
- Some microcontrollers use a single data line for transmitting and receiving the data.
- In general, It has one start bit, 8-bit data and one stop bit mean the 8-bit data transfer ones signal is high to low.
- UART transmits data is organized into packets. Each packet contains 1 start bit, 5 to 9 data bits (depending on the UART), an optional parity bit, and 1 or 2 stop bits.

- Start Bit

The UART data transmission line is normally held at a high voltage level when it's not transmitting data. To start the transfer of data, the transmitting UART pulls the transmission line from high to low for one clock cycle. When the receiving UART detects the high to low voltage transition, it begins reading the bits in the data frame at the frequency of the baud rate.

- Data Frame

The data frame contains the actual data being transferred. It can be 5 bits up to 8 bits long if a parity bit is used. If no parity bit is used, the data frame can be 9 bits long. In most cases, the data is sent with the least significant bit first.

- Parity

Parity describes the evenness or oddness of a number.

The parity bit is a way for the receiving UART to tell if any data has changed during transmission.

Bits can be changed by electromagnetic radiation, mismatched baud rates, or long distance data transfers.

After the receiving UART reads the data frame, it counts the number of bits with a value of 1 and checks if the total is an even or odd number. If the parity bit is a 0 (even parity), the 1 bits in the data frame should total to an even number. If the parity bit is a 1 (odd parity), the 1 bits in the data frame should total to an odd number.

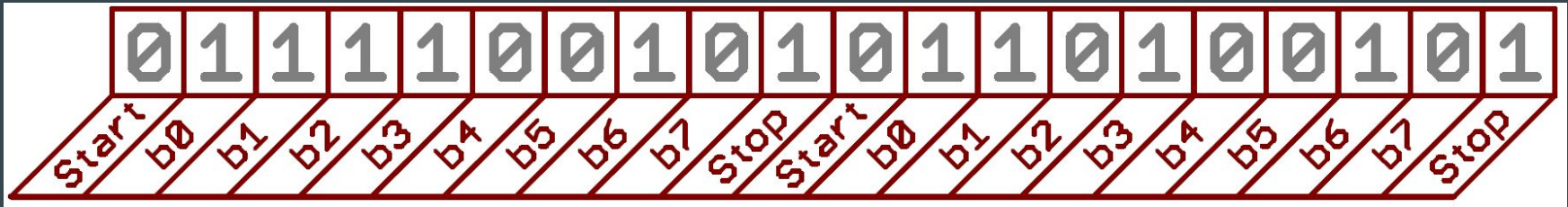
When the parity bit matches the data, the UART knows that the transmission was free of errors. But if the parity bit is a 0, and the total is odd; or the parity bit is a 1, and the total is even, the UART knows that bits in the data frame have changed

- Stop Bits

To signal the end of the data packet, the sending UART drives the data transmission line from a low voltage to a high voltage for at least two bit durations.

9600 8N1 (AN EXAMPLE)

- 9600 8N1 - 9600 baud, 8 data bits, no parity, and 1 stop bit - is one of the more commonly used serial protocols.
- We wish to transmit 2 packets of data O and K. The ASCII value of O(that's uppercase) is 79, which breaks down into an 8-bit binary value of 01001111, while K's binary value is 01001011.
- Data transferred has least-significant bit first.

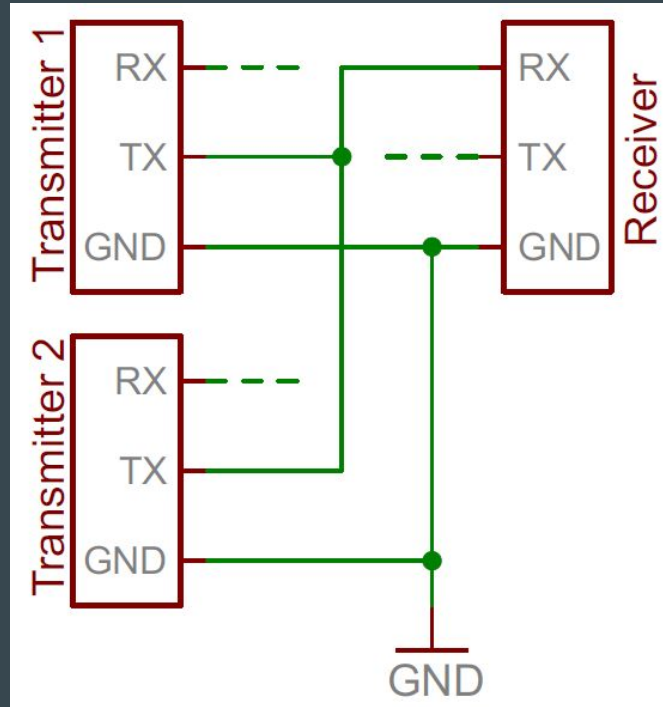


- Since we're transferring at 9600 bps, the time spent holding each of those bits high or low is $1/(9600 \text{ bps})$ or 104 μs per bit.
- Since we're transferring at 9600 bps, the time spent holding each of those bits high or low is $1/(9600 \text{ bps})$ or 104 μs per bit.

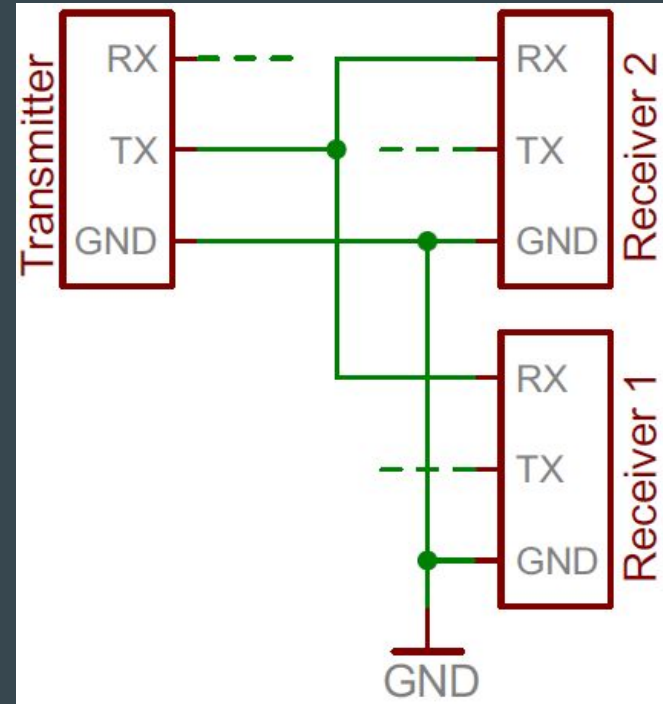
Baud Rate Mismatch

- Baud rates are like the languages of serial communication. If two devices aren't speaking at the same speed, data can be either misinterpreted, or completely missed. If all the receiving device sees on its receive line is garbage, check to make sure the baud rates match up.

ERROR IN COMMUNICATION



COMMUNICATION IS POSSIBLE



USART PROTOCOL

- USART stands for universal synchronous and asynchronous transmitter and receiver.
- This protocol is used to transmitting and receiving the data byte by byte along with the clock pulses.
- It is a full-duplex protocol means transmitting and receiving data simultaneously to different board rates.

USB PROTOCOL (UNIVERSAL SERIAL BUS)

- Again it is a serial communication of two wire protocol. The data cable signal lines are labelled D+ and D-.
- This protocol is used to communicate with the system peripherals. USB protocol is used to send and receive the data serially to the host and peripheral devices.
- USB communication requires a driver software which is based on the functionality of the system.USB device can transfer data on the bus without any request on the host computer

INTRA SYSTEM PROTOCOL

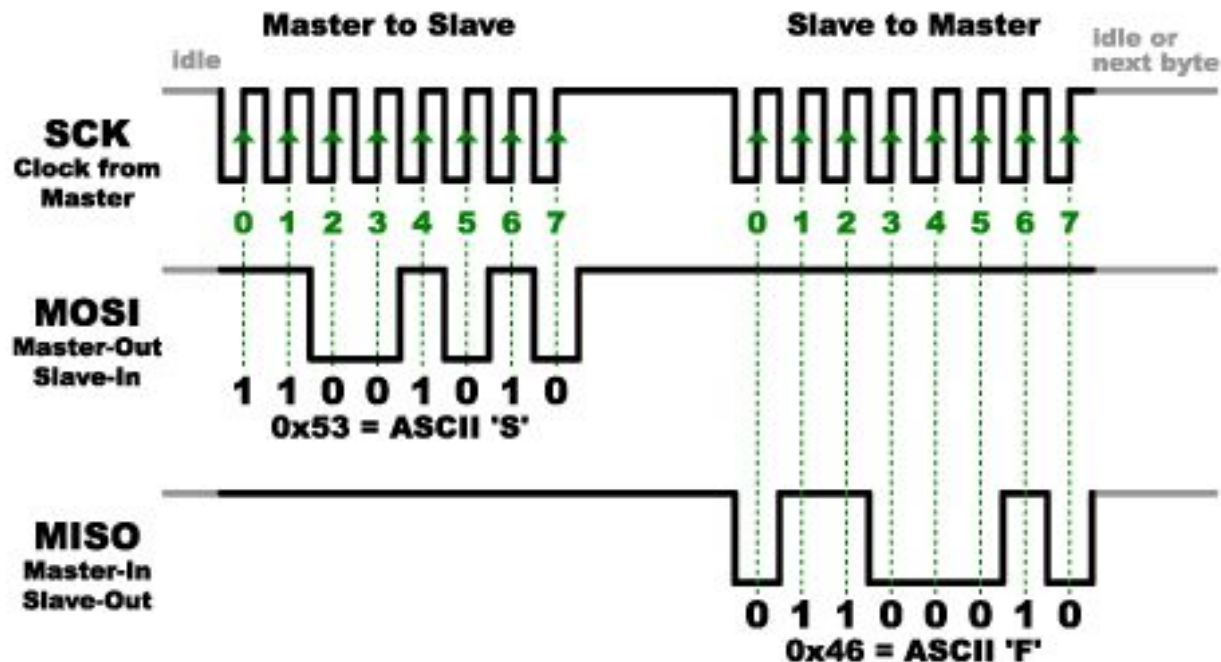
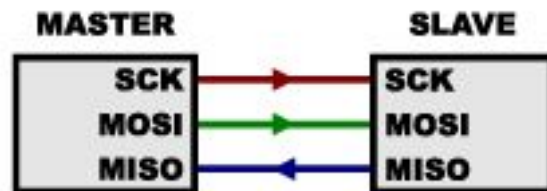
- The Intra system protocol is used to communicate the two devices within the circuit board.
- Intra system protocols, are used expand the peripherals of the microcontroller. The circuit complexity and power consumption will be increases by using intra system protocol.

WHAT'S WRONG WITH SERIAL PORTS?

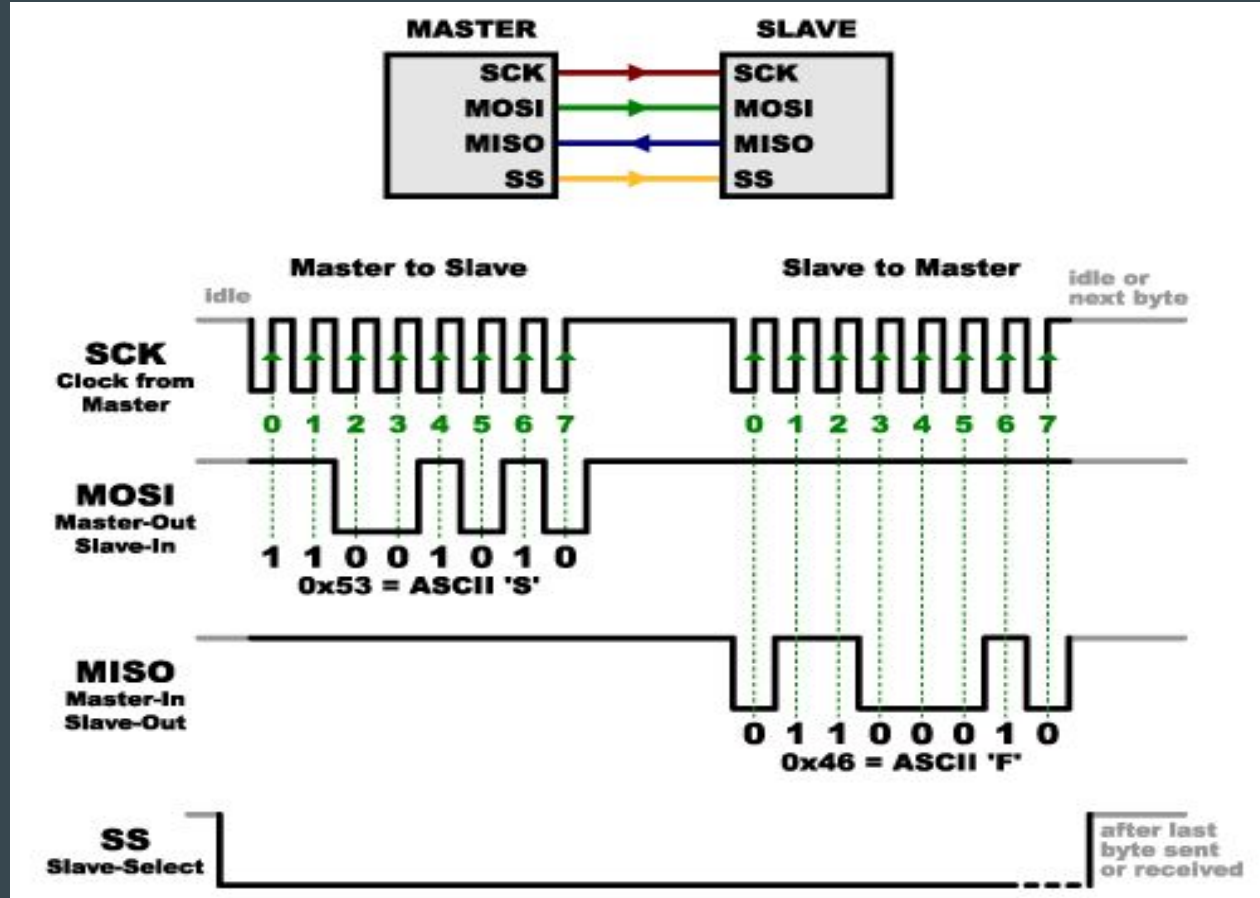
- The communication between the two devices is asynchronous.
- Since computers normally rely on everything being synchronized to a single “clock” (the main crystal attached to a computer that drives everything), this can be a problem when two systems with slightly different clocks try to communicate with each other.

SERIAL PERIPHERAL INTERFACE (SPI)

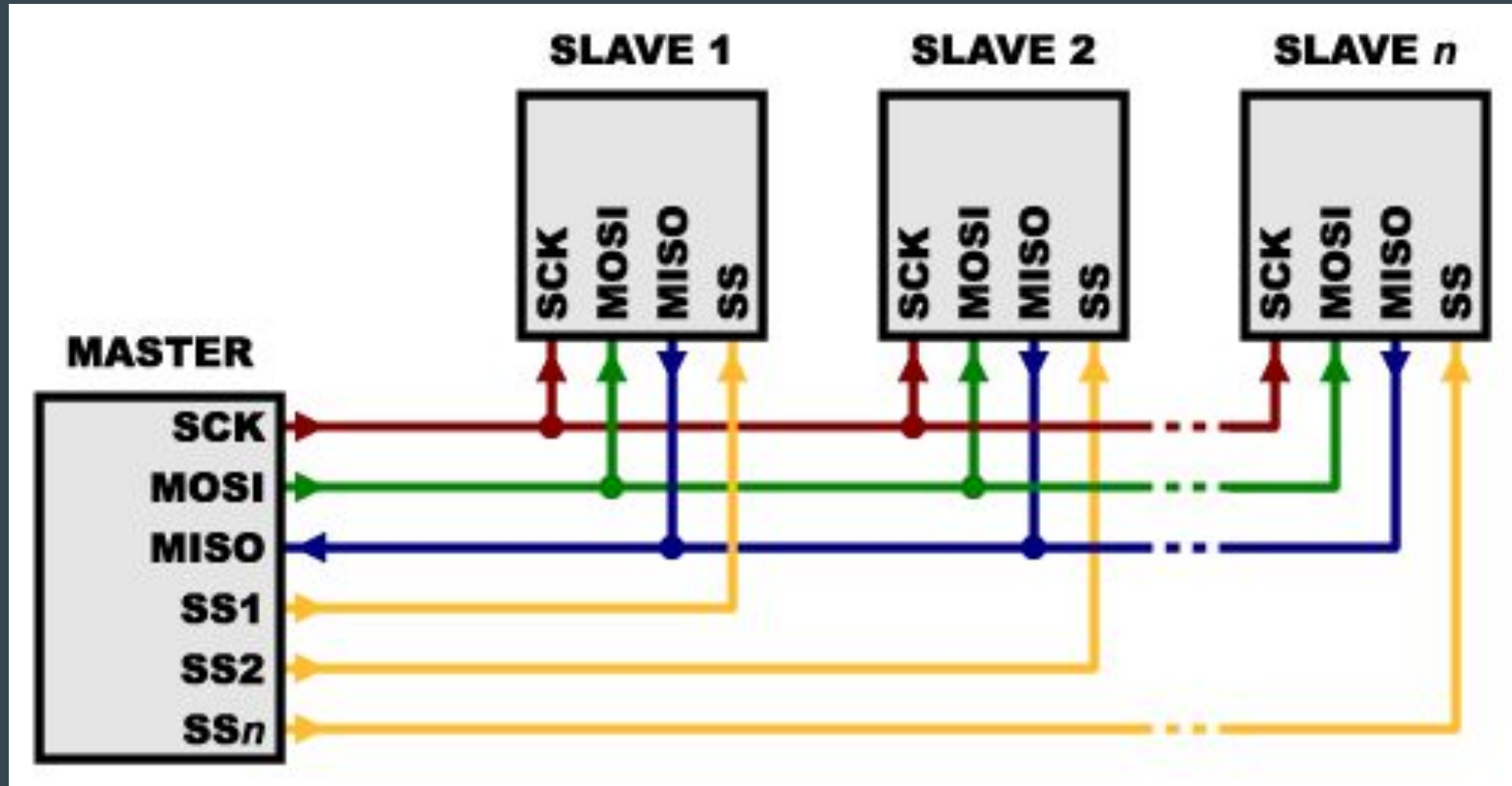
- SPI is a synchronous data bus.
- The clock is an oscillating signal that tells the receiver exactly when to sample the bits on the data line.
- In SPI, only one side generates the clock signal (usually called CLK or SCK for Serial Clock). The side that generates the clock is called the “master”, and the other side is called the “slave”.
- There is always only one master, but there can be multiple slaves.
- MOSI : “Master Out / Slave In” , MISO : “Master In / Slave Out”.



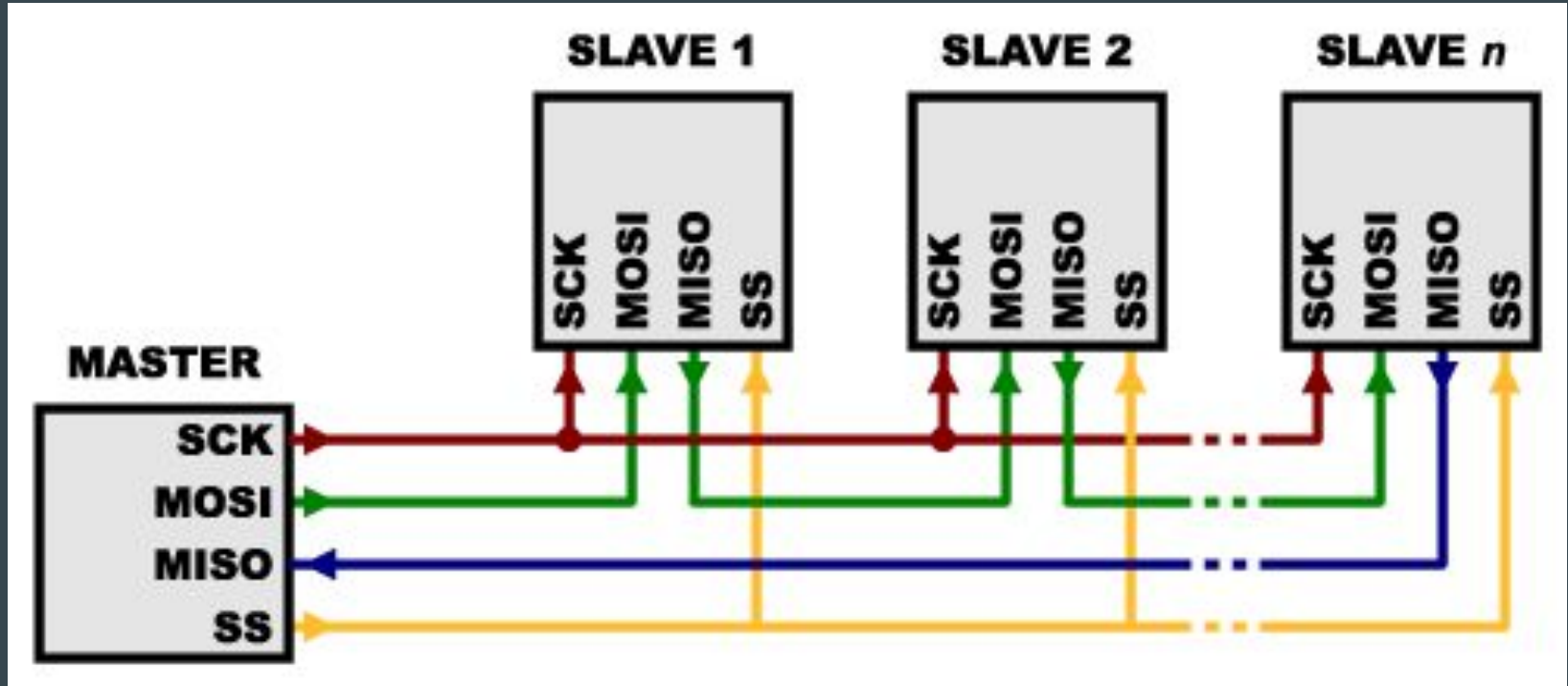
SLAVE SELECT (SS)



MULTIPLE SLAVES : CASE 1



MULTIPLE SLAVES : CASE 2 (DAISY-CHAIN LOOP)



ADVANTAGES OF SPI:

- It's faster than asynchronous serial
- The receive hardware can be a simple shift register
- It supports multiple slaves

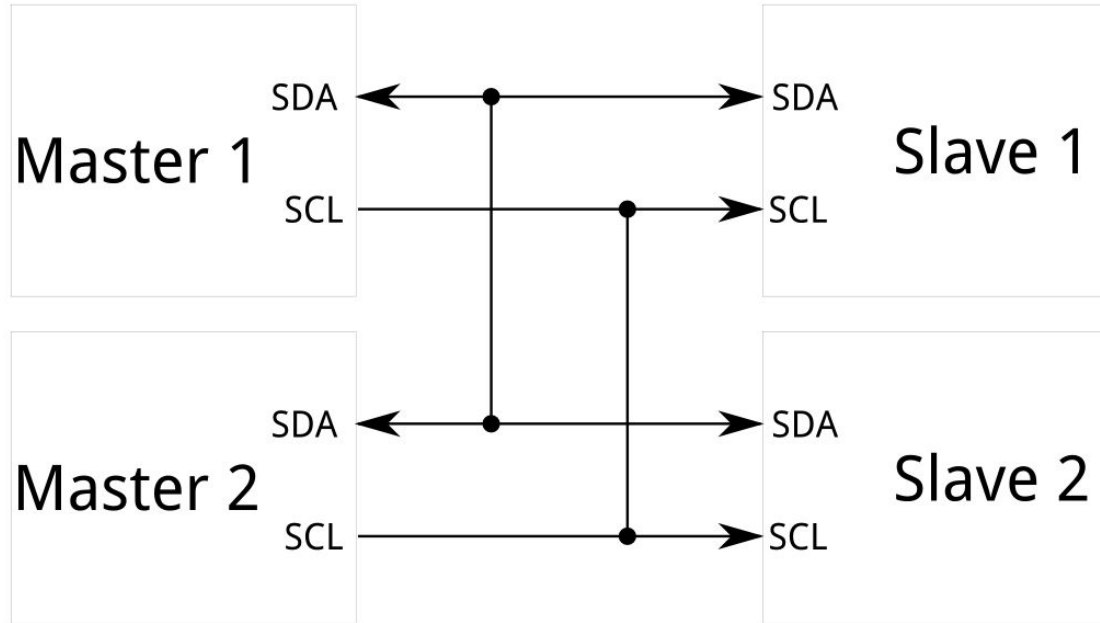
DISADVANTAGES OF SPI:

- It requires more signal lines (wires) than other communications methods
- The communications must be well-defined in advance (you can't send random amounts of data whenever you want)
- The master must control all communications (slaves can't talk directly to each other)
- It usually requires separate SS lines to each slave, which can be problematic if numerous slaves are needed.

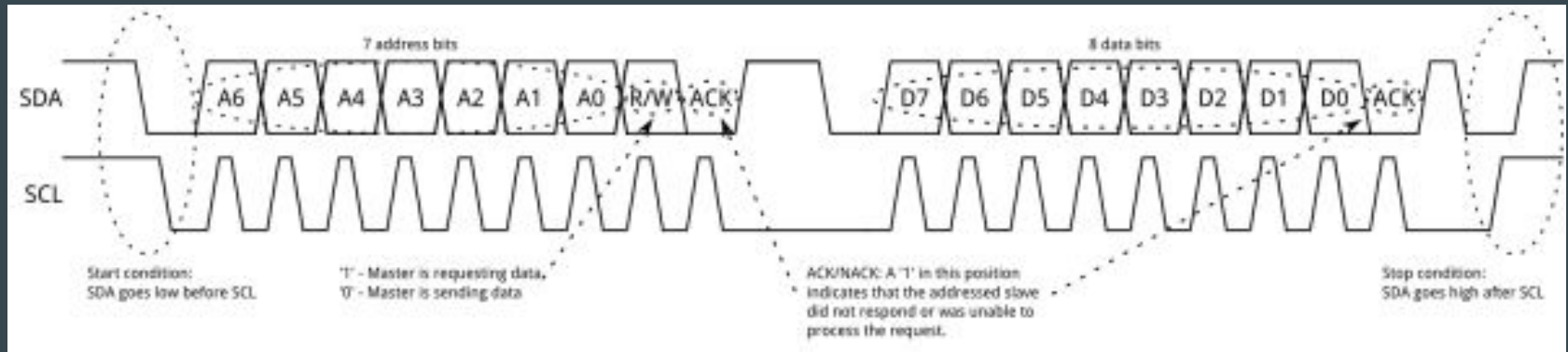
I2C – INTER INTEGRATED CIRCUIT

- I2C requires two wires, like asynchronous serial, but those two wires can support up to 1023 slave devices.
- Unlike SPI, I2C can support a multi-master system, allowing more than one master to communicate with all devices on the bus.
- Master devices can't talk to each other over the bus and must take turns using the bus lines).
- Each I2C bus consists of two signals: SCL and SDA. SCL is the clock signal, and SDA is the data signal.
- I2C is half duplex.

I²C - THE BEST OF BOTH WORLDS!



- Messages are broken up into two types of frame: an address frame, where the master indicates the slave to which the message is being sent, and one or more data frames, which are 8-bit data messages passed from master to slave or vice versa.
- Data is placed on the SDA line after SCL goes low, and is sampled after the SCL line goes high. The time between clock edge and data read/write is defined by the devices on the bus and will vary from chip to chip.



REFERENCES

- <https://learn.sparkfun.com/tutorials/serial-communication/rules-of-serial>
- <https://web.stanford.edu/class/cs140e/notes/lec4/uart-basics.pdf>
- <https://www.elprocus.com/communication-protocols/>
- <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi>
- <https://learn.sparkfun.com/tutorials/i2c>

Basic Arduino Programming

Parts of the environment :

Save - ctrl/cmd+S

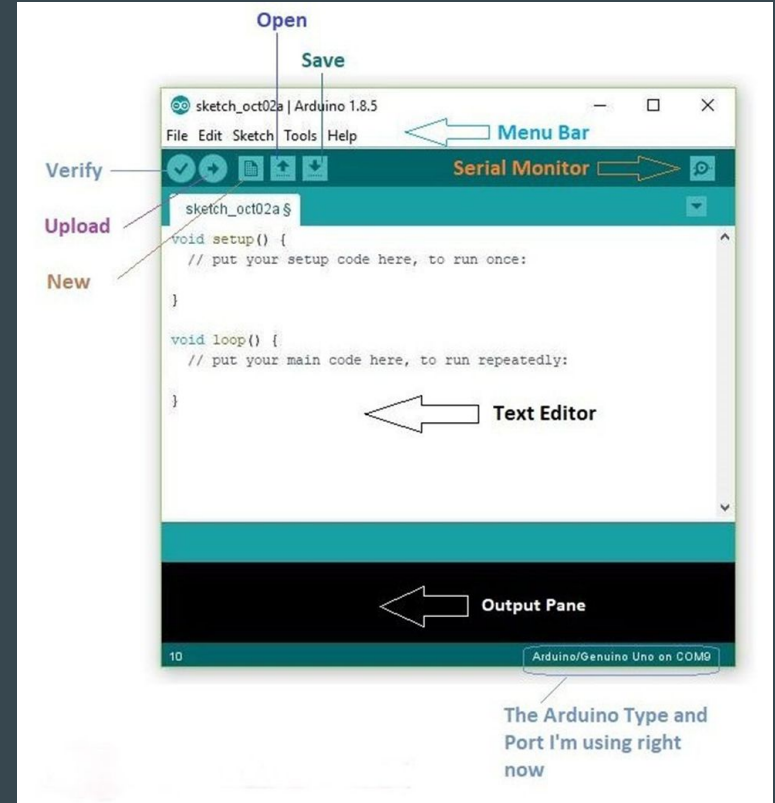
Verify - ctrl/cmd+R

Upload - ctrl/cmd+U

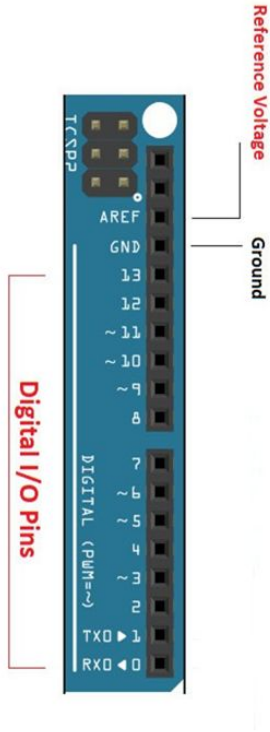
Text editor

Output pane

Board and port used



WORKING WITH PINS



pins in arduino

```
int led = 12;
int button = 1;

void setup() {
  pinMode(led, OUTPUT); // e.g. speakers, leds, motors
  pinMode(button, INPUT); // e.g. buttons, IR sensors, microphones, accelerometers
}

void loop() {
  digitalWrite(led, LOW);
  delay(1000);
  digitalWrite(led, HIGH);
  delay(1000);
}
```

example code

Code Explanation :

```
void setup(){
```

```
// One Time Setup and Pin Initialisations
```

```
}
```

```
void loop(){
```

```
// Loops forever
```

```
}
```

Code Explanation : Continued

```
int led = 2; // Define the Pin number
```

```
void setup(){
```

```
pinMode(led, OUTPUT ) ; // Set the Mode the Pin is going to function  
}
```

```
void loop(){
```

```
digitalWrite(2 , HIGH); // Set the Value of the Pin to HIGH ( 3.3V)
```

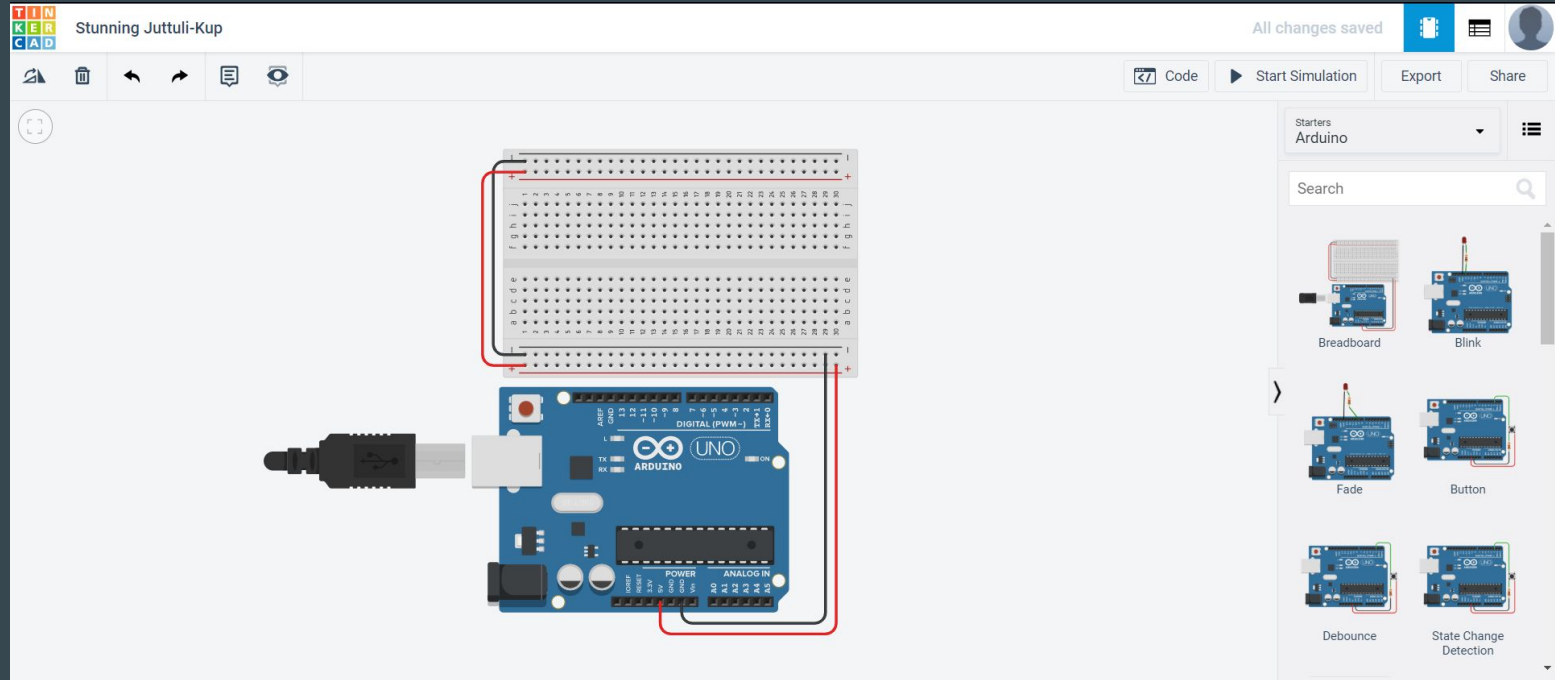
```
delay(200) ; // wait for 200 milliseconds
```

```
digitalWrite(2 , LOW); // Set the Value of Pin to LOW ( 0 V )
```

```
delay(200); // wait for 200 milliseconds ( Question : what happens if i delete this line ? )
```

```
}
```

TinkerCAD Environment :



Next Steps :

- Go to tinkercad.com/circuits
- Launch Circuits
- Create your account
- Create a New Circuit

Problem - 1

Let's understand the Tinkercad Environment and solve the following problems :

1. Blink an In-build LED (Pin 13)
2. Blink an External LED
3. Attach an push button and Turn off and on LED based on the inputs
4. Attach a Potentiometer and get Input Values from it.
5. Rotate a Servo by giving some custom input values.
6. Rotate servo using inputs from potentiometer.
7. Attach a Photo resistor and use them to control the brightness of a led.

Blink In-build LED :

- Drag and Drop an Arduino from the Components List
- Click on Code and select Text
- Write the following code
- Click on Start Simulation to see the output

CODE :

```
void setup(){  
    pinMode(13, OUTPUT); // 13 is also the LED_BUILTIN  
}  
  
void loop(){  
    digitalWrite(13, HIGH);  
    delay(1000); // Wait for 1000 millisecond(s)  
    digitalWrite(13, LOW);  
    delay(1000); // Wait for 1000 millisecond(s)  
}
```

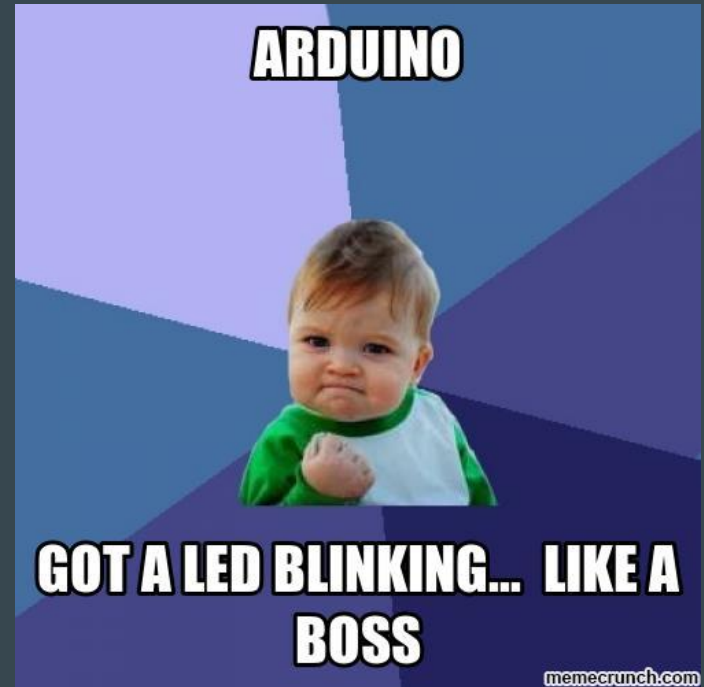
Blink an External LED :

- Drag and Drop an Arduino from the Components List
- Drag and Drop a LED and Resistor and connect the LED to 13th Pin in the Arduino.
- Click on Code and select Text
- Write the following code
- Click on Start Simulation to see the output

CODE :

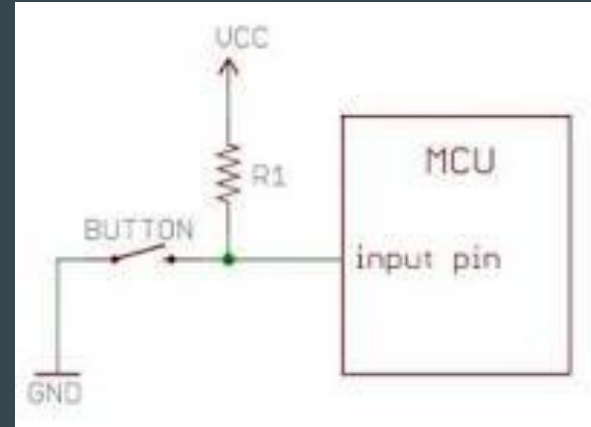
```
void setup(){  
    pinMode(13, OUTPUT); // 13 is also the LED_BUILTIN  
}  
  
void loop(){  
    digitalWrite(13, HIGH);  
    delay(1000); // Wait for 1000 millisecond(s)  
    digitalWrite(13, LOW);  
    delay(1000); // Wait for 1000 millisecond(s)  
}
```

Congrats for writing your first Arduino Code :



Attach a Button :

- INPUT_PULLUP
 - Instead of Attaching an External Pull-up resistor , we will use the Pull-up resistor in the Arduino by Specifying it in the pinMode
 - `pinMode(button , INPUT_PULLUP);`
- digitalRead()
 - Reads the value from a specified digital pin, either HIGH or LOW.
 - Input parameters : Pin number
 - `val = digitalRead(button);`



Gentle Reminder :

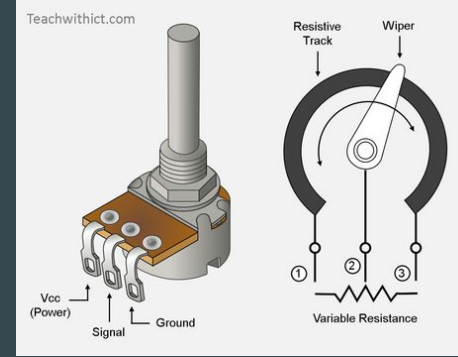
1. It is not necessary to remember or mug up any code. You would automatically start remembering them by practice.
2. All the Code will be shared to you
3. You only need to understand the basics as you can easily build upon that.
4. Each Component / Sensor is different , and you can go through their respective datasheets to understand on how to work with it.
5. There are also online tutorials for all the popular sensors in case you use need to use them in the future.
6. Feel free to Google while doing your exercise. (Don't copy paste)
7. The Goal of this Workshop is to break the ice for you and our end goal is that afterwards given any problem statement , you can research about various solutions and come up with a circuit and code for the same

Attach a Potentiometer and get Input Values from it.

Serial Monitor : The Serial Monitor is part of the Arduino IDE software. Its job is to allow you to both send messages from your computer to an Arduino board (over USB) and also to receive messages from the Arduino.

`Serial.begin(baud_rate);` //Starts the Serial comms

`Serial.println(value);` // Prints Data



`analogRead()` : Reads the value from the specified analog pin. Arduino boards contain a multichannel, 10-bit analog to digital converter. This means that it will map input voltages between 0 and the operating voltage(5V or 3.3V) into integer values between 0 and 1023.

Rotate a Servo by giving some custom input values.

Servo motors are great devices that can turn to a specified position. Usually, they have a servo arm that can turn 180 degrees.

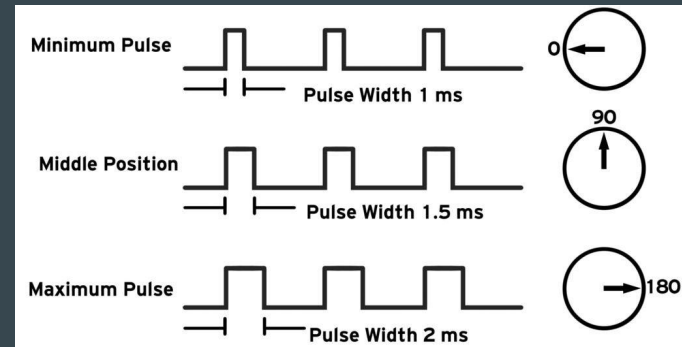
But Instead of sending Signals , we will using an Library for the same :

```
#include<Servo.h>
```

```
Servo servo_1; // Create a Servo Object
```

```
servo_1.attach(servo_pin); //Define Servo pin
```

```
servo_1.write(90); // Rotate to 90 degrees
```



Get Input from Serial Monitor

```
int pos = 0;

if (Serial.available()){          // Check if data from serial monitor is available

    pos = Serial.parseInt();      // Convert String to Integer

}
```


Rotate servo using inputs from potentiometer.

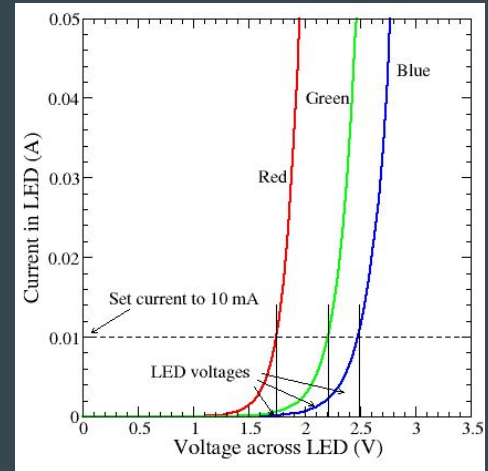
Try this Problem on your own.

PWM - Brightness of the LED

We saw about Pulse width Modulation in Servo , let us see how that helps us in controlling brightness of an LED.

LED's are diodes , which implies they don't follow Ohm's law ($V = IR$).

We can control the brightness but by controlling the current , that implies having a variable resistor. Yes , Go ahead try to simulate if that works

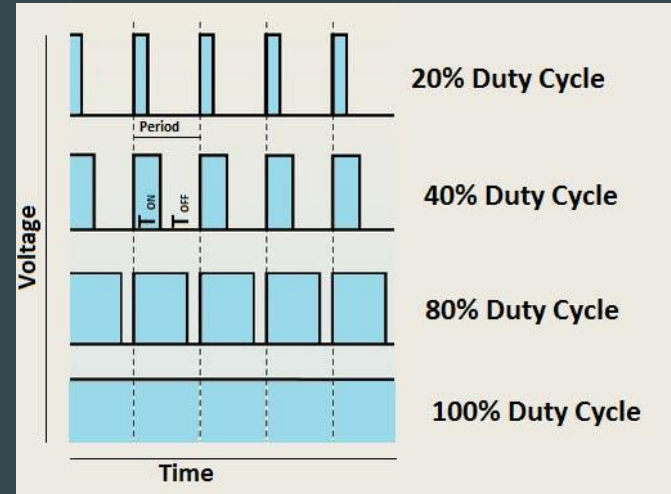


PWM - Continued

Hmm ... So you must have found out that you need to set the values of the Potentiometer appropriately, and not just that, it also has a drawback of power loss, so there's an alternate way of controlling the brightness, it is done by controlling the time which the pin is turned on.

PWM - Pulse Width Modulation.

Let's now see how to set the Duty Cycle and how it affects the brightness of the LED.



PWM - Continued

Remember first project (LED blinking) ?

CODE :

```
void setup(){
  pinMode(13, OUTPUT); // 13 is also the LED_BUILTIN
}

void loop(){
  digitalWrite(13, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(13, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}
```

This is a PWM signal with 50 % Duty Cycle at a Frequency of 0.5Hz , if we increase the frequency to 500 Hz or above , you could clearly see how the brightness changes from the blink code.

Go to the Blink code and Have two LEDs side by side and compare how their brightness varies with different frequencies.

PWM - Alternate Method

```
analogWrite();
```

Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds.

```
analogWrite(ledPin, duty_cycle); // Duty Cycle between 0 - 255
```

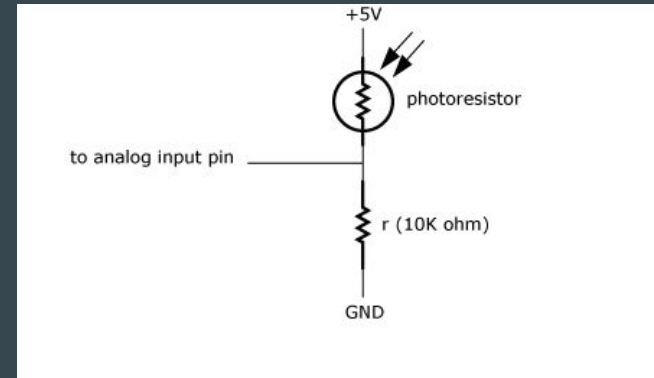
Control Brightness of LED using Photoresistor

Photoresistor is an active component that decreases resistance with respect to receiving luminosity (light) on the component's sensitive surface.

Using the Circuit Provided , Hookup the Potentiometer and Control the Brightness of the LED using the Pulse width signal estimating the Duty cycle from the Analog Input

Hint :) : You might need to google for the `map()` function.

`map(value, fromLow, fromHigh, toLow, toHigh)`



Problem - 2

The Air conditioner in your house is not functioning properly , you take a closer look at what happens and find out the PID controller of your Air Conditioner has been damaged and once you turn on the AC , it cools down your room to 16 degree celsius , but being a person from Chennai , you find it appropriate to sleep at 24 ± 1 degree celsius , so with your passion for electronics , you start designing a circuit which turns off the AC when the temperature falls below 23 degree celsius and turn it on when the temperature rises above 25 degree celsius.

Design a circuit for the same.

Problem Statement - 3

Your parents have left for a small vacation , so your elder sibling promises to your parents that he / she will cook the meals for both of you until your parents come back , but you know that your sibling is very forgetful ,so for your own safety and for the safety of the family ,you need to design a circuit using Arduino which alerts you or your sibling if they ever forget to turn off the gas after cooking.

Design a Circuit with an appropriate logical explanation for choosing your components.



Problem Statement -4

Assuming you are an Electronics Engineer in a reputed automobile company , now you are part of a team working on the automated parking system of a newly developing self-driving car , you are asked to design an system which would measure the clearance distance in the rear left and rear right of the car and report to the automated system to take further decisions, and upon the distance falling behind a certain threshold , you are supposed to blink an led which is connected to the Dashboard of the car to alert the passengers that the automated system is failing and manual control needs to be taken.

Design an Arduino circuit for the same.

Note for the Afternoon Session :

You will be given the hardware components required and we will start with simple circuits build our way up to complex ones.

LEDs are very sensitive to high current , so kindly verify the circuit twice before powering on the LEDs

You can also use the Simulation circuit to verify , before powering on the circuit.

BUILTIN_LED is pin 2 for the ESP32

Here are the list of projects what we'll work on during the afternoon session.

ESP32 Circuits

- Blink LED
- PWM with LED
- Button with LED
- Capacitive Touch - DIY Project
- LCD
- Button with LCD
- Bluetooth with LED
- WIFI with LED
- Insta Followers on LCD
- Twitter Tweet from Serial Monitor
- Tweet your Insta Followers count - DIY Project

ESP32 Installation Instruction :

Prerequisites :

Install Arduino IDE, get it from Arduino website.(<https://www.arduino.cc/en/Main/Software>)

Internet connection

Micro USB cable and fully charged laptop.

Instructions :

Start Arduino IDE and open File -> Preferences window.

Enter https://dl.espressif.com/dl/package_esp32_index.json into Additional Board Manager URLs field and Press Ok.

Open Boards Manager from Tools > Board menu and find esp32platform.

Select the latest version from a drop-down box.

Click install button.

Congrats, Your Software Setup is complete.

Blink LED :

- Type the code into your Arduino IDE.
- Connect your ESP32 and Go to
Tools -> Board -> DOIT ESP32
- Then Go to Tools -> Port -> Select
COM Port
- Click Upload
- While you get Connecting in the
bottom , Press on the Boot Button in
the ESP32 , and remove while it start
uploading

CODE :

```
void setup(){  
  pinMode(2, OUTPUT); // 13 is also the LED_BUILTIN  
}  
  
void loop(){  
  digitalWrite(2, HIGH);  
  delay(1000); // Wait for 1000 millisecond(s)  
  digitalWrite(2, LOW);  
  delay(1000); // Wait for 1000 millisecond(s)  
}
```

Brightness of LED : A different Approach (PWM) :

1. First, you need to choose a PWM channel. There are 16 channels from 0 to 15.

2. Then, you need to set the PWM signal frequency. For an LED, a frequency of 5000 Hz is fine to use.

3. You also need to set the signal's duty cycle resolution: you have resolutions from 1 to 16 bits. We'll use 8-bit resolution, which means you can control the LED brightness using a value from 0 to 255.

4. Next, you need to specify to which GPIO or GPIOs the signal will appear upon.

5. Finally, to control the LED brightness using PWM

```
// setting PWM properties
```

```
const int freq = 5000;
```

```
const int ledChannel = 0;
```

```
const int resolution = 8;
```

```
// configure LED PWM functionalitites
```

```
ledcSetup(ledChannel, freq, resolution);
```

```
// attach the channel to the GPIO to be controlled
```

```
ledcAttachPin(ledPin, ledChannel);
```

```
// changing the LED brightness with PWM
```

```
ledcWrite(ledChannel, dutyCycle);
```

Button :

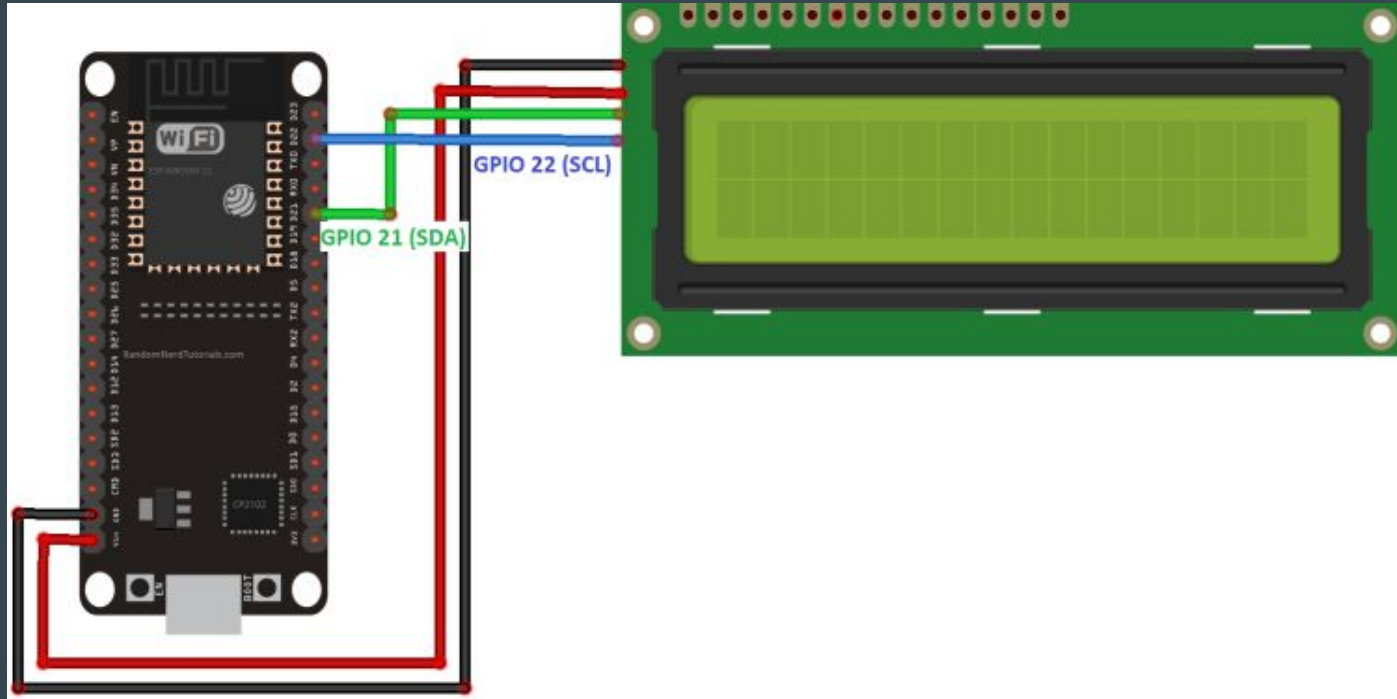
Try this out from the Arduino Example.

Also Try our using `INPUT_PULLDOWN` and see how the flow changes.

Capacitive Touch :

Feel free to Google for this and your end goal is to make the LED glow up when you touch the wire and turn off when you remove your finger from the wire.

LCD : Wiring



LCD : Software

Let's get the I2C address first and then download the LCD Library

Tools -> Manage Libraries and Type -> LiquidCrystal_I2C Library by Marco Schwartz.

```
#include <LiquidCrystal_I2C.h>
// set the LCD number of columns and rows
int lcdColumns = 16;
int lcdRows = 2;
// set LCD address, number of columns and rows
LiquidCrystal_I2C lcd(0x27, lcdColumns, lcdRows);
void setup(){
  // initialize LCD
  lcd.init();
  // turn on LCD backlight
  lcd.backlight();
}
```

LCD : Software

```
void loop(){  
  // set cursor to first column, first row  
  lcd.setCursor(0, 0);  
  // print message  
  lcd.print("Hello, World!");  
  delay(1000);  
  // clears the display to print new message  
  lcd.clear();  
  // set cursor to first column, second row  
  lcd.setCursor(0,1);  
  lcd.print("Hello, World!");  
  delay(1000);  
  lcd.clear();  
}
```

Button with LCD :

Your Goal is to attach the button such that the LCD turns on only when the button is pressed.

This is given as an exercise to you.

Bluetooth and WiFi with LED :

Install RoboRemo App from Android Play store.

The Code will be explained to you.

Insta Followers on LCD :

Install the “InstagramStats” library from Tools -> Manage Libraries

Thank you