**Robotics**

**Robot: - R.O BOT (remote operated bot)**

* **How to check whether the Device is Robot or Not?**

**3 check points**

* **It is programmable**
* **It is movable (at least some part)**
* **It is also do some task by itself**
* **Robot is made of 3 components.**

**1) Electronics:-**

**2) Machines & mechanism:-**

**3) Coding:-**

**\*Electronics\***

* **Current**
* **Voltage**
* **Resistance**
* **Switches**
* **Digital Base / Analog**
* **PWM (Pulse width modulation)**
* **PPM (node packager manager)**
* **Switch**
* **Transistors**
* **Circuits**
* **Inductors**
* **What is current?**

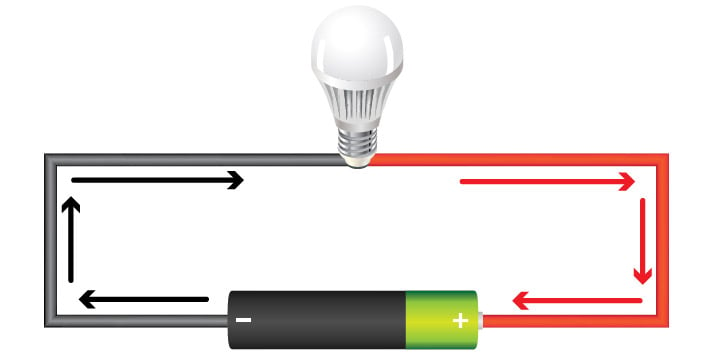
**Current is a flow of electron**

**\*voltage\* (measure in volt)**

**1 volt potential difference**

**1 joule of work is done in transferring 1 coulomb of charge from one point to the other point.**

**1 volt= 1 joule/ 1 coulomb**



1. In this dc circuit, the switch is closed (turned ON).
2. Voltage in the power source—the "potential difference" between the battery's two poles—is activated, creating pressure that forces electrons to flow as current out the battery's negative terminal.
3. Current reaches the light, causing it to glow.
4. Current returns to the power source.

**V = Change in Potential Energy/Charge = ΔU/Q.**

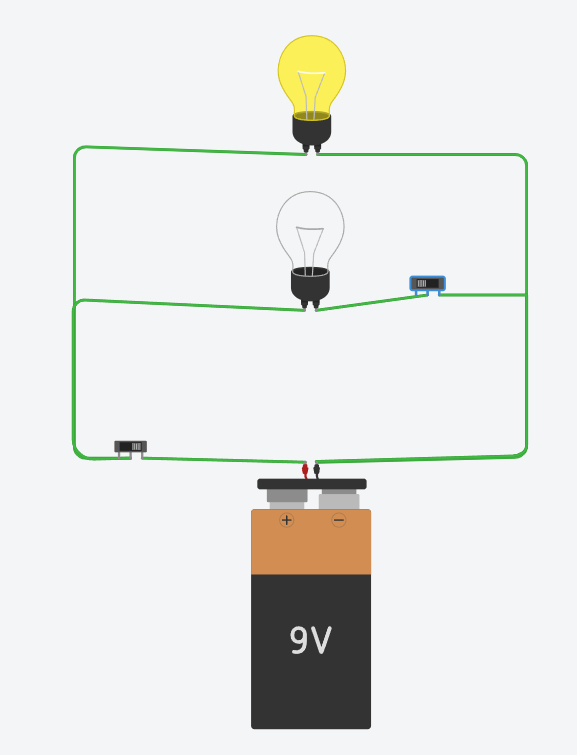
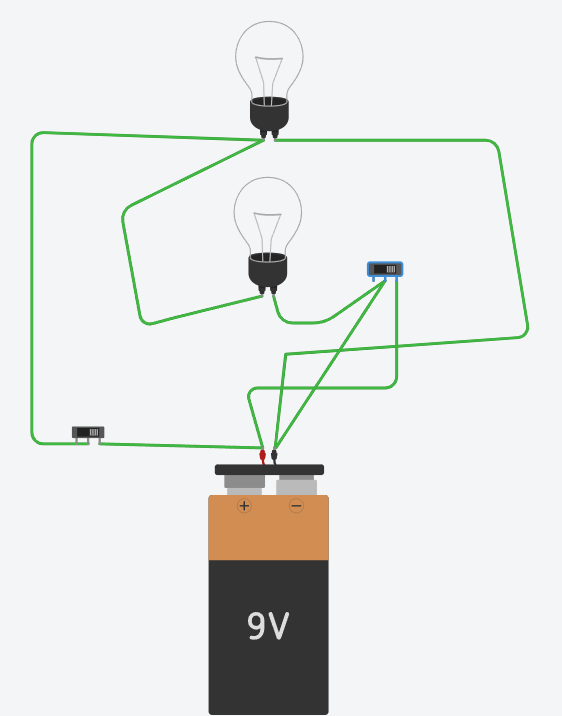
**\*conductor\***

**Element which have free electron**

**\*Current\***

**Electron current is a flow of Charge.**

[**https://www.tinkercad.com/dashboard**](https://www.tinkercad.com/dashboard)

** **

**Truth table**

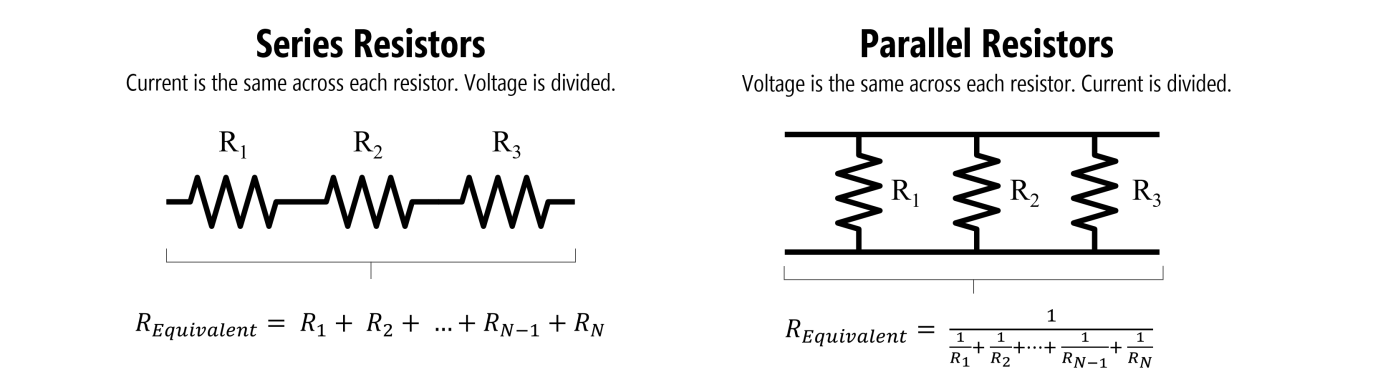
|  |  |  |  |
| --- | --- | --- | --- |
| **S1** | **S2** | **B1** | **B2** |
| **on** | **On** | **off** | **off** |
| **off** | **Off** | **off** | **off** |
| **on** | **Off** | **on** | **on** |
| **off** | **On** | **off** | **off** |

* **Types of learner ?**

**Class**

**Self-study**

* **Series Circuit/parallel circuit**



* **Register in series**

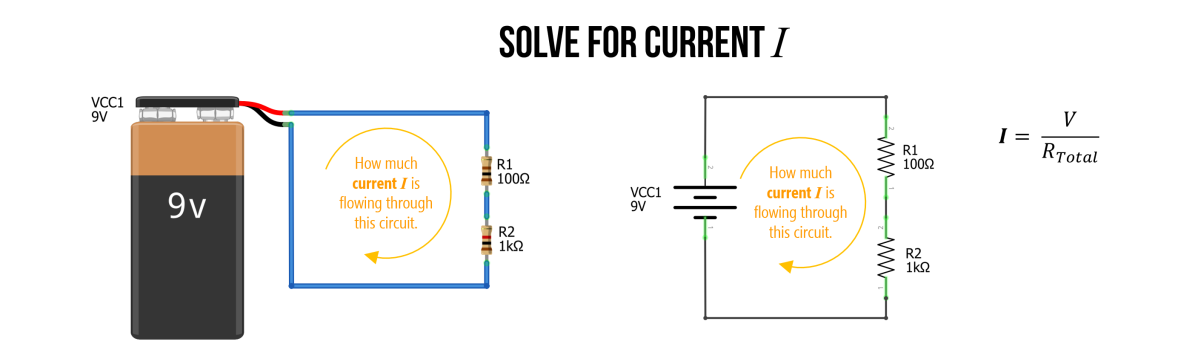


**Register opposes the flow of current**

**To control the flow of current Resister is used**

### Series example 1: Solve for current

Let’s begin with the simplest series resistor circuit: a 9V battery with 100Ω and 1kΩ resistors in series.



**Figure.** A simple circuit with two series resistors (100Ω and 1kΩ) and a 9V battery. How much current II is flowing through this circuit?

#### Step 1: Solve for total resistance

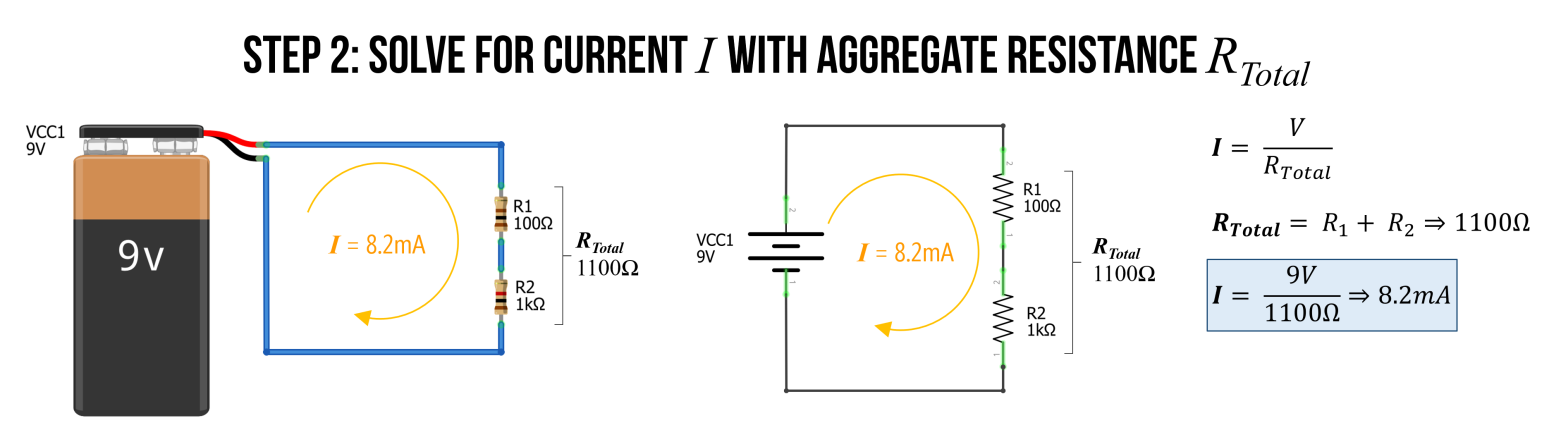
The first step is to solve for the total resistance in our circuit. We know that we sum resistances in series, so: RTotal=R1+R2⇒100Ω+1000Ω⇒1100ΩRTotal=R1+R2⇒100Ω+1000Ω⇒1100Ω. The total resistance is 1100Ω1100Ω.



**Figure.** To find the equivalent resistance of this circuit (let’s call this RTotalRTotal), we can combine series resistors by summing them.

#### Step 2: Solve for current I with equivalent resistance

We can now use this equivalent resistance value RTotalRTotal to solve for the current II by using Ohm’s Law: I=9V/1100Ω⇒0.0082A⇒8.2mAI=9V/1100Ω⇒0.0082A⇒8.2mA



**Figure.** We now solve by current II simply by Ohm’s Law: I=9V/1100Ω⇒8.2mAI=9V/1100Ω⇒8.2mA

That’s it. We did it! The total current is I=8.2mAI=8.2mA.

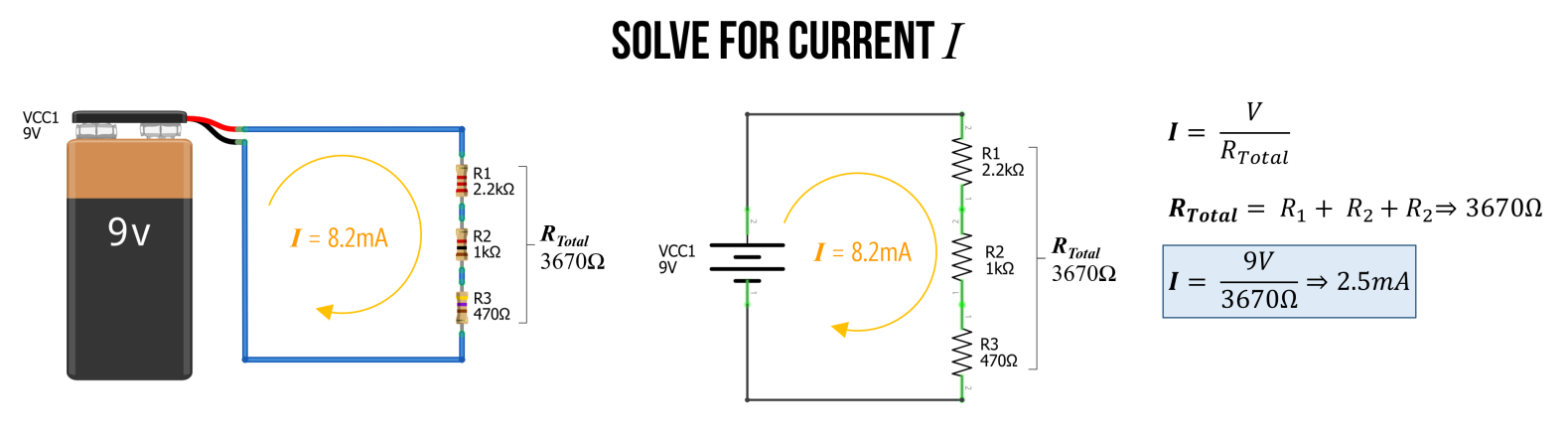
### Series example 2: Solve for current

To reinforce understanding, let’s try again but with three resistors instead of two. This time, R1=2.2kΩR1=2.2kΩ, R2=1kΩR2=1kΩ, and R3=470ΩR3=470Ω.

Again, we start by finding RTotalRTotal, which is:

RTotal=R1+R2+R3RTotal=2200Ω+1000Ω+470ΩRTotal=3670ΩRTotal=R1+R2+R3RTotal=2200Ω+1000Ω+470ΩRTotal=3670Ω

We can then use this equivalent resistance value to solve for current II, which is I=9V3670Ω⇒0.0025A⇒2.5mAI=9V3670Ω⇒0.0025A⇒2.5mA.



**Figure.** In the image above, we solve for current with three series resistors. First, sum the resistances (because they are in series) and then use this aggregate resistance (RTotalRTotal) to determine current with Ohm’s Law: I=VRTotal⇒9V3670Ω$⇒2.5mAI=VRTotal⇒9V3670Ω$⇒2.5mA

#### Check our work in a circuit simulator

We can check our work in our favorite circuit simulator, which is whatever you like. :)

I will use the open-source tool [CircuitJS](https://www.falstad.com/circuit/circuitjs.html). The specific simulation is [here](https://www.falstad.com/circuit/circuitjs.html?ctz=CQAgjCAMB0l3BWcMBMcUHYMGZIA4UA2ATmIxAUgpABZsKBTAWjDACgA3cYlWwm7rzCEqo2lWJQpMBGwDug8CIoo8SquwBOKtWAyEdIFAgNUUaSG20JVRjLxtrsNqWHhXD2Qmse0aaqhoMSwUwHj4BXxp+cHlDYSpfPQMUON9jA3T7IzTbZwc87xyFKP9DaIFUoA).

We can click on the wires to magically show how much current is traveling through them or to show their electric potential (voltage) with respect to ground. And sure enough, you’ll see that indeed 2.5mA2.5mA is traveling through the circuit. What else do you observe?

Well, remember how we’ve been emphasizing that voltages are split or divided across resistors in series. You can clearly see this as well! The voltage is at 9V9V at the top node but drops by 5.4V5.4V over the 2.2kΩ2.2kΩ resistor to 3.6V3.6V, which then drops by 2.4V2.4V over the 1kΩ1kΩ resistor leaving just 1.2V1.2V of electric potential before finally dropping down to 0V0V or GNDGND across the 470Ω470Ω resistor. We’ll talk more about this next!

**Figure.** This video shows a [CircuitJS](https://www.falstad.com/circuit/circuitjs.html) simulation of a basic three resistor series circuit. You can play with the circuit [here](https://www.falstad.com/circuit/circuitjs.html?ctz=CQAgjCAMB0l3BWcMBMcUHYMGZIA4UA2ATmIxAUgpABZsKBTAWjDACgA3cYlWwm7rzCEqo2lWJQpMBGwDug8CIoo8SquwBOKtWAyEdIFAgNUUaSG20JVRjLxtrsNqWHhXD2Qmse0aaqhoMSwUwHj4BXxp+cHlDYSpfPQMUON9jA3T7IzTbZwc87xyFKP9DaIFUoA).

## Voltage dividers

The notion that **series resistors** split voltages is a critical concept when working with microcontrollers. So, it deserves its own emphasis.

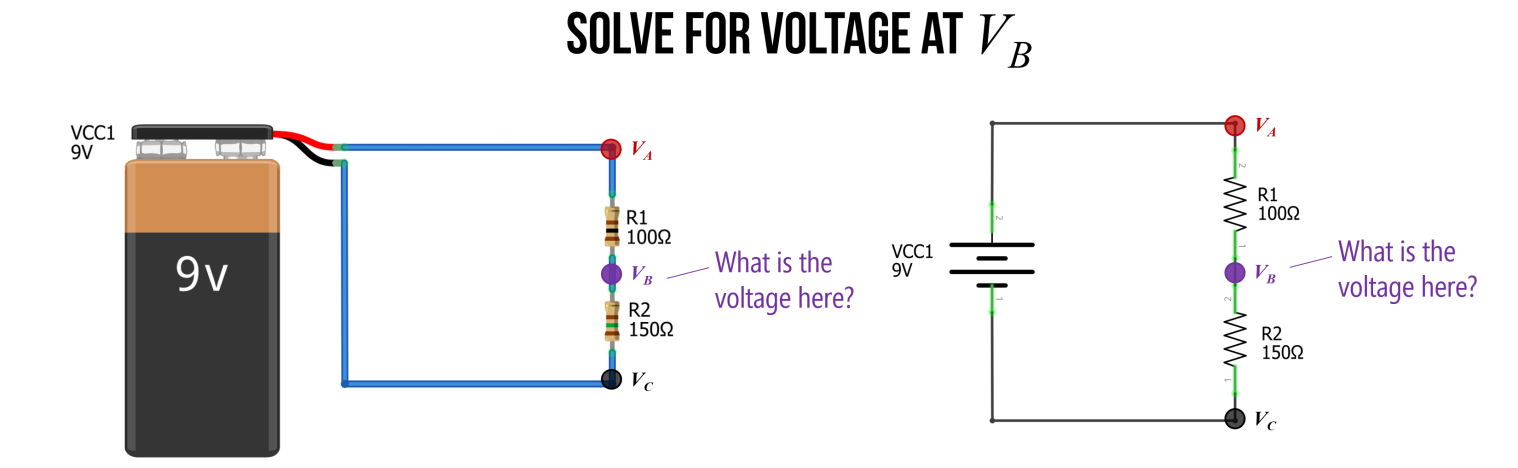
The key thing to remember: there is a voltage drop across each resistor (this is always the case, not just in a series circuit configuration). Thus, between each resistor we have a different electric potential or voltage. And because microcontrollers “read” voltage rather than current, we can use this property to control dynamic input into our microcontroller!

Let’s go over some examples.

### Example 1: Solve for voltage at VB

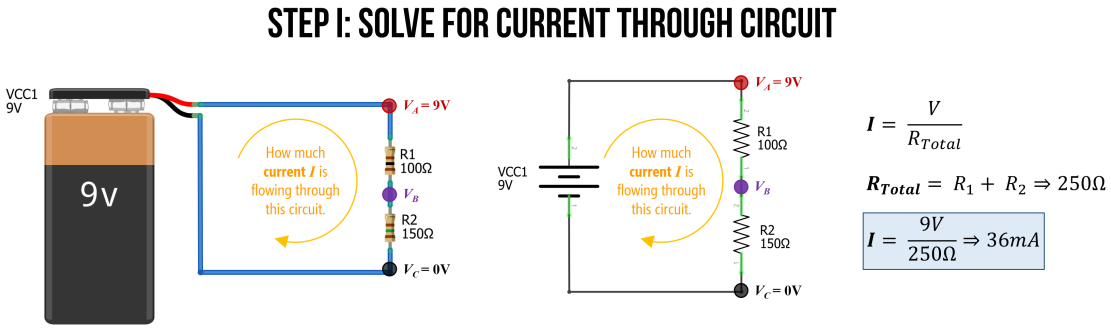
With this idea of voltages dropping across each resistor, let’s look at how to calculate the voltage at the node VBVB with respect to ground (and remember, a node is just any junction point with two or more connections in a circuit).

Before moving through our example, stop and ask yourself: how would you calculate the voltage at VBVB?



#### Step 1: Solve for the current through the circuit

As before, the first step is to solve for the current through the circuit. We do this, again, by finding the equivalent resistance RTotalRTotal and using Ohm’s Law. So, I=VRTotal⇒9V250Ω⇒36mAI=VRTotal⇒9V250Ω⇒36mA.



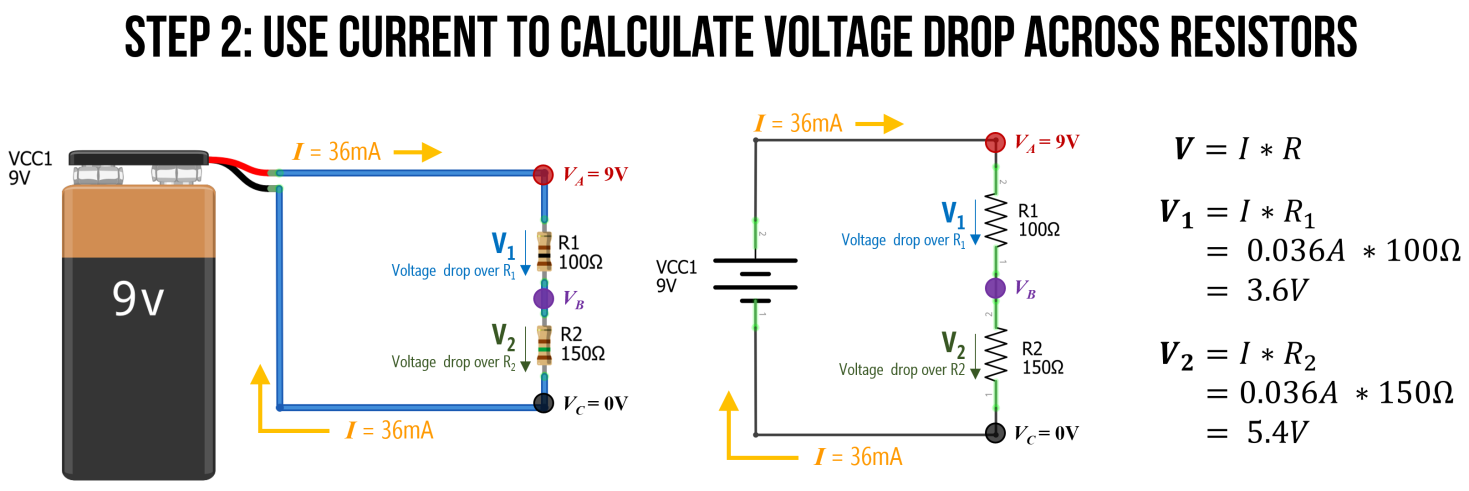
#### Step 2: Calculate voltage drop across resistors

Now that we know the total current flowing through our circuit (36mA36mA), we can use this to calculate the specific voltage drop across each resistor. Let’s call the voltage drop over R1R1: V1V1 and the voltage drop over R2R2: V2V2. And because we are interested in calculating voltage, we will use this formulation of Ohm’s Law: V=I∗RV=I∗R.

Thus:

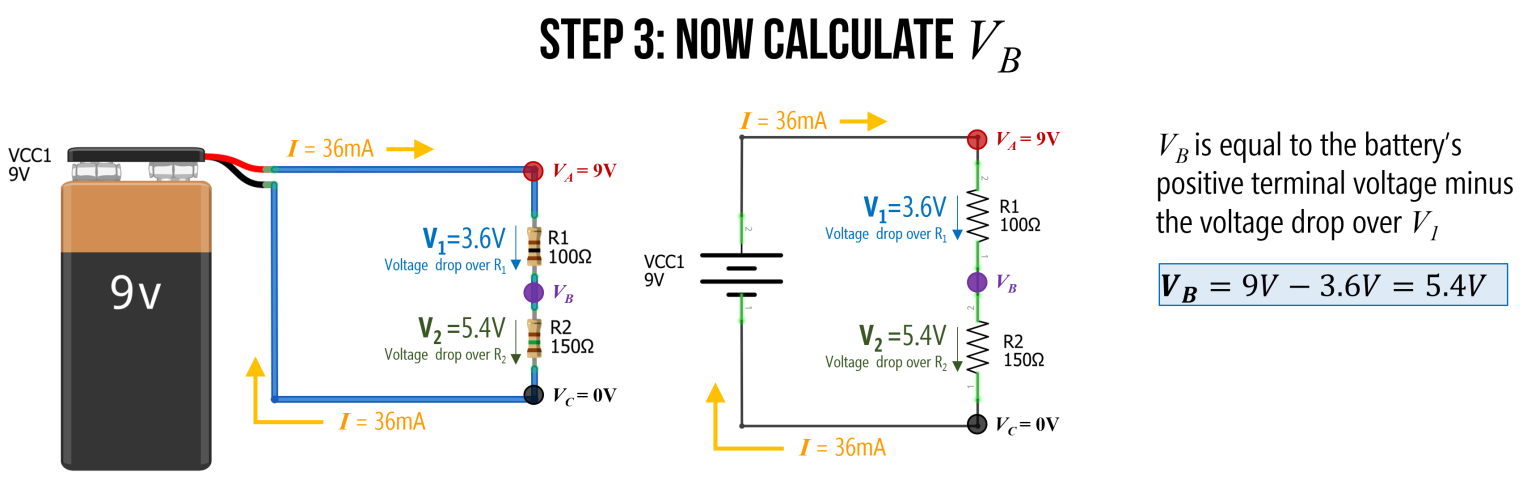
V1=I∗R1⇒0.0036A∗100Ω⇒3.6VV2=I∗R2⇒0.0036A∗150Ω⇒5.4VV1=I∗R1⇒0.0036A∗100Ω⇒3.6VV2=I∗R2⇒0.0036A∗150Ω⇒5.4V

And, just as a quick check on our work (and without going into too much detail), we know from [Kirchoff’s circuit laws](https://www.khanacademy.org/science/physics/circuits-topic/circuits-resistance/a/ee-kirchhoffs-laws), that VTotal=V1+V2⇒9V=3.6V+5.4V⇒9V=9VVTotal=V1+V2⇒9V=3.6V+5.4V⇒9V=9V. So, things are looking good so far!



#### Step 3: Now calculate VB

Now it is trivial to calculate VBVB. We know that VA=9VVA=9V and that R1R1 causes a 3.6V3.6V voltage drop. So, VBVB must be equal to 9V−3.6V9V−3.6V, which is 5.4V.



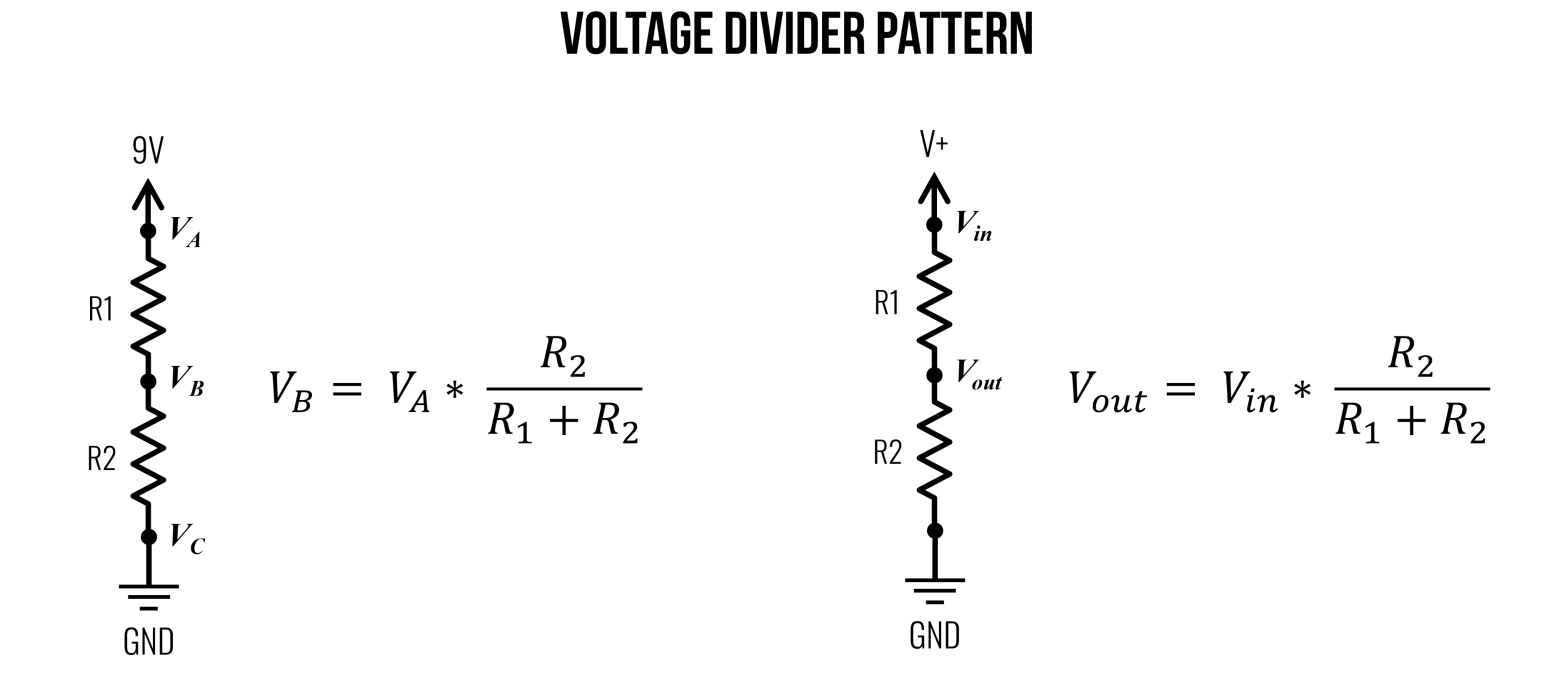
### The voltage divider pattern

We call a two-resistor configuration like this a **voltage divider** precisely because, as you can see, it divides the voltages. In this case, we used 100Ω100Ω and 150Ω150Ω resistors in series to output a 5.4V5.4V at VBVB.

Using Ohm’s Law, we can [derive the voltage divider equation](https://www.khanacademy.org/science/electrical-engineering/ee-circuit-analysis-topic/ee-resistor-circuits/a/ee-voltage-divider) for VBVB in terms of the input voltage (VAVA) into our voltage divider network and the two resistors: the top resistor R1R1 and the bottom resistor R2R2.

This voltage divider equation is thus: VB=VA∗R2R1+R2VB=VA∗R2R1+R2

Or more commonly written as: Vout=Vin∗R2R1+R2Vout=Vin∗R2R1+R2



**Figure.** The voltage divider pattern and equation. Image made in PowerPoint. See [Khan Academy](https://www.khanacademy.org/science/electrical-engineering/ee-circuit-analysis-topic/ee-resistor-circuits/a/ee-voltage-divider) for more.

Importantly, as you can tell from the equation, it is not the absolute resistances that matter but rather \*the ratio \*\*of R1R1 to R2R2 that controls VoutVout. Thus, for the purposes of \*dividing voltages, setting R1=100ΩR1=100Ω and R2=100ΩR2=100Ω would be the same as R1=2.2kΩR1=2.2kΩ and R2=2.2kΩR2=2.2kΩ, they would both divide the voltages evenly. So, VoutVout would equal 4.5V4.5V if Vin=9VVin=9V.

However, the amount of current between the two circuits would be significantly different with the former: I=9V200Ω⇒45mAI=9V200Ω⇒45mA and the latter: I=9V4.4kΩ⇒2.0mAI=9V4.4kΩ⇒2.0mA.

Wouldn’t it be cool to dynamically control one of those resistor values to output a variable voltage at VoutVout? Yes! And this is the basis of a [potentiometer](https://makeabilitylab.github.io/physcomp/electronics/variable-resistors.html), which we will learn about in a later lesson.

#### Deriving the voltage divider equation

Given what you are learning about circuits, you now have the knowledge to derive the voltage divider equation or, at the very least, understand how it is derived. Let’s take a look!

 **Figure.** A derivation of the voltage divider equation. See [Khan Academy](https://www.khanacademy.org/science/electrical-engineering/ee-circuit-analysis-topic/ee-resistor-circuits/a/ee-voltage-divider) for more.

Using the figure above, let’s identify and write down what we know. We know that the voltage drop over R2R2 is equal to VoutVout (indeed, they are the same thing) and that VR2=I∗R2VR2=I∗R2:

Vout=VR2=I∗R2Vout=VR2=I∗R2

We also know that VinVin is equal to VR1+VR2VR1+VR2 given [Kirchhoff’s Voltage Law](https://www.khanacademy.org/science/physics/circuits-topic/circuits-resistance/a/ee-kirchhoffs-laws).

Vin=VR1+VR2Vin=VR1+VR2

Using Ohm’s Law, we can substitute I∗R1I∗R1 for VR1VR1 and I∗R2I∗R2 for VR2VR2.

Vin=I∗R1+I∗R2Vin=I∗R1+I∗R2

Now, rearrange the VinVin equation using algebra:

Vin=I∗(R1+R2)⇒I=Vin(R1+R2)Vin=I∗(R1+R2)⇒I=Vin(R1+R2)

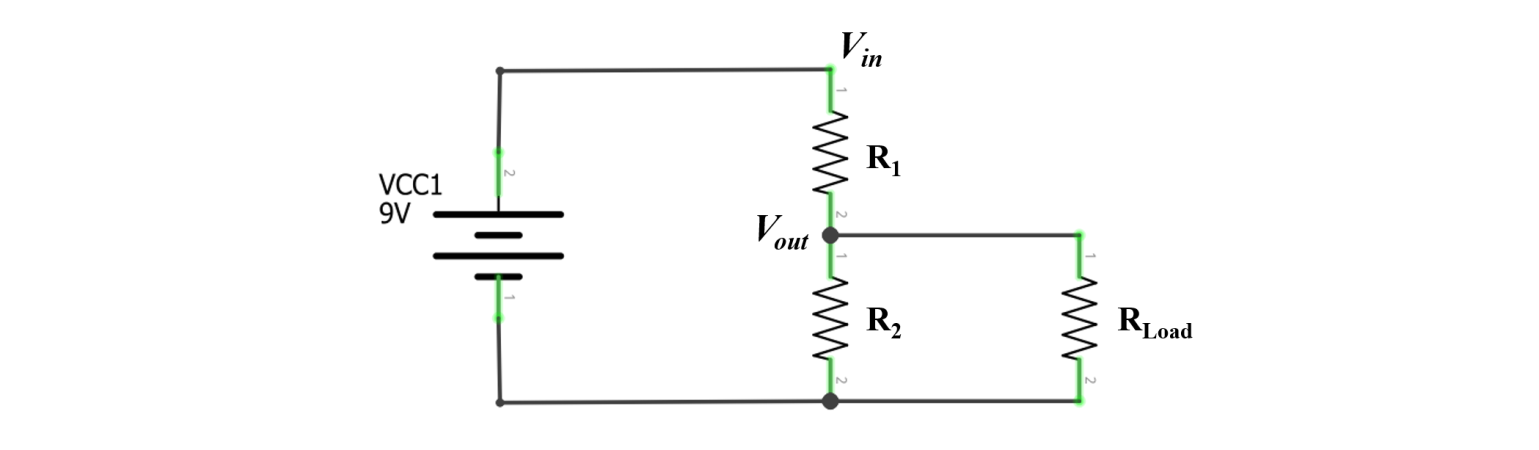
Returning to Vout=I∗R2Vout=I∗R2, we can substitute II given the formulation above:

Vout=I∗R2=Vin(R1+R2)∗R2Vout=I∗R2=Vin(R1+R2)∗R2

Finally, rearrange the above to achieve the popular voltage divider equation:

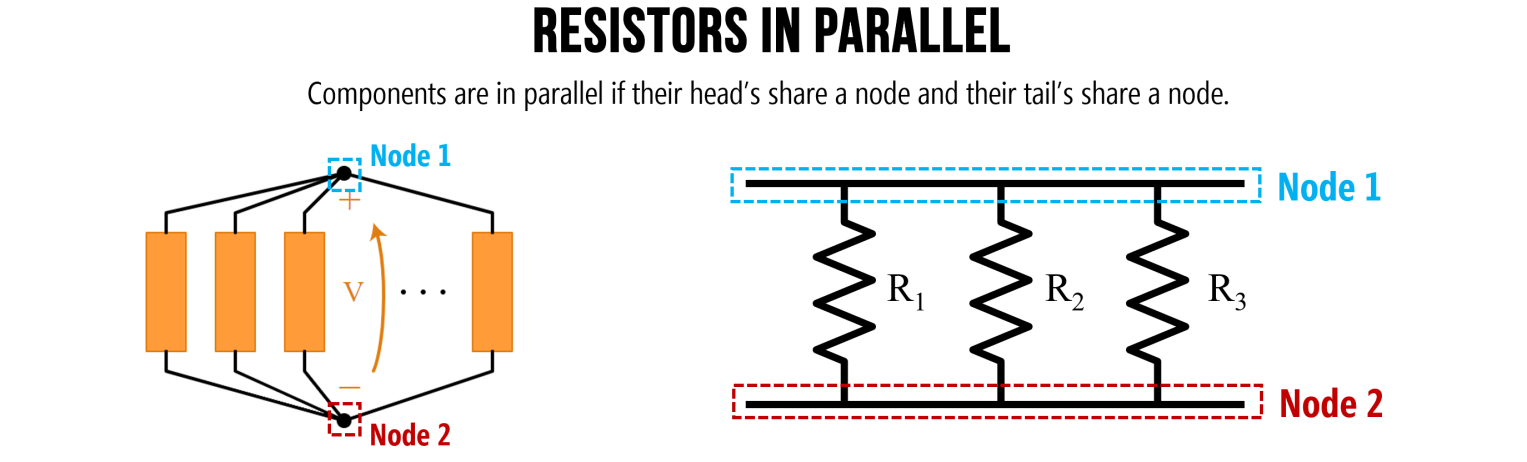
Vout=Vin∗R2(R1+R2)Vout=Vin∗R2(R1+R2)

Note: for this voltage divider equation to hold true, the current II flowing through R1R1 must be (largely) equal to R2R2. That is, if we hook up a branch to VoutVout, as we’ve done below, then this branch must have very high resistance so that very little current “leaks” out into that branch. That is, RLoadRLoad must be magnitudes greater than R1+R2R1+R2. However, in the case of microcontroller inputs, this is fortunately the case, which we will return to later.

 **Figure.** The voltage divider equation only holds when RLoadRLoad is large, which it will be when we start using microcontrollers (which read changes in voltage levels and have “high input impedance”)

## Parallel resistors

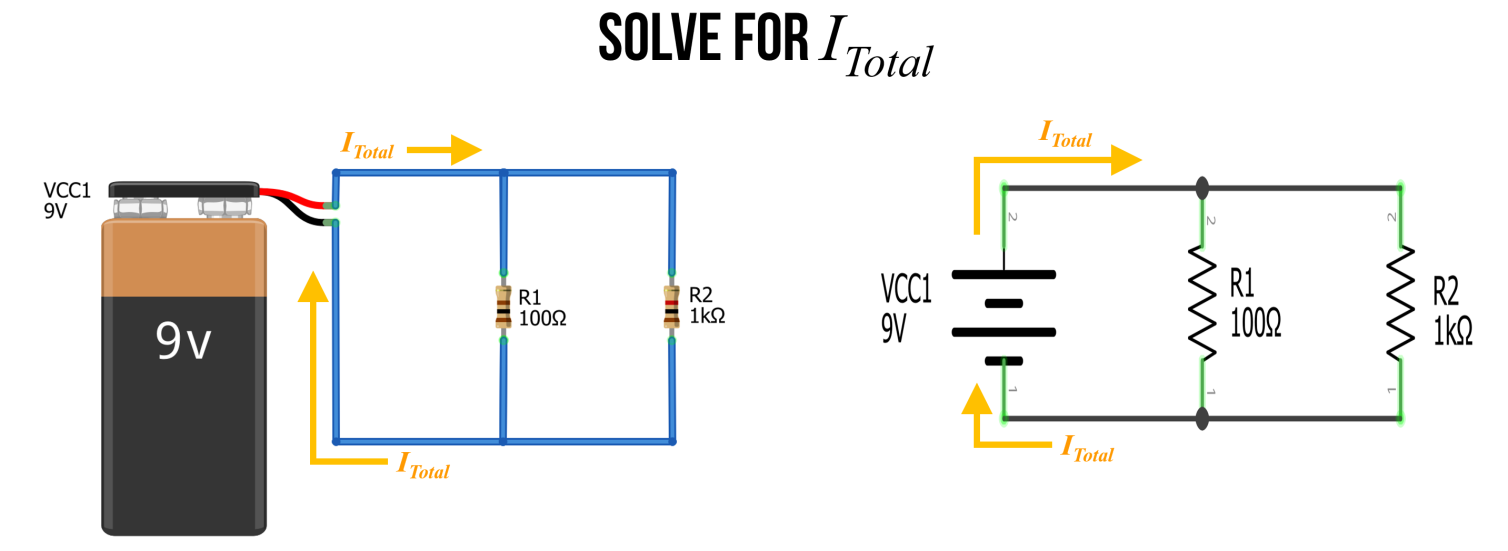
Whereas **series resistors** have the same current but divide voltage, [**parallel resistors**](https://www.khanacademy.org/science/electrical-engineering/ee-circuit-analysis-topic/ee-resistor-circuits/a/ee-parallel-resistors) have the same voltage but divide current. Components in parallel look like this:



**Figure.** Components are in parallel if their head’s share a node and their tail’s share a node. Image on left from [Khan Academy](https://www.khanacademy.org/science/electrical-engineering/ee-circuit-analysis-topic/ee-resistor-circuits/a/ee-parallel-resistors). Image made in PowerPoint.

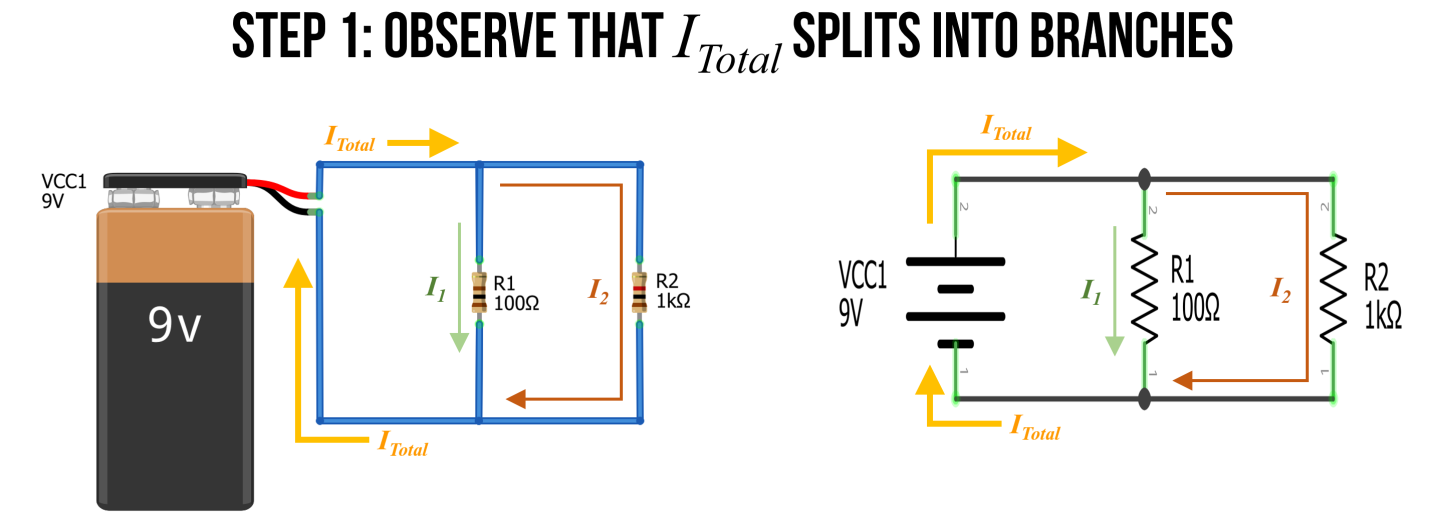
### Parallel example 1: Solve for ITotalITotal

In the circuit below, we have two parallel resistors R1=100ΩR1=100Ω and R2=1kΩR2=1kΩ. Let’s solve for the total current ITotalITotal in the circuit.



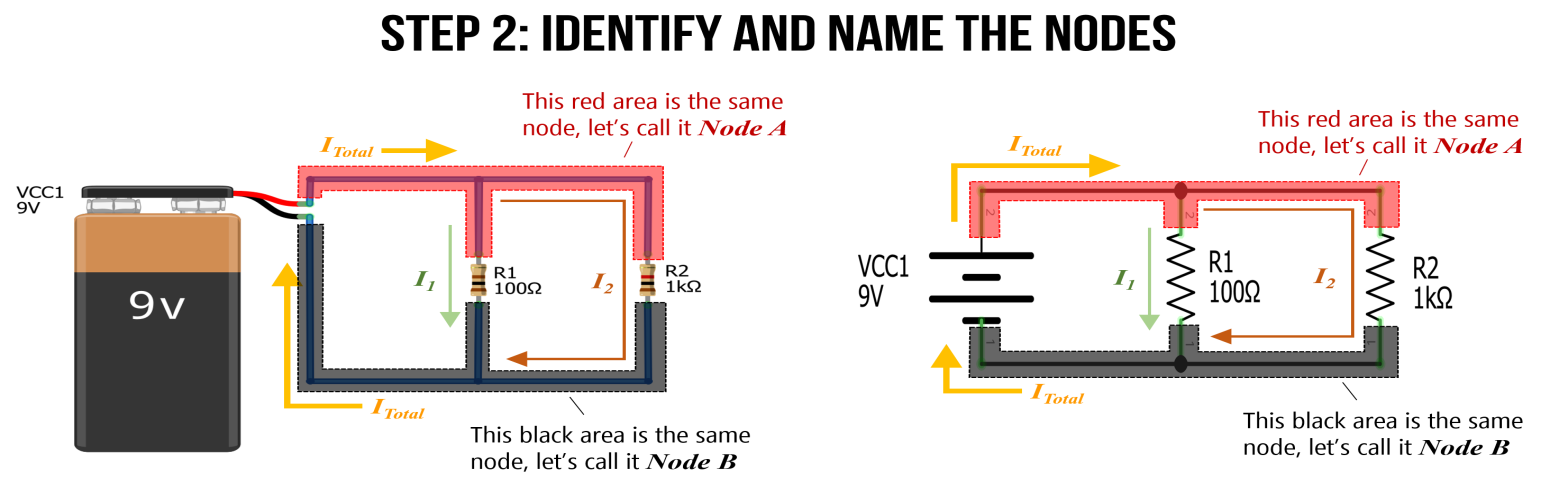
#### Step 1: Observe that ITotalITotal splits into branches

The first thing to recognize is that ITotalITotal splits into two branches. Let’s call the current down those two branches I1I1 and I2I2. From Kirchoff’s Laws, we know that ITotal=I1+I2ITotal=I1+I2. This is due to the conservation of energy—no charges are lost in our circuit (they simply flow around and around).



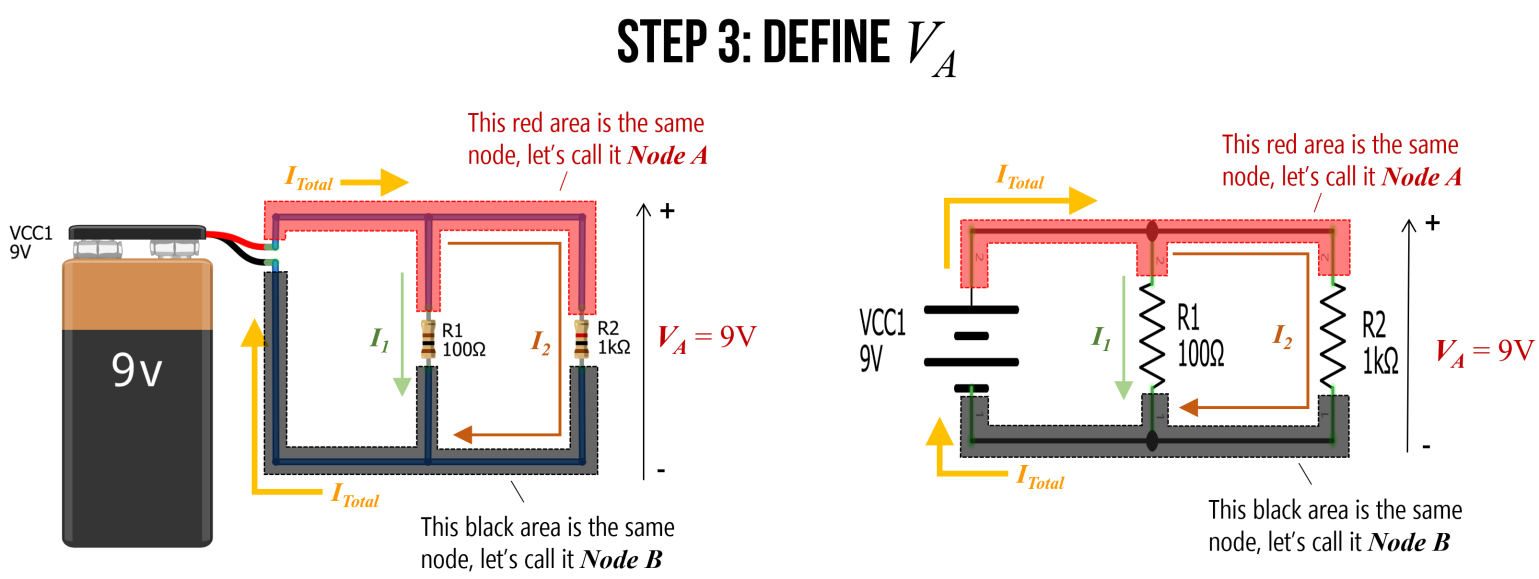
#### Step 2: Identify and name nodes

Also recognize that there are only two nodes in our circuit. We can label them NodeANodeA and NodeBNodeB.



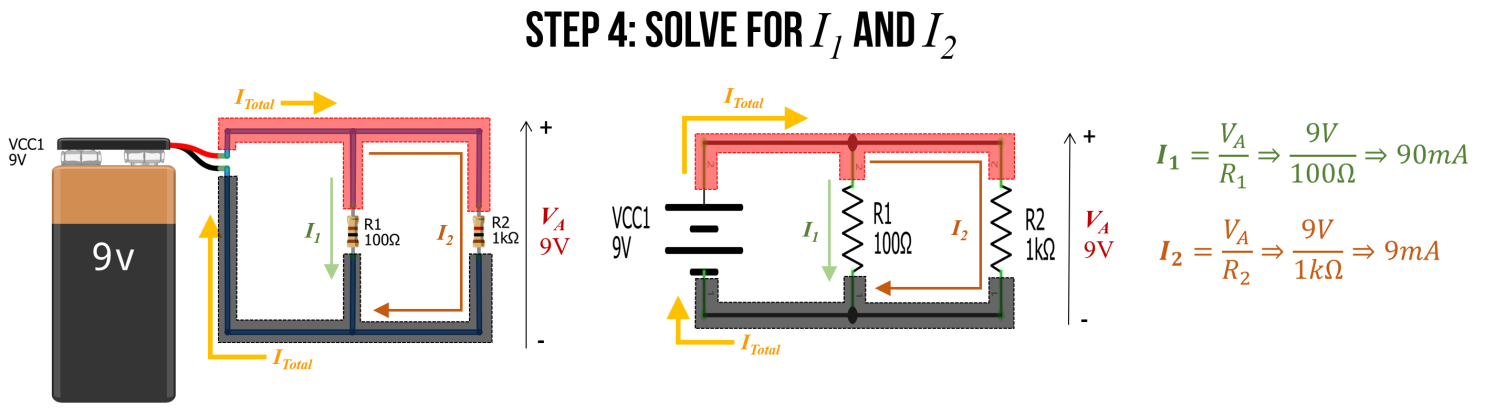
#### Step 3: Define VAVA

Because NodeANodeA is directly connected to the positive terminal of the battery, it has an electric potential of 9V. Let’s call this VA=9VVA=9V. Similarly, NodeBNodeB is directly connected to the negative terminal of the battery, so let’s refer to this as GNDGND or OVOV.



#### Step 4: Solve for I1I1 and I2I2

Using Ohm’s Law, we can now solve for I1I1 and I2I2 where: I1=VAR1I1=VAR1 and I2=VAR2I2=VAR2. Thus, I1=9V100Ω⇒90mAI1=9V100Ω⇒90mA and I2=9V1000Ω⇒9mAI2=9V1000Ω⇒9mA.

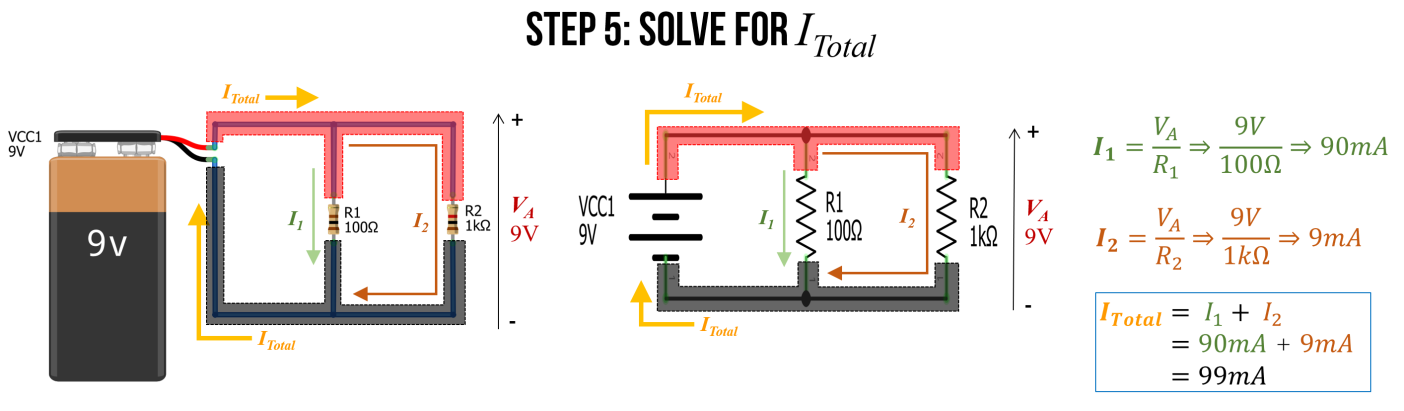


Stop for a moment. Think about these results. Do they conceptually make sense?

Using Ohm’s Law, we found that **10 times** as much current flows through the I1I1 branch as the I2I2 branch. Indeed, this exactly matches the ratio of the two resistors R1 and R2: R1 is 10 times smaller than R2 and thus, more current will flow through I1I1 branch (10x more!). This makes sense: just as more water will flow through a branch with less resistance, so too will more current flow through the path of less resistance.

#### Step 5: Finally, solve for ITotalITotal

Finally, we can use ITotal=I1+I2ITotal=I1+I2 to solve for ITotalITotal. In this case, ITotal=90mA+9mA⇒99mAITotal=90mA+9mA⇒99mA.



#### Step 6: Use equivalent resistance to check our work

Remember how we introduced an equation for equivalent resistance in parallel resistor circuits? The equation is:

Requivalent=11R1+1R2+...+1RN−1+1RNRequivalent=11R1+1R2+...+1RN−1+1RN

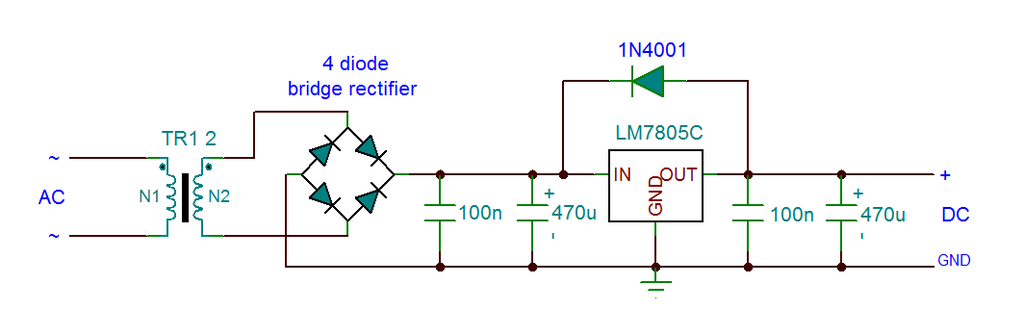
As an aside, if you’re curious about its derivation, see this [Khan Academy lesson](https://www.khanacademy.org/science/electrical-engineering/ee-circuit-analysis-topic/ee-resistor-circuits/a/ee-parallel-resistors)—but, in short, you can derive it from Ohm’s Law (and the steps that we followed above).

We can use this equation to more quickly solve for ITotalITotal, which is ITotal=VARequivalentITotal=VARequivalent.

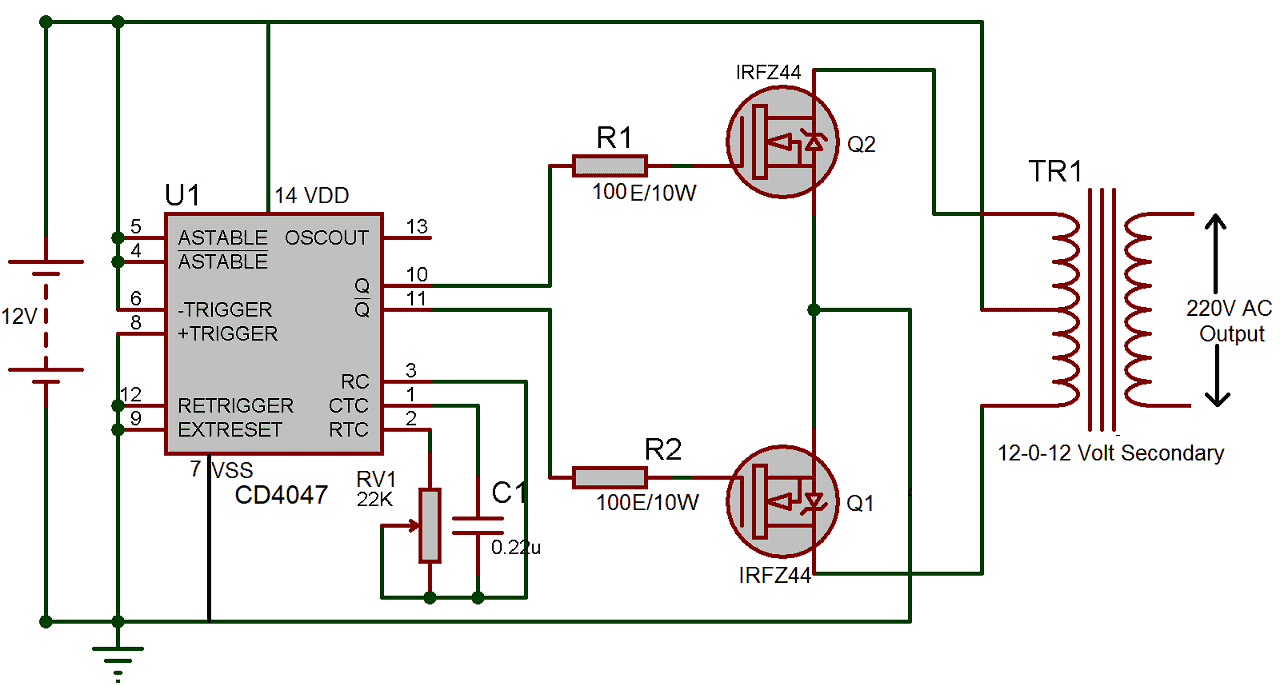
We know that Requivalent=11100Ω+11kΩ⇒90.9ΩRequivalent=11100Ω+11kΩ⇒90.9Ω

Thus, ITotal=9V90.91Ω⇒99mAITotal=9V90.91Ω⇒99mA.

**AC to DC converter**



**DC to AC converter**



## What is a Bridge Rectifier?

**Bridge Rectifiers** are circuits that convert [alternating current](https://www.electrical4u.com/alternating-current/) (AC) into [direct current](https://www.electrical4u.com/dc-current/) (DC) using diodes arranged in the bridge circuit configuration. Bridge rectifiers typically comprise of four or more diodes. The output wave generated is of the same polarity irrespective of the polarity at the input.

Bridge rectifiers are similar to [half-wave rectifiers](https://www.electrical4u.com/half-wave-rectifiers/) and [full-wave rectifiers](https://www.electrical4u.com/full-wave-rectifiers/). They use four [diodes](https://www.electrical4u.com/diode-working-principle-and-types-of-diode/) (D1, D2, D3, and D4). The input is supplied across terminals A and B, and the output is collected across a load [resistor](https://www.electrical4u.com/types-of-resistor/) (RL) connected between terminals C and D.

When a positive pulse appears at the AC input (terminal A is positive, terminal B is negative), diodes D1 and D3 become forward biased, while diodes D2 and D4 are reverse biased.

As a result, the [current](https://www.electrical4u.com/electric-current-and-theory-of-electricity/) flows along the short-circuited path created by the diodes D1 and D3 (considering the diodes to be ideal), as shown by Figure 2a. Thus the [voltage](https://www.electrical4u.com/voltage-or-electric-potential-difference/) developed across the load resistor RL will be positive towards the end connected to terminal D and negative at the end connected to the terminal C.

When a negative pulse appears at the AC input (terminal A is negative, terminal B is positive), diodes D2 and D4 become forward biased, while diodes D1 and D3 are reverse biased.

It’s important to note that the voltage across RL remains the same polarity regardless of whether the AC input pulse is positive or negative. Thus, the output of the bridge rectifier always has the same polarity, as shown by the waveforms in Figure 3.

However, it is to be noted that the bridge rectifier’s DC will be pulsating in nature. In order to obtain a pure form of DC, one has to use a [capacitor](https://www.electrical4u.com/what-is-capacitor/) in conjunction with the bridge circuit (Figure 4).  
  
In this design, the positive pulse at the input causes the capacitor to charge through the diodes D1 and D3. However as the negative pulse arrives at the input, the charging action of the capacitor ceases and it starts to discharge via RL.

This results in the generation of DC output which will have ripples in it as shown in the figure. This ripple factor is defined as the ratio of the AC component to the DC component in the output voltage. In addition, the mathematical expression for the ripple voltage is given by the equation  
https://www.electrical4u.com/images/march16/1464440425.GIF?ezimgfmt=rs:79x53/rscb38/ng:webp/ngcb38  
Where,  
Vr represents the ripple voltage.  
Il represents the load current.  
f represents the frequency of the ripple which will be twice the input frequency.  
C is the [Capacitance](https://www.electrical4u.com/what-is-capacitor/).

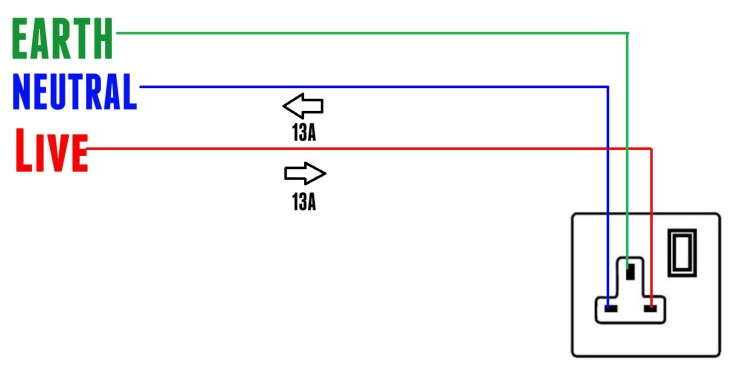
Further, the bridge rectifiers are primarily of two types, viz., Single-Phase Rectifiers and Three-Phase Rectifiers. In addition, each of these can be either Uncontrolled or Half-Controlled, or Full-Controlled.

**Bridge rectifiers** for a particular application are selected by considering the load current requirements. These bridge rectifiers are quite advantageous as they can be constructed with or without a [transformer](https://www.electrical4u.com/what-is-transformer-definition-working-principle-of-transformer/) and are suitable for high voltage applications.

However, here two diodes will be conducting for every half-cycle and thus the [voltage drop](https://www.electrical4u.com/voltage-drop-calculation/) across the diodes will be higher. Lastly one has to note that apart from converting AC to DC, bridge rectifiers are also used to detect the amplitude of modulated radio signals and to supply polarized voltage for welding applications.

* **Difference between inverter and converter ?**

**A converter can turn AC power into DC. It can change the voltage level from one level to another, such as from 110v to 12v. On the other hand, an inverter transforms DC power into AC power.**



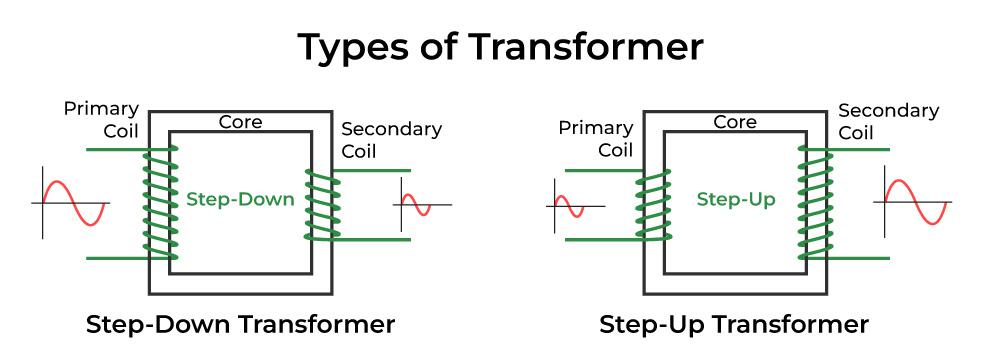
**According to international convention,**

## The colour of the live wire in a cable is brown.

## The colour of the neutral wire in a cable is light blue.

## The colour of the earth wire in a cable is green.

* **Transformer, device that transfers electric energy from one alternating-current circuit to one or more other circuits, either increasing (stepping up) or reducing (stepping down) the voltage.**

****

**A step-up transformer is used in power transmission to convert low voltage AC into high voltage AC, to prevent power loss during transmission. On the other hand,**

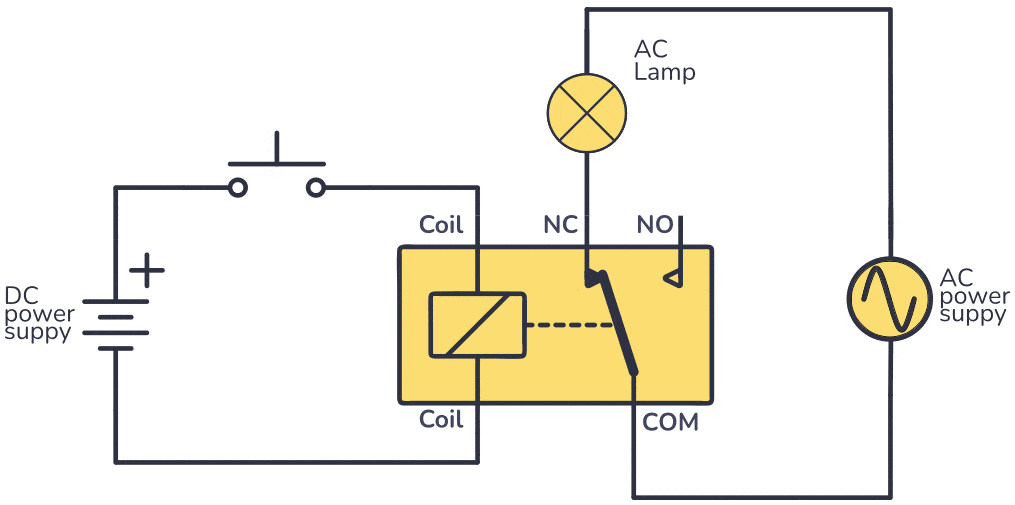
**a step-down transformer is used to convert high voltage AC into low voltage AC for the successful operation of low voltage devices.**

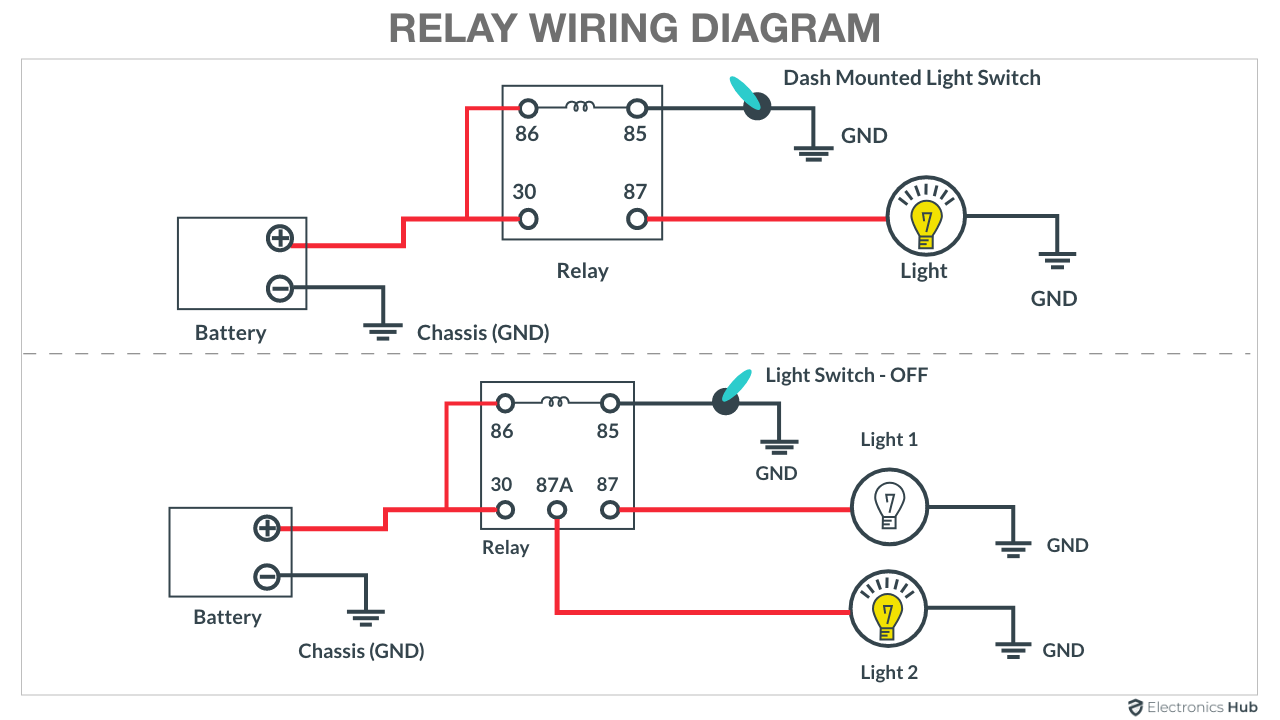
* **Relay**

**Relays are the switches that aim at closing and opening the circuits electronically as well as electromechanically. It controls the opening and closing of the circuit contacts of an electronic circuit. When the relay contact is open (NO), the relay isn't energized with the open contact.**

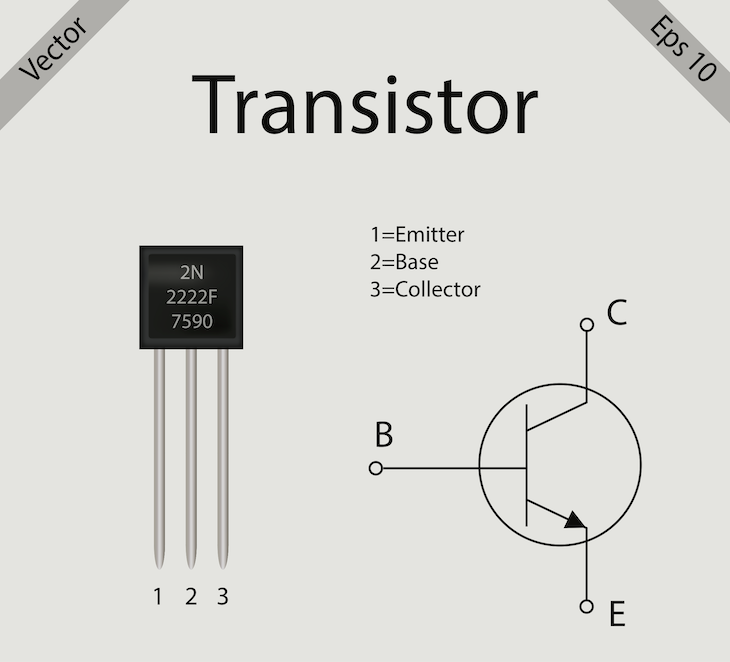
**High voltage switching Device**

**Relay is widely used in Robotics**

****

****

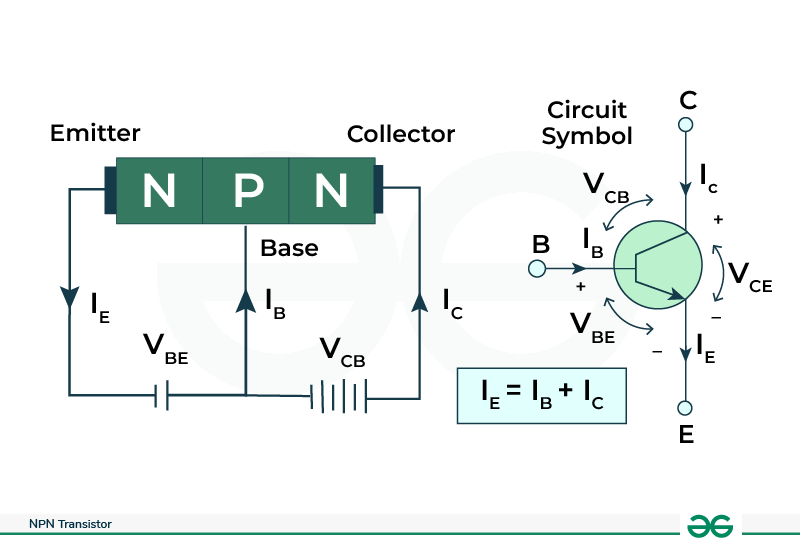
* **Transistor**

****

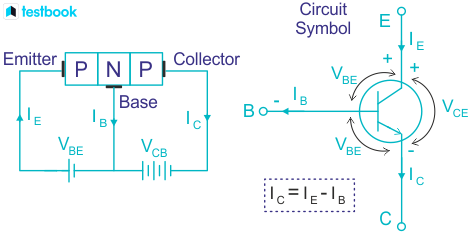
**What is a transistor?**

A transistor is a type of [semiconductor](https://byjus.com/jee/semiconductors/) device that can be used to conduct and insulate electric current or voltage. A transistor basically acts as a switch and an amplifier. In simple words, we can say that a transistor is a miniature device that is used to control or regulate the flow of electronic signals.

**What is a PNP and NPN transistor?**

****

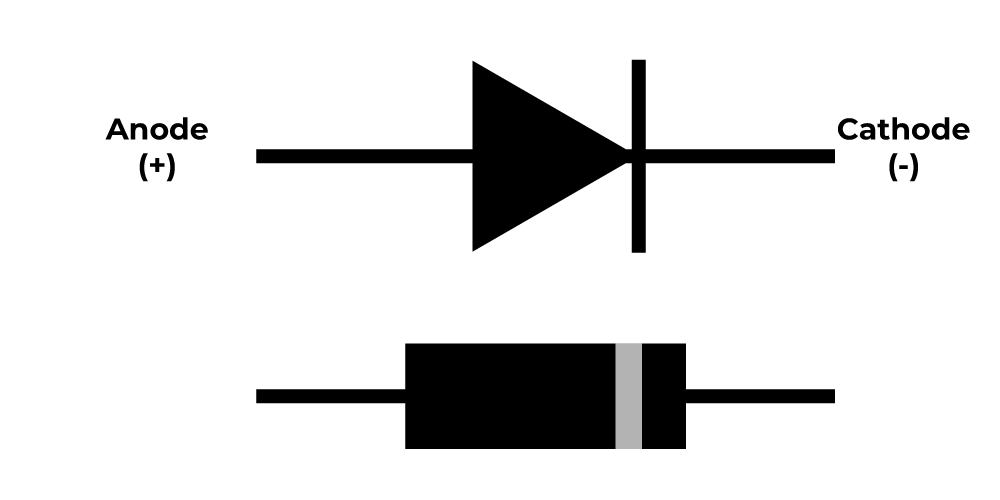
**Bipolar Junction Transistors are further divided into NPN and PNP transistors. The NPN transistor consists of two n-type semiconductor materials separated by a thin layer of p-type. In contrast, the PNP transistor consists of two p-type semiconductors separated by a thin layer of n-type.**

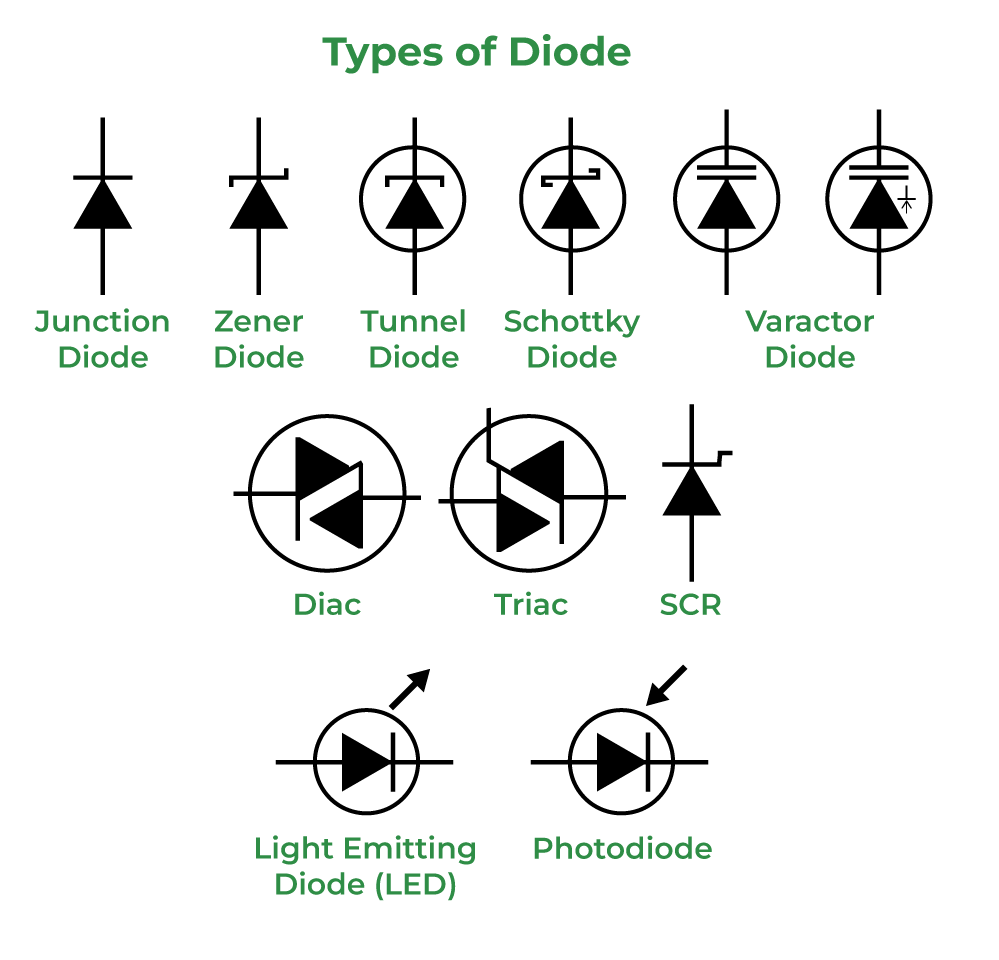
****

* **Diode**

**What is a diode and its function?**

**A diode is a semiconducting electrical device with two terminals that ideally allows current to flow easily through in one direction but restricts the flow of current in the opposite direction. A diode can be thought of as a switch that controls the direction of electrons through the electrical circuit.**

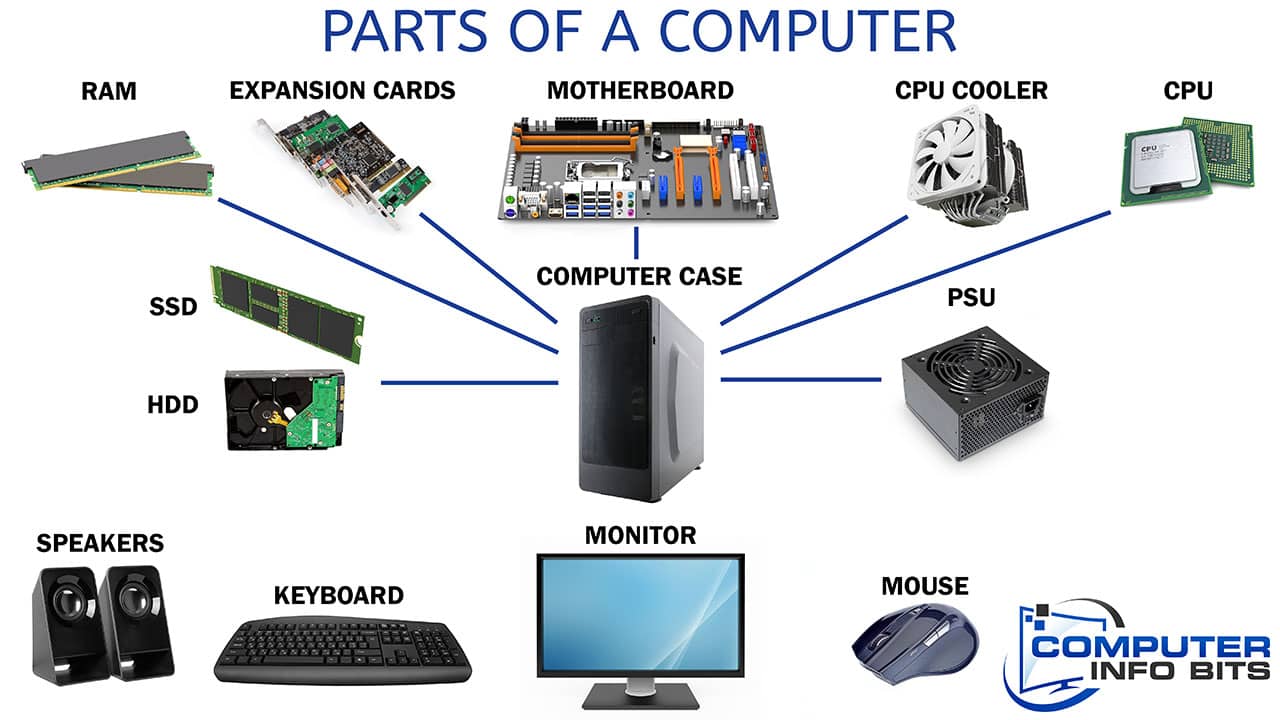
****

****

* **arduino uno r3 datasheet**

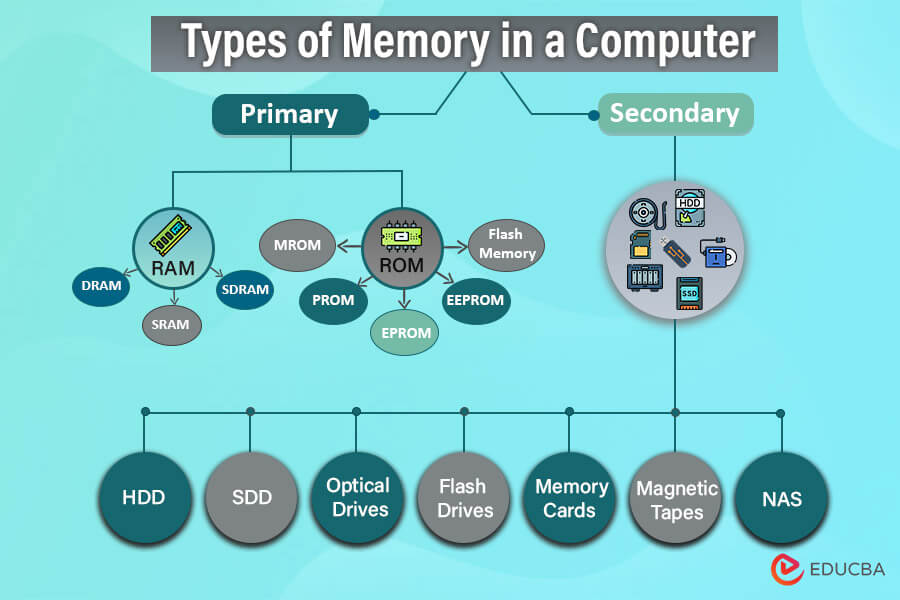
[**https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf**](https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf)

* **CPU**

****

* **Types of memory in computer**

**random access memory (RAM), read-only memory (ROM), cache memory, virtual memory, and various types of secondary storage devices like hard disk drives (HDDs) and solid state drives (SSDs).**

****

* **SMPS**

**Stand-alone switched-mode power supply An adjustable switched-mode power supply for laboratory use. Like other power supplies, an SMPS transfers power from a DC or AC source (often mains power, see AC adapter) to DC loads, such as a personal computer, while converting voltage and current characteristics.**

* **What is meant by CMOS memory?**

**The full form of CMOS is Complementary Metal-Oxide-Semiconductor. CMOS is an integrated circuit built on a printed circuit board. It is a battery-powered memory chip that effortlessly holds the initialisation data. The BIOS uses this data to turn on the device, i.e., during the bootup process.**

* **What is firmware and BIOS?**

**Firmware is a special class of software, so called because it is more or less permanently stored on chips. Firmware is often referred to generically as a BIOS (Basic Input/Output System) because the only firmware contained in early PCs was the main system ROM-BIOS (Read-Only Memory BIOS).**

* **What is POST on a PC?**

**POST stands for Power-On Self-Test. It's a set of diagnostic tests that a computer runs every time it's powered on to make sure all of its components are working properly.**

* **What is boot Event on a PC?**

**To boot (to boot up, to start up or booting) a computer is to load an operating system (OS) into the computer's main memory or RAM. Once the OS is loaded (for example, on a PC, you will see the initial Windows or Mac desktop screen), it's ready for users to run applications.**

* **What is a microcontroller used for?**

**A microcontroller is a compact integrated circuit designed to govern a specific operation in an embedded system. A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip.**

**Microcontroller is a compressed micro computer manufactured to control the functions of embedded systems in office machines, robots, home appliances, motor vehicles, and a number of other gadgets. A microcontroller is comprises components like – memory, peripherals and most importantly a processor.**

* **What is memory controller?**

**A memory controller, also known as memory chip controller (MCC) or a memory controller unit (MCU), is a digital circuit that manages the flow of data going to and from a computer's main memory.**

## ****Difference between Microprocessor and Microcontroller****

|  |  |
| --- | --- |
| **Microprocessor** | **Microcontroller** |
| Since memory and I/O are connected externally, the circuit becomes large in size. | Since memory and I/O are present together, the internal circuit is small in size. |
| It cannot be used in compact systems | It can be used in compact systems. |
| Cost is high | Cost is low |
| It is not suitable for devices that run on stored power since total power consumption is high due to external components. | It can be used on devices that use stored power since total power consumption is low due to less external components. |
| RAM, ROM, I/O units, and other peripherals are not embedded on a single chip. | RAM, ROM, CPU and other peripherals are embedded on a single chip. |
| Do not have power saving mode. | Have a power-saving mode. |
| Used in personal computers. | Used in embedded systems. |
| Less number of registers. | More number of registers. |
| Uses an external bus. | Uses an internal controlling bus. |
| Based on the Von Neumann model | Based on the Harvard architecture |
| It is a central processing unit on a single silicon-based integrated chip. | It is a byproduct of the development of microprocessors with a CPU along with other peripherals. |
| Complex and expensive due to a large number of instructions to process. | Simple and inexpensive due to less number of instructions to process. |
| Can run at a very high speed. | Can run up to 200MHz or more. |

## Detailed Differences Between Microprocessor and Microcontroller

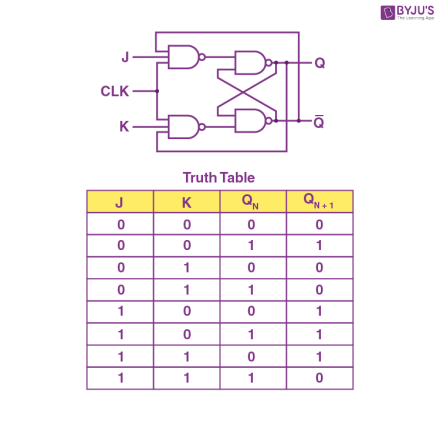
**Microprocessor and Microcontroller are different in the following ways:**

* **Microprocessor only consists of Central Processing Unit, whereas Microcontroller has memory, a CPU and I/O. All these are integrated into one chip.**
* **A microprocessor uses external bus to interface to  ROM, RAM, and other peripherals. Microcontroller, on the other hand, uses internal controlling bus.**
* **A microprocessor is used in personal computers whereas microcontroller is used in embedded system.**
* **Microprocessor is based on the Von Neumann model whereas Microcontroller is based on the Harvard architecture.**
* **A microprocessor is complicated in nature, with a large set of instructions to process. Microcontroller is not that expensive and less complex with fewer instructions to process.**

## ****What is a microprocessor?****

**Microprocessor is a type of computer**[**processor**](https://www.shiksha.com/online-courses/articles/what-is-processor/)**in which both the data processing logic and control are included on a single integrated circuit or on small numbers of integrated circuits. These processors consist of logic, control and arithmetic circuits. Its integrated circuit is capable of interpreting and executing program instructions. These are multiple-purpose, clock-driven and register-based digital integrated circuits that accept input in binary data and process it as per the instruction stored in its memory.**

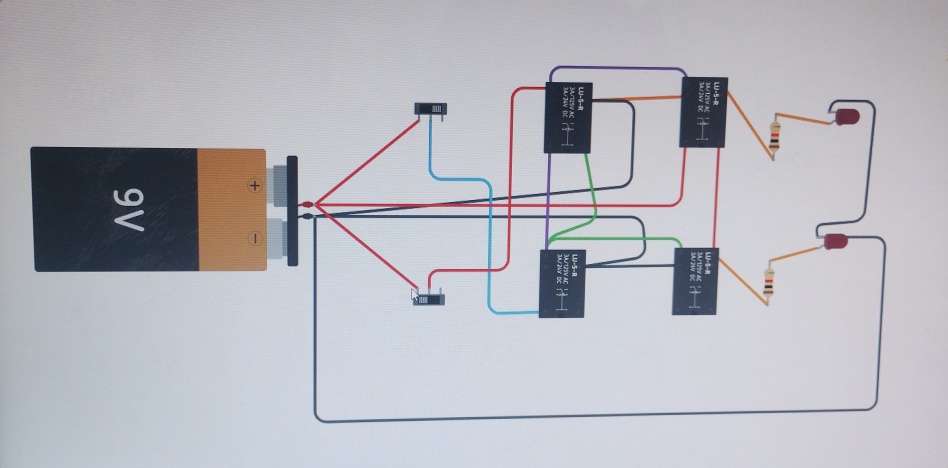
* **What is the JK flip flop?**

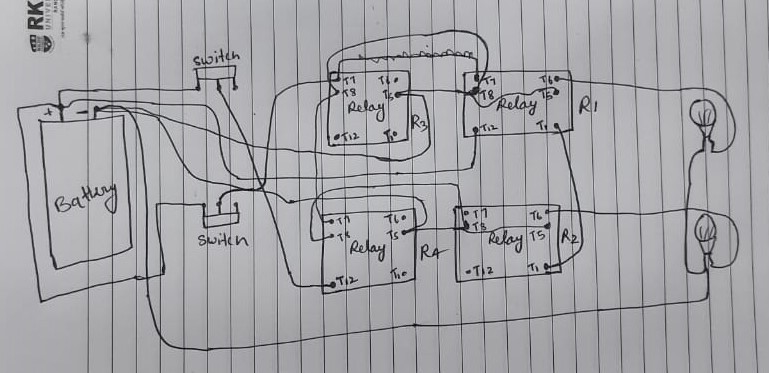
****

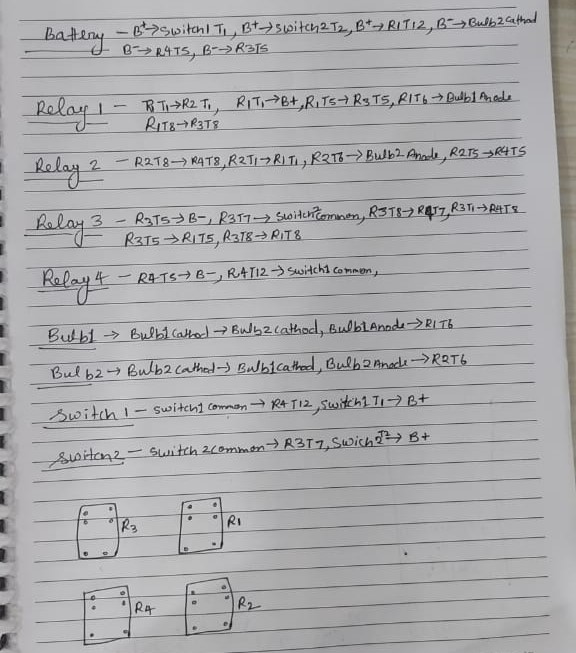
**JK flip flop operates on sequential logic principle, where the output is dependent not only on the current inputs but also on the previous state.**

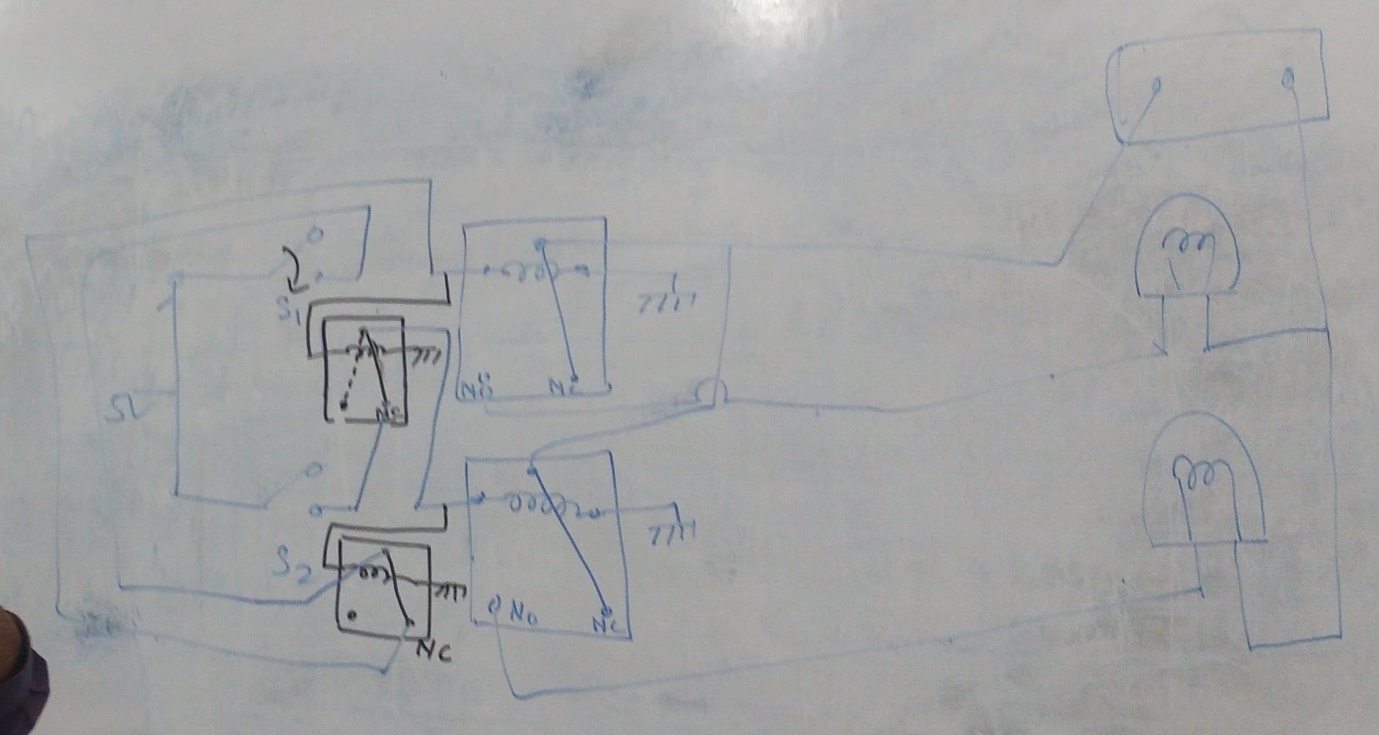
**Projects to make:-**

* 1. **First finger first glow buzzer for Quiz**

****

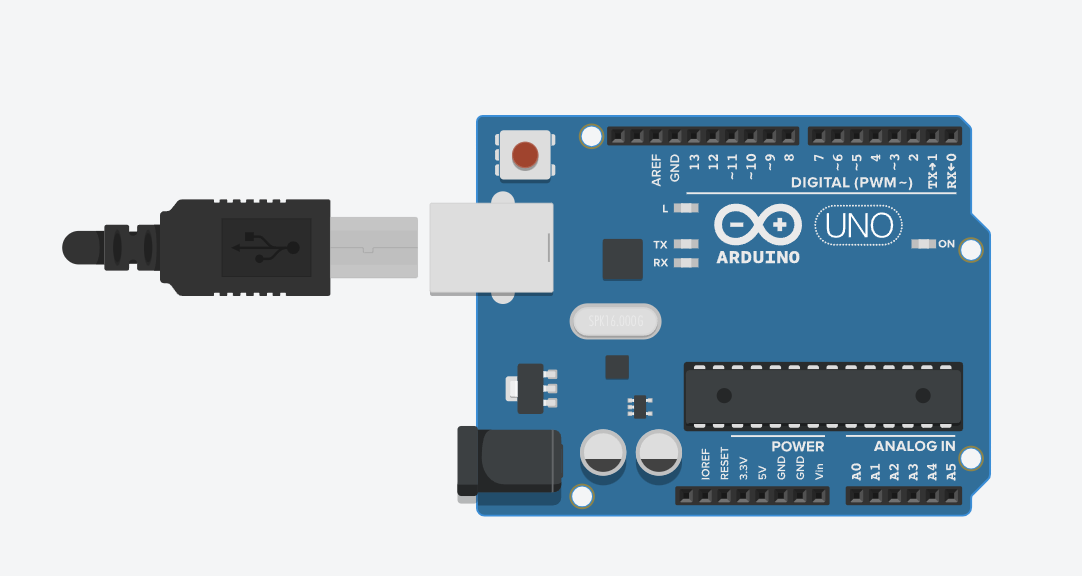
****

****

****

* 1. **Lie detector using Transistor and circuit**

**Hardware**

****

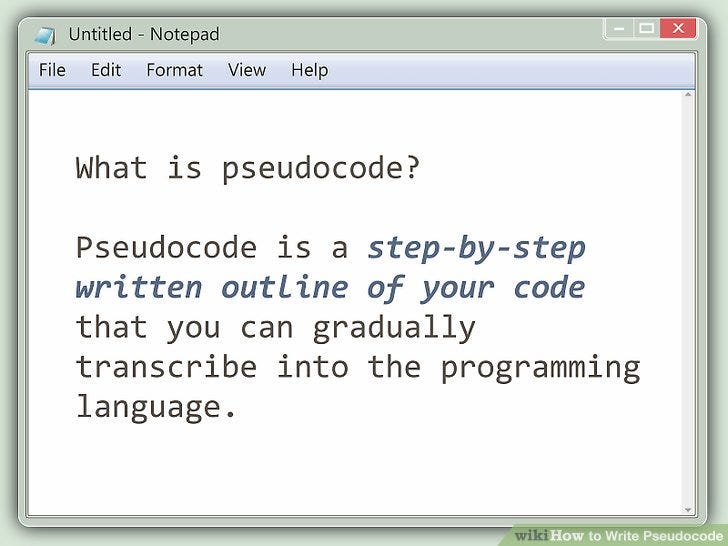
**  **

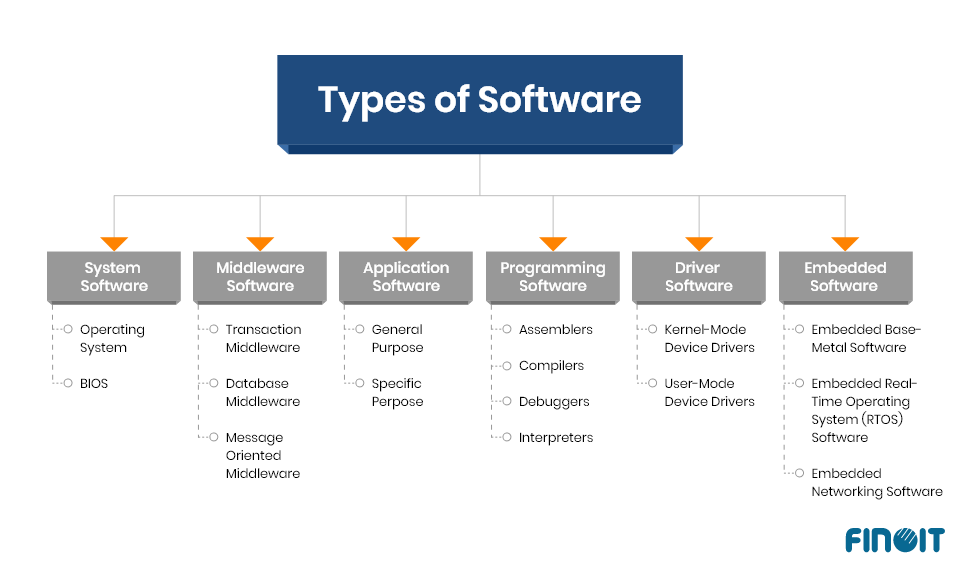
* **PIN Distributed in 3 sides**
* **VIN is used if we have to use more than 5 volt**
* **Arduino Uno board have on board regulator**
* **More than 12 volt will damage Arduino Uno board**
* **Digital pin/Analog pin are used for input/output**
* **Power pin should be used in last**
* **Header pin is connected in arduino to program in another way/internally**

**Software**

* **What is software?**

**The software is basically a set of instructions or commands that tell a computer what to do. In other words, the software is a computer program that provides a set of instructions to execute a user's commands**



****

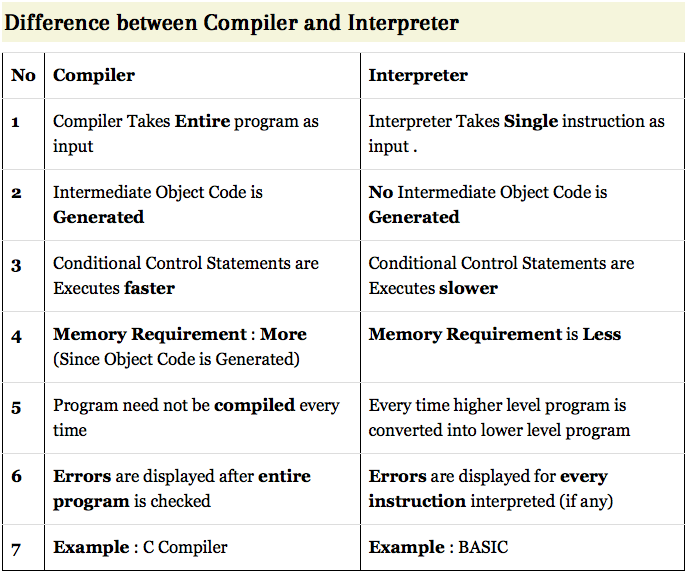
* **Assemble ,compiler ,Debugger and interpreter are language processor**
* **Difference between compiler and interpreter?**

**Interpreters**

* **An interpreter is a program that analyzes and executes a program line-by-line**
* **This means that instead of building the whole program before running, it runs each line sequentially**
* **The overall execution time is slower using an interpreter**
* **Interpreters are great for debugging, as they will execute the program line-by-line up until an error occurs**
* **This makes bug finding easy and provides useful error feedback**
* **Languages such as Java, Ruby, Python and Visual Basic use interpreters**

**Compilers**

* **A compiler is a program that translates and executes another program in whole**
* **The compiler works to first translate the entire program into machine code**
* **Execution time is generally faster than using interpreters as the code is compiled as one block**
* **Compilers use less system resources (memory) than interpreters**
* **Error messages are only produced after the whole program has been analyzed, hence debugging is comparatively harder**
* **Langauges such as BASIC, C++ and Objective-C use compilers**



* **Arduino uno R3 Clock Speed: 16 MHz on Uno board, though IC is capable of 20 MHz maximum at 5 Volts. Flash Memory: 32 KB, of which 0.5 KB used by the bootloader.**

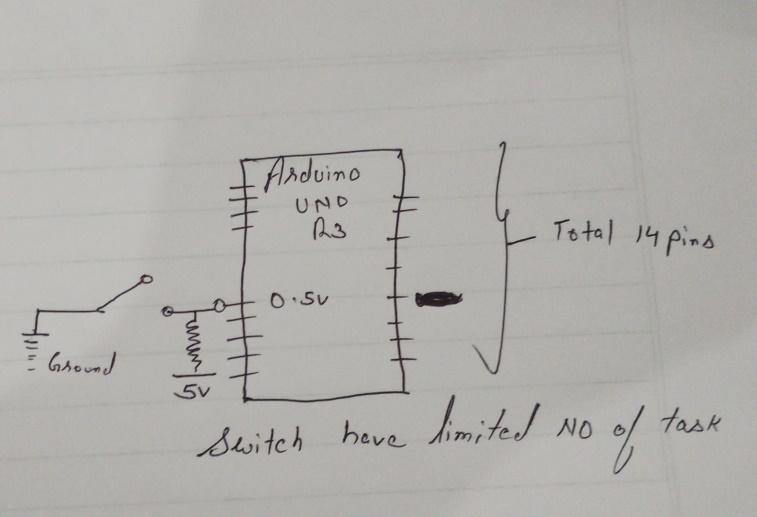
**Work in Single thread**

* **Swarm robotics** **is an approach to the coordination of multiple robots as a system which consist of large numbers of mostly simple physical**[**robots**](https://en.wikipedia.org/wiki/Robot)**. In a robot swarm, the collective behavior of the robots results from local interactions between the robots and between the robots and the environment in which they act. It is supposed that a desired**[**collective behavior**](https://en.wikipedia.org/wiki/Collective_behavior)**emerges from the interactions between the robots and interactions of robots with the environment. This idea emerged on the field of**[**artificial swarm intelligence**](https://en.wikipedia.org/wiki/Artificial_swarm_intelligence)**, as well as the studies of insects, ants and other fields in nature, where**[**swarm behaviour**](https://en.wikipedia.org/wiki/Swarm_behaviour)**occurs.**
* **How multiple arduino can talk each other or work together ?**

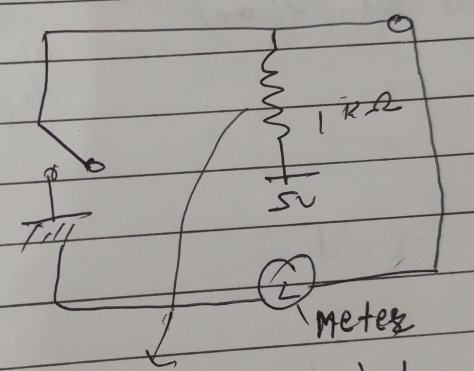
**We will start with two Arduinos, one Master and one Slave, than we will add to this another Slave and adjust the Master code, than we will add yet another Slave to this and adjust the Master code again. That we show how to connect multiple Arduinos via I2C where one Arduino is the Master and has multiple Slaves.**

* **Psudo codes are also called as False code.**
* **First we write psudo code than change to codes.**
* **Psudo codes/ False code are Algorithm.**

**1st project Hardware**

****

* **When we make arduino pin input it become or goes to high impudent state**

** pull up Resister supply voltage to pin**

**1st project coding**

* **How to read button value Arduino?**

**To read the button's state, we use the digitalRead() function, with one argument: the button's pin. The result of that function is going to be either HIGH or LOW. Here, because we are in a pull up configuration (with INPUT\_PULLUP), when the button is not pressed you will read HIGH.**

* **# Being good Programmer it is compulsory to add comments on code.**
* **#How to write code for switch in Arduino?Example**

**The code is given below:**

**const int pinOFswitch = 3;**

**const int LED = 8;**

**void setup( ) {**

**pinMode(pinOFswitch, INPUT);**

**pinMode(LED, OUTPUT);**

**}**

**void loop( )**

**{**

* **# coded by \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**
* **Blank sketch or code to Delete past code**

**void setup() {**

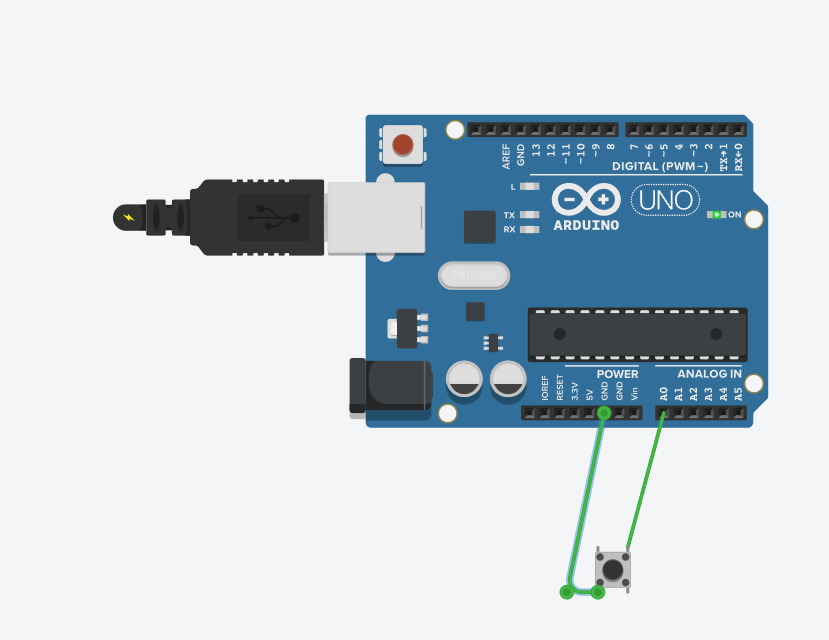
**// put your setup code here, to run once**

**}**

**void loop() {**

**// put your main code here, to run repeatedly**

**}**

****

**// C++ code**

**//**

**void setup()**

**{**

**pinMode(A0, INPUT\_PULLUP);**

**Serial.begin(9600);**

**}**

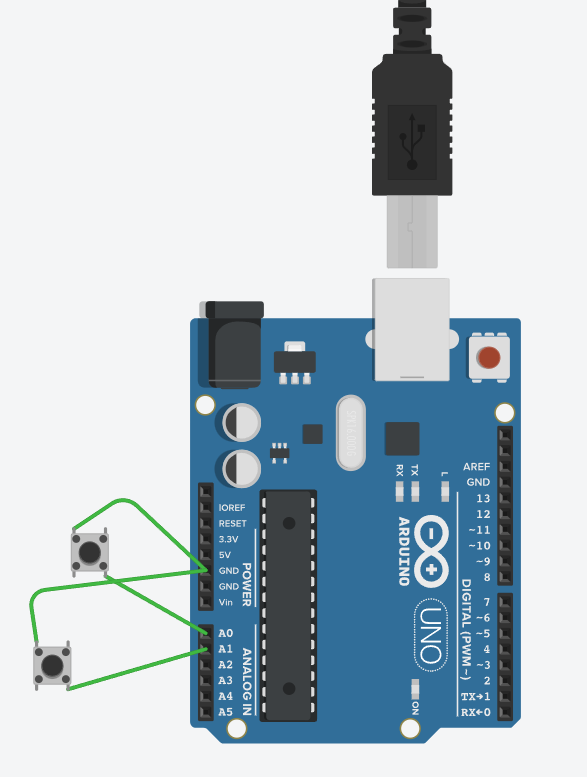
**void loop()**

**{**

**Serial.println(digitalRead(A0));**

**delay(1000); // Wait for 1000 millisecond(s)**

**}**

****

// C++ code

//

int c=0;

void setup()

{

pinMode(A0,INPUT\_PULLUP);

pinMode(A1,INPUT\_PULLUP);

Serial.begin(9600);

}

void loop()

{

digitalRead(A0);

delay(1000); // Wait for 1000 millisecond(s)

if (digitalRead(A0)==0);

c++;

if(c%2!=0)

{

Serial.println("hello");

**}**

**else**

**{**

**Serial.println("hi");**

**}**

**}**

**…………………………………………………………………………………………………………………**

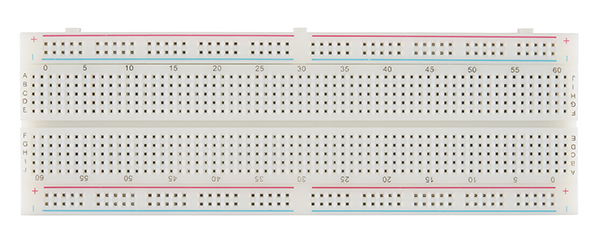
* **Input/ INPUT\_PULLUP(if we are not using for hardware setup)**
* **Serial.print/ Serial.println**
* **what is debouncing problem in arduino**

**Debouncing is a technique where you delay the execution of a function until after a certain amount of time has passed. checking twice in a short period of time to make sure the pushbutton is definitely pressed.**

**Basic knowledge of programming**

1. **Input/output**
2. **Variables**
3. **Data types**
4. **Operators**
5. **Control structure**
6. **Advance Data Types**

**What is breadboard used for?**

****

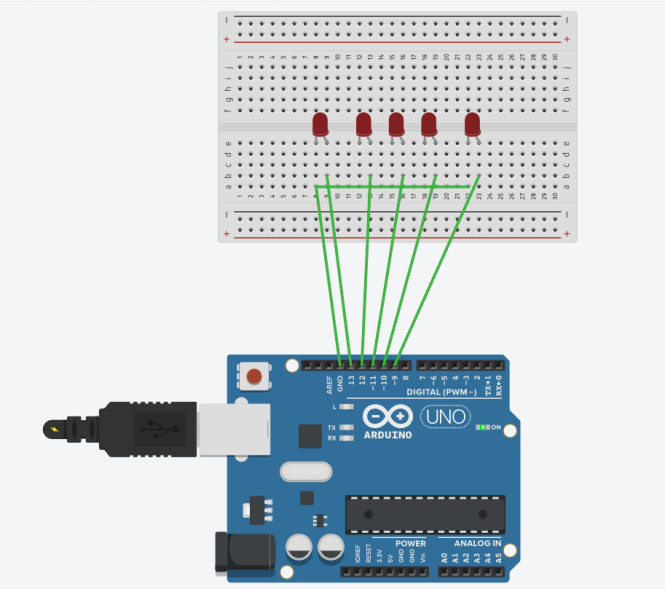
**A breadboard (sometimes called a plugblock) is used for building temporary circuits. It is useful to designers because it allows components to be removed and replaced easily. It is useful to the person who wants to build a circuit to demonstrate its action, then to reuse the components in another circuit.**

**There are two major types of breadboards; these are solder and solderless boards. Solder boards are boards you have to solder components onto (per the name). These are most of your standard circuit boards, and if you flip one over you'll notice that all of the connections are soldered to the board itself.**

**Solderless Breadboard with Full, Half and Mini Sizes**

1. **Half (400 Tie Points)**
2. **Mini (Tiny)**
3. **Full (830 Tie Points)**

* **Blinking 5 led together**

****

**void setup()**

**{**

**pinMode(13, OUTPUT);**

**pinMode(12, OUTPUT);**

**pinMode(11, OUTPUT);**

**pinMode(10, OUTPUT);**

**pinMode(9, OUTPUT);**

**}**

**void loop()**

**{**

**digitalWrite(13, HIGH);**

**digitalWrite(12, HIGH);**

**digitalWrite(11, HIGH);**

**digitalWrite(10, HIGH);**

**digitalWrite(9, HIGH);**

**digitalWrite(13, LOW);**

**digitalWrite(12, LOW);**

**digitalWrite(11, LOW);**

**digitalWrite(10,LOW);**

**digitalWrite(9, LOW);**

**}**

**…………………………………………………………………………………………………………………**

**void setup()**

**{**

**pinMode(13, OUTPUT);**

**pinMode(12, OUTPUT);**

**pinMode(11, OUTPUT);**

**pinMode(10, OUTPUT);**

**pinMode(9, OUTPUT);**

**}**

**void loop()**

**{**

**digitalWrite(13, HIGH);**

**digitalWrite(12, HIGH);**

**digitalWrite(11, HIGH);**

**digitalWrite(10, HIGH);**

**digitalWrite(9, HIGH);**

**delay(1000);**

**digitalWrite(13, LOW);**

**digitalWrite(12, LOW);**

**digitalWrite(11, LOW);**

**digitalWrite(10,LOW);**

**digitalWrite(9, LOW);**

**delay(500);**

**}**

**……………………………………………………………………………………………………………………**

**// Blink led alternatively**

**void setup()**

**{**

**for (int i=13;i>8;i--){**

**pinMode(i, OUTPUT);**

**}**

**}**

**void loop()**

**{**

**delay (500);**

**for (int i=13;i>8;i--){**

**digitalWrite(i, HIGH);**

**delay(500);**

**}**

**for (int i=13;i>8;i--){**

**digitalWrite(i, LOW);**

**//delay(500);**

**}**

**……………………………………………………………………………………………………………………**

**//BLINK LED Alternatively**

**void setup()**

**{**

**for (int i=13;i>9;i--){**

**pinMode(i, OUTPUT);**

**}**

**}**

**void loop()**

**{**

**delay (500);**

**for (int i=13;i>8;i--){**

**digitalWrite(i, HIGH);**

**delay(500);**

**}**

**for (int i=13;i>8;i--){**

**digitalWrite(i, LOW);**

**// delay (500);**

**}**

**}**

**………………………………………………………………………………………………………………………**

**int i,j;**

**void setup()**

**{**

**for (int i=13;i>8;i--){**

**pinMode(i, OUTPUT);**

**}**

**}**

**void loop()**

**{**

**for (int i=13;i>8;i--,j++){**

**digitalWrite(i, HIGH);**

**delay(500);**

**}**

**for (int k=13;k>8;k--){**

**digitalWrite(k, LOW);**

**}**

**// delay (500);**

**}**

**…………………………………………………………………………………………………………………..**

**//Glow led from left to right and vice versa**

**const int ledPins[] = {2, 3, 4, 5, 6}; // Define the pins for the LEDs**

**const int numLeds = sizeof(ledPins) / sizeof(ledPins[0]); // Calculate the number of LEDs**

**int delayTime = 100; // Delay time in milliseconds**

**void setup() {**

**for (int i = 0; i < numLeds; i++) {**

**pinMode(ledPins[i], OUTPUT); // Set each LED pin as an output**

**}**

**}**

**void loop() {**

**// Glow LEDs from left to right**

**for (int i = 0; i < numLeds; i++) {**

**digitalWrite(ledPins[i], HIGH); // Turn the current LED on**

**delay(delayTime); // Wait for the specified delay**

**digitalWrite(ledPins[i], LOW); // Turn the current LED off**

**}**

**………………………………………………………………………………………………………………..**

**// Glow LEDs from right to left**

**for (int i = numLeds - 1; i >= 0; i--) {**

**digitalWrite(ledPins[i], HIGH); // Turn the current LED on**

**delay(delayTime); // Wait for the specified delay**

**digitalWrite(ledPins[i], LOW); // Turn the current LED off**

**}**

**}**

**,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,**

**//Glow led in converging pattern**

**const int ledPins[] = {2, 3, 4, 5, 6}; // Define the pins for the LEDs**

**const int numLeds = sizeof(ledPins) / sizeof(ledPins[0]); // Calculate the number of LEDs**

**int delayTime = 100; // Delay time in milliseconds**

**void setup() {**

**for (int i = 0; i < numLeds; i++) {**

**pinMode(ledPins[i], OUTPUT); // Set each LED pin as an output**

**}**

**}**

**void loop() {**

**// Glow LEDs from both ends towards the center**

**for (int i = 0; i < numLeds / 2; i++) {**

**digitalWrite(ledPins[i], HIGH); // Turn the current LED on from the left**

**digitalWrite(ledPins[numLeds - i - 1], HIGH); // Turn the current LED on from the right**

**delay(delayTime); // Wait for the specified delay**

**digitalWrite(ledPins[i], LOW); // Turn the current LED off from the left**

**digitalWrite(ledPins[numLeds - i - 1], LOW); // Turn the current LED off from the right**

**}**

**……………………………………………………………………………………………………………………**

**// Glow LEDs from the center towards both ends**

**for (int i = numLeds / 2 - 1; i >= 0; i--) {**

**digitalWrite(ledPins[i], HIGH); // Turn the current LED on from the left**

**digitalWrite(ledPins[numLeds - i - 1], HIGH); // Turn the current LED on from the right**

**delay(delayTime); // Wait for the specified delay**

**digitalWrite(ledPins[i], LOW); // Turn the current LED off from the left**

**digitalWrite(ledPins[numLeds - i - 1], LOW); // Turn the current LED off from the right**

**}**

**}**

**,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,**

**// Glow led in diverging pattern**

**const int ledPins[] = {2, 3, 4, 5, 6}; // Define the pins for the LEDs**

**const int numLeds = sizeof(ledPins) / sizeof(ledPins[0]); // Calculate the number of LEDs**

**int delayTime = 100; // Delay time in milliseconds**

**void setup() {**

**for (int i = 0; i < numLeds; i++) {**

**pinMode(ledPins[i], OUTPUT); // Set each LED pin as an output**

**}**

**}**

**void loop() {**

**// Glow LEDs from the center towards both ends**

**for (int i = numLeds / 2; i >= 0; i--) {**

**digitalWrite(ledPins[i], HIGH); // Turn the current LED on from the center to the left**

**digitalWrite(ledPins[numLeds - i - 1], HIGH); // Turn the current LED on from the center to the right**

**delay(delayTime); // Wait for the specified delay**

**digitalWrite(ledPins[i], LOW); // Turn the current LED off from the center to the left**

**digitalWrite(ledPins[numLeds - i - 1], LOW); // Turn the current LED off from the center to the right**

**}**

**,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,**

**// Glow LEDs from both ends towards the center**

**for (int i = 0; i < numLeds / 2; i++) {**

**digitalWrite(ledPins[i], HIGH); // Turn the current LED on from the left**

**digitalWrite(ledPins[numLeds - i - 1], HIGH); // Turn the current LED on from the right**

**delay(delayTime); // Wait for the specified delay**

**digitalWrite(ledPins[i], LOW); // Turn the current LED off from the left**

**digitalWrite(ledPins[numLeds - i - 1], LOW); // Turn the current LED off from the right**

**}**

**}**

**………………………………………………………………………………………………………………..**

**// Glow led in randomly**

**int i;**

**void setup()**

**{**

**for (int i=13;i>8;i--){**

**pinMode(i, OUTPUT);**

**}**

**}**

**void loop()**

**{**

**i-random (9,13),**

**digitalWrite(i, HIGH);**

**delay(500);**

**digitalWrite(i, LOW);**

**delay (500);**

**}**

**…………………………………………………………………………………………………………………**

**//led cube making 4\*4?**

**https://www.instructables.com/LED-CUBE-4X4X4-USING-ARDUINO-UNO/**

**// code to turn on led while button is pressed**

**const int ledPin = 13; // Choose a pin for the LED**

**const int buttonPin = 2; // Choose a pin for the button**

**void setup() {**

**pinMode(ledPin, OUTPUT); // Set the LED pin as an output**

**pinMode(buttonPin, INPUT); // Set the button pin as an input**

**}**

**void loop() {**

**int buttonState = digitalRead(buttonPin); // Read the state of the button**

**if (buttonState == HIGH) {**

**digitalWrite(ledPin, HIGH); // Turn on the LED if the button is pressed**

**} else {**

**digitalWrite(ledPin, LOW); // Turn off the LED if the button is not pressed**

**}**

**}**

**…………………………………………………………………………………………………………………..**

**//toggle the led state of led**

**const int ledPin = 13; // Choose a pin for the LED**

**void setup() {**

**pinMode(ledPin, OUTPUT); // Set the LED pin as an output**

**}**

**void loop() {**

**digitalWrite(ledPin, HIGH); // Turn on the LED**

**delay(1000); // Wait for 1 second**

**digitalWrite(ledPin, LOW); // Turn off the LED**

**delay(1000); // Wait for 1 second**

**}**

**………………………………………………………………………………………………………………..**

**Alternatively, you can use the toggle function to toggle the state of the LED:**

**const int ledPin = 13; // Choose a pin for the LED**

**bool ledState = LOW; // Initialize the LED state to LOW**

**void setup() {**

**pinMode(ledPin, OUTPUT); // Set the LED pin as an output**

**}**

**void loop() {**

**ledState = !ledState; // Toggle the LED state**

**digitalWrite(ledPin, ledState); // Set the LED state**

**delay(1000); // Wait for 1 second**

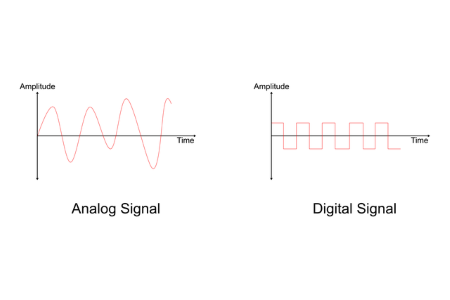
**}**

**……………………………………………………………………………………………………………………**

**Analog Sensors**

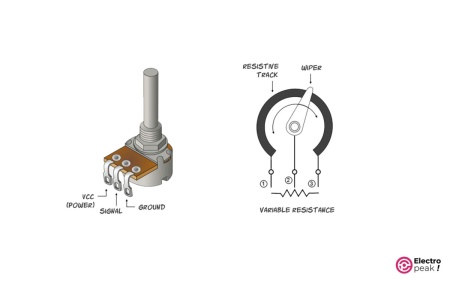
**Analog sensors create analog signals from the quantity that is sensed. In comparison, digital sensors produce digital signals when performing a measurement. As mentioned previously, the most common types of analog sensors include sound sensors, light sensors, temperature sensors, and pressure sensors.**

**What is the difference between analog and digital devices?**



**Analog and Digital are two types of signals. The difference between Analog and Digital is how the data is transmitted. Digital signals are transmitted as 1s and 0s, whereas analog signals are transmitted in waves.**

**What is a Potentiometer?**



**A potentiometer is a manually adjustable variable resistor with 3 terminals. Two of the terminals are connected to the opposite ends of a resistive element, and the third terminal connects to a sliding contact, called a wiper, moving over the resistive element.**

**What is a potentiometer used for?**

**A potentiometer is used as a voltage divider or variable resistor in a circuit. Some applications include dimmer switches for lights, brightness controls in televisions, and faders in audio equipment.**



**Psudo code**

* **Read the sensor**
* **Print the reading in serial monitor**
* **Map the input/ output**
* **Convert 0-1024 or 1024 –to 0**
* **3 phase low, average and high**

**Code**

**Void setup {**

**}**

**PinMode(**

**Int sensor =0;**

**Sensor =analogRead(A0);**

**Sensor= map(sensor,o,10);**

**Serial.println(sensor),**



**Int sum=0;**

**Void setup ()**

**{**

**PinMode(A0,input);**

**Serial.begin(9600);**

**}**

**Void loop()**

**Sensor =analogRead(A0);**

**Serial.println(sensor);**

**Delay(250);**

**}**

**………………………………………………………………………………………………………………………………………………………**

**Int sum=0;**

**Void setup ()**

**{**

**PinMode(A0,input);**

**Serial.begin(9600);**

**}**

**Void loop()**

**Sensor =analogRead(A0);**

**Sensor=map(sensor,0,1024,0,10);**

**Serial.println(sensor);**

**Delay(250);**

**}**

**……………………………………………………………………………………………………………………………………………………**

**Int sum=0;**

**Void setup ()**

**{**

**PinMode(A0,input);**

**Serial.begin(9600);**

**}**

**Void loop()**

**Sensor =analogRead(A0);**

**Sensor=map(sensor,0,1024,1024,0);**

**Serial.println(sensor);**

**Delay(250);**

**}**

**………………………………………………………………………………………………………………………………………………….**

**Int sum=0;**

**Void setup ()**

**{**

**PinMode(A0,input);**

**Serial.begin(9600);**

**}**

**Void loop()**

**Sensor =analogRead(A0);**

**Sensor=map(sensor,0,1024,1024,-1);**

**Serial.println(sensor);**

**Delay(250);**

**}**

**……………………………………………………………………………………………………………………………………………………**

**Int sum=0;**

**Void setup ()**

**{**

**PinMode(A0,input);**

**Serial.begin(9600);**

**}**

**Void loop()**

**Sensor =analogRead(A0);**

**Sensor=map(sensor,0,1024,1024,-3);**

**Serial.println(sensor);**

**Delay(250);**

**}**

**……………………………………………………………………………………………………………………………………………………**

**Int sum=0;**

**Void setup ()**

**{**

**PinMode(A0,input);**

**Serial.begin(9600);**

**}**

**Void loop()**

**Sensor =analogRead(A0);**

**Sensor=map(sensor,0,1024,0,10);**

**Serial.println(sensor);**

**If(sensor<5){**

**Serial.println(“Low”);**

**}**

**If(sensor>8){**

**Serial.println(“High”);**

**}**

**Else{**

**If (sensor>5)**

**Serial.println(“average”);**

**}**

**Delay(250);**

**}**

………………………………………………………………………………………………………………………………………………………….

**int sens=0;**

**void setup()**

**{**

**pinMode(A0, INPUT);**

**pinMode(6, OUTPUT);**

**Serial.begin(9600);**

**}**

**void loop()**

**{**

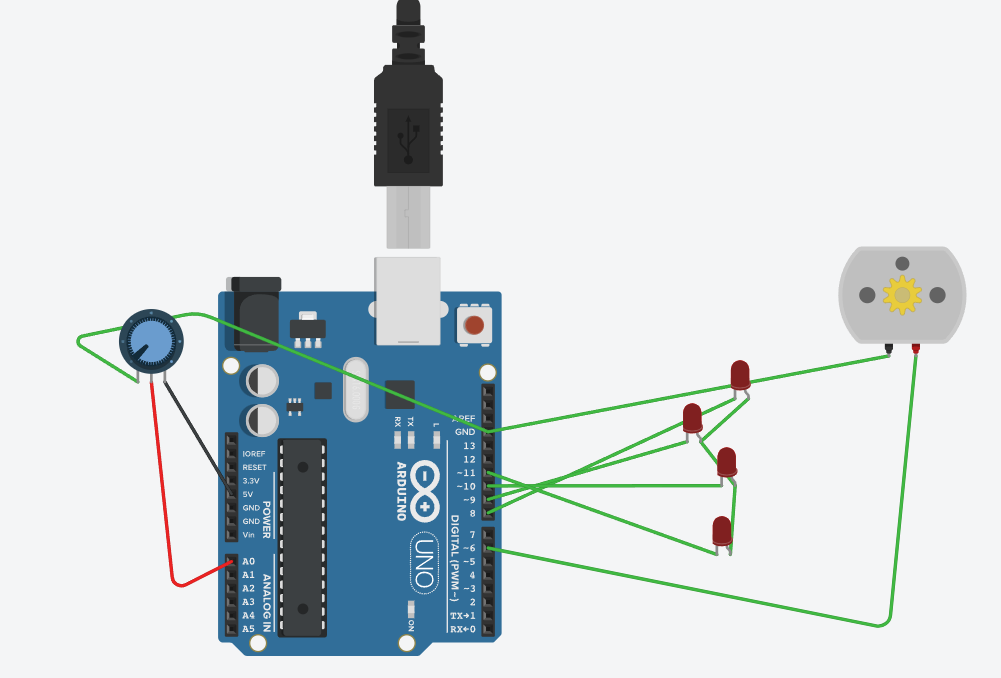
**sens=analogRead(A0);**

**//sens=map(sens,0,1024,0,10);**

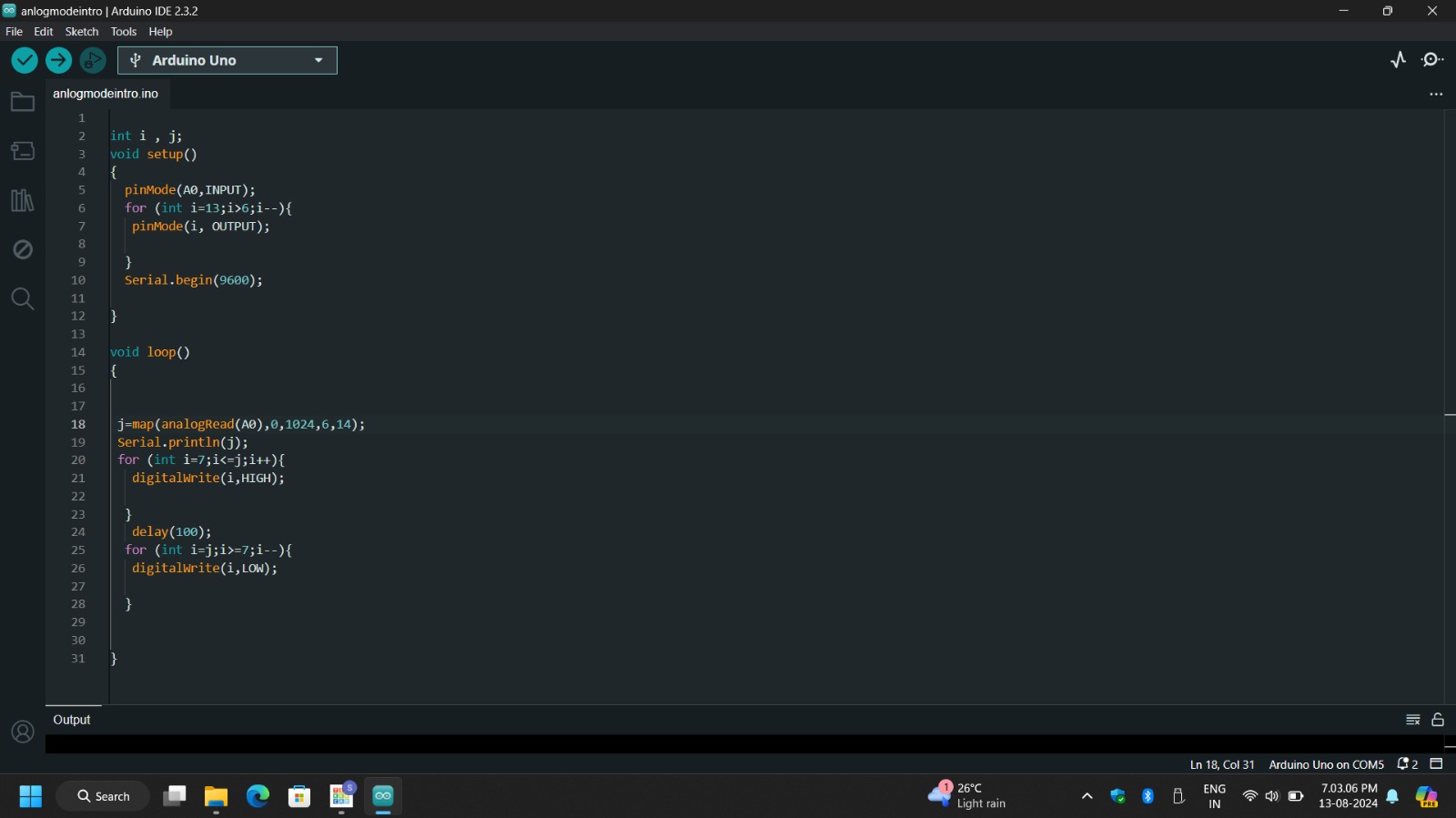
**//Serial.println(sens);**

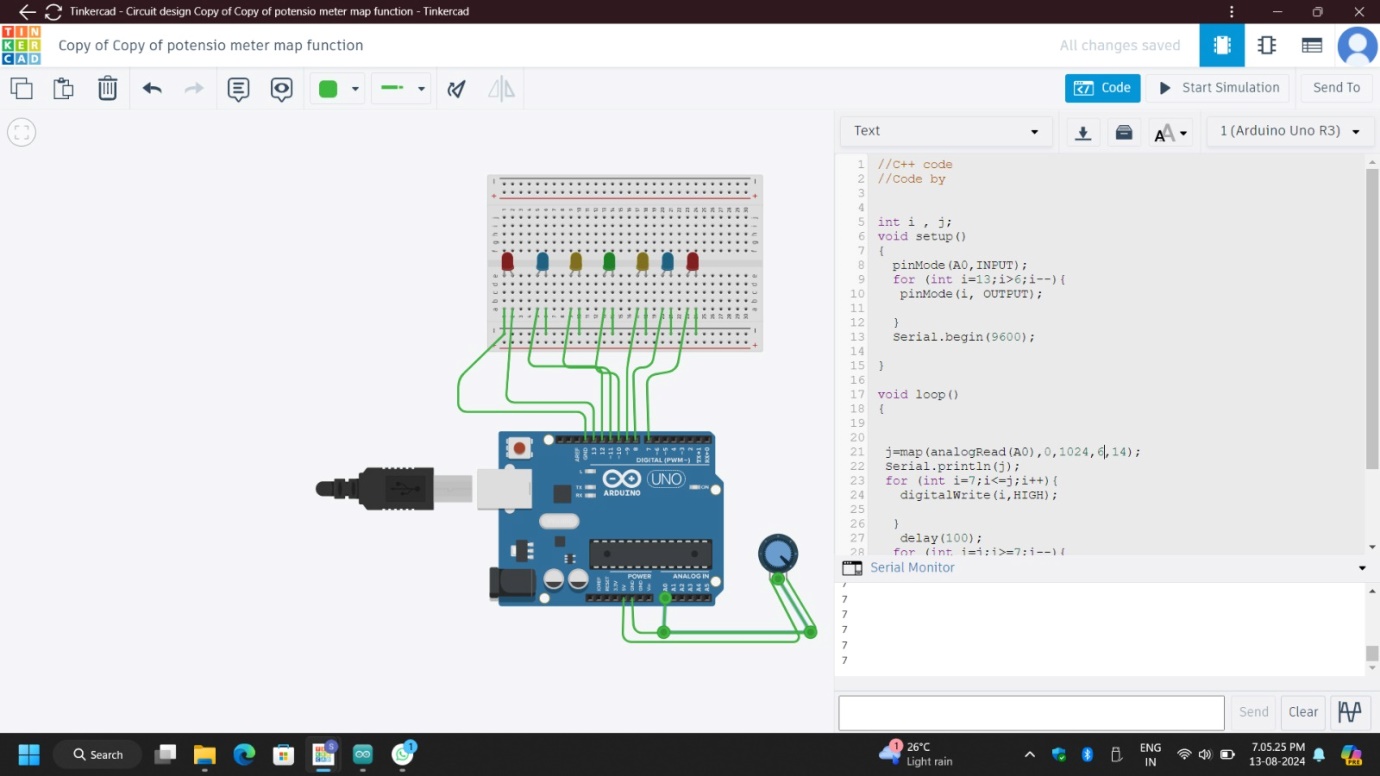
**analogWrite(6,sens);**

**}**

****

**……………………………………………………………………………………………………………………..**

****

****

**int i , j;**

**void setup()**

**{**

**pinMode(3,OUTPUT);**

**pinMode(A0,INPUT);**

**for (int i=13;i>6;i--){**

**pinMode(i, OUTPUT);**

**}**

**Serial.begin(9600);**

**}**

**void loop()**

**{**

**j=map(analogRead(A0),0,1024,6,14);**

**Serial.println(j);**

**for (int i=7;i<=j;i++){**

**digitalWrite(i,HIGH);**

**}**

**delay(100);**

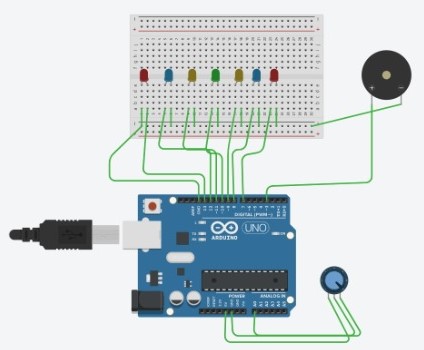
**for (int i=j;i>=7;i--){**

**digitalWrite(i,LOW);**

**}**

**analogWrite(3,analogRead(A0));**

**}**

****

**………………………………………………………………………………………………………………………**

**How to Read Serial Data in Arduino**

**TX= transmit**

**RX= receive**

**C=Serail.read();**

**Void Setup(){**

**PinMode(3,Output);**

**Serail,begin(9600);**

**}**

**Void loop(){**

**//prompt a message**

**Serail.println(“1 to on 2 to off”);**

**//wait for user to answer**

**While (! Serial.aviable())**

**Serial.read();**

**//read Serial Data**

**If (C==”1”){**

**Digitalwrite(7,High);**

**}**

**If(c==”2”){**

**}**

**}**

**………………………………………………………………………………………………………………………**

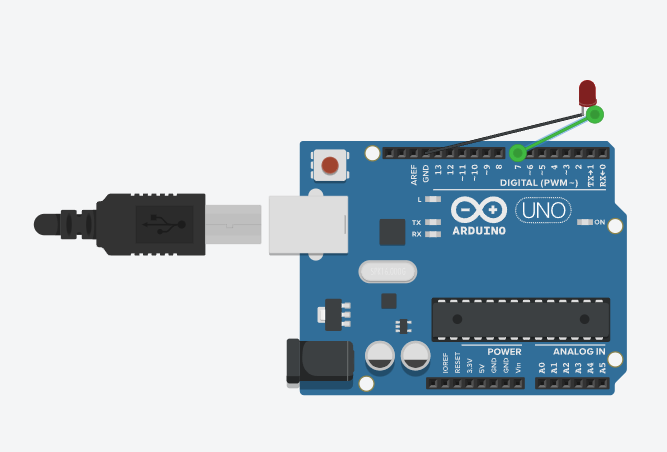
**What is prompt communication in Arduino?**

In the context of Arduino, "prompt communication" typically refers to the ability of the Arduino to send and receive data quickly and efficiently between itself and other devices or systems. This communication can occur via various methods, such as serial communication, I2C (Inter-Integrated Circuit), SPI (Serial Peripheral Interface), or wireless communication like Bluetooth or Wi-Fi.

Here are some common aspects of prompt communication in Arduino:

1. **Serial Communication**: Arduino boards commonly use UART (Universal Asynchronous Receiver/Transmitter) for serial communication. This involves sending data between the Arduino and a computer or other serial devices. It is often used for debugging, logging data, or interacting with other hardware.
   * **Serial Monitor**: A tool within the Arduino IDE that allows you to view data being sent from the Arduino board in real-time.
   * **Serial.begin()**: Initializes serial communication at a specified baud rate.
2. **I2C Communication**: This is a synchronous, multi-master, multi-slave communication protocol. It's often used to connect multiple devices with just two wires: SDA (data line) and SCL (clock line).
   * **Wire Library**: The standard library for I2C communication in Arduino. It provides functions to communicate with I2C devices.
3. **SPI Communication**: A synchronous serial communication protocol that uses separate lines for data input, data output, clock, and a select line for each device.
   * **SPI Library**: Provides functions to set up and manage SPI communication.
4. **Wireless Communication**: For remote or wireless communication, you might use modules like Bluetooth (e.g., HC-05), Wi-Fi (e.g., ESP8266 or ESP32), or LoRa.
   * **WiFi Library**: For connecting Arduino to the internet or local networks using Wi-Fi.
   * **Bluetooth Library**: For establishing serial communication over Bluetooth.
5. **Interrupts**: These are used for handling high-priority tasks promptly without needing to constantly check for conditions. Interrupts can help in handling events like button presses or sensor readings with minimal delay.

Prompt communication involves ensuring that the data is transmitted and received with minimal delay, and that the Arduino can handle the communication efficiently without significant lag or loss of information. This is crucial in applications where real-time response or high-speed data exchange is required.

****

**// C++ code**

**//**

**String c;**

**void setup()**

**{**

**pinMode(7,OUTPUT);**

**Serial.begin(9600);**

**}**

**void loop()**

**{**

**Serial.println("enter 1 for on and 2 for off");**

**while(!Serial.available()){**

**}**

**while(Serial.available()>0){**

**c= Serial.readString();**

**Serial.println(c);**

**delay(500);**

**}**

**if(c == "on" ){**

**digitalWrite(7,HIGH);**

**}**

**if(c == "off"){**

**digitalWrite(7,LOW);**

**}**

**}**

**//Serial.parseInt();**

**//Serial.parseFloat();**

**………………………………………………………………………………………………………………………**

**BY sir**

**String C=” “;char c;**

**void setup()**

**{**

**pinMode(7,OUTPUT);**

**Serial.begin(9600);**

**}**

**void loop()**

**{**

**Serial.println("enter 1 for on and 2 for off");**

**while(!Serial.available()){**

**}**

**while(Serial.available()>0){**

**c+=c; or c=c+c; or c+=c ;**

**c= Serial.readString();**

**Serial.println(c);**

**delay(500);**

**if (c==’1’){**

**digitalwrite(7,High);**

**}**

**if (c==’2’){**

**digitalwrite(7,low);**

**}**

**}**

**Serial.parseInt();**

**…………………………………………………………………………………………………………………….**

1. **Chat Bot using Arduino using serial Data**

**// C++ code**

**//**

**void answer(int x)**

**{**

**switch(x){**

**case 1:**

**Serial.println("Hello Welcome How Can I help you ! ");**

**break;**

**case 2:**

**Serial.println("Goodbye! Have a great day!");**

**break;**

**case 3:**

**Serial.println("I am Bot... devloped by Rider ");**

**break;**

**case 4:**

**Serial.println("My Name is NextGenDevHub! a cute chat Bot for entertanment ");**

**break;**

**case 5:**

**Serial.println("No i do not save our chat even i don't know what is your previous question");**

**break;**

**case 6:**

**Serial.println("I can't speek i am only give some programed question's Answer ! ");**

**break;**

**case 7:**

**Serial.println(" !!!!");**

**break;**

**case 8:**

**Serial.println("");**

**break;**

**}**

**}**

**void setup()**

**{**

**pinMode(13, OUTPUT);**

**Serial.begin(9600);**

**Serial.println("User I am UNO Bot . ");**

**digitalWrite(13, HIGH);**

**delay(1000);**

**}**

**void loop()**

**{**

**if (Serial.available()>0){**

**String prompt =Serial.readString();**

**if (prompt=="hii bot"||prompt=="hello bot"||prompt=="hello"||prompt=="hii")**

**{**

**answer(1);**

**}**

**else if (prompt=="Bye!"||prompt=="thankyou" ||prompt=="Bye")**

**{**

**answer(2);**

**}**

**else if (prompt=="Are you a human"||prompt=="Are you a robot")**

**{**

**answer(3);**

**}**

**else if (prompt=="What is your name"||prompt=="your name"||prompt=="tell me about you"||prompt=="tell me about yourself")**

**{**

**answer(4);**

**}**

**else if (prompt=="do you save what i say"||prompt=="do you save what we talk")**

**{**

**answer(5);**

**}**

**else if (prompt=="which languages can you speak"||prompt=="Which languages can you speak")**

**{**

**answer(6);**

**}**

**else if (prompt=="Kaa Bol raha hai")**

**{**

**answer(8);**

**}**

**else{**

**answer(7);**

**}**

**delay(2000);**

**}**

**}**

1. **Ludo Dice using Serial data**
2. **Password protected led using Relay**
3. **Locker using Potentiometer**
4. **IVR using Arduino**
5. **Student Record Using Read Serial data**

**……………………………………………………………………………………………………………………**

* **ambient light sensor phototransistor**
* **Ir Sensor**

****

**How motor is controlled?**

**The motor is controlled applying a voltage on the motor leads. The higher the voltage, the higher the speed. The direction is changed reversing the polarity on the leads. The maximum torque is limited by the current rating of the motor and it is obtained at zero speed (start-up).**

**Line flowing robot**

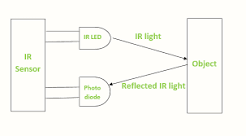
**IR sensor**



**What is the IR sensor?**

**An infrared sensor (IR sensor) is a radiation-sensitive optoelectronic component with a spectral sensitivity in the infrared wavelength range 780 nm … 50 µm. IR sensors are now widely used in motion detectors, which are used in building services to switch on lamps or in alarm systems to detect unwelcome guests.**

**What is the working principle of IR?**

****

**The photo-diode's resistance and output voltage change in proportion to the IR light received. This is the underlying working principle of the IR sensor. When the IR transmitter emits radiation, it reaches the object and some of the radiation reflects back to the IR receiver.**

**Single sensor Line flowing robot**

**// Define motor control pins**

**#define LEFT\_MOTOR\_PIN 9 // PWM pin for left motor speed**

**#define RIGHT\_MOTOR\_PIN 10 // PWM pin for right motor speed**

**// Define sensor pin**

**#define LINE\_SENSOR\_PIN A0 // Analog pin for IR sensor (assuming analog sensor)**

**// Define motor speed**

**#define MOTOR\_SPEED 255 // Max speed**

**// Function prototypes**

**void setup();**

**void loop();**

**void moveForward();**

**void turnLeft();**

**void turnRight();**

**void stopMotors();**

**void setup() {**

**// Initialize motor pins**

**pinMode(LEFT\_MOTOR\_PIN, OUTPUT);**

**pinMode(RIGHT\_MOTOR\_PIN, OUTPUT);**

**// Initialize sensor pin**

**pinMode(LINE\_SENSOR\_PIN, INPUT);**

**// Start with motors stopped**

**stopMotors();**

**}**

**void loop() {**

**int sensorValue = analogRead(LINE\_SENSOR\_PIN); // Read sensor value**

**if (sensorValue > 512) { // Threshold to detect line (black)**

**moveForward(); // If over line, move forward**

**} else {**

**// If not over line (white), turn**

**turnLeft(); // Adjust turning based on requirement**

**}**

**}**

**void moveForward() {**

**analogWrite(LEFT\_MOTOR\_PIN, MOTOR\_SPEED); // Set speed of left motor**

**analogWrite(RIGHT\_MOTOR\_PIN, MOTOR\_SPEED); // Set speed of right motor**

**}**

**void turnLeft() {**

**analogWrite(LEFT\_MOTOR\_PIN, MOTOR\_SPEED / 2); // Slow down left motor**

**analogWrite(RIGHT\_MOTOR\_PIN, MOTOR\_SPEED); // Full speed on right motor**

**}**

**void turnRight() {**

**analogWrite(LEFT\_MOTOR\_PIN, MOTOR\_SPEED); // Full speed on left motor**

**analogWrite(RIGHT\_MOTOR\_PIN, MOTOR\_SPEED / 2); // Slow down right motor**

**}**

**void stopMotors() {**

**analogWrite(LEFT\_MOTOR\_PIN, 0); // Stop left motor**

**analogWrite(RIGHT\_MOTOR\_PIN, 0); // Stop right motor**

**}**

**………………………………………………………………………………………………………………………**

**Double sensor line following**

**Truth table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sensor left** | **Sensor right** | **Action** | **Motor left** | **Motor right** |
|  |  |  |  |  |
| **0** | **0** | **forward** | **forward** | **forward** |
| **1** | **1** | **stop** | **Release** | **Release** |
| **0** | **1** | **right** | **forward** | **Release** |
| **1** | **0** | **left** | **Release** | **forward** |

**// Define motor control pins**

**#define LEFT\_MOTOR\_PIN 9 // PWM pin for left motor speed**

**#define RIGHT\_MOTOR\_PIN 10 // PWM pin for right motor speed**

**// Define sensor pins**

**#define LEFT\_SENSOR\_PIN A0 // Analog pin for left IR sensor**

**#define RIGHT\_SENSOR\_PIN A1 // Analog pin for right IR sensor**

**// Define motor speed**

**#define MOTOR\_SPEED 255 // Max speed**

**#define TURN\_SPEED 150 // Speed for turning**

**// Function prototypes**

**void setup();**

**void loop();**

**void moveForward();**

**void turnLeft();**

**void turnRight();**

**void stopMotors();**

**void setup() {**

**// Initialize motor pins**

**pinMode(LEFT\_MOTOR\_PIN, OUTPUT);**

**pinMode(RIGHT\_MOTOR\_PIN, OUTPUT);**

**// Initialize sensor pins**

**pinMode(LEFT\_SENSOR\_PIN, INPUT);**

**pinMode(RIGHT\_SENSOR\_PIN, INPUT);**

**// Start with motors stopped**

**stopMotors();**

**}**

**void loop() {**

**int leftSensorValue = digitalRead(LEFT\_SENSOR\_PIN); // Read left sensor value**

**int rightSensorValue = digitalRead(RIGHT\_SENSOR\_PIN); // Read right sensor value**

**if (leftSensorValue == HIGH && rightSensorV**

**……………………………………………………………………………………………………………………**

**3 sensor line following**

**Truth table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sensor left** | **Sensor middle** | **Sensor right** | **Action** | **Motor left** | **Motor right** |
| **0** | **0** | **0** | **Left turn** | **Release** | **Forward** |
| **0** | **0** | **1** | **Right turn** | **Forward** | **Release** |
| **0** | **1** | **0** | **forward** | **Forward** | **Forward** |
| **0** | **1** | **1** | **Right turn** | **Forward** | **Release** |
| **1** | **0** | **0** | **Left turn** | **Release** | **Forward** |
| **1** | **0** | **1** | **omit** | **omit** | **omit** |
| **1** | **1** | **0** | **Left turn** | **Release** | **Forward** |
| **1** | **1** | **1** | **Left turn** | **Release** | **Forward** |

**// Define motor control pins**

**#define LEFT\_MOTOR\_PIN 9 // PWM pin for left motor speed**

**#define RIGHT\_MOTOR\_PIN 10 // PWM pin for right motor speed**

**// Define sensor pins**

**#define LEFT\_SENSOR\_PIN A0 // Analog pin for left IR sensor**

**#define CENTER\_SENSOR\_PIN A1 // Analog pin for center IR sensor**

**#define RIGHT\_SENSOR\_PIN A2 // Analog pin for right IR sensor**

**// Define motor speeds**

**#define MOTOR\_SPEED 255 // Max speed**

**#define TURN\_SPEED 150 // Speed for turning**

**// Function prototypes**

**void setup();**

**void loop();**

**void moveForward();**

**void turnLeft();**

**void turnRight();**

**void stopMotors();**

**void setup() {**

**// Initialize motor pins**

**pinMode(LEFT\_MOTOR\_PIN, OUTPUT);**

**pinMode(RIGHT\_MOTOR\_PIN, OUTPUT);**

**// Initialize sensor pins**

**pinMode(LEFT\_SENSOR\_PIN, INPUT);**

**pinMode(CENTER\_SENSOR\_PIN, INPUT);**

**pinMode(RIGHT\_SENSOR\_PIN, INPUT);**

**……………………………………………………………………………………………………………………**

* **In IR sensor 0 is white and 1 is black**
* **PID(personality integration and derivative)**
* **In line following robot atlest 5 sensor maximum 8 sensors should be used**

**What is the PID algorithm?**

**In simple terms, the PID algorithm regulates a process variable by calculating a control signal that is the sum of three terms: proportional, integral, and derivative. Hence its name. As a result, it can return a process variable into the acceptable range.**

**What is IR sensor array?**

**The IR sensor array is a device with 7 mounted infrared sensors. Each IR sensor is capable of detecting black and white colors. The array is capable of emitting sounds. The IR array can perform detection at proper distances. The infrared sensors transmitters/receivers have optimal operating range of 0-5CM.**