ROBOTICS ISLAND

Contents

Lesson No. 1 - Basic Robotics Concepts

1- What is a Robot?

A robot is typically defined as a programmable machine capable of carrying out a series of actions autonomously or semi-autonomously.

2- Applications of Robotics

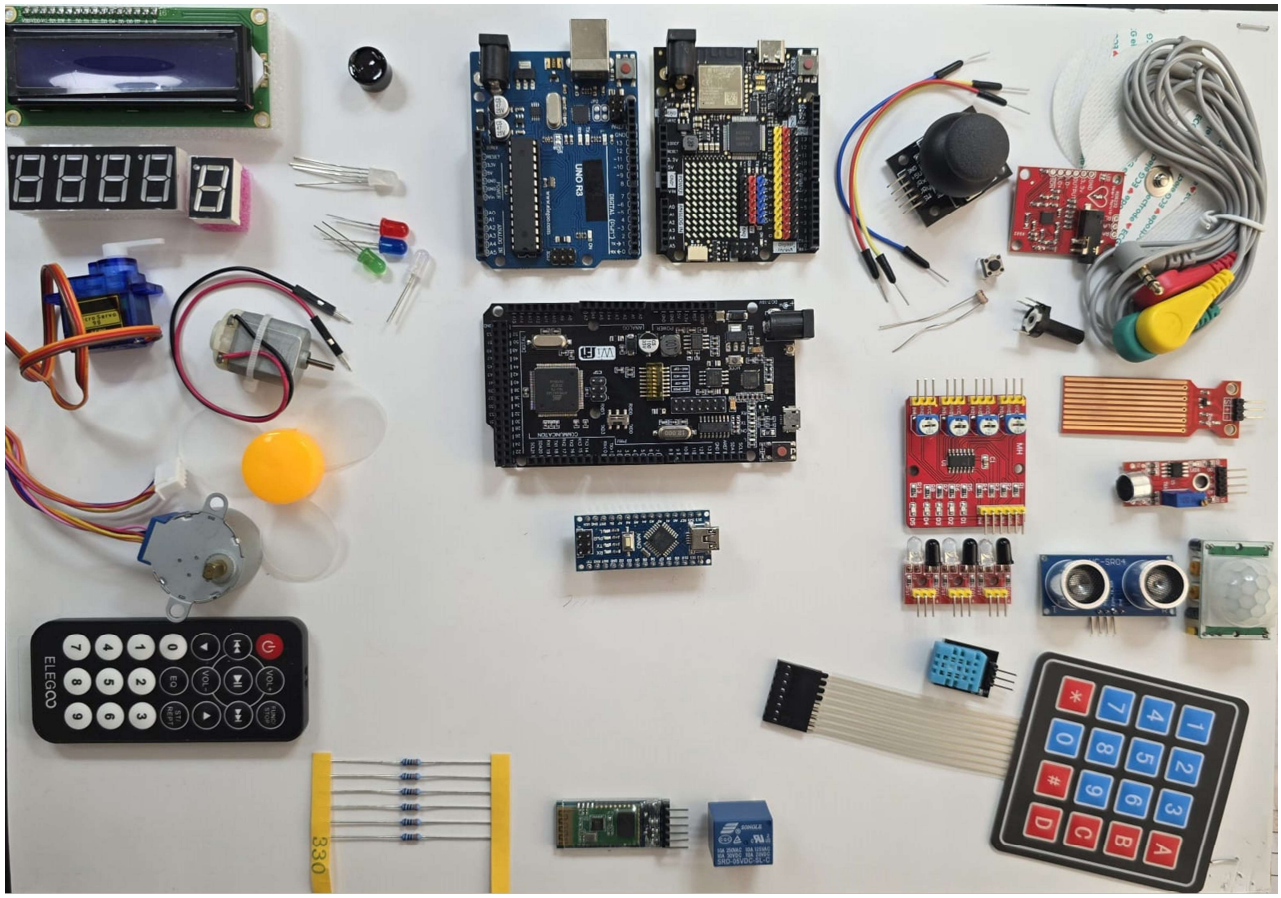
Robotics has a wide array of applications across various industries:

- **Manufacturing**: Robots are used for assembly, welding, painting, and packaging.
- **Healthcare**: Surgical robots assist in precise operations; rehabilitation robots aid patient recovery.
- **Service**: Robots perform tasks like cleaning, delivery, and customer service.
- **Exploration**: Drones and underwater robots explore difficult-to-reach environments (space, ocean depths).
- **Education**: Educational robots help teach programming and engineering concepts.

3- Key Components of Robotics

- **Sensors (Input)**: is a device that detects changes in the environment and converts that information into a readable signal or data (e.g., cameras, ultrasonic sensors).
- **Actuators(Output)**: actuator is a device that converts a control signal into physical motion (e.g., Motors or servos that allow the robot to move)
- **Controller**: A microcontroller or computer that executes the robot's programs (e.g., Arduino, ..)
- **Power Supply**: Batteries or other power sources that provide energy.
- **Software**: Programming languages and frameworks used to control the robot.

Outputs **Controllers** **Inputs**



Lesson No. 2 - Introduction to Arduino

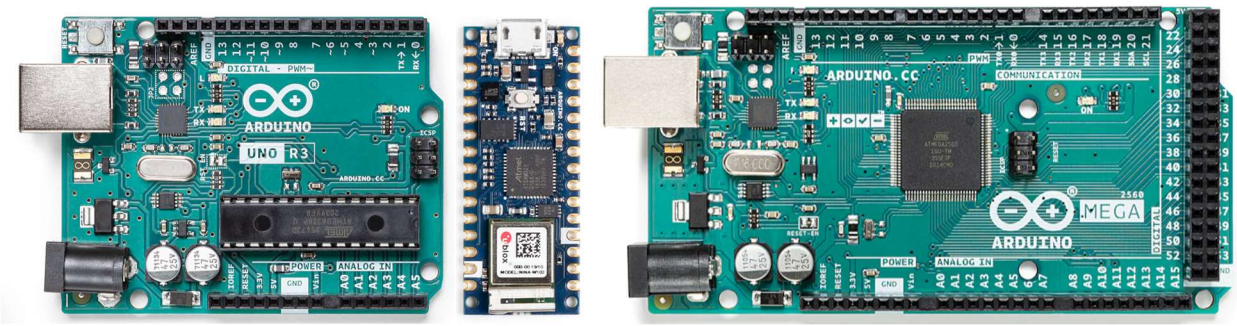
Arduino is an open-source electronics platform designed to make it easy for anyone to create interactive projects and prototypes. It consists of two main components: hardware and software.

- **Hardware**

Arduino boards come equipped with a microcontroller, which is the brain of the device, and a variety of input/output (I/O) pins. These pins can connect to sensors, motors, LEDs, and other electronic components, allowing you to create a wide range of projects.

Popular Arduino models include:

- **Arduino Uno**: The most widely used board, ideal for beginners.
- **Arduino Nano**: A compact version suitable for small projects.
- **Arduino Mega**: Offers more pins and memory, ideal for complex projects.



- **Software**

The Arduino Integrated Development Environment (IDE) is a user-friendly platform where you can write code in a **simplified version of C/C++.** The IDE allows you to upload your code to the Arduino board, making it easy to program and control your hardware.

- **Key Features**

- **User-Friendly**: Designed for simplicity, making it accessible to beginners and hobbyists.
- **Open Source**: Both hardware and software are open-source, encouraging community collaboration and innovation.
- **Versatile**: Supports a wide range of sensors, modules, and shields, which can be easily added to expand functionality.
- **Cross-Platform**: Available for various operating systems, including Windows, macOS, and Linux.

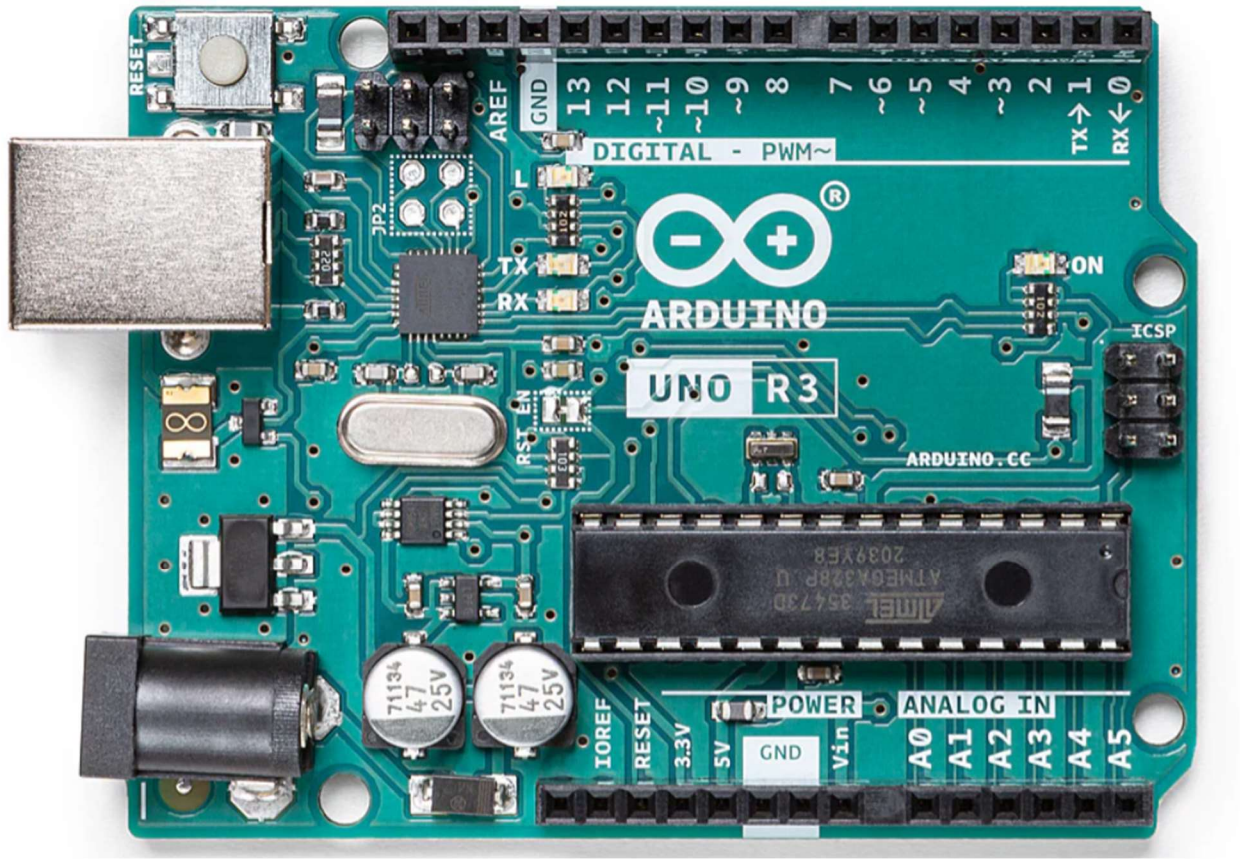
- **Applications**

Arduino is used in a variety of applications, including:

- **Prototyping**: Rapidly create and test electronic projects.
- **Education**: Used in schools to teach programming and electronics.
- **Home Automation**: Control devices like lights and thermostats.
- **Robotics**: Build robots that can perform tasks autonomously.
- **Art and Design**: Create interactive installations and wearable technology.

Arduino Uno R3 Board

The Arduino Uno R3 is one of the most popular and widely used boards in the Arduino family. It serves as an excellent entry point for beginners while also being versatile enough for more complex projects. Here's a detailed overview of the Arduino Uno R3 board:



1. Key Features

- **Microcontroller**: ATmega328P
- **Operating Voltage**: 5V
- **Input Voltage (recommended)**: 7-12V
- **Digital I/O Pins**: 14 (6 can be used as PWM outputs)
- **Analog Input Pins**: 6
- **Flash Memory**: 32 KB (of which 0.5 KB is used by the bootloader)
- **SRAM**: 2 KB
- **EEPROM**: 1 KB
- **Clock Speed**: 16 MHz

2. Pin Configuration

- **Digital Pins**: Used for input or output of digital signals. Pins 0-13 can be used for various tasks such as reading switches or controlling LEDs. (Try to avoid pin0 and pin1 as digital pins)
- **Pins 0 and Pin 1:**

  Pin0- RX: Receive pin for serial communication

  Pin1- TX: Transmit pin for serial communication

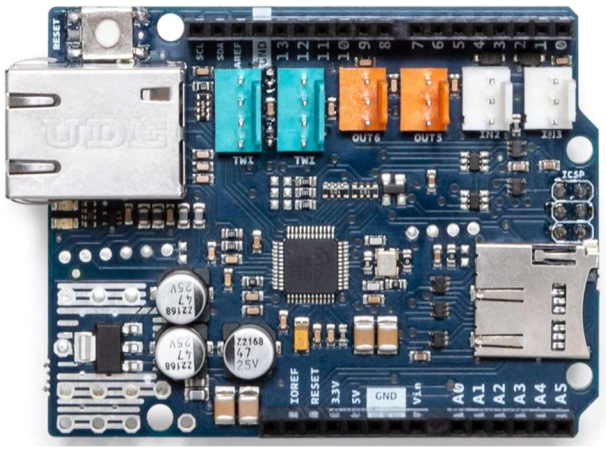
- **PWM Pins**: Pins 3, 5, 6, 9, 10, and 11 can provide Pulse Width Modulation (PWM) signals to simulate analog output.
- **Analog Pins**: Pins A0 to A5 are used for reading analog signals from sensors.

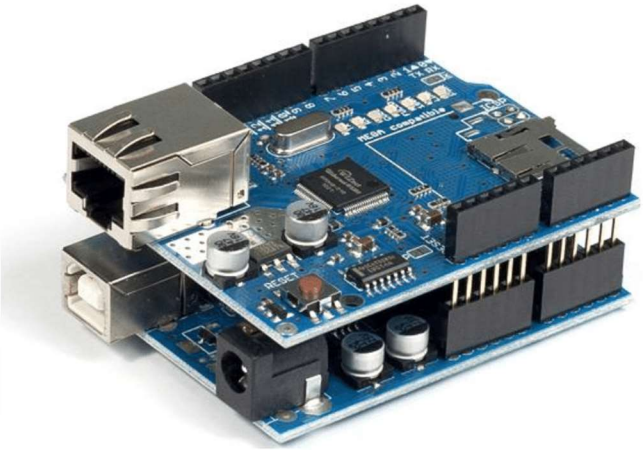
3. Connectivity

- **USB Connection**: For programming the board and providing power.
- **Power Jack**: For external power supply (7-12V).
- **Reset Button**: Used to reset the board.

4. Compatibility

The Arduino Uno R3 is compatible with a vast range of **shields** (add-on boards) and libraries, allowing users to extend its functionality easily. It also supports various sensors, motors, and other peripherals, making it suitable for diverse project

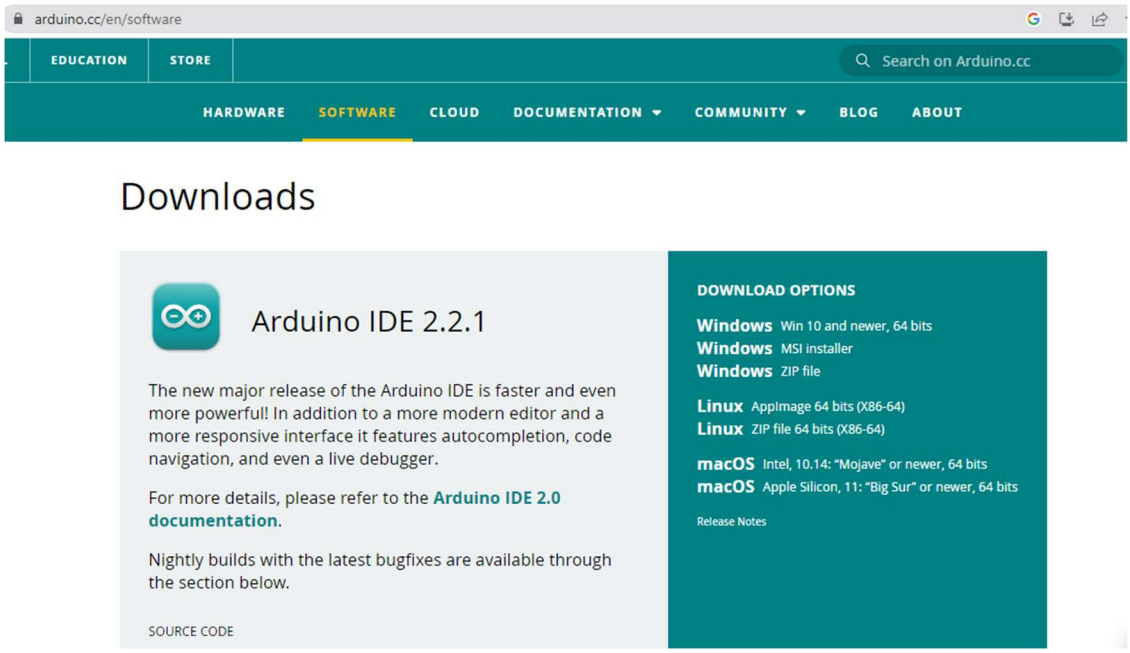




Software and Code

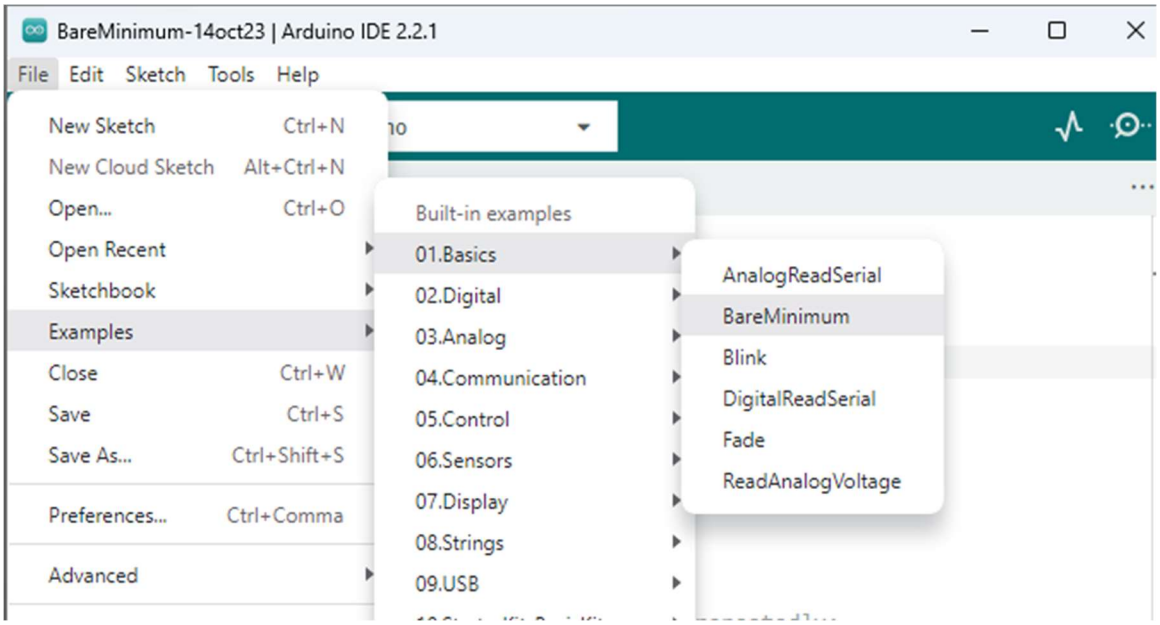
Software Setup – IDE (https://www.arduino.cc/en/software)

Click below on the one that match your operating system.

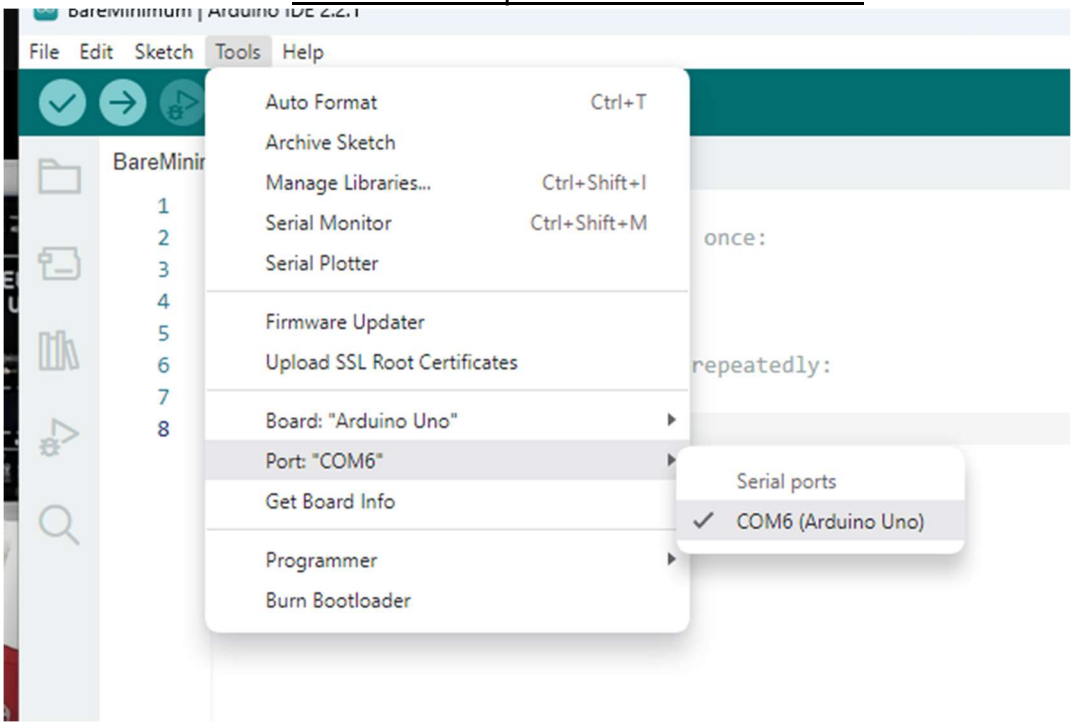


- **Writing the first program, Uploading code to the Arduino board and Controlling an LED:**

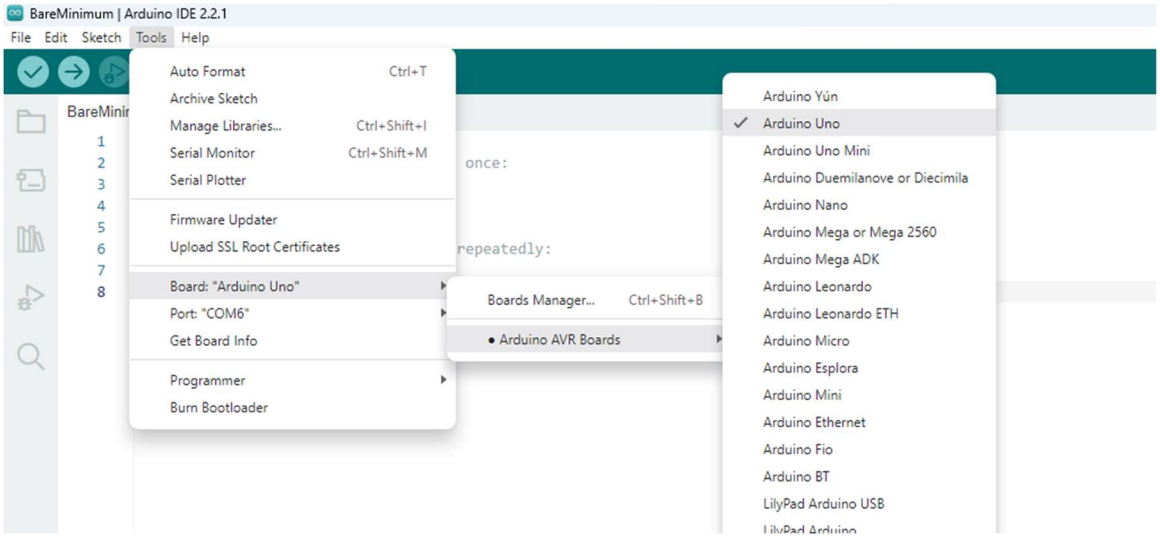
1- Open IDE Software:
2- Connect Arduino with Laptop
3- Then select BareMinimum as below:



4- Make sure the port is selected as below:



5- Select the type of ARDUINO that you use as below:



Structure The basic structure of the Arduino programming language is fairly simple and runs in at least two parts. These two required parts, or functions, enclose blocks of statements.

- **void setup ( ) { }:** This function runs once when the Arduino is powered on or reset. It is used to initialize variables, set pin modes, and start communication.
- **void loop ( ) { }**: This function runs continuously in a loop after the `setup()` function has completed. It's where the main logic of your program is executed.

```
void setup ( )
{
// put your setup code here, to run once:
}


void loop ( )
{
  // put your main code here, to run repeatedly:
}
```

Functions:

pinMode(pin, mode**) : pin** is the number of the pin**, Mode** is INPUT or an OUTPUT.

digitalWrite(pin**,** value**)** : **pin** is the number of the pin, **value** either logic level HIGH or LOW.

delay(ms): for timing and the number will be in (ms).

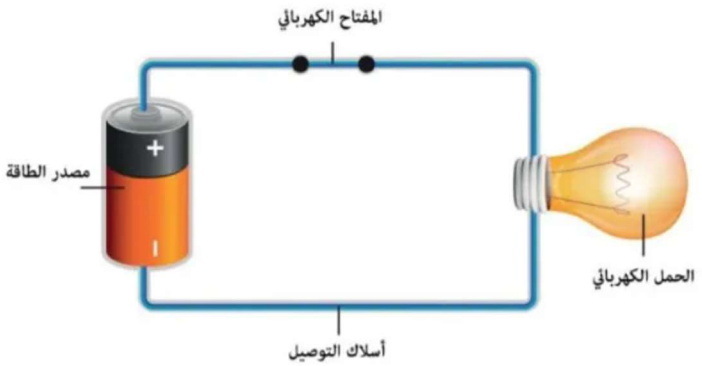
Code1: To turn on the built in LED - Pin13

```
void setup() {
  // put your setup code here, to run once:
  pinMode(13, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(13, HIGH);
}
```

Code2: To blink the built in LED - Pin13 - with timing 1000 ms

```
void setup() {
  // put your setup code here, to run once:
  pinMode(13, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

Lesson No. 3 - Basic Electronics Concepts



Voltage (V)

- **Definition**: Voltage, also known as electric potential difference, is the force that pushes electric charges through a circuit. It's measured in volts (V).
- **Analogy**: Think of voltage as water pressure in a hose. Higher pressure means more potential to push water through.

Current (I)

- **Definition**: Current is the flow of electric charge, measured in amperes (A). It represents how many electrons are flowing past a point in a circuit per second.
- **Analogy**: Current is like the flow rate of water in the hose. More water flowing means a higher current.

Resistance (R)

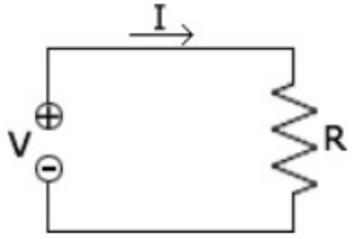
- **Definition**: Resistance is the opposition to the flow of current in a circuit, measured in ohms (Ω). It determines how much current will flow for a given voltage.

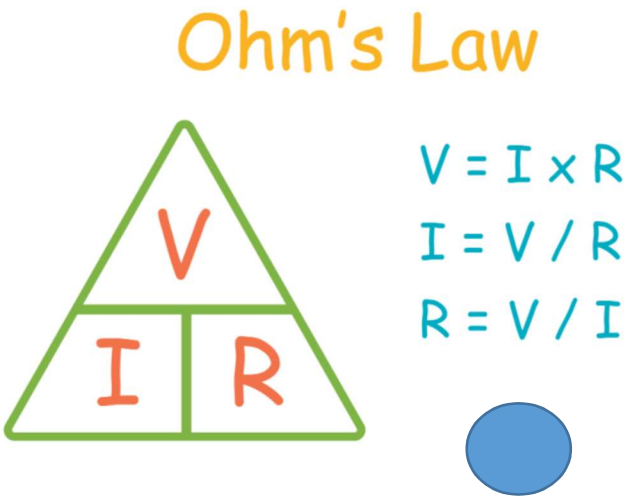
- **Analogy**: Resistance is like a narrowing in the hose that restricts water flow. The more narrow the hose, the higher the resistance.

Quantity	Ohm's Law symbol	Unit of measure (abbreviation)
Voltage	E	Volt (V)
Current	I	Ampere, amp (A)
Resistance	R	Ohm (Ω)

Ohm's Law

Ohm's Law connects these three concepts with the formula: V=I×R





- **Interpretation**:
  - If you increase the voltage (V) while keeping resistance (R) constant, the current (I) will increase.
  - If you increase resistance while keeping voltage constant, the current will decrease.

Key Takeaways

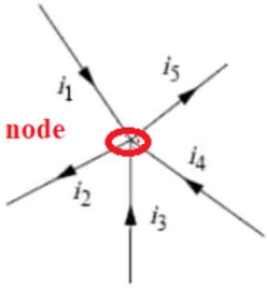
- **Voltage** provides the energy to move charges.

- **Current** is the movement of charges.

- **Resistance** restricts that movement

- **Kirchhoff's Current Law (KCL)**: This law states that the total current entering a junction (or node) in an electrical circuit must equal the total current leaving that junction. Essentially, it reflects the conservation of electric charge.

  ∑ I in = ∑ I out

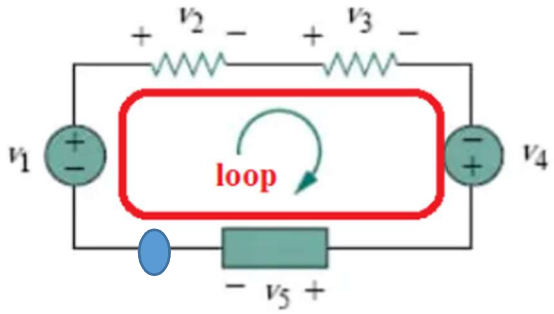


$$i_1 + i_3 + i_4 = i_2 + i_5$$

(KCL)

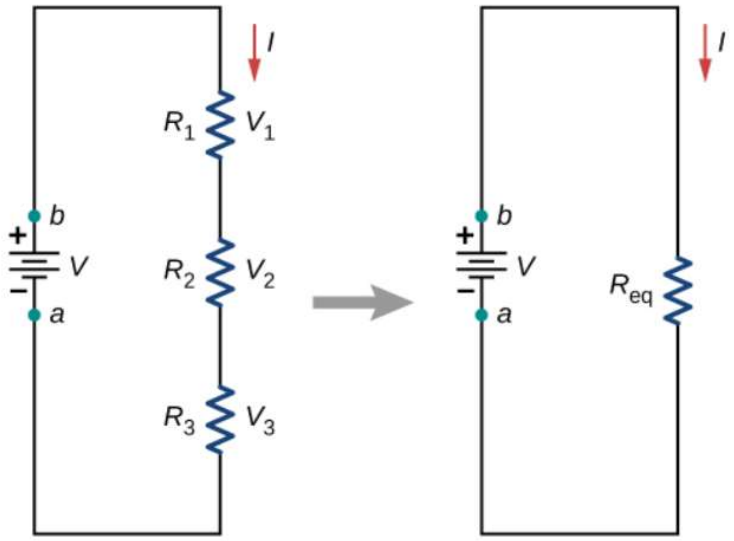
- **Kirchhoff's Voltage Law (KVL)**: This law states that the sum of the electrical potential differences (voltages) around any closed loop in a circuit must equal zero. This is based on the principle of conservation of energy.

  ∑ V = 0



$$-v_1 + v_2 + v_3 - v_4 + v_5 = 0$$

KVL



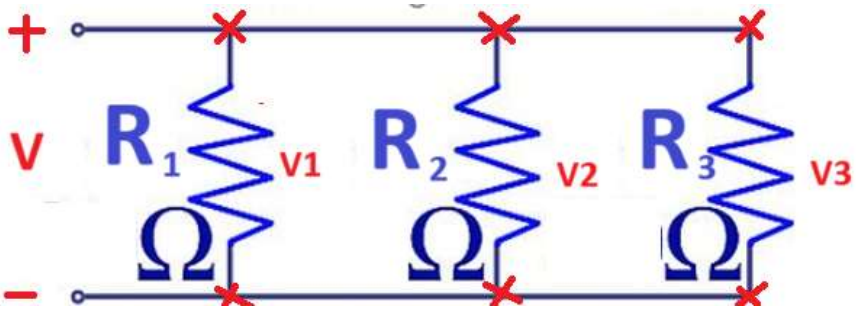
(a) Original circuit (b) Equivalent circuit

Series Resistors:

$R_{eq} = R_1 + R_2 + ... R_n$

$V = V1 + V2 + V3$

$I = I1 = I2 = I$



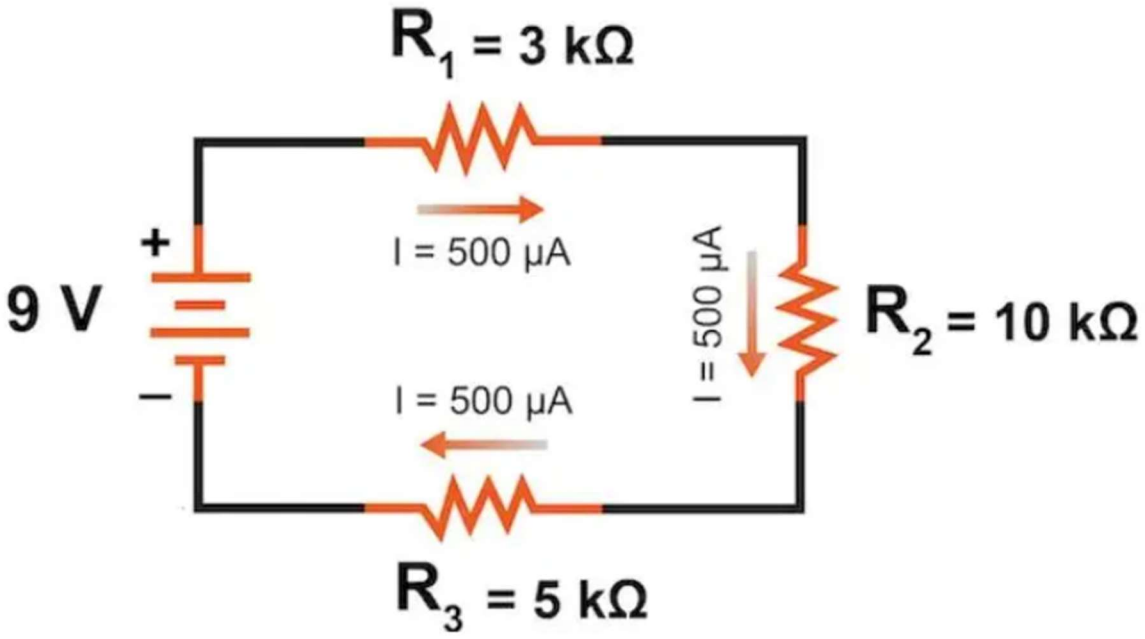
Parallel Resistors:

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

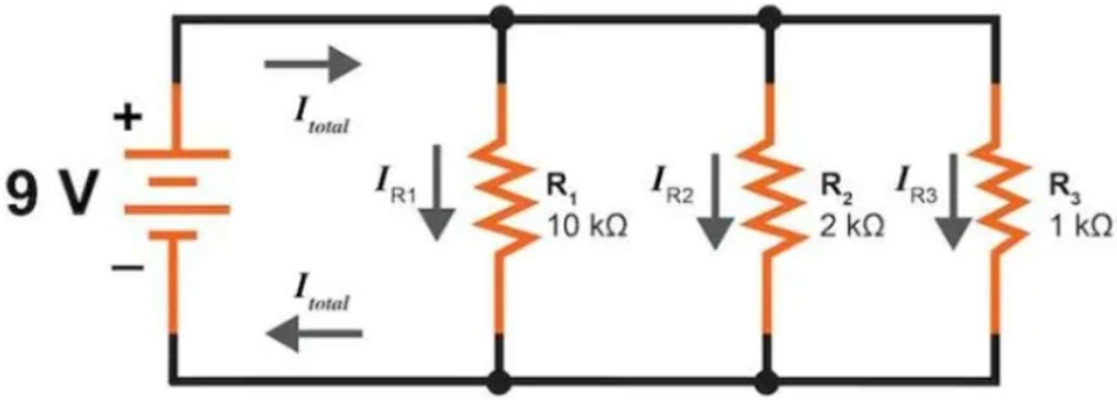
$V = V1 = V2 = V3$

$I = I1 + I2 + I3$

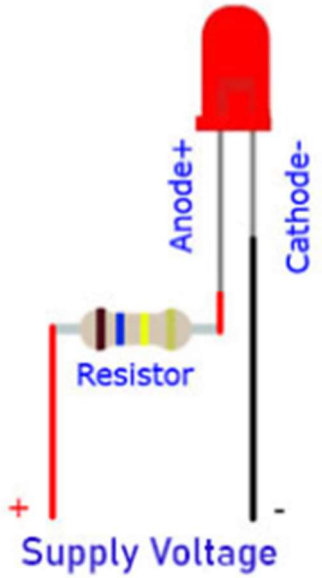
Homework 1: Calculate the voltage across R1 , R2 and R3

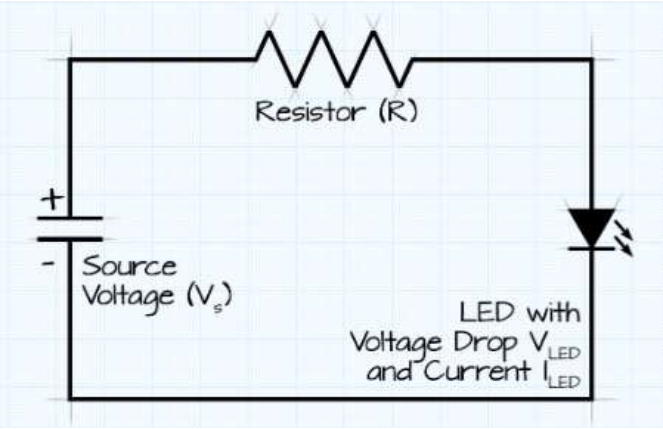


Homework 2: Calculate the Current I1 , I2 , I3 and I Total



Working with basic components: LEDs, resistors (330 Ohm)



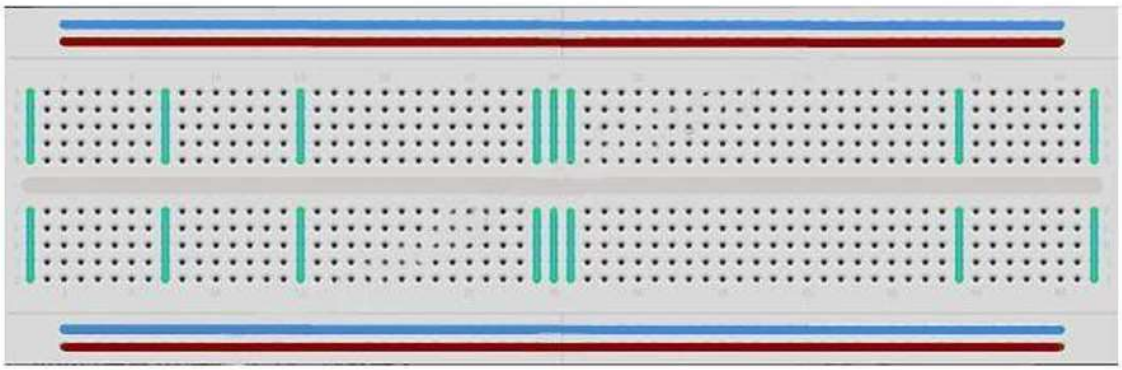


Diode LED (Light Emitting Diode): is a type of semiconductor material (p-n junction) that emits light when an electric current passes and this When a voltage is applied across the LED and the **forward voltage**, typically ranges from about **1.8V to 3.3V**, depending on the type and color of the LED.

Note: **P-type (Anode)** is a Positive side and **N-type (Cathode)** is a Negative side

What is breadboards

A breadboard enables you to prototype circuits quickly, without having to solder the connections. Below is an example



Wiring Connection of first program: "Blink" (blinking an LED)

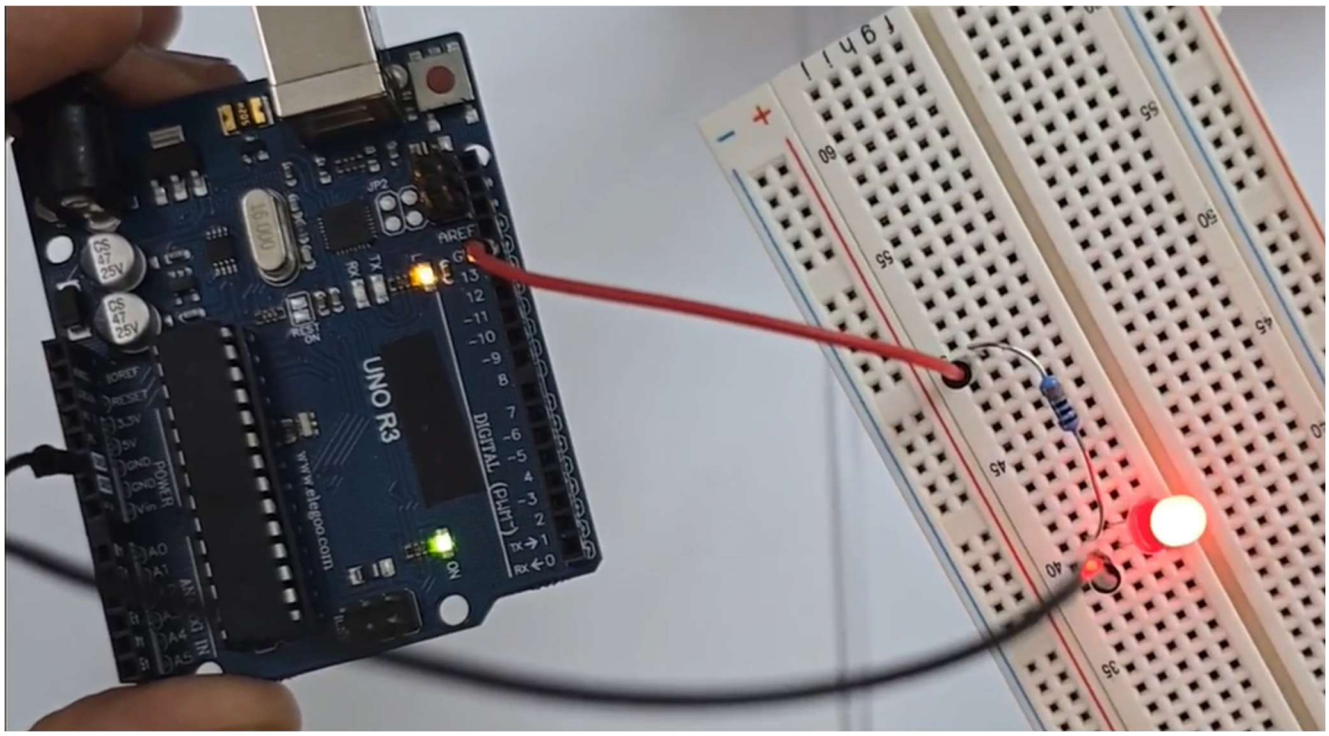
Blinking LED using Breadboard and PIN13 (Same code of blinking in Lesson 2)

Materials Needed:

1. Arduino Uno R3
2. LED (any color)
3. 330Ω resistor (to limit current through the LED)
4. Breadboard
5. Jumper wires

Connection Summary

1- PIN 13 on Arduino Connected to → One end of 330Ω resistor
2- another end of 330Ω resistor Connected to → Anode of LED (Long leg – Positive)
3- Cathode of LED (Short Leg – Negative) Connected to → Arduino GND



Code:

```
void setup() {
  // put your setup code here, to run once:
  pinMode(13, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

Lesson No. 4 - Variables

Variables are used to store data that your program can manipulate. In Arduino, you can use variables to keep track of information like sensor readings, user inputs, or any other data you need to work with.

Types of Variables

Here are some common types of variables in Arduino:

1. **int**: For integer values (e.g., -2, 0, 42).

   int myNumber = 10;

2. **float**: For floating-point numbers (decimal values).

   float temperature = 23.5;

3. **char**: For a single character.

   char letter = 'A';

4. **String**: For a sequence of characters (text).

   String message = "Hello, Arduino!";

5. **bool**: For boolean values (true/false).

   bool isOn = true;

Declaring Variables

You declare a variable by specifying its type followed by its name, and you can also initialize it with a value.

Scope of Variables

Variables can be declared in different scopes:

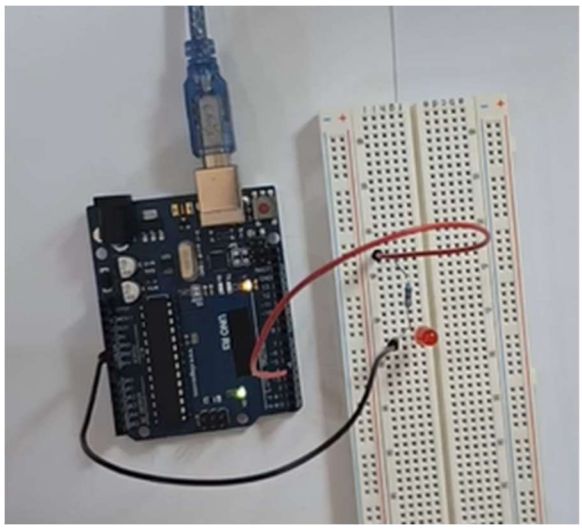
- **Global Variables**: Declared outside of functions and accessible from anywhere in the code.
- **Local Variables**: Declared inside a function and only accessible within that function.

Materials Needed:

- Arduino Uno R3
- LED (any color)
- 330Ω resistor (to limit current through the LED)
- Breadboard
- Jumper wires

Connection Summary

- PIN 5 on Arduino Connected to → One end of 330Ω resistor
- another end of 330Ω resistor Connected to → Anode of LED (Long leg – Positive)
- Cathode of LED (Short Leg – Negative) Connected to → Arduino GND



Code: Blink LED 3 times with delay 100 ms and 3 times with 500 ms.

```
int delayTS = 100;
int delayTL = 500;
int rLED = 5;
float Pi = 3.14;
String myName = "Robotics Island";

void setup() {
  // put your setup code here, to run once:
  pinMode(rLED, OUTPUT);
}
```

```
void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(rLED, HIGH);
  delay(delayTS);
  digitalWrite(rLED, LOW);
  delay(delayTS);

  digitalWrite(rLED, HIGH);
  delay(delayTS);
  digitalWrite(rLED, LOW);
  delay(delayTS);

  digitalWrite(rLED, HIGH);
  delay(delayTS);
  digitalWrite(rLED, LOW);
  delay(delayTS);

  digitalWrite(rLED, HIGH);
  delay(delayTL);
  digitalWrite(rLED, LOW);
  delay(delayTL);

  digitalWrite(rLED, HIGH);
  delay(delayTL);
  digitalWrite(rLED, LOW);
  delay(delayTL);

  digitalWrite(rLED, HIGH);
  delay(delayTL);
  digitalWrite(rLED, LOW);
  delay(delayTL);
}
```

Lesson No. 5 - Binary Numbers

1. **Definition**: Binary language consists of sequences of 0s and 1s (bits) that represent data, instructions, and operations in a form that computers can understand.

2. **How It Works**:
    - **Data Representation**: Characters, numbers, images, and sound are all encoded in binary. For instance, the letter 'A' is represented as 01000001 in ASCII (a common encoding scheme).
    - **Machine Code**: Programs are ultimately translated into binary machine code, which the computer's CPU executes directly

Bit

- **Definition**: The smallest unit of data in computing, represented as either 0 or 1.
- **Function**: Represents two states (off/on, true/false).

Byte

- **Definition**: A group of 8 bits.
- **Function**: The standard unit used to encode a single character of text in computer systems (e.g., letters, numbers).
- **Values**: Can represent 256 different values (from 0 to 255).

Larger Units

- **Kilobyte (KB)**: 1,024 bytes.
- **Megabyte (MB)**: 1,024 KB.
- **Gigabyte (GB)**: 1,024 MB.
- **Terabyte (TB)**: 1,024 GB.

Summary

- **Bit**: Single binary digit (0 or 1).
- **Byte**: 8 bits, used to represent a character or a small amount of data.

Converting between binary and decimal

Decimal - Binary - Octal - Hex – ASCII Conversion Chart

Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII
0	00000000	000	00	NUL	32	00100000	040	20	SP	64	01000000	100	40	@	96	01100000	140	60	`
1	00000001	001	01	SOH	33	00100001	041	21	!	65	01000001	101	41	A	97	01100001	141	61	a
2	00000010	002	02	STX	34	00100010	042	22	"	66	01000010	102	42	B	98	01100010	142	62	b
3	00000011	003	03	ETX	35	00100011	043	23	#	67	01000011	103	43	C	99	01100011	143	63	c
4	00000100	004	04	EOT	36	00100100	044	24	$	68	01000100	104	44	D	100	01100100	144	64	d
5	00000101	005	05	ENQ	37	00100101	045	25	%	69	01000101	105	45	E	101	01100101	145	65	e
6	00000110	006	06	ACK	38	00100110	046	26	&	70	01000110	106	46	F	102	01100110	146	66	f
7	00000111	007	07	BEL	39	00100111	047	27	'	71	01000111	107	47	G	103	01100111	147	67	g
8	00001000	010	08	BS	40	00101000	050	28	(	72	01001000	110	48	H	104	01101000	150	68	h
9	00001001	011	09	HT	41	00101001	051	29	)	73	01001001	111	49	I	105	01101001	151	69	i
10	00001010	012	0A	LF	42	00101010	052	2A	*	74	01001010	112	4A	J	106	01101010	152	6A	j
11	00001011	013	0B	VT	43	00101011	053	2B	+	75	01001011	113	4B	K	107	01101011	153	6B	k
12	00001100	014	0C	FF	44	00101100	054	2C	,	76	01001100	114	4C	L	108	01101100	154	6C	l
13	00001101	015	0D	CR	45	00101101	055	2D	-	77	01001101	115	4D	M	109	01101101	155	6D	m
14	00001110	016	0E	SO	46	00101110	056	2E	.	78	01001110	116	4E	N	110	01101110	156	6E	n
15	00001111	017	0F	SI	47	00101111	057	2F	/	79	01001111	117	4F	O	111	01101111	157	6F	o
16	00010000	020	10	DLE	48	00110000	060	30	0	80	01010000	120	50	P	112	01110000	160	70	p
17	00010001	021	11	DC1	49	00110001	061	31	1	81	01010001	121	51	Q	113	01110001	161	71	q
18	00010010	022	12	DC2	50	00110010	062	32	2	82	01010010	122	52	R	114	01110010	162	72	r
19	00010011	023	13	DC3	51	00110011	063	33	3	83	01010011	123	53	S	115	01110011	163	73	s
20	00010100	024	14	DC4	52	00110100	064	34	4	84	01010100	124	54	T	116	01110100	164	74	t
21	00010101	025	15	NAK	53	00110101	065	35	5	85	01010101	125	55	U	117	01110101	165	75	u
22	00010110	026	16	SYN	54	00110110	066	36	6	86	01010110	126	56	V	118	01110110	166	76	v
23	00010111	027	17	ETB	55	00110111	067	37	7	87	01010111	127	57	W	119	01110111	167	77	w
24	00011000	030	18	CAN	56	00111000	070	38	8	88	01011000	130	58	X	120	01111000	170	78	x
25	00011001	031	19	EM	57	00111001	071	39	9	89	01011001	131	59	Y	121	01111001	171	79	y
26	00011010	032	1A	SUB	58	00111010	072	3A	:	90	01011010	132	5A	Z	122	01111010	172	7A	z
27	00011011	033	1B	ESC	59	00111011	073	3B	;	91	01011011	133	5B	[	123	01111011	173	7B	{
28	00011100	034	1C	FS	60	00111100	074	3C	<	92	01011100	134	5C	\	124	01111100	174	7C	|
29	00011101	035	1D	GS	61	00111101	075	3D	=	93	01011101	135	5D	]	125	01111101	175	7D	}
30	00011110	036	1E	RS	62	00111110	076	3E	>	94	01011110	136	5E	^	126	01111110	176	7E	~
31	00011111	037	1F	US	63	00111111	077	3F	?	95	01011111	137	5F	_	127	01111111	177	7F	DEL

Note : HIGH/LOW These constants define pin levels as HIGH or LOW and are used when reading or writing to digital pins in software.

HIGH is defined as logic level 1, ON, or 5 volts

LOW is defined as logic level 0, OFF, or 0 volts.

Lesson No. 6 - Digital Output for

Working with multiple LEDs for **Binary Counting 4 Bits (0 to 15 in decimal)**

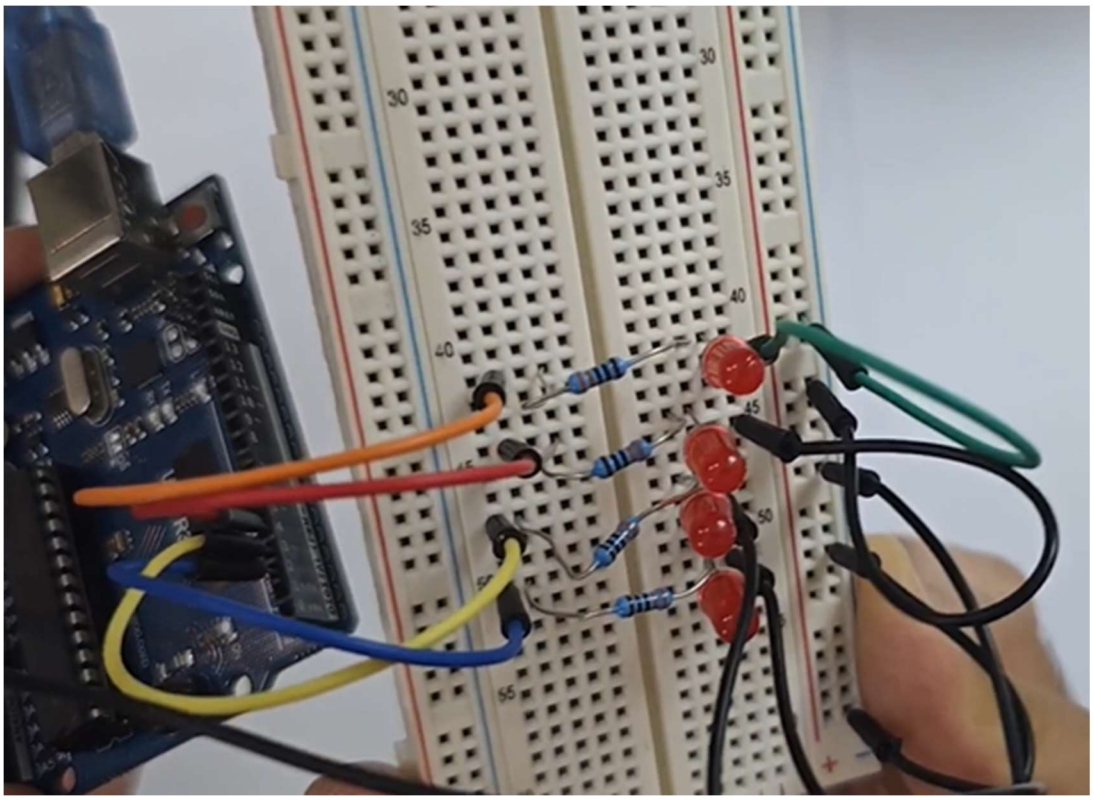
- Function: **digitalWrite()**

Materials Needed:

- Arduino Uno R3
- LED (any color) * 4 pcs
- 330Ω resistor (to limit current through the LED) * 4 pcs
- Breadboard
- Jumper wires

Connection Summary

- PIN 2 on Arduino Connected to → One end of 330Ω resistor
- another end of 330Ω resistor Connected to → Anode of LED (Long leg – Positive)
- Cathode of LED (Short Leg – Negative) Connected to → Arduino GND
- Repeat the above for pins 3 , 4 and 5



Code:

```
int bit1 = 2;
int bit2 = 3;
int bit3 = 4;
int bit4 = 5;
int dt = 1000;

void setup() {
  pinMode(bit1, OUTPUT); // put your setup code here, to run once:
  pinMode(bit2, OUTPUT);
  pinMode(bit3, OUTPUT);
  pinMode(bit4, OUTPUT);
}
void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(bit1, LOW);
  digitalWrite(bit2, LOW);
  digitalWrite(bit3, LOW);
  digitalWrite(bit4, LOW);
  delay(dt);

  digitalWrite(bit1, HIGH);
  digitalWrite(bit2, LOW);
  digitalWrite(bit3, LOW);
  digitalWrite(bit4, LOW);
  delay(dt);

  digitalWrite(bit1, LOW);
  digitalWrite(bit2, HIGH);
  digitalWrite(bit3, LOW);
  digitalWrite(bit4, LOW);
  delay(dt);

  digitalWrite(bit1, HIGH);
  digitalWrite(bit2, HIGH);
  digitalWrite(bit3, LOW);
  digitalWrite(bit4, LOW);
  delay(dt);
```

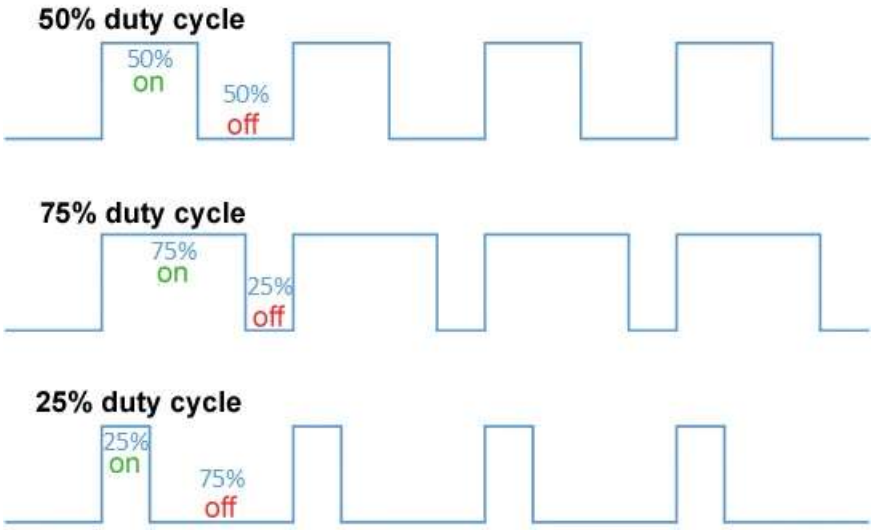
```
  digitalWrite(bit1, LOW);
  digitalWrite(bit2, LOW);
  digitalWrite(bit3, HIGH);
  digitalWrite(bit4, LOW);
  delay(dt);

  digitalWrite(bit1, HIGH);
  digitalWrite(bit2, LOW);
  digitalWrite(bit3, HIGH);
  digitalWrite(bit4, LOW);
  delay(dt);

  digitalWrite(bit1, LOW);
  digitalWrite(bit2, HIGH);
  digitalWrite(bit3, HIGH);
  digitalWrite(bit4, LOW);
  delay(dt);

  digitalWrite(bit1, HIGH);
  digitalWrite(bit2, HIGH);
  digitalWrite(bit3, HIGH);
  digitalWrite(bit4, LOW);
  delay(dt);

  digitalWrite(bit1, LOW);
  digitalWrite(bit2, LOW);
  digitalWrite(bit3, LOW);
  digitalWrite(bit4, HIGH);
  delay(dt);

  digitalWrite(bit1, HIGH);
  digitalWrite(bit2, LOW);
  digitalWrite(bit3, LOW);
  digitalWrite(bit4, HIGH);
  delay(dt);

  digitalWrite(bit1, LOW);
  digitalWrite(bit2, HIGH);
  digitalWrite(bit3, LOW);
```

```
  digitalWrite(bit4, HIGH);
  delay(dt);

  digitalWrite(bit1, HIGH);
  digitalWrite(bit2, HIGH);
  digitalWrite(bit3, LOW);
  digitalWrite(bit4, HIGH);
  delay(dt);

  digitalWrite(bit1, LOW);
  digitalWrite(bit2, LOW);
  digitalWrite(bit3, HIGH);
  digitalWrite(bit4, HIGH);
  delay(dt);

  digitalWrite(bit1, HIGH);
  digitalWrite(bit2, LOW);
  digitalWrite(bit3, HIGH);
  digitalWrite(bit4, HIGH);
  delay(dt);

  digitalWrite(bit1, LOW);
  digitalWrite(bit2, HIGH);
  digitalWrite(bit3, HIGH);
  digitalWrite(bit4, HIGH);
  delay(dt);

  digitalWrite(bit1, HIGH);
  digitalWrite(bit2, HIGH);
  digitalWrite(bit3, HIGH);
  digitalWrite(bit4, HIGH);
  delay(dt);
}
```

Lesson No. 7 - Analog Output and PWM

Introduction to Pulse Width Modulation (PWM)

1. **Basic Concept**: PWM works by varying the width (duration) of the pulses in a signal while keeping the frequency constant. The duty cycle, which is the ratio of the pulse "on" time to the total period of the signal, determines the average power delivered.
2. **Duty Cycle**:
   - Expressed as a percentage.
   - A 50% duty cycle means the signal is on half the time and off half the time.
   - A higher duty cycle increases the average voltage/power delivered to a load.
3. **Signal Representation**:
   - A typical PWM signal alternates between high (1) and low (0) states.
   - By changing the duration of the high state, you can control the effective voltage and power supplied.



- Using **analogWrite(pin,value)** to control LED brightness value from 0-255.

analogWrite is a function in Arduino used to output a PWM (Pulse Width Modulation) signal on certain digital pins. It allows you to control the brightness of LEDs, the speed of motors, and other applications where variable voltage is needed.

- Function **: analogWrite(pin, value);**

Parameters

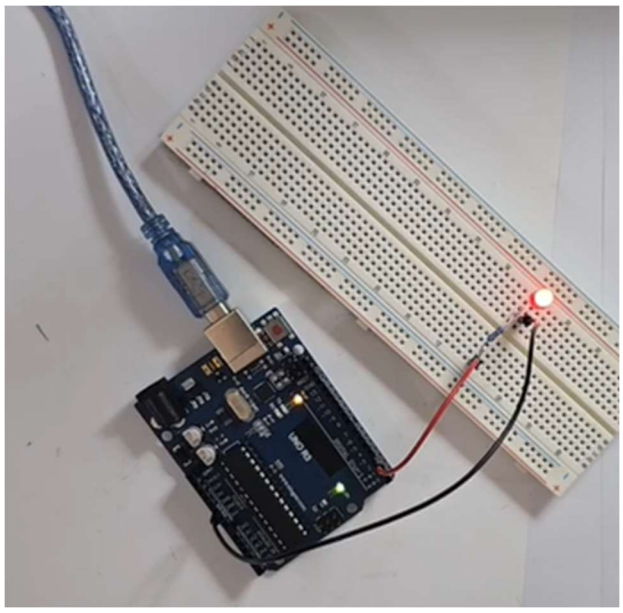
- **pin**: The number of the pin you want to write to (must support PWM).
- **value**: A number between 0 (0% duty cycle) and 255 (100% duty cycle).
  - 0 turns the output off.
  - 255 outputs a constant high signal.

Materials Needed:

- Arduino Uno R3
- LED (any color)
- 330Ω resistor (to limit current through the LED)
- Breadboard
- Jumper wires

Connection Summary

- PIN 3 on Arduino Connected to → One end of 330Ω resistor
- another end of 330Ω resistor Connected to → Anode of LED (Long leg – Positive)
- Cathode of LED (Short Leg – Negative) Connected to → Arduino GND

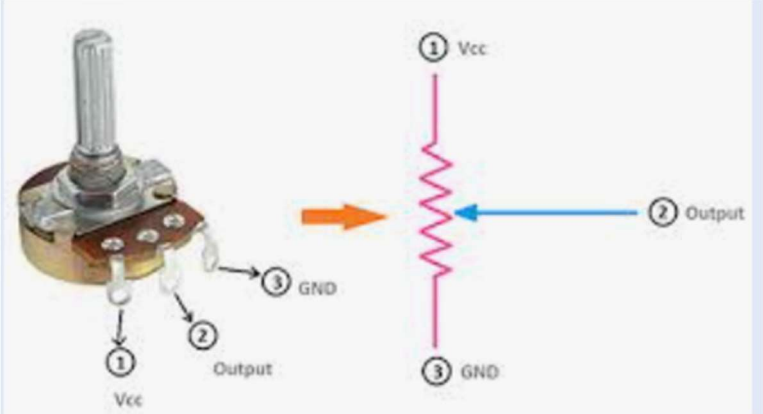


Code:

```
int LEDpin = 3;
int LEDbright = 125;

void setup() {
  pinMode(LEDpin, OUTPUT);  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
  analogWrite(LEDpin, LEDbright);
}
```

Lesson No. 8 - Analog Input

Potentiometer



A potentiometer is a **variable resistor** that has three terminals:

- **Two fixed terminals** connected to a resistive element.
- **One adjustable terminal** (the wiper) that moves along the resistive element.

Operation: By turning the knob or sliding the lever, you change the position of the wiper, which adjusts the resistance between the wiper and the fixed terminals. This allows you to vary the output voltage.

Reading from analog sensors (like a potentiometer)

- Function: **analogRead()**
- variable = analogRead(pin);

The **analogRead(potPin)** function reads the voltage from the potentiometer and gives a value between 0 and 1023.

- Functions: **Serial.begin(), Serial.print(), Serial.println()**

Using the serial monitor to view sensor data and debugging

To read analog value we have to use pins (A0 to A5)

Note:

1- Analog input from 0 to 1023 , so 5 volt = 1023 , so if we need to know the value of input voltage we use the below equation:

```
readvalue=analogRead(readpin);
Vin=(5./1023.)*readvalue;
```

2- If we want to use print we will do the below:
Serial.begin(9600); and should equal the baud rate in software.

We use Serial.println(Vin); to print and then new line
And we use Serial.print(Vin); to print next each other.

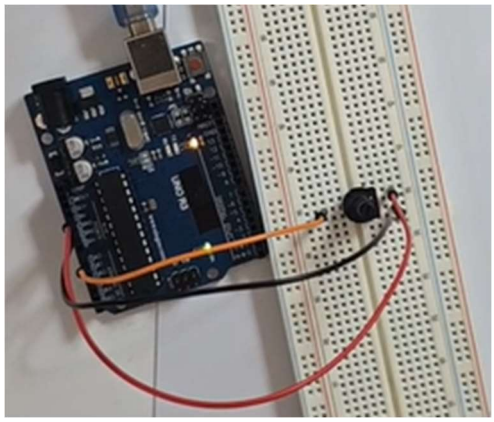
Wiring a Potentiometer with Arduino

Components Needed

- Arduino board
- Potentiometer (e.g., 10k ohm)
- Breadboard and jumper wires

Wiring Diagram

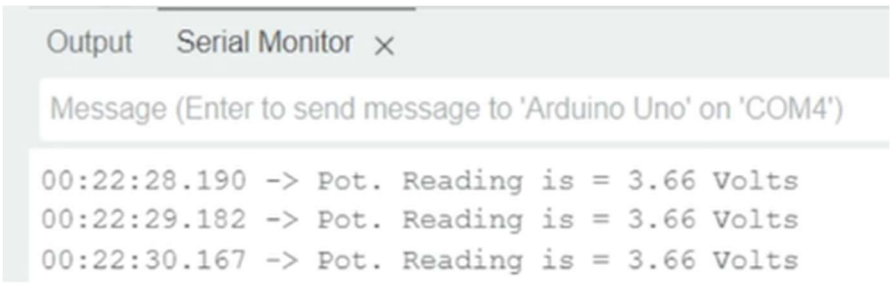
1. Connect one outer terminal of the potentiometer to **GND** (ground).
2. Connect the other outer terminal of the potentiometer to **VCC** (usually 5V).
3. Connect the middle terminal (wiper) to an **analog pin** on the Arduino (e.g., **A0**).



Code:

```
int v2 = A0;
int readVal;
int dt = 1000;
float readVolt;
String msg1 = "Pot. Reading is = ";
String msg2 = " Volts";

void setup() {
  pinMode(v2, INPUT);
  Serial.begin(9600);
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
  readVal = analogRead(v2);
  readVolt = (readVal / 1023.) * 5.;
  Serial.print(msg1);
  Serial.print(readVolt);
  Serial.println(msg2);
  delay(dt);
}
```

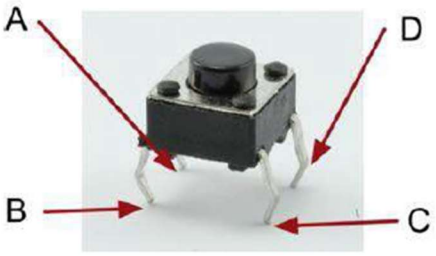
Serial Reading:

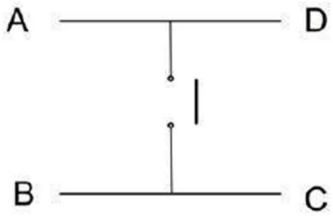


Lesson No. 9 - Digital Input

- Connecting and reading a **pushbutton**
  - Functions: **pinMode(), digitalRead()**

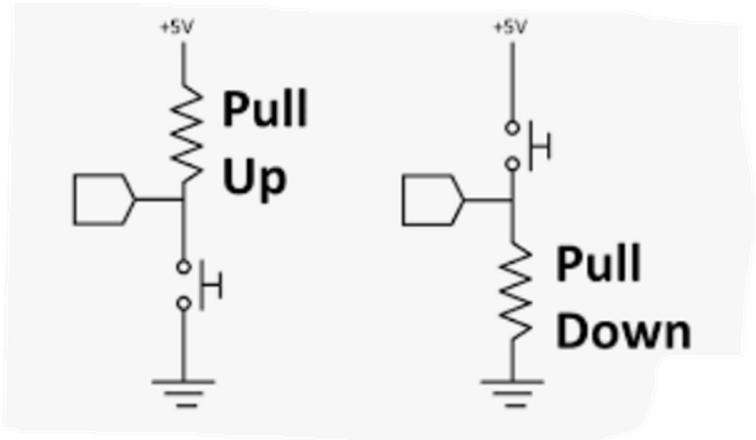
variable = digitalRead(Pin);



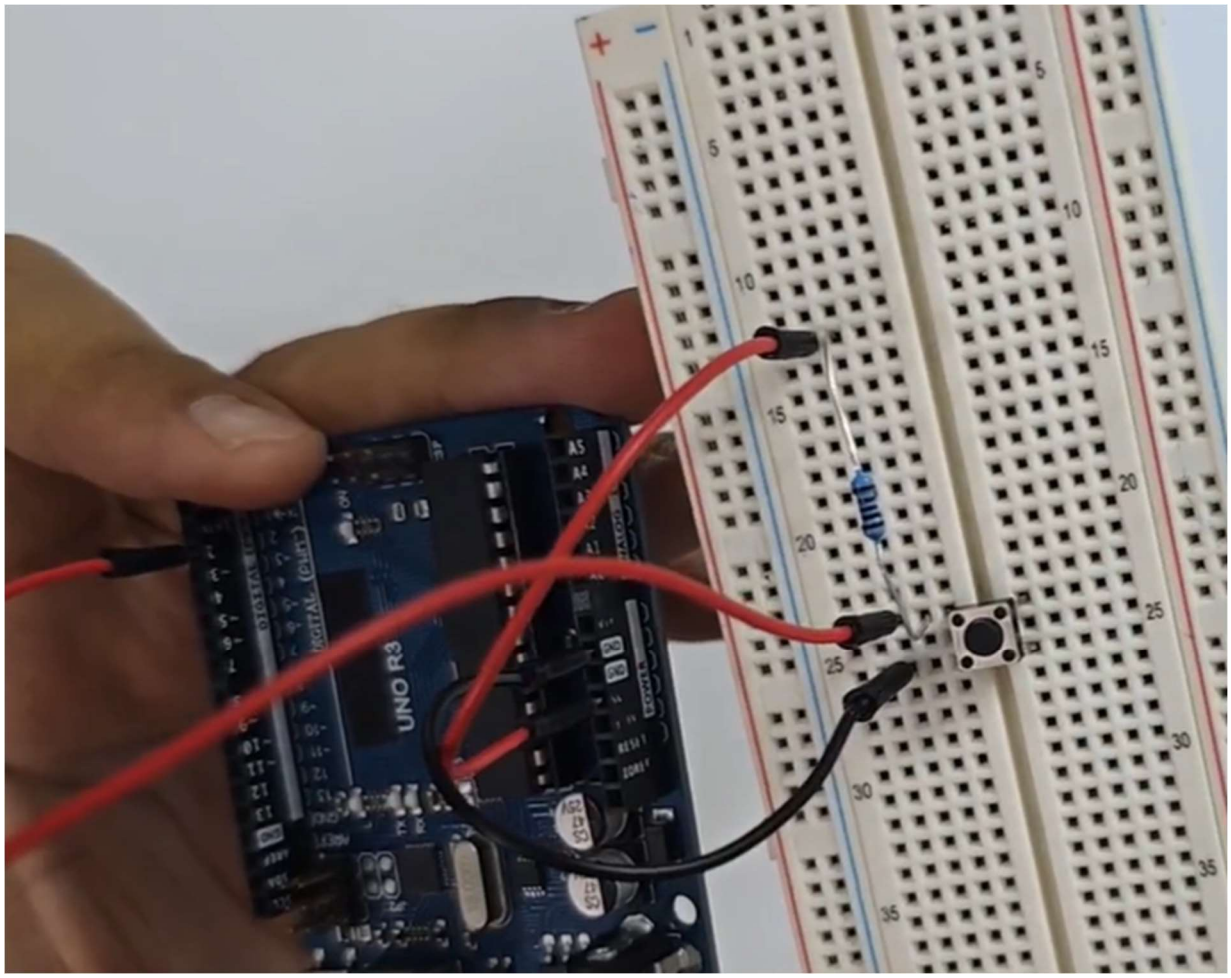


Actually, there are only really two electrical connections. Inside the switch package, pins B and C are connected together, as are A and D.

- **Pull up and Pull Down Resistor**
- **R=10K ohm**

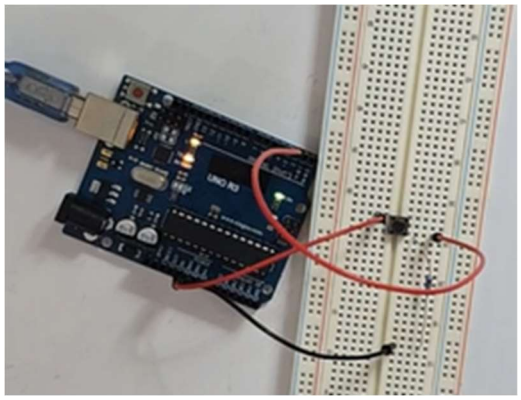


Pull Up Resistor

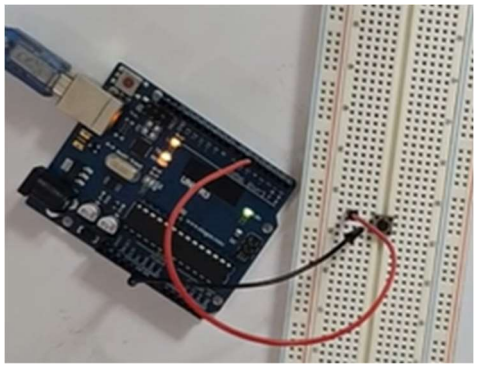



```
int PB = 5;
int PBStatus;

void setup() {
  // put your setup code here, to run once:
  pinMode(PB, INPUT);
  digitalWrite(PB, HIGH);
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  PBStatus = digitalRead(PB);
  Serial.println(PBStatus);
}
```

Pull Down Resistor



Simplest way to use pushbutton without external resistor



```
int PB=5;
int PBStatus;

void setup() {
// put your setup code here, to run once:
pinMode(PB,INPUT);
digitalWrite(PB,HIGH);
Serial.begin(9600);
}
void loop() {
 // put your main code here, to run repeatedly:
PBStatus=digitalRead(PB);
Serial.println(PBStatus);
}
```

Lesson No. 10 - Control Structures – (if) and (if...else) statements

Order executed with conditions (ex: turn RED LED on when the Voltage is 3 to 4 Volts)

```
if (someVariable ?? value)
{
doSomething;
}
else
 {
doThingB;
}
```

comparison operators

if(<) Less
if(>) Greater
if(==) Equal
if(>=) Greater than or equal
if(<=) Less than or equal

logical operators

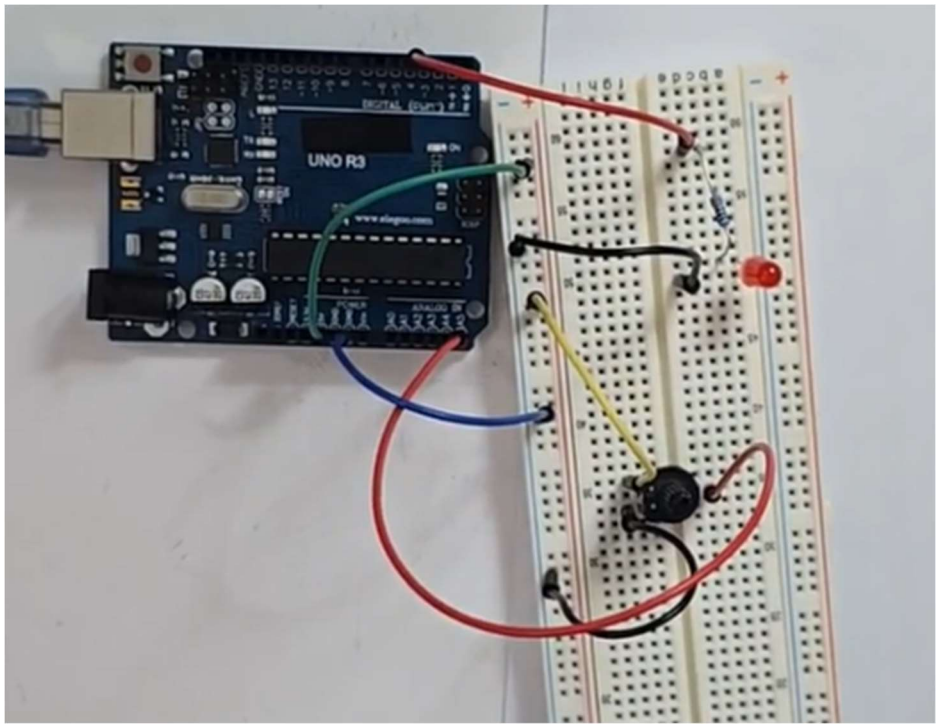
if(!=) Not equal
if(&&) AND
if(||) OR

Materials Needed:

- Arduino Uno R3
- LED (any color)
- 330Ω resistor (to limit current through the LED)
- Potentiometer (10k ohm)
- Breadboard
- Jumper wires

Connection Summary

- Connect one outer terminal of the potentiometer to **GND** (ground).
- Connect the other outer terminal of the potentiometer to **VCC** (usually 5V).
- Connect the middle terminal (wiper) to an **analog pin** on the Arduino (**A5**).
- PIN 3 on Arduino Connected to → One end of 330Ω resistor
- another end of 330Ω resistor Connected to → Anode of LED (Long leg – Positive)
- Cathode of LED (Short Leg – Negative) Connected to → Arduino GND



If voltage >=3 and <= 4 , RED LED turn on.

Code:

```
int potPin = A5;
int potVal;
float potVoltage;
int LEDPin = 3;

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

void loop() {
  // put your main code here, to run repeatedly:
  potVal = analogRead(potPin);
  potVoltage = potVal * (5. / 1023.);
  Serial.println(potVoltage);

  if (potVoltage >= 3 && potVoltage <= 4) {
    digitalWrite(LEDPin, HIGH);
  } else {
    digitalWrite(LEDPin, LOW);
  }
}
```

Lesson No. 11 - Control Structures - for loops

```
for (initialization; condition; expression)
{
doSomething;
 }
```

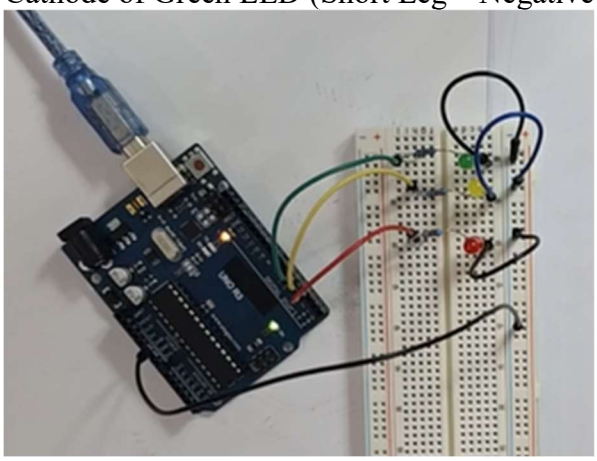
To do :
Blink Red LED, Yellow LED & Green LED as per number you determine (ex: Red LED 6 times , Yellow LED 4 times and Green LED 2 times)

Materials Needed:

- Arduino Uno R3
- LED (Red , Yellow and Green)
- 330Ω resistor (to limit current through the LED) * 3 pcs
- Breadboard
- Jumper wires

Connection Summary

- **PIN 3** on Arduino Connected to → One end of 330Ω resistor
- another end of 330Ω resistor Connected to → Anode of Red LED (Long leg – Positive)
- Cathode of Red LED (Short Leg – Negative) Connected to → Arduino GND
- **PIN 4** on Arduino Connected to → One end of 330Ω resistor
- another end of 330Ω resistor Connected to → Anode of Yellow LED (Long leg – Positive)
- Cathode of Yellow LED (Short Leg – Negative) Connected to → Arduino GND
- **PIN 5** on Arduino Connected to → One end of 330Ω resistor
- another end of 330Ω resistor Connected to → Anode of Green LED (Long leg – Positive)
- Cathode of Green LED (Short Leg – Negative) Connected to → Arduino GND



Code:

```
int rLEDpin = 3;
int yLEDpin = 4;
int gLEDpin = 5;
int dt = 500;
int j;
int blinkR = 6;
int blinkY = 4;
int blinkG = 2;

void setup() {
  // put your setup code here, to run once:
  pinMode(rLEDpin, OUTPUT);
  pinMode(yLEDpin, OUTPUT);
  pinMode(gLEDpin, OUTPUT);
}
void loop() {
  // put your main code here, to run repeatedly:
  for (j = 1; j <= blinkR; j = j + 1) {
    digitalWrite(rLEDpin, HIGH);
    delay(dt);
    digitalWrite(rLEDpin, LOW);
    delay(dt);
  }

  for (j = 1; j <= blinkY; j = j + 1) {
    digitalWrite(yLEDpin, HIGH);
    delay(dt);
    digitalWrite(yLEDpin, LOW);
    delay(dt);
  }

  for (j = 1; j <= blinkG; j = j + 1) {
    digitalWrite(gLEDpin, HIGH);
    delay(dt);
    digitalWrite(gLEDpin, LOW);
    delay(dt);
  }
}
```

Lesson No. 12 - Control Structures - while loops

```
while (someVariable ?? value)
{
doSomething;
 }
```

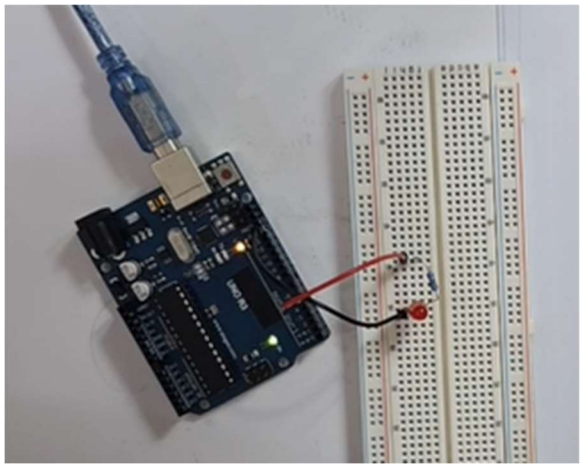
To do :
Blink Red LED as per number you determine (ex: Red LED 6 times)

Materials Needed:

- Arduino Uno R3
- LED (Red)
- 330Ω resistor (to limit current through the LED)
- Breadboard
- Jumper wires

Connection Summary

- **PIN 5** on Arduino Connected to → One end of 330Ω resistor
- another end of 330Ω resistor Connected to → Anode of Red LED (Long leg – Positive)
- Cathode of Red LED (Short Leg – Negative) Connected to → Arduino GND



Code:

```
int j;
int dt = 500;
int dt2 = 2000;
int rLED = 5;

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

void loop() {
  // put your main code here, to run repeatedly:
  j = 1;
  while (j <= 5) {
    digitalWrite(rLED, HIGH);
    delay(dt);
    digitalWrite(rLED, LOW);
    delay(dt);
    j = j + 1;
  }
  delay(dt2);
}
```

Lesson No. 13 - Serial Communication

- Using the serial monitor for debugging
  - Functions: **Serial.begin(), Serial.print(), Serial.println()**
- Sending and receiving simple serial data
  - Functions: **Serial.parseInt(),Serial.parseFloat(),Serial.available()**
- **Reading Int Number:**

```
while (Serial.available() == 0) {
  }
  blink = Serial.parseInt();
```

- **Reading Float Number:**

```
while (Serial.available() == 0) {
  }
  blink = Serial.parseFloat();
```

- **Reading String**

```
while (Serial.available() == 0) {
  }
name=Serial.readString();
```

To do :
Blink Red LED as per number you determine from your KEYBOARD (Serial Read)

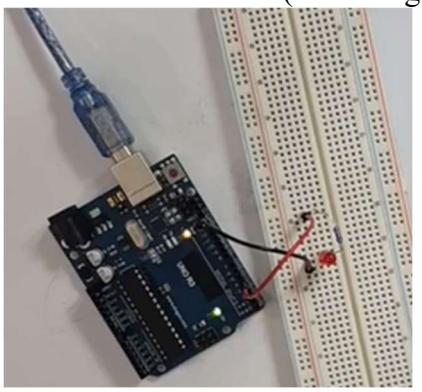
Materials Needed:

- Arduino Uno R3
- LED (Red)
- 330Ω resistor (to limit current through the LED)
- Breadboard
- Jumper wires

Connection Summary

- **PIN 4** on Arduino Connected to → One end of 330Ω resistor
- another end of 330Ω resistor Connected to → Anode of Red LED (Long leg – Positive)

- Cathode of Red LED (Short Leg – Negative) Connected to → Arduino GND



Code:

```
int rLEDpin = 4;
int j;
int blink;
int dt = 500;
String msg1 = " How many times to blink LED";

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

void loop() {
  // put your main code here, to run repeatedly:
  Serial.println(msg1);
  while (Serial.available() == 0) {
  }
  blink = Serial.parseInt();

  for (j = 1; j <= blink; j = j + 1) {
    digitalWrite(rLEDpin, HIGH);
    delay(dt);
    digitalWrite(rLEDpin, LOW);
    delay(dt);
  }
}
```

Reading String: `Serial.readString();`

Code:

```
String msg1="What is your name ?";
String msg2="Welcome to Robotics ISland ";
String name;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  Serial.println(msg1);
  while (Serial.available() == 0) {
  }
name=Serial.readString();
Serial.print(msg2);
Serial.println(name);
}
```