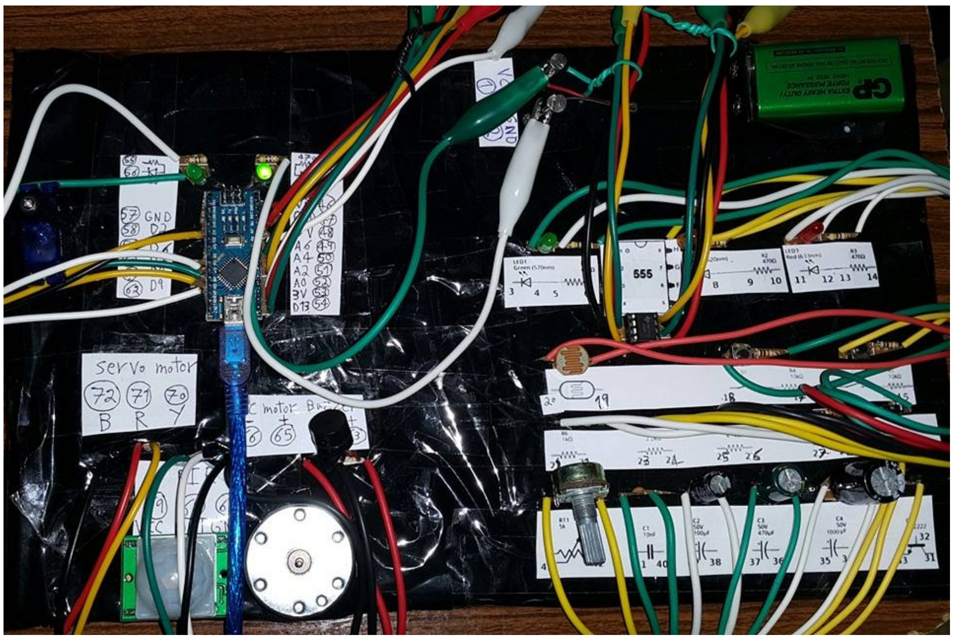
Teach Yourself Electronics !



(Model AL-AY-A-1)

Electric Circuits Learning Kit Explanation Booklet

Learn By Doing. Build A Working Circuit first then learn the theory behind it ! Build More than 15 Real Life Projects.

Start From the basics (turning on LEDs) to more advanced projects (programming a small robot) to a fully functional project (a motion-detector house alarm).

No Soldering Required. Suitable For Ages : 12 and above

(9V Battery Not Included)

Prepared By : Ahmed Mo’nis

Contents Index

Circuits Index

Introduction

So You Decided To Learn Electronics, Congratulations ! Hopefully This Guide Will Walk You Through The Different Circuits You’ll Be Building Soon. While Learning The Basics Of Electric Circuits Isn’t Something You Can Do Overnight, The Kit That Accompanies This Booklet Was Designed To Make Your Learning Journey As Simple As Possible While Maintaining All The Necessary Information That Will Carry You Later On In Case You Decide To Learn More On Your Own Or With The Help Of Others. As The Case With Any Learning Guide, Feel Free To Ask Any Questions To Your Instructors/Educators Or Send Me An E-mail at : ahmed.mo2nis@gmail.com. However, I strongly recommend That You Try To Search The Web Before You Start Asking Around As This Will Improve Your Skills Of Problem Solving And Increase Your Self Reliance. Now Let’s Begin Building Circuits !

License

How To Use This Booklet ?

If You look at the kit, you will find a lot of wires that has a big part at the end of it. These are called crocodile clips, by using these crocodile clips you can attach them to each other which I found to be an easier manner of connecting circuits than any other way. You simply follow the sequence of each crocodile wire as the numbers are ordered here in this booklet and that’s how you connect them to each other. With the exception of 1 and 2 which are points that have nails instead of crocodile clips, all other connections are just open a crocodile clip and attach it to the other, the same thing can be done when connecting to the 2 nails. Either way make sure to connect everything gently, you don’t want to damage the kit or tear up wires !

So obviously you can use the booklet to build the circuits in the kit. However, I have tried as much as possible to make this booklet useful for people who want to build their own circuits separately from the kit. The schematic diagrams (If you don’t know what it means don’t worry it will be explained in the next few lines) are provided for each circuit so that it can help you in case you find this booklet without the kit and you can learn from it by building each circuit on your own without the kit. However the purpose of the kit was to spare you just that so that you can find most of the basic circuits in one place. There are no per-requisites for you to know before reading the scientific material provided in this booklet. Some general knowledge of Maths and Physics would be helpful though. For example how to solve an unknown in an algebraic equation and the units of conversion (milli = 10^-3 ,micro = 10^-6 and k or kilo = 10^3 = 1000). If you don’t know what this is then I suggest you read about them a little on the Internet it won’t take much but you probably learned about them in school or you can simply understand them as you go along the booklet since they are not that hard. If you are a bit advanced and know some of the concepts discussed here feel free to skip some parts or just take a look at these circuits to catch up quickly. Either way make sure you don’t move from one concept to another, or from one circuit to the next without fully grasping it.

While I tried to simplify everything in this booklet, it's far far from perfect ! So don't hesitate if you find anything you don't understand to ask around any instructor or someone with a knowledge base in the area of electronics and electricity or e-mail me ahmed.mo2nis@gmail.com. But I advice you before asking around to try to search online and try to find the answer yourself. It will be more fulfilling and you will learn more. Either way take your time in learning.

Troubleshooting

PLEASE READ THIS PART CAREFULLY BEFORE PROCEEDING !

The Kit Is Composed Of Many Electronic Components That Are Connected To Crocodile Clips. To Build A Circuit You Follow The Order Of The Wires As Described In Each Circuit (For Example In Circuit 1 Order Of Wires is : 1-3, 2-6. So You Connect The Crocodile Clip At The Component Numbered 3 To The Number 1 -i.e. The Green LED To The VCC Red Wire-. And The Component Numbered 6 To The Number 2 -i.e. The 470 Ohm Resistor To The GND Black Wire). Note That Only Numbers 1 & 2 Are Not Connected To Crocodile Clips But Are Actually Metal Nails Connected To The 9V Battery Box. Also Note That Some Components May Be Interconnected With Each Other, This Is Done To Simplify Your Connections So That Your Circuit Wouldn't Be Overcrowded With Crocodile Clips And To Prevent Confusion. However Always Look At The Schematic Diagram To See How The Circuit Can Be Constructed Away From The Kit. Don't Worry If You Don't Know What The Term "Schematic Diagram" Means, It'll Be Explained In The Next Few Pages.

Besides The Kit And The Booklet You're Reading Right Now, You Will Be Needing A 9V Battery For The Circuits To Work. You Can Buy It From Any Super Market. For The Micro-controller /Arduino Related Circuits You Will Be Needing A PC Or A Laptop With Arduino Software Installed On It (See The Arduino Section For More Details). Obviously The Computer Needs To Have An Empty USB Hub To Be Connected To The Arduino Cable. If You Don't Have That, You've Got A Problem.

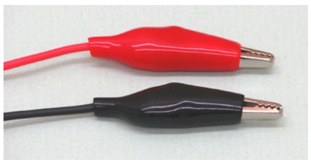
In Case Any Circuit Doesn't Work As Described Here Don't Get Frustrated Or Upset. Just Take A Deep Breath And Take Another Look At The Wiring Order (i.e The Order Of Wires). Start Asking Yourself These Questions : Did I Follow The Wiring Order Correctly ? Did I Make Sure That The 9V Battery Is Properly Inserted And Connected Properly Into The Battery Box Or Did I Accidentally Switched The Positive and Negative Pins ? Is This Battery New Or Used (Old) ? If It Isn't New Did I Test It On Something Else To See If It's Working Or Not ? If It Is New And Worked On Previous Circuits In The Kit Before The Current Circuit Ask Yourself : Could It Be Drained By Now ? Note That Some Battery Types Perform Better In Certain Projects Than Others (ex. With Buzzer). Always Make Sure To Disconnect The Crocodile Clips From Each Other By The End Of Each Circuit In Order To Prevent Unnecessary Battery Loss. If The Circuit Still Doesn't Work & You're 100% Sure The Problem Isn't From The Battery Or From Wrong Connections Then Check The Schematic Diagram And The Title (Name) Of The Circuit And What It's Supposed To Do. Search Online For Similar Circuits Or Ask Around Or Send Me An E-mail Since There Might Be A Possibility Of A Typo. In The Case Of An Arduino Related Circuit Not Working Check The Code, Make Sure The Code Was Verified And Uploaded To Your Board, Or Change The USB Port If Needed (See The Arduino Section For More Details).

Safety Notice

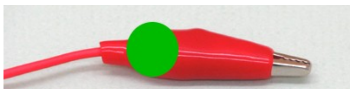
This kit is safe since its power source is only a 9 V battery meaning that there are no dangers of getting an electric shock from making a wrong connection so no need to worry. While you won't get electrocuted there is some chance of burning components by making a wrong connection which will result in the component being too hot for a while or even exploding which might be dangerous if your face was too close to the circuit so while you are working always make sure to put your safety first. And if you need to look closely while making a connection no problem but **DO NOT CONNECT** the battery while doing that !!! Don't insert the battery until the very end and only after you are 100% sure of your other connections. If you have a magnifying glass around use it instead of looking too close. Always make sure you follow the connection sequence described here and if you follow it correctly and something wrong happens with your circuit, check the schematic, if something is still wrong and you have revised your connections over and over many times then ask an instructor or e-mail me ahmed.mo2is@gmail.com

Circuit 1

If You Look closely next to the battery box there are 2 nails, they are labeled VCC and GND and they are numbered 1 and 2. Underneath them there are other components with labels and numbers as well, the only difference is they have wires attached to them with each wire having a big part at the end of it that looks like this.



Try holding this visible metal end and walk with your hand back wards through this wire until you reach and area where you can feel another metal part covered in plastic. It's probably the same place as this green circle in the following picture. If it's not exactly there keep searching with your hand until you feel it.

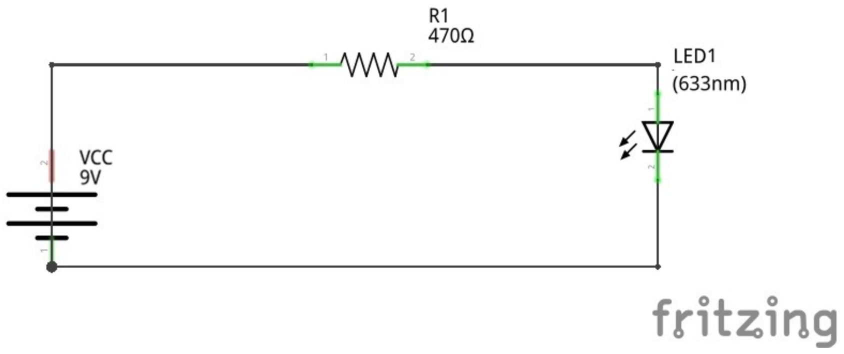


Press that covered metal part gently and you will notice that the visible metal part opens. It will look something like a crocodile mouth with zigzag metal parts resembling its teeth. These type of wires are called crocodile clips and they are used for connecting electrical parts just the same way you have opened it, only you need to close it by releasing your hand after you make sure it closes on another crocodile wire or a nail such as those labeled 1 and 2. This is how you are going to connect all the circuits in this kit.

Now take another look at the components under the battery box and the nails, notice the numbers each component is given and connect the crocodile clips to the nails according to the following sequence where each dash '-' represents a wire connection and each connection is separated by a comma ','. The connection sequence for circuit 1 is:-

1-3, 2-6

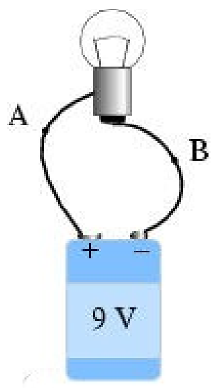
If you are sill confused and don't know what to connect yet don't worry, take a look at the following diagram which is used to draw circuits. It's called a schematic diagram. You may not understand how it works yet, you will in a few lines but for now it may help you as you can easily spot the similarities between some symbols on this diagram and some symbols under some numbers and labels and that should help you know which crocodile wire to connect to where in case you found the above numbers and labels not clear enough. For each circuit I will provide a schematic diagram to help you understand more.



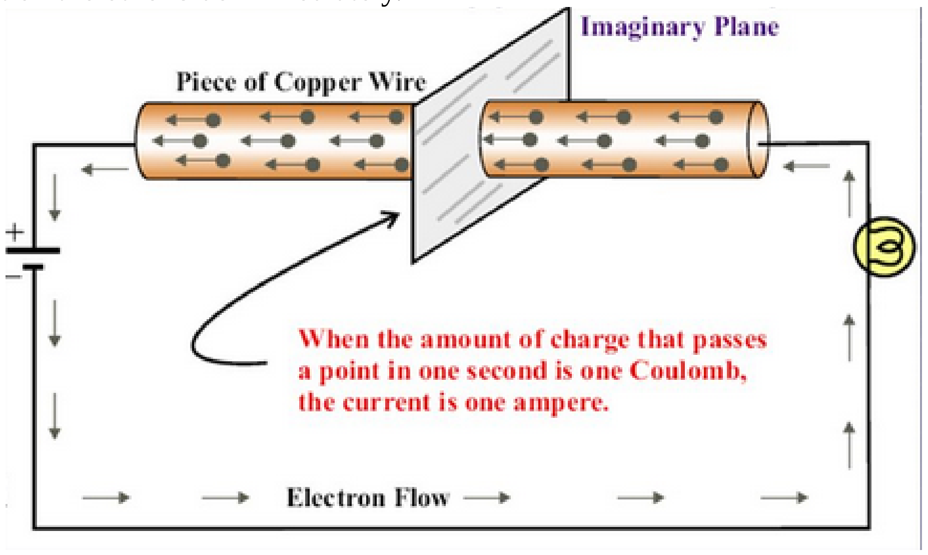
Basic Electricity

You’ve often heard the phrase “electrical current” and you may have wondered what is that ? Well, electrical current is the flow of electrons (small particles) in a wire.

Electrons flow when you have a “closed loop” - a path from the negative to the positive terminal of a battery. For example, When you connect a small light bulb to the Positive (+ve) and Negative (-ve) sides of a battery, you will get a closed circuit (a closed loop) where electrons can flow between the terminals and make the lamp shine. Inside the wires of this circuit, you will have electrons flowing.



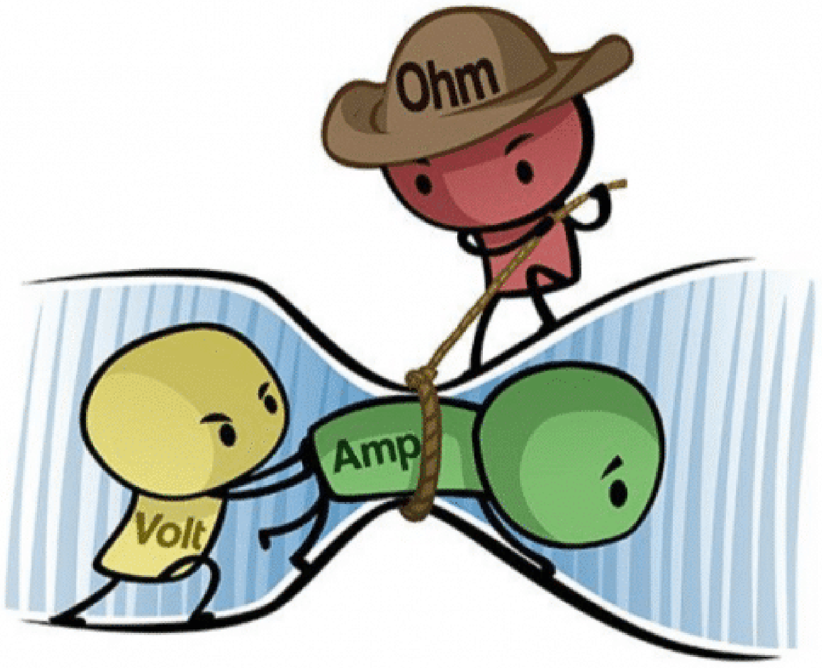
A Wire already has electrons in it. And When you connect a battery & make a closed circuit, they start to move. It’s Like a pipe filled with marbles. When you put a marble in on one side, another comes out on the other side immediately.


Imaginary Plane
Piece of Copper Wire
+
–
When the amount of charge that passes a point in one second is one Coulomb, the current is one ampere.
Electron Flow

“Electronics” control electrical currents by combining different components to enable different types of functions in a circuit.

But Does the current flow from the positive to the negative terminal of a circuit or is it the other way around ? It doesn’t really matter ! All you need to do when making your calculations is to decide on a direction and make your calculations based on that direction. No matter which direction you choose – the results will be the same. Current can flow in both directions, depending on if there are positive or negative charge carriers at work. In metals, the negative charge carriers are called electrons. They flow from the negative to the positive. So in a normal electric circuit based on metal wires, electrons will flow from the negative terminal to the positive. But the conventional way of talking about current direction is from positive to negative.

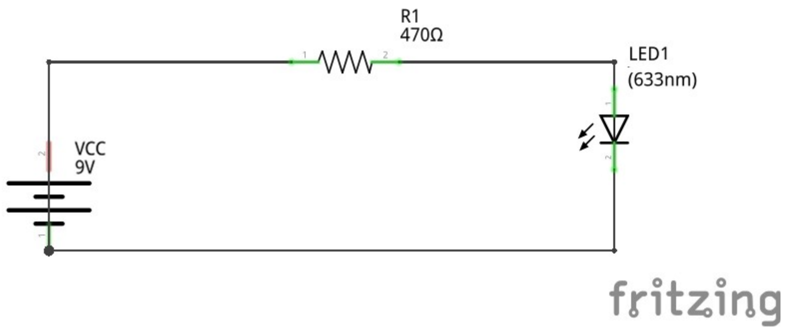
In a circuit, current is the flow of electrons (the amount of electrons running). Voltage is the electrical potential difference between 2 points (just like you measure your height between 2 points : the ground and your shoulders, voltage is always measured as a voltage difference between 2 points). Resistance is something that resists the flow of electrons. The relationship between I, the current (measured in Amps A), Voltage (measured in Volts V) and Resistance (measured in Ohms or Ω) is called Ohm's law. It can be formulated into this equation V = IR.



A component which the current flows through it (like wires) is called a conductive component while a component which doesn't allow current to pass through it (like the plastic covering the wire to protect humans from electric shock) is called an insulating component.

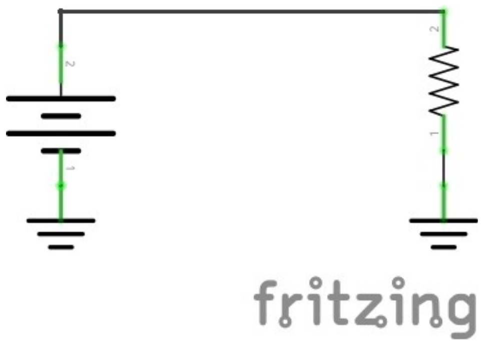
To make any electronic circuit, you start with a schematic diagram. A schematic is a drawing of a circuit. It tells you which components are needed and how to connect these components. You can design your own schematics or find them online. To read a schematic diagram you need to know :
1- The electronic symbols of the components 2-That the lines drawn between the components represent wires.

To be able to read schematics you must know the symbols of the components, among the most used symbols of components in circuit diagrams are: battery, capacitor, resistor, diode, transistor, Integrated Circuit, Logic Gates, Switches. We will be learning about each of these components and what does their symbols look like in the following circuits that we are gonna build, but for now let's take another look at the schematic for the simple circuit we have just built.

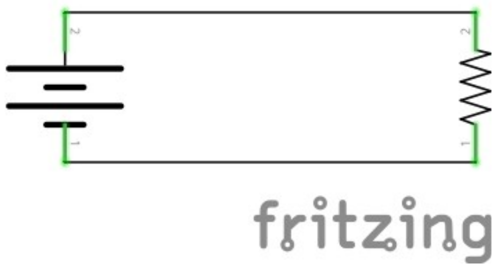


The One with the label VCC is the schematic symbol for the battery, it has the value of 9 Volts which the voltage of the battery. The one that looks like a Zig-Zag line with a label R1 is the symbol for a resistor and the one that looks like a triangle represents that small thing that gave a small green light when we followed the connection sequence above is called and labeled LED.

We mentioned measuring voltage from the ground as you would measure someone's height from their shoulders to the ground. But how do you represent ground in a schematic diagram ? There are 2 ways. One where ground symbols are used. The other one where you don't use ground symbols is the more popular one since it simplifies the circuit also because everyone knows how voltage is measured.



A schematic using ground symbols

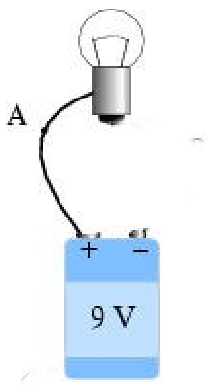


The same schematic shown without ground symbols

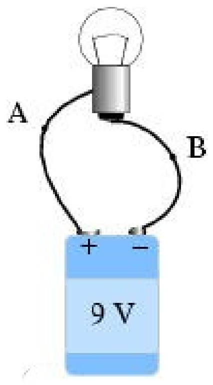
Note that ground is usually given the abbreviation GND in schematics

Open And Closed Circuits

A path between the positive and the negative terminals of the battery through the light bulb is called a closed loop and you need to establish a closed loop (closed connection) to make current flow in the circuit. An Open circuit will never work.



Open Loop
(Open Circuit)



Closed Loop
(Closed Circuit)

Short Circuit

A short circuit is a connection that was not meant to be there. For example, if you accidentally connect the plus to the minus of a battery, you will have a short circuit between plus and minus of the battery. Which is NOT GOOD. Some batteries will explode if you short circuit them.

If you built the circuit, most probably it's YOU who caused the short circuit ! Types of errors that cause short circuits are:

1- Connection Errors: check if you have put every component in the correct place before connecting the power. Compare the connections in the circuit diagram to those on your board.

2-Solder Bridges: If you are not careful when you solder, you can get something called a solder bridge between 2 pads on your board. For example, if you add too much solder and the solder flows over to a pad close by.

3-Pieces of Metal: Another reason for having a short circuit could be that a little piece of metal fell onto your circuit. For example, when you cut the legs of a component after you solder it, the pieces you cut off can easily go flying off in an unknown direction. If it lands on your circuit, it can give you a short circuit.

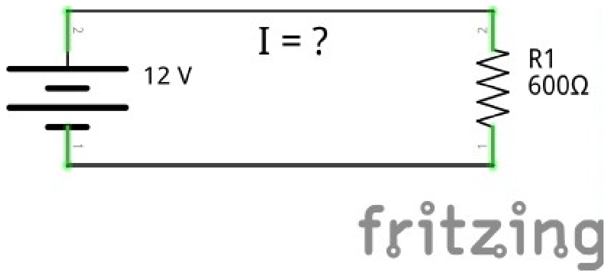
4-Conducting surface: This happens when you place your circuit onto something made of metal. If you're using a breadboard you won't have that problem. But if you have a circuit board with connections underneath your board, a metal surface will create a lot of short circuits for you. Note that it doesn't have to be a surface either. Maybe you have placed your circuit on top of a screw driver, some pieces of metal after clipping component legs, or something else.

5-Damaged Components: Another error can be a damaged component.

The best way to find short circuits is to use a multimeter to perform a continuity check. It will beep every time there is a connection between 2 points touched by the measurement probes. So if it beeps when the probes touch 2 points that were not supposed to be connected, you've found a short circuit.

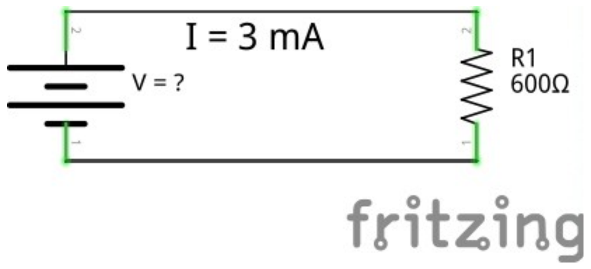
How To Use Ohm's law ?

Above we mentioned Ohm's law. Below is a very simple circuit with a battery and a resistor. The battery is a 12V battery and the resistance of the resistor is 600 ohm. How much current flows through the circuit ?



V = IR
I = V/R = 12/600 = 0.02 A = 20 mA

Below we have a circuit with a resistor and a battery. But this time we don't know the voltage of the battery. Instead we have measured the current in the circuit and found it to be 3mA

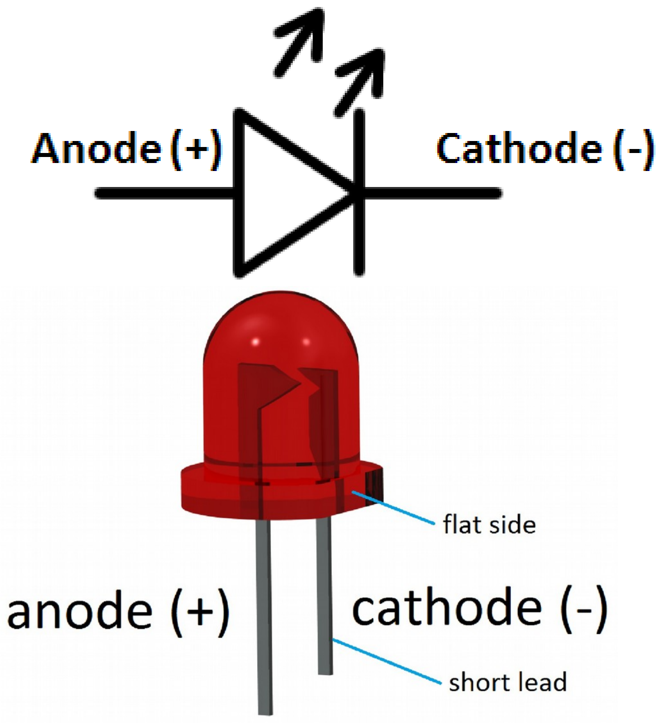


V = IR = 600 * 3 * 10^-3 = 1.8 V

For More complicated calculations, you can go to http://www.ohmslawcalculator.com/

What's a LED ?

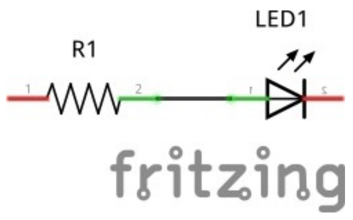
So far you have built a circuit that consists of a battery (voltage source), a resistor and a LED. But what exactly is a LED ? A LED is a small component that gives light, LEDs are cheap, simple to use and can be used to indicate if the circuit is working or not. LED stands for Light-Emitting Diode which means that a LED is a type of diodes (Diodes are going to be explained later) that emits light when it has current flowing through itself.



The LED has 2 pins : Anode and Cathode, the anode (positive pin) is the longest pin. This is the pin you connect to the most positive Voltage. The Cathode (negative pin) is the pin you connect to the most negative voltage. They must be connected correctly for the LED to work. A LED always has a resistor connected next to it. But Why is that ?

Current Limiting Resistor

A current limiting resistor is a resistor that is used to reduce the current in a circuit. A simple example is a resistor connected in series with a LED. You would usually want to have a current limiting resistor in a series (Series connection will be explained in a following section) with your LED so that you can control the amount of current through the LED.



If too much current is going through your LED, the LED will burn out too fast. If too little current is going through, the current might not be enough to light the LED. But How do we select an appropriate value of a current limiting resistor you might ask ? Well, depending on the type of the LED and how much current it needs to light up and how much voltage it drops (loses) you choose an appropriate value of resistance using Ohm’s law to calculate the value. Now that you’ve learned the importance of current limiting resistors, let’s move to building our next circuit !

Circuit 2

This time we are going to build a circuit that is exactly like circuit 1 we did earlier, BUT we are going to change one thing only : the resistance. So start by doing this connection sequence :-

1-3, 2-6

But that’s exactly the same connection for circuit 1 ! That’s correct we only did this to remind ourselves before making some changes. Now disconnect the previous connection and do the following connection sequence:-

1-3, 2-21, 22-4

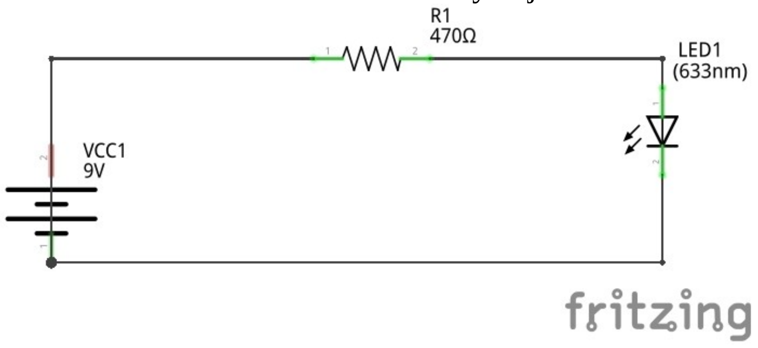
Note that points 4 and 5 are interconnected (they both are connected to the same crocodile clip) so don’t get confused when you look at the schematic and only see one crocodile clip over two points with two labels and two number, just connect the 2 crocodile clips together by opening both of them and closing them inside each other, notice anything different with the LED ? It’s brightness has changed !

Now let’s disconnect the previous connection and do one more connection sequence and see what happens:-

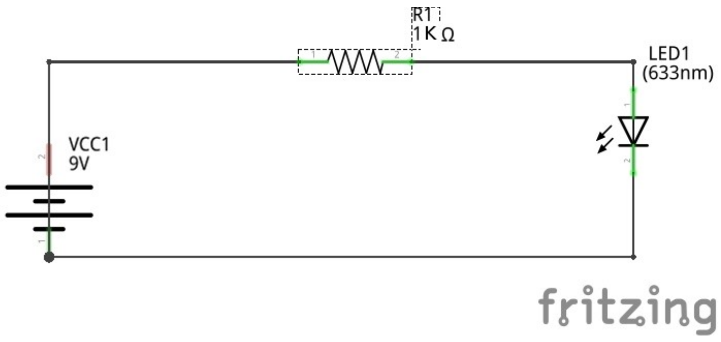
1-3, 2-23, 24-4

Again, the LED brightness has changed with the change of value of the resistance ! By now you already know some theoretical knowledge that makes you understand what just happened. The change in the value of the current limiting resistor causes a change on how the LED light looks.

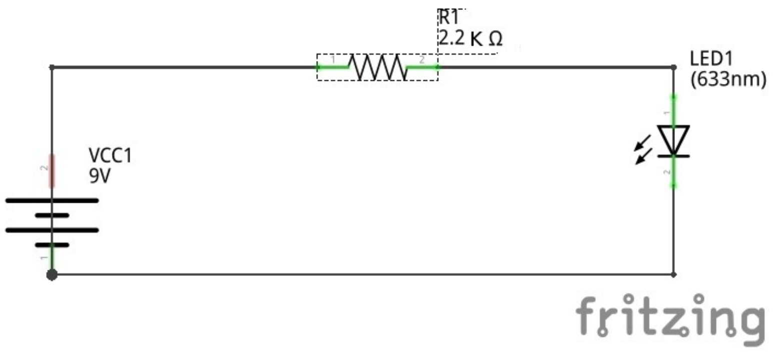
Now take a look at the schematics for the three circuits you just built.



In the first connection sequence we used a 470 Ω resistor which resulted in a certain brightness coming from the LED.



In the second connection sequence we replaced the previous resistor with another resistor with a value of 1 k Ω (k = 10^3 = 1000 which means 1000 Ω). This change of value of the resistor causes a change in the brightness of the LED



In the third and last connection sequence the resistor value is 2.2 k Ω. Again this time the change of the value of the resistor causes a change in the amount of current flowing into the LED which means a change in the brightness of the LED according to Ohm’s Law.

Resistor Types

By now you have understood how resistors with different values can affect a simple circuit. So resistors come in different values but they also come in different types based on the material they were made of and based on how they function. The different types have different properties which makes them useful for different applications. Some are very accurate, some can withstand high temperatures, some can handle high power and some are very cheap.

But How do you choose a resistor for your circuit ? The most important thing to consider is the power that the resistor needs to dissipate. Power dissipation in a resistor can be calculated with this equation:-

P = IV and since I = V/R from Ohm’s Law
P = (V/R) V = V^2 / R

The Unit For measuring power is Watts (W)

And based on the result of your calculation you get a resistor with a power rating which equals at least the value you calculated. But preferably more, a rule of thumb is to find a resistor with twice the power rating. The most common power value used is 250 mW resistor, since those are the standard ones. Usually you can just use the cheapest resistor you can find with the CORRECT power rating. But sometimes you need special types of resistors depending on your application.

Remember: Always see the schematic diagram before building any circuit, see if the schematics specify the resistor type or search online to find out.

Remember: Even though the resistor is a passive component that doesn’t do anything actively to your circuit, by selecting a resistance value you are using the resistor to control the voltages and the current in your circuit.

To learn more about resistors go to http://www.resistorguide.com/

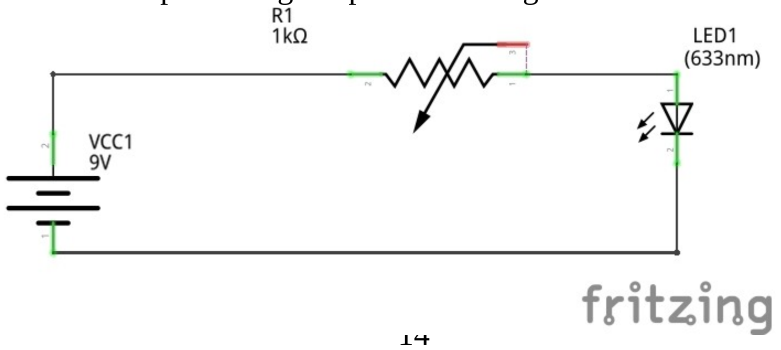
Circuit 3

Do the following connection sequence:-

1-3, 2-42, 43-4

After doing this connection sequence take a look at the component between points 42 and 43. You’ll see a long knob that you can move back and forth using your fingers. Try moving it in both directions gently using your fingers and notice what happens to the LED. The LED brightness change as you move the knob ! (If you don’t see it then you need to look very closely or move the knob till you can’t move it anymore in the end of one direction or the other). What is that ? Well that component is called a variable resistor and as the name suggests it’s a special type of resistor where the value of resistance can be changed by the user by turning the knob in one way or the other. Resistors of this type are used in applications such as controlling the volume of an audio system.

While it has many looks in real life, the variable resistor (sometimes called potentiometer) has a specific symbol when drawn in a schematic diagram which is just like a regular resistor symbol but with an arrow drawn over it representing the part that changes its value of resistance.



Circuit 4

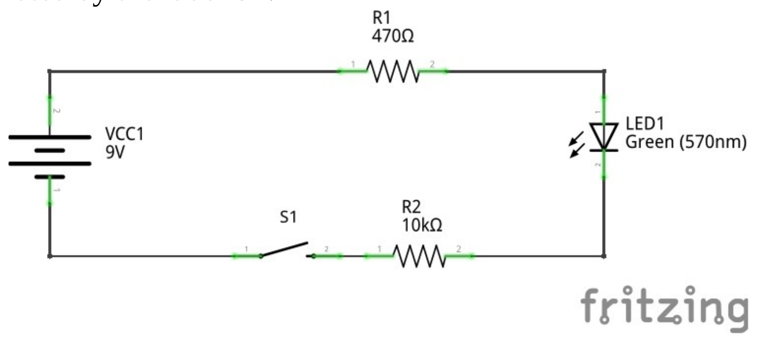
In this circuit we're going to use a switch. The switch isn't really an electronic component, it's mechanical but it's used in electronics all the time. Inside it's just a piece of metal that connects or disconnects 2 pins. For example, to turn the power of a circuit on or off.

Do the following connection sequence:-

1-3, 2-15, 16-6

After making sure your connection sequence is correct you'll notice there is a small button above the point numbered 16. Press this button several times and see what happens. As you press the button the LED lights up. When you stop pressing it, the LED turns off. As long as you are pressing the button the LED is turned on but the moment you release this button it turns off.

This switch has only 2 pins. It can only connect or disconnect one pin from the other. This is called a "single pole, single throw" (SPST). Sometimes it's also called a push-button switch. There are different types of switches depending on their functions and uses. In the following schematic, the switch symbol is denoted by the label S1.



You may have noticed something in this schematic, that there is a resistor connected to the switch. And you may have wondered why is that ? That's what we're gonna find out!

Pull-up resistors – By : Kaiha Tsusha

(Source:https://www.reddit.com/r/arduino/comments/2wzbe2/why_do_i_have_to_use_a_220_ohm_resistor_on_a_push/?st=jedkp9il&sh=9c195965)

Imagine yourself in a very dark room where you can't see anything, naturally you will use your hands to guide you around. When your hands aren't touching anything high or low, you don't know your location in the room. As soon as your hands touch something on the ground, then you know a lot more about your surroundings and where exactly you are in reference to the ground.

The same thing applies to an input pin in a digital circuit. When the input pin is not connected to anything (Open circuit, such as an unpressed momentary normally-open push-button), the electrical signal is basically guessing without any idea if it should read what is coming as a high input (button pressed) or low input (button not pressed) value.

When you connect the input pin to the ground (GND) or to the power source (VCC), the input pin immediately knows what to read and will give a good result.

To make this more obvious for the input pin, we want the circuit to be connected to the ground when the button is pushed and VCC when it is released (or vice-versa) but never guessing blindly.

If we use a good value for what is called a “pull-up” resistor on the near side of the button (like 10 k ohm), then the input pin will always read a weak but definite “high” signal when the button is open (not pressed). If the button is closed (pressed), there is 0 ohm to ground so this is a very strong definite “low” signal to the input. If you use a piece of wire instead of that pull-up resistor, then your button would directly connect ground to VCC, or short circuit the whole power supply and burn out something. The 10 k ohm resistor insulates the VCC from the ground while still giving that input a tenuous weak connection to VCC whenever it’s not connected more firmly to ground. The opposite of a pull-up resistor is a pull-down resistor.

Circuit 5

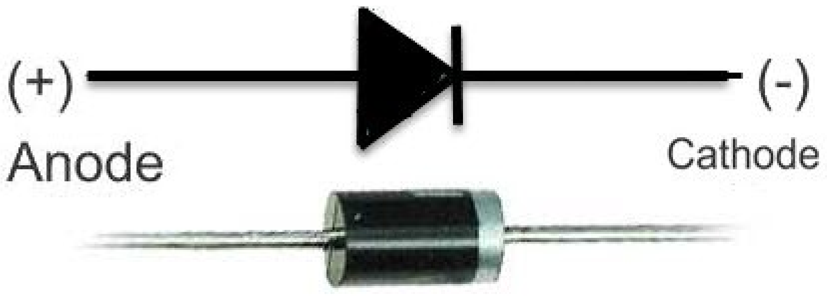
Follow this connection sequence:-

1-3, 2-30, 6-29

When you do this connection sequence, the LED lights up right ? Now, try doing this connection sequence and see what happens to the LED:-

1-3, 2-29, 6-30

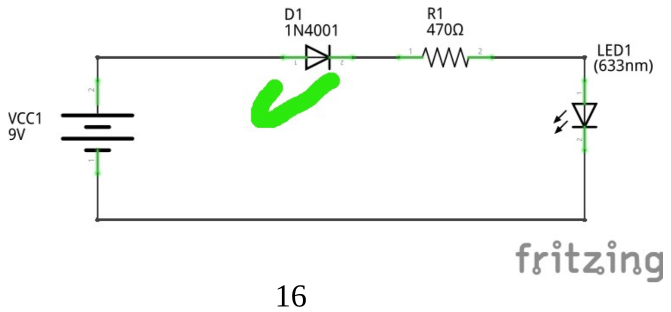
What happened ?! Nothing, nothing at all ! The LED didn’t light, but why is that ? it’s because we reversed the connection of this component between points 29 and 30. What exactly is that thing ? It’s called a diode and this is how its symbol looks in schematic diagram in comparison to what it looks like in real life.



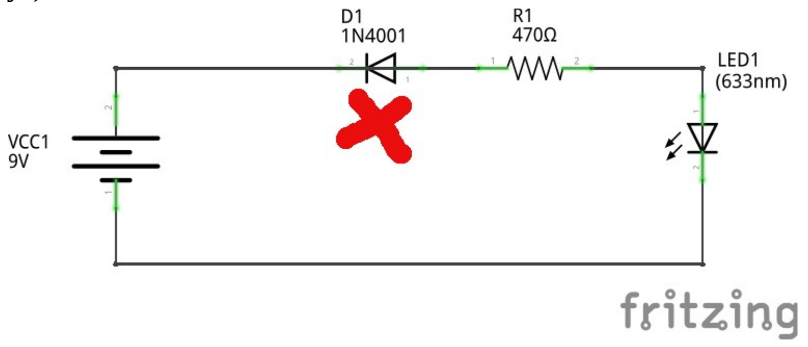
What Is A Diode ?

We mentioned briefly that the LED is a type of diode that emits light when current passes through it but what exactly is a diode ? A Diode is an electric component that conducts current in one direction and blocks current from flowing in the other direction.

How To connect a diode ?

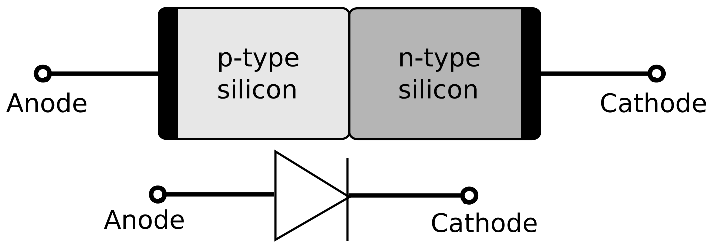


In the circuit above, the diode is connected in the right direction. This means current can flow through it so that the LED will light up. But what happens if we connect it the other way around (the wrong way!) ?



In this 2nd circuit the diode is connected the wrong way. This means that no current will flow in the circuit and the LED will be turned OFF.

But how does a diode work ? We mentioned before that there are conductive materials (like wires) and insulating materials (like plastic), there is also a 3rd type of materials which are called semiconductors, as the name suggests they can act as a conductive material sometimes and as an insulating material sometimes else, depending on many factors. This 3rd type of materials have been quite useful in the field of electronics. The diode is created from something called a PN junction. 2 semiconductor materials put together.



At the intersection of these 2 materials something called a “depletion region” appears. This depletion region acts as an insulator and refuses to let any current pass. When you apply a voltage from the positive side to the negative side, the “depletion region” between the 2 materials disappears and the current can flow from the positive to the negative side. However, when you apply a voltage in the other direction, from the negative to the positive side, the depletion region expands and resists any current flowing.

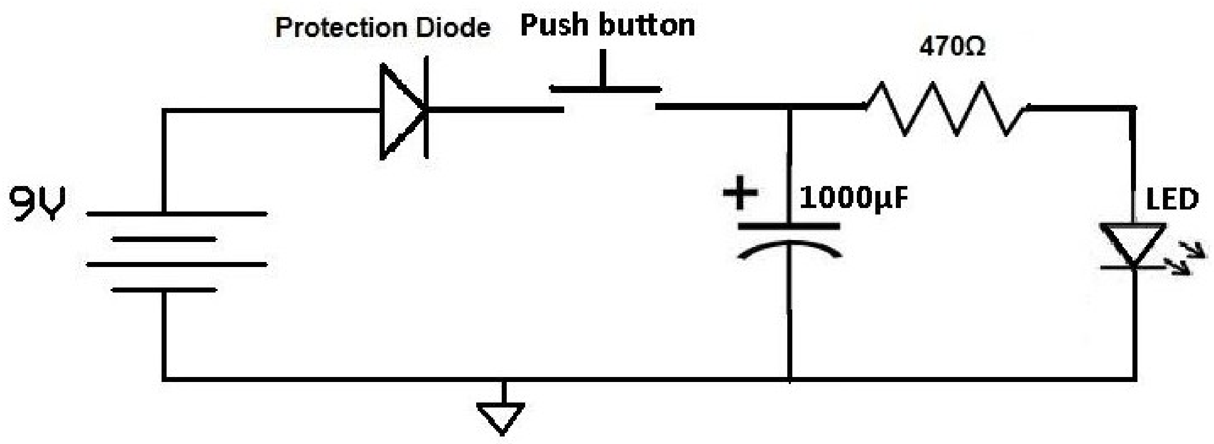
There are many types of diodes such as signal diodes, rectifier diodes (which are used in power supplies), zener diodes and light-emitting diodes (LEDs).

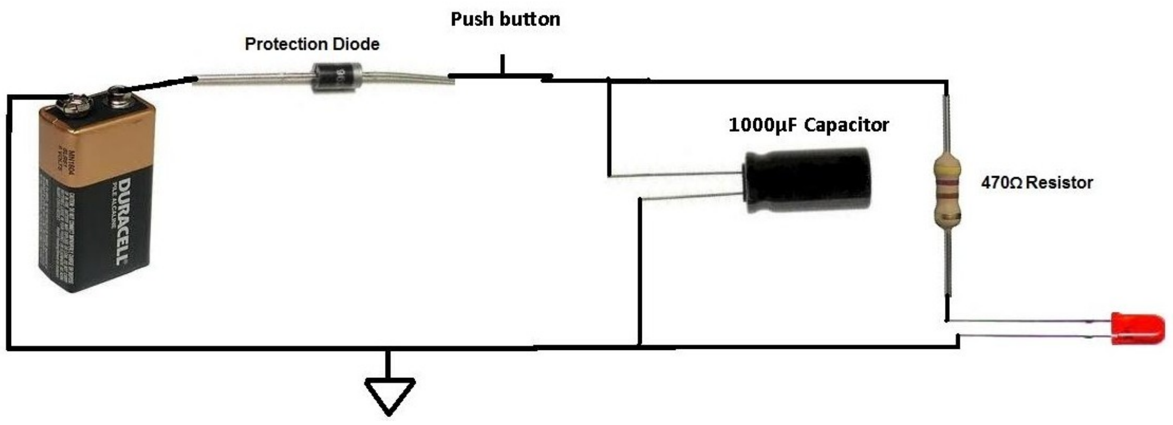
Circuit 6

Connect the crocodile clips according to the following sequence:-

1-35-3, 2-15, 16-34-6

This time might be a bit confusing with all the wires but REMEMBER: this time some connections require you to connect THREE CROCODILE CLIPS TO EACH OTHER. So do that carefully and make sure you have connected the right clips according to the sequence above before inserting the battery like we mentioned earlier. After you're done and the battery is connected properly look at the top of the LED and press the push-button and wait…..you'll notice that the LED slowly lights up! Try removing your finger from the push-button and you will see the light fading away slowly until the LED turns off completely. But how and why is that ? This is because of the component we have added into our circuit build this time. You've probably noticed it since it's kinda big and hard to miss. It's called the capacitor and we're going to learn everything about it in just a second but first let's take a look at how it looks like in a schematic diagram in comparison to what it actually looks like in real life.





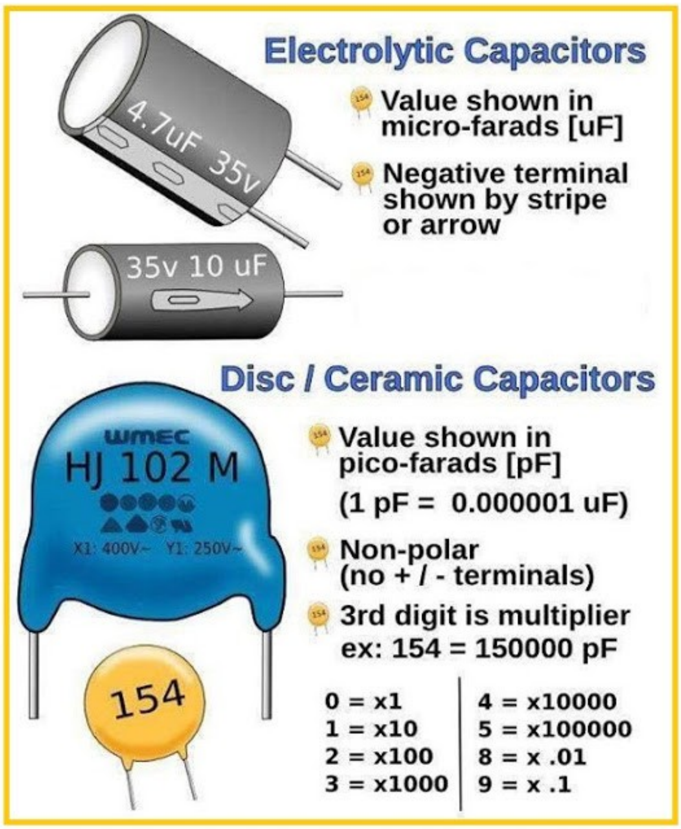
Note: This schematic has something additional than what we built which is a diode for protection. This is completely optional but its better to do that for more security when you go into building more advanced circuits in the future. But for the purpose of simplification, ignore it and treat it like a regular wire.

Now let's find out more about capacitors.

Capacitors

A capacitor works like a tiny rechargeable battery with a very very very low capacity. The time it takes to discharge a capacitor is usually only a split second. And so is the time to recharge it. So now you know that the capacitor can store a charge and it can release the charge when needed. In the circuit we have just built, when we pressed the push-button, the current passed on the capacitor before going to the LED which made the LED light more slowly. However, when we remove our finger from the push-button the capacitor (which stored a charge when current passed through the circuit before) discharged the charge it had to the LED which made it appear as if the LED took some time to turn off since it had some light left in it from the discharging of the capacitor. If you connected the other capacitors in this kit instead of the 1000 micro F (or Farad which is the unit for measuring capacitance), the only difference will be the time it takes to charge or discharge which would be observable on the time it takes the LED to be completely bright or completely dark.

There are different types of capacitors, the following picture is a handy guide on how to differentiate between them.

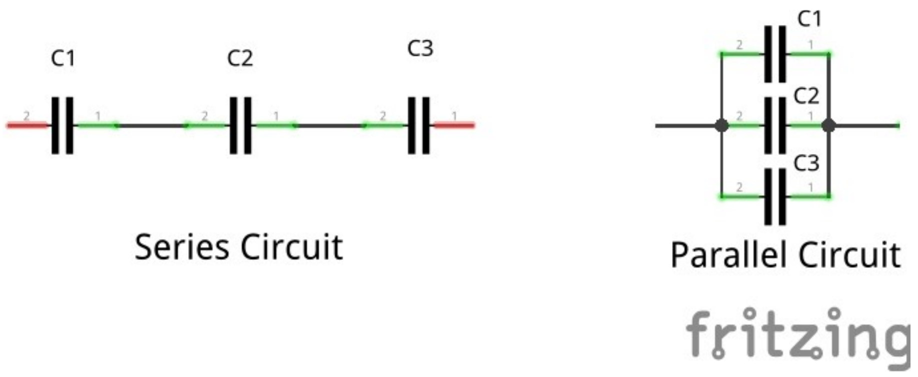


A capacitor is commonly used for filtering. But what is filtering? Lets look at an example : Many alarm clocks are powered by the electrical outlet on the wall in a house. Sometimes, the power goes down. Most alarm clocks have a backup battery that will power the alarm clock until the power comes back on so that the time is not reseted. In electronic circuits capacitors can be used in the same way. If you have a circuit with a micro-controller (we will discuss micro-controllers in a following section) running a program, if the voltage for the micro-controller drops for only a split second, the micro-controller restarts and you don’t want that. By using a capacitor, the capacitor can supply power for the micro-controller in the split second that the voltage drops so that the micro-controller doesn’t restart. This way it will filter out the “noise” on the power line. This type of filtering is called “decoupling” and a capacitor used for that purpose is called a “decoupling capacitor”. It’s also called a “bypass capacitor”.

Now before we continue building more circuits, we need to expand our theoretical knowledge a little bit.

Series And Parallel Circuits

In electronics. We can find both series and parallel circuits

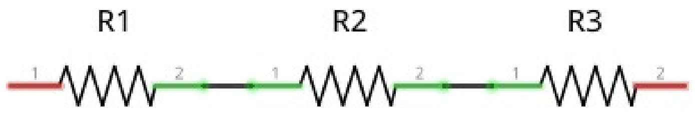


A series circuit is a circuit where the components are connected in a consecutive chain. This gives the current only one path to take.

A parallel circuit is a circuit where the components are connected parallel to each other. So the current will flow in several paths.

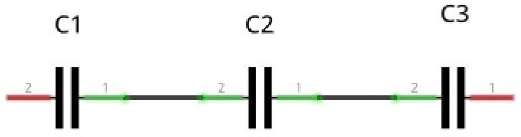
Often a circuit is a mix of series and parallel circuits.
The current in a series circuit is equal everywhere in the path. The same amount of current flows through all the components. To find the current use Ohm’s law V=IR.

If you have several resistors connected in series, it’s very easy to find the total resistance. All you have to do is to find the sum of all the resistance values.



Rtotal = R1 + R2 + R3

Capacitors connected in series are a little more complicated to calculate

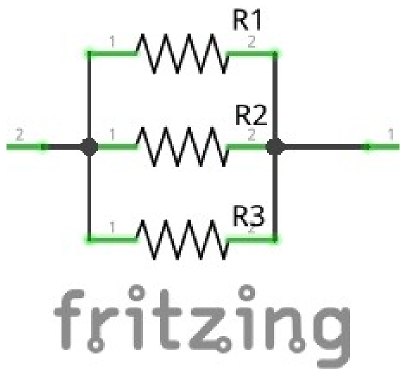


1/Ctotal = 1/C1 + 1/C2 + 1/C3

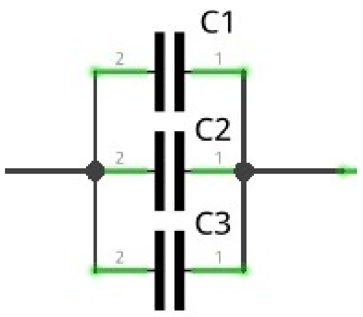
To calculate the resistance and capacitance in a parallel circuit just switch the equations from the series circuits.

The current in parallel circuits can be different from path to path. To find the current of a path, use Ohm's law V=IR

To calculate the resistance in a parallel circuit, you use the following equation
1/Rtotal = 1/R1 + 1/R2 + 1/R3



To calculate capacitance in a parallel circuit, simply take the sum of all the capacitance values



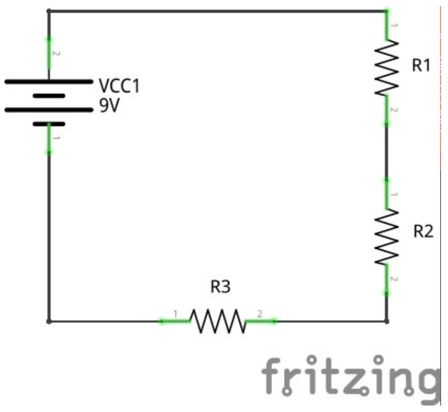
Ctotal = C1 + C2 + C3

Kirchhoff's Law Of Voltage And Current

Kirchhoff's law of current and voltage are 2 laws that are useful when you are working with circuits. Knowing them will make it much easier to understand circuit diagrams, design electronics, repair electronics, and everything related to electronics. Understanding these laws helps you when looking and understanding circuit diagrams.

Kirchhoff's Voltage Law (KVL) says that if you add all the voltage drops (losses) in a circuit you get the voltage of the power supply

Let's say you have a 9 Volt battery connected to 3 resistors in series. If you measure the voltages over the components the sum of all them will be 9 Volts



VR1 + VR2 + VR3 = 9V

Kirchhoff's Current Law (KCL) says : All the current going into a node is equal to all the current that goes out from the node. A simpler way of saying it is : "What goes in must come out".

Examples of Kirchhoff's Current Law:
- The current that flows into a circuit must come out from the circuit
- The current that flows into a resistor must come out of the resistor
- The current that flows into 4 resistors in parallel must come out of the 4 resistors in parallel

How does this make it easier for you to understand what is going o in a circuit ? When you know this, you can simplify a lot. If you have a big circuit with lots of components in parallel and series mixed, it can be hard to find the individual currents. But sometimes it's good enough to just know that if 500 mA goes into this section of the circuit, 500 mA will come out of it.

Circuit 7

Connect the crocodile clips according to the following sequence:-

1-31, 2-6, 3-33, 32-hand 1, 31-hand2

Note that this time there is a component that has THREE crocodile clips attached to it so only two of them are on the same line.

BUT what is hand1 and hand2 you might ask ? Simply, you will leave the crocodile clip at point 32 (at the top of the component which has three clips attached to it) unconnected and after you finish connecting all the other clips you touch the clip at point 32 and the clip coming from point 31 (which is connected to point 1) with your hands, one hand touching each clip. Look at the top of the Green LED and watch what happens.

When you held one crocodile clip in between your fingers and touched the other one with your fingers from your other hand, the green LED suddenly turned on ! But Why and how did that happen ? And what is that component which has three pins attached to three crocodile wires ? Let's find out.

But first, congratulations you've just used a transistor to build this circuit ! Yes that three-legged component is called a transistor. You might have heard about transistors before since they are the cornerstone of modern electronics. Without transistors we wouldn't have things like : TVs, Computers, Mobile Phones,..etc. But what exactly is a transistor and how does it work. In order to get a good understanding of how transistors work we need to go back in time before the invention of transistors and understand another component that was and still used today. The Relay.

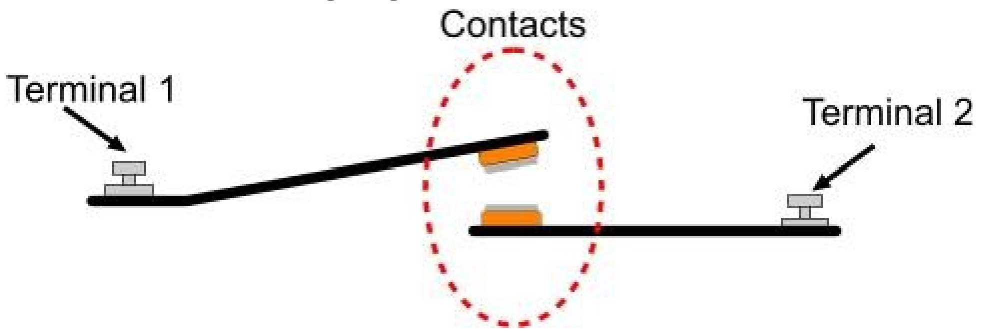
How Relays Work ? By : Giorgos Lazaridis

(Source: http://pcbheaven.com/wikipages/How_Relays_Work/)

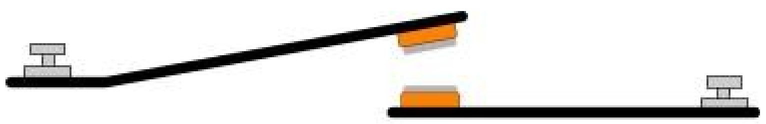
A relay is an electromechanical switch. It works in the same way as a manual switch that people use to turn on/off things, but instead of having to manually push the switch, you apply power to it to change the switch position. A relay consists of an electromagnet and a mechanical switch. An electromagnet is a simple device made up of a wire wound in a coil around a core of ferromagnetic material such as iron.

Basic Relay Operation:

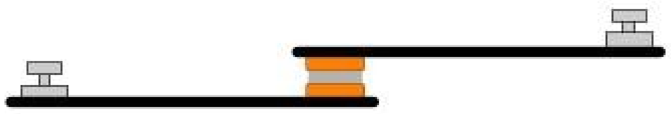
Each relay has 2 mechanical parts inside. The 1st part is the contacts of the relay. The contacts operate similarly to the contacts of a simple switch or push-button. You should consider the contacts as a pair of metals like the following diagram



The 2 terminals operate as a switch. When the contacts are 'in contact' then the current flows from Terminal 1 to Terminal 2. There are 2 types of contacts: The Normal Open (NO) and The Normal Closed (NC). The NO is a contact like the one showed in the previous diagram. When the contact is still, then no current flows through it (because it's an open circuit, not a closed one). On the other hand, a NC contact allows the current to flow when the contact is still.

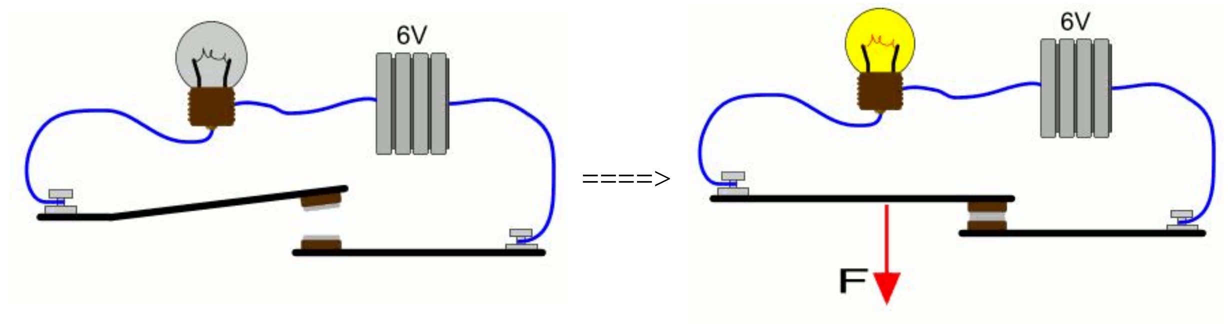


A Normal Open contact (NO)

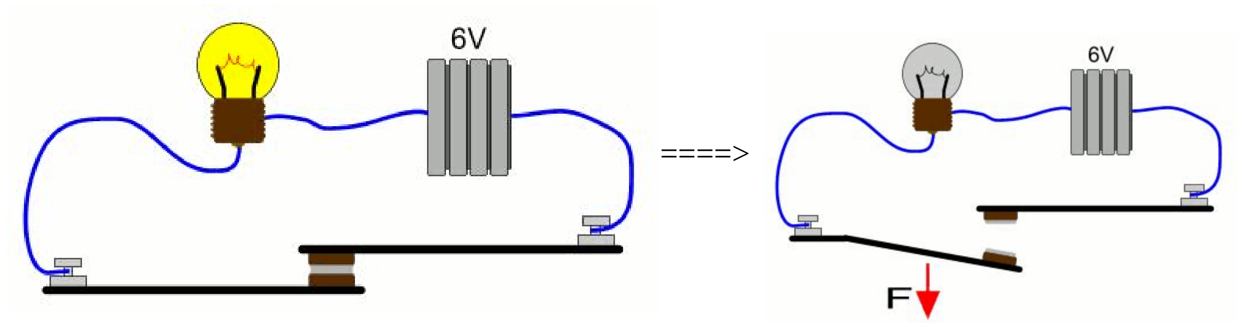


A Normal Closed contact (NC)

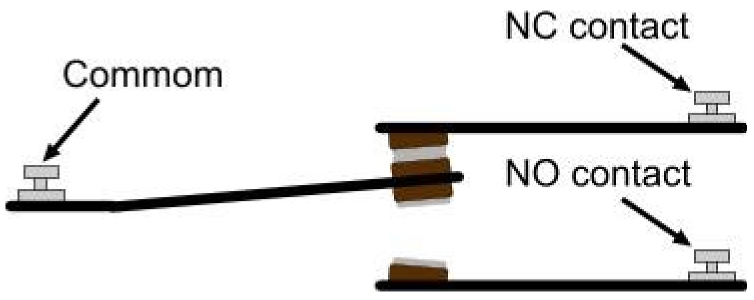
You may notice that the NC contact is turned upside down compared to the NO contact. This is done in purpose. This way, both contacts (NO and NC) will change state if a force is applied to the left metal heading from UP to DOWN. The following diagrams show how a NO contact operates by lighting a light bulb:



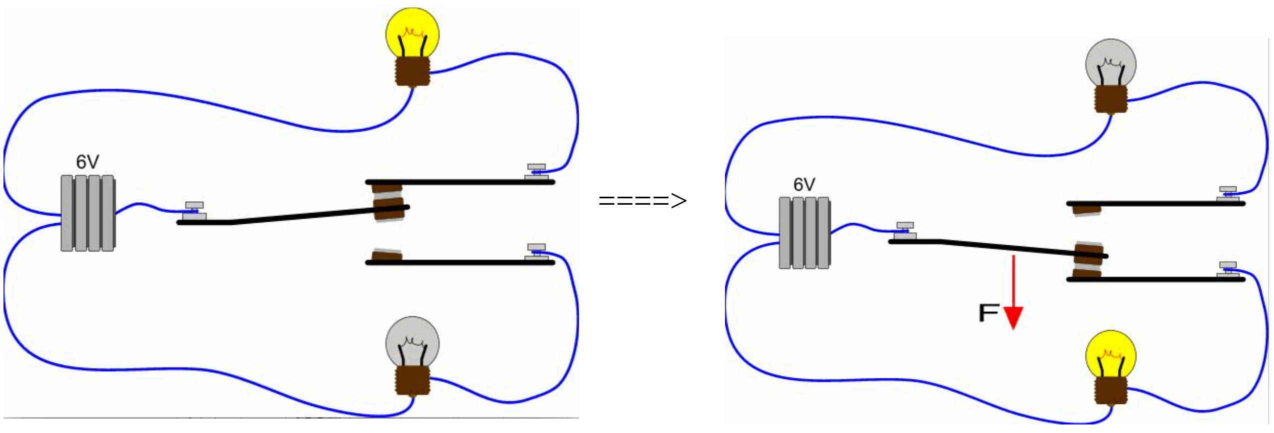
As for the NC contacts, it works exactly opposite the NO contacts. Just like the following diagrams:



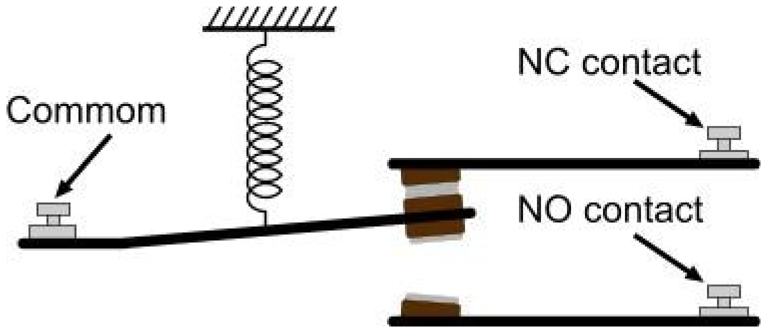
A relay may have a combination of the above contacts such as the following diagram



In this case, there is a 3rd terminal called 'common'. The NO and NC are referred to the common terminal. Between the NC and the NO contact, there is no contact at any time. The following diagram shows how this pair works

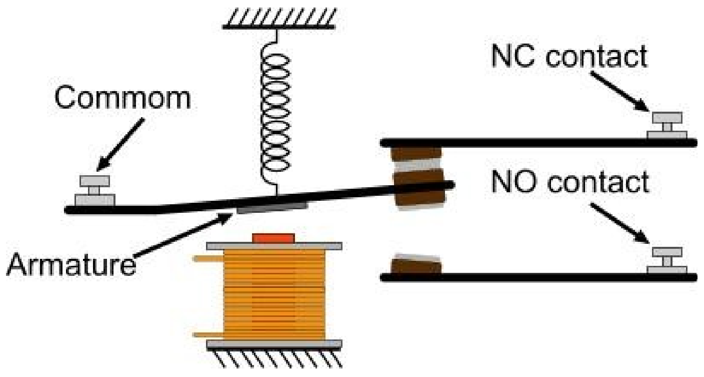


So we have the NO contact and the NC contact. But which state is considered as normal? Inside the relay there is a spring which defines the normal position of the common contact. In the previous diagram there was a F force applied to the common terminal and at the other instance it wasn't applied to the common. HOWEVER, there is another force that pulls the contact UPWARDS and this force is applied ALL the time. This force comes from the spring like the following diagram:

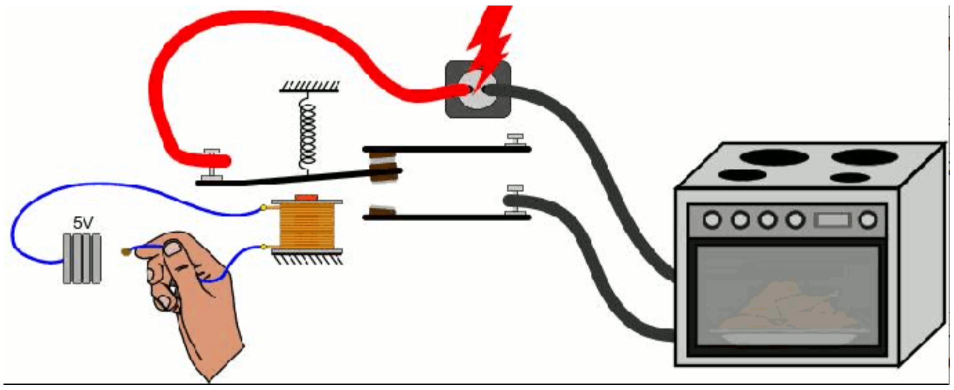


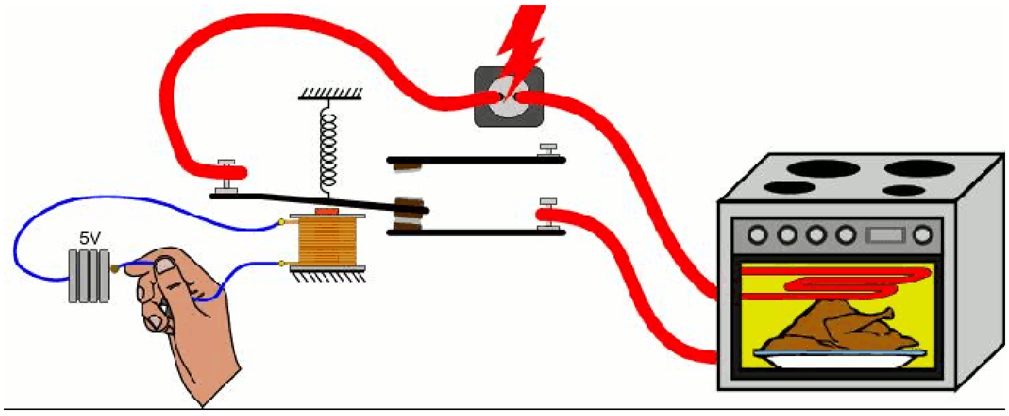
Now you can see who is pulling the common terminal UP all the time. So the spring defines what is the Normal State, and thus defines which contact is the NO and which is the NC. In other words, the normal state is defined as the state that there is no other force applied to the common terminal except the one from the spring.

What moves the common contact of the relay? This is the last part of the relay operation. The device that forces the terminal to move is the electromagnet we mentioned before. A coil is placed right under the contact. When current is flown through this coil, a magnetism is created. This magnetism can overcome the force of the spring and can pull the contact towards it, thus it changes its position. And due to the fact that the contact is usually a small piece of metal not capable to be pulled by the electromagnet, another piece of metal is attached to the common. This piece of metal is called 'Armature'. The following diagram is the complete illustration of the basic relay:



Now imagine that someone wants to control a 220V 1 k Watt load with a command that comes from a 5 V battery. A load-Relay should be used for this application. The coil of the relay is driven with the 5 V. The contacts from this relay (NO) will be connected in series with the power supply of the load, therefore the load will only operate when the relay is actuated. In the following example, we are going to turn on an electric oven using a relay:





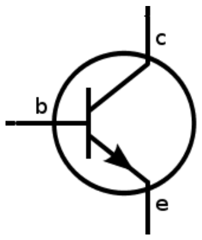
While we are not using any relays in this kit, it was very important to understand them to understand transistors. So what are transistors and how do they work ?

How Transistors Work ? By : Øyvind Nydal Dahl

(Source : https://www.build-electronic-circuits.com/how-transistors-work/)

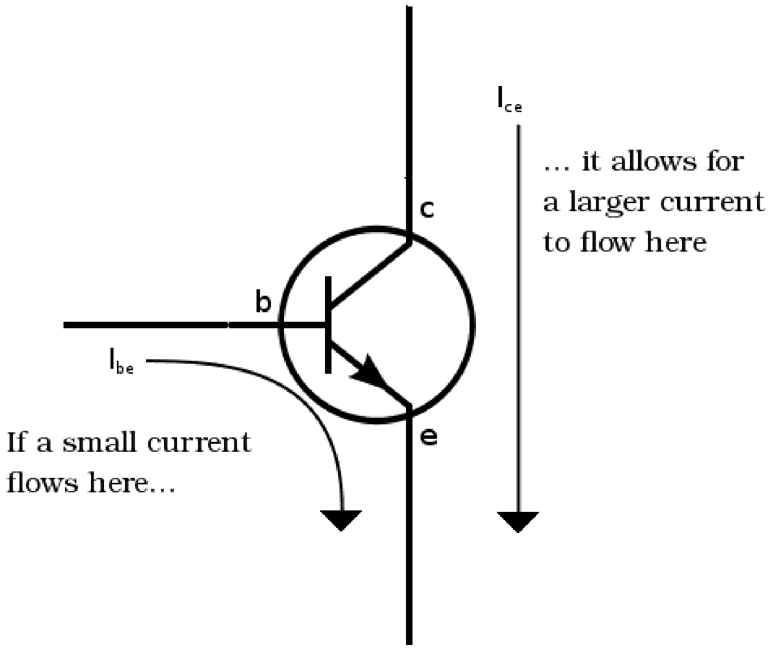
The transistor is like an electronic switch. It can turn a current on and off. A simple way you can think of it is to look at the transistor as a relay without any moving parts. A transistor is a similar to a relay in the sense that you can use it to turn something ON and OFF.

There are different types of transistors. A very common one is the “bipolar junction transistor” (BJT). It has 3 pins: Base (b), collector (c) and emitter (e). And it comes in two versions: NPN and PNP. The schematic symbol for the NPN looks like this:-



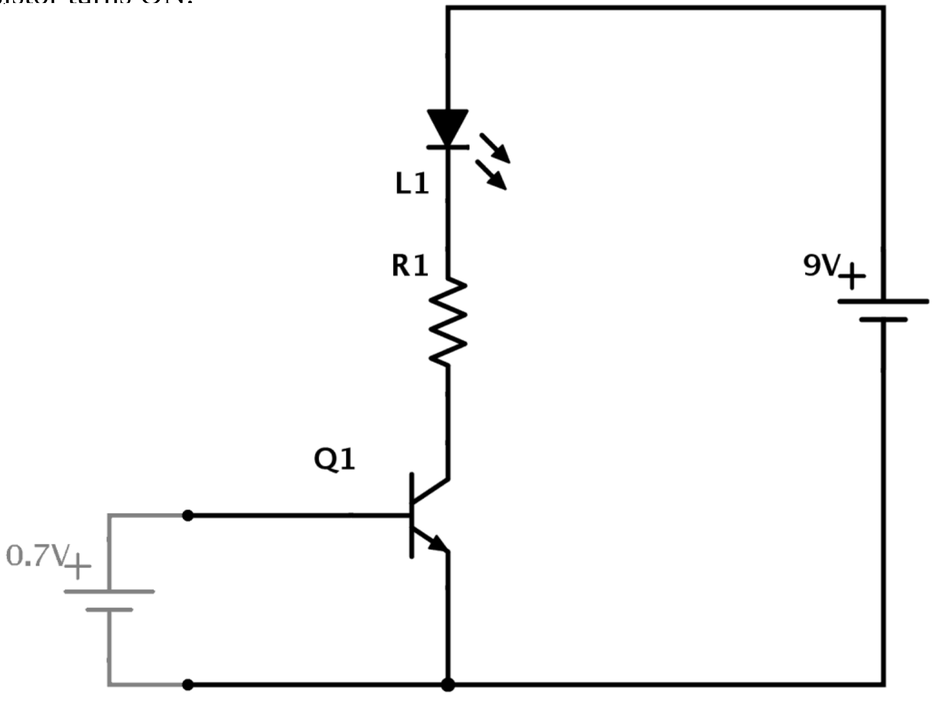
The transistor works because of something called a semi-conducting material. Semi-conductors have distinct electrical characteristics. They are of high electrical resistance, higher than typical resistance materials but still of much lower resistance than insulators.

A current flowing from the base (b) to the emitter (e) ‘opens’ the flow of current from the collector (c) to the emitter (e)



In a standard NPN transistor, you need to apply a voltage of about 0.7 V between the base (b) and the emitter (e) to get the current flowing from base (b) to emitter (e) . When you apply 0.7 V from base (b) to emitter (e) you will turn the transistor ON and allow a current to flow from collector (c) to emitter (e).

In this example you can see how transistors work. A 9 V battery connects to a LED and a resistor. But it connects through the transistor, this means that no current will flow in that part of the circuit until the transistor turns ON.



To turn the transistor ON you need to apply 0.7 V from base (b) to emitter (e) of the transistor. Imagine you have a small 0.7 battery (In a practical circuit you would use resistors to get the correct voltage from whatever voltage source you have).

When you apply the 0.7 V battery from base (b) to emitter (e), the transistor turns ON. This allows current to flow from the collector (c) to the emitter (e) and thereby turning the LED ON.

We use transistors in almost all electronics and it's probably the most important component in electronics.

If you want to design circuits with transistors, you need also to know about PNP transistors. For example, PNP can be used in circuits that automatically turn on a light when it gets dark.

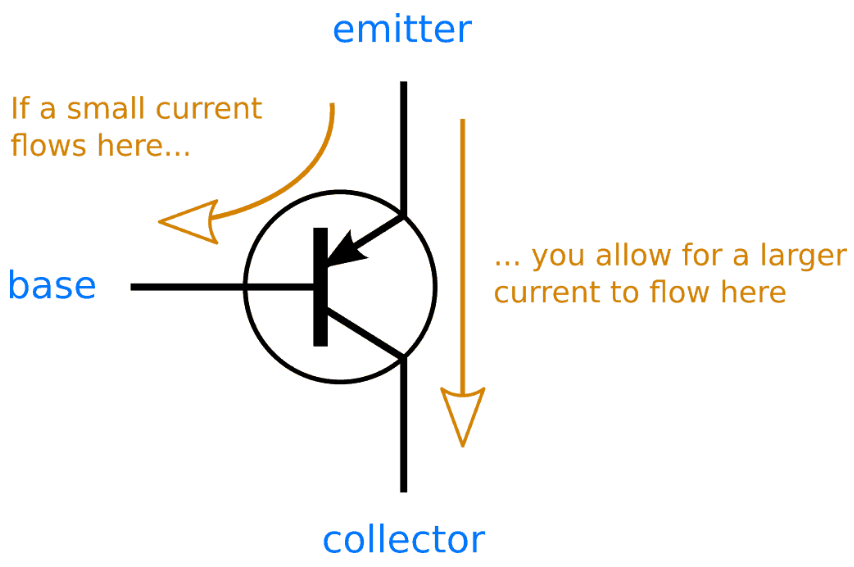
If you understood the NPN transistor, it will be easier to understand the PNP transistor. They work pretty much in the same way, with one major difference: The current in the PNP transistor flows in the opposite direction of the current in the NPN transistor.

How PNP Transistors work ? By : Øyvind Nydal Dahl

(Source: https://www.build-electronic-circuits.com/pnp-transistor/)

The PNP transistor has the same leg names as the NPN : Base, Emitter and Collector.

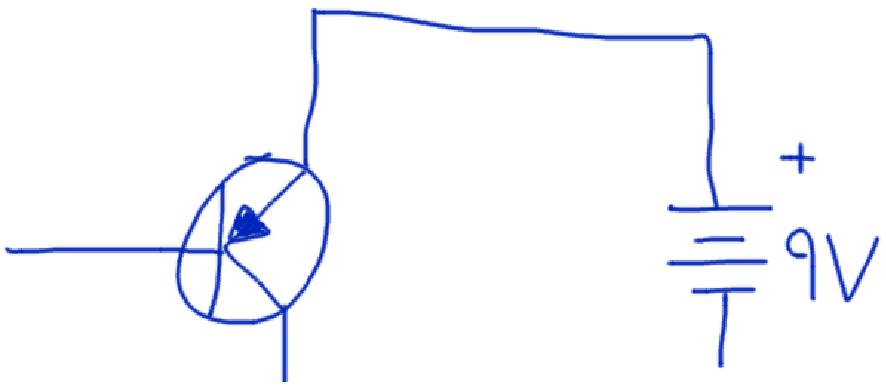
A PNP transistor will "turn on" when you have a small current running from emitter (e) to base (b) of the transistor, "turn on" means that the transistor will open up a channel between emitter (e) and collector (c). And this channel can carry a much larger current.


emitter
If a small current flows here...
base
... you allow for a larger current to flow here
collector

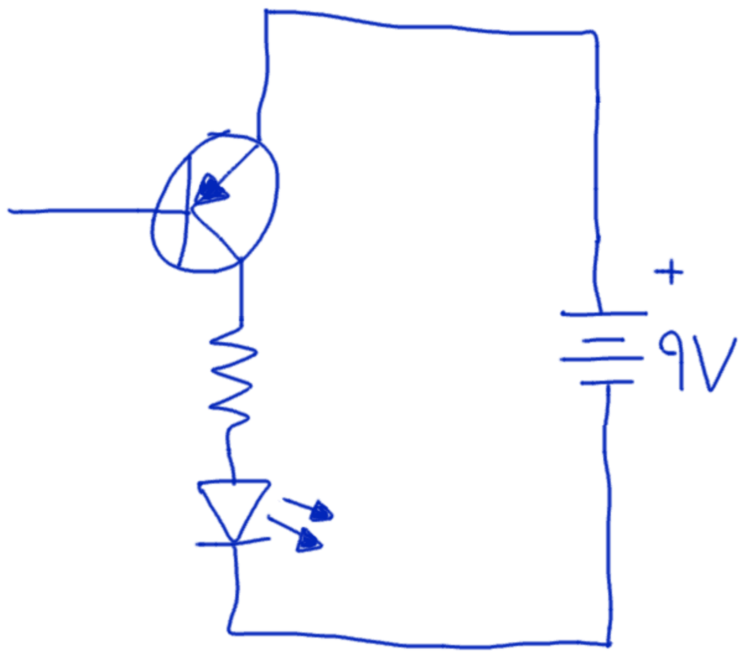
To get current running from emitter (e) to base (b), you need a voltage difference of about 0.7 V. Since the current goes from emitter (e) to base (b), the base (b) needs to be 0.7 V lower than the emitter (e). By setting the base (b) voltage of a PNP transistor to 0.7 V lower than the emitter (e), you "turn the transistor on" and allow for current to flow from emitter (e) to collector (c).

Let’s see how to create a simple PNP transistor circuit, with this circuit you can turn on a LED when it gets dark.

A-Step 1: The Emitter: to turn on the PNP transistor, you need the voltage on the base (b) to be lower than the emitter (e). For a simple circuit like this, it’s common to connect the emitter (e) to the plus from your source (VCC). This way, you know what voltage you have on the emitter (e).



B-Step 2: What you want to control : When the transistor turns on, the current can flow from the emitter (e) to the collector (c). So, let’s connect what we want to control: A LED, since a LED should always have a resistor in series with it therefore add a resistor as well (a current limiting resistor)



You can replace the LED and resistor with whatever you want to control.

C-Step 3: The Transistor Input: To turn on the LED, you need to turn on the transistor so that the channel from emitter (e) to collector (c) opens. To turn on the transistor you need to get the voltage on the base (b) to be 0.7 V lower than the emitter (e), which is 9 V – 0.7 V = 8.3 V. For example, you can make the LED turn on when it gets dark by using a photoresistor (also known as LDR and we will explain it in the next section).

Circuit 8

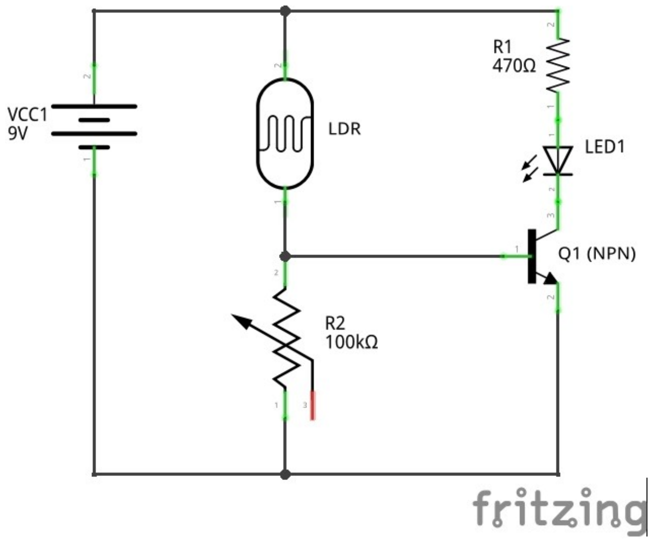
Do the following connection sequence:-

1-21, 2-6, 22-20, 3-19

As soon as you finish connecting the wires correctly you will see the green LED lighting, now try moving your finger above the component between the two points 19 and 20. Now try touching that component by your finger until your finger completely is above that component, what happened ? The LED turns OFF ! Now move your finger away again, the LED turns On again ! Try Moving your finger slowly away from that component and slowly above it and observe the changes in the light of the LED. But what is that thing that affects the circuit like that ? That component is called a Light-Dependant Resistor or simply a LDR.

So you've used a LDR in your circuit, you are probably wondering right now about what is this thing. LDR stands for Light Dependant Resistor. This is a specific type of resistors, like the variable resistor (potentiometer) you used before in Circuit 3 it has a variable resistance but unlike the potentiometer its resistance varies depending on the light. You've tried using different intensities of light from hight brightness to total darkness and you've seen how the light coming from the LED varied in each of these intensities. Below is a further explanation of how this circuit works.

This LDR circuit diagrams shows you how to make a light detector. A LDR or "Light Dependant Resistor" is a resistor where the resistance decreases with the strength of the light. Please note that -like many symbols- the LDR schematic symbol may be drawn in different ways in some circuits so if you ever feel confused when reading a schematic diagram perform an online search to see if the symbol stands for something you know.



Light Dependant Resistors (LDRs) are also called photo resistors. They are made of high resistance semi-conductor material. When light hits the device, the electrons gain energy from the light which makes them jump into the conductive band and therefore conduct electricity.

The LDR circuit works like this: when it’s dark, the LDR has high resistance. This makes the voltage at the base (b) of the transistor too low to turn the transistor ON. Therefore, no current will go from the collector (c) to the emitter (e) of the transistor. All the current will instead pass through the LDR and the potentiometer. When it’s light, the LDR has low resistance, this makes the voltage at the base (b) of the transistor higher. High enough to turn the transistor ON. Because the transistor is turned on, current flows through the transistor. It flows from the positive battery terminal, through R1, the LED and the transistor down to the negative battery terminal. This makes the LED light up.

The resistor R1 controls the amount of current going through the LED. What if you want to power the circuit with something other than a 9 V battery ? Then you need to change the resistor value to get the right amount of current flowing through the LED. The variable resistor R2 is used to change the trigger point for the LED. That is, how much light that is needed for the LED to turn ON and OFF. You can probably pass with a 10 k ohm potentiometer. It depends on the resistance of your LDR. But with a 100 k ohm potentiometer you will have room for a wider range of LDR values.

You can also make the LED turn On when it’s dark like an automatic street light. To do this, you need to use a PNP transistor instead of a NPN transistor.

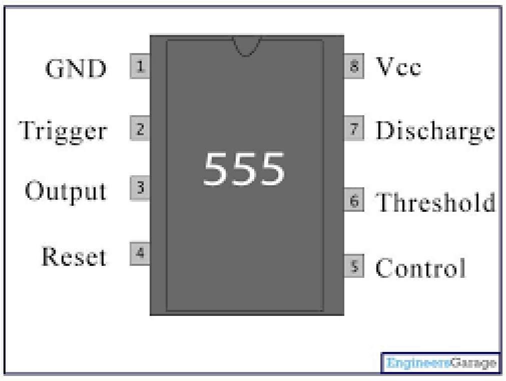
Even though LDRs are technically a type of resistors, they are also considered a type of sensors.

Sensors: are devices that respond electrically to a physical change. Often this response is a change in resistance, the sensor converts one analog (physical) condition to another analog (electrical) condition.

We have already used LDR which is a light sensor, in a coming section in this kit we will be also using something called PIR sensor (also called motion sensor).

Before we build our next circuit, let’s recap what we have known so far. We’ve managed to go from basic electronics concepts into building simple circuits using LEDs and resistors until we have used transistors. Before we said that transistors are considered the most important single component in electronics. If you’re not convinced yet hopefully you will change your mind in this next section as we are going to discuss a huge step forward in the advancement of electronics which was built on top of transistors. The Integrated Circuits !

Integrated Circuits (ICs): are usually shown as rectangular boxes with pin names. Below is an example of an IC which is the 555 timer



An integrated circuit is simply any type of circuit made to fit onto a chip. Sometimes we call it a chip, a microchip or an IC. The function of the IC in your circuit depends on the type of circuit. It can be a radio transmitter ,an amplifier or a micro-controller or any other circuit you can think of.

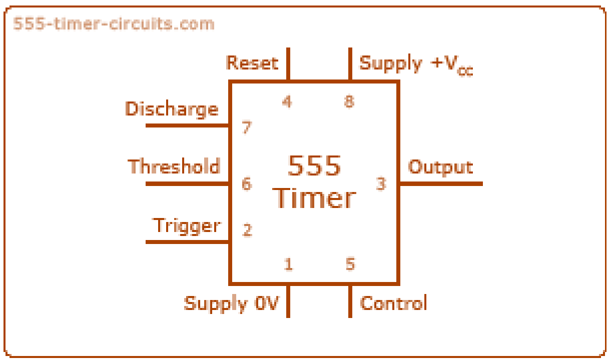
By making an integrated circuit on a small chip, it's much easier to make advanced projects. If you have an IC but don't know it's function just take the model number that you find on the top of the chip and search the web by typing the model number with the word "data sheet" after it. Usually, you'll find a PDF file of the data sheet. The data sheet is a technical document that describes the component. By opening and reading the 1st page of the data sheet, you should be able to get an understanding of what the IC does. However, if your chip doesn't have anything written on it, it's almost impossible to know what the function of the IC is, so make sure when you buy an IC that it has proper printing on it.

In the next circuit we will be using one of the most famous ICs and probably the most used, sold and successful IC all over the world. The 555 Timer.

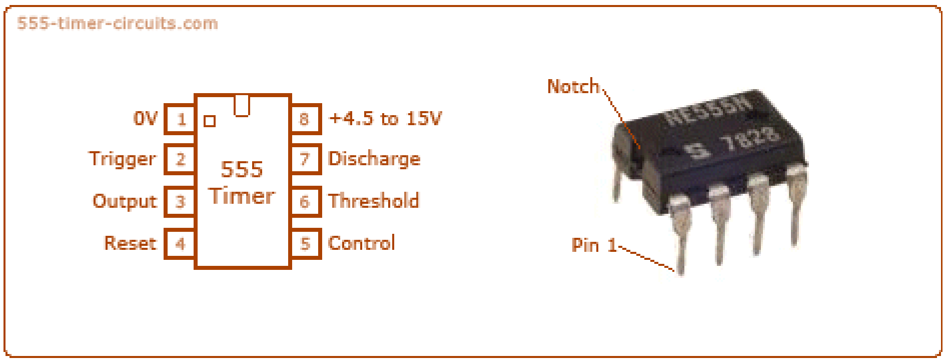
BONUS CIRCUIT 1

By now you have acquired a very good knowledge base concerning electronics, but there is still a few more things to learn. Since this circuit has a lot of wiring maybe it won't be simple to connect it using the crocodile clips in the kit. So you may want to try connecting it outside the kit using a breadboard which is the simplest way to test a circuit or try a schematic diagram since all you have to do is put the components and remove them to try other configurations or to correct a mistake you made. The following diagrams will include th breadboard view of the connection and the schematic diagram as always. You can still try to challenge yourself and make it using the kit but the most important thing however, is to understand the following ideas.

The 555 (the reason for that name is that it's composed inside from three 5 k ohm resistors) integrated circuit is an easy to use timer that has many applications. In a circuit diagram, the 555 timer chip is often drawn like the illustration below. Notice how the pins aren't in the same order as the actual chip, this is because it's much easier to recognize the function of each pin and it makes drawing circuit diagrams much easier.

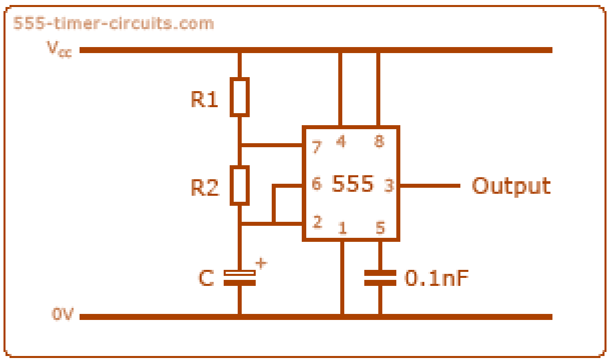


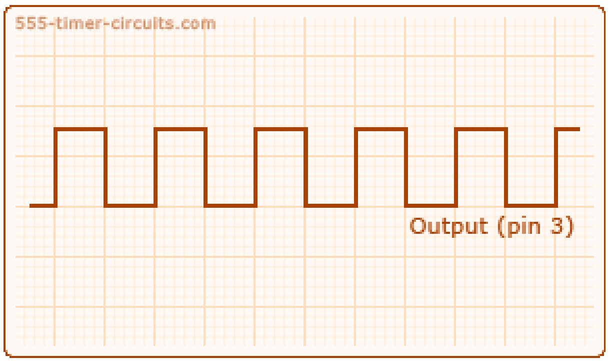
Here is how the 555 timer IC looks in real life compared to how it's drawn to identify its pins



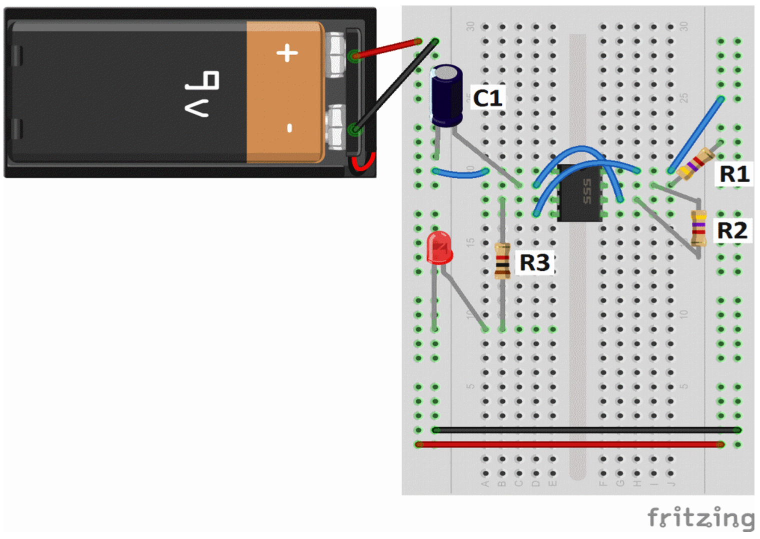
The output of the 555 can be one of 2 states at any time, the 1st state is the 'low' state, which is 0 V. The 2nd state is the 'high', which is the voltage VCC (the voltage of your power supply which can be anything from 4.5 to 15 V. The Absolute Maximum is 18 V). There are 3 main operating modes of the 555 timer, each mode represents a different type f circuit that has a particular output, they are:-

1- Astable Mode: An Astable circuit has no stable state – hence the name "astable". The output continually switches state between HIGH and LOW without any intervention from the user, the output is called a 'square wave'. This type of circuit could be used to switch a motor ON and OFF at regular time intervals, to flash lamps and LEDS, and this circuit is useful as a 'clock' pulse for other digital ICs and circuits.

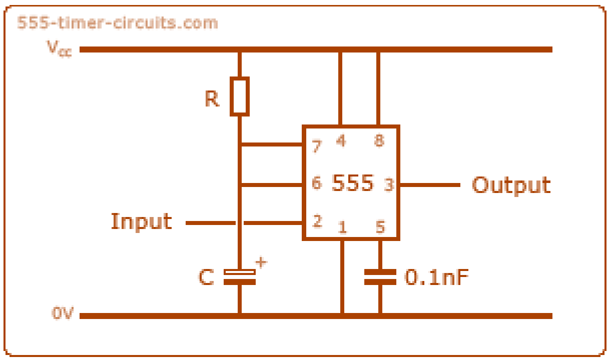


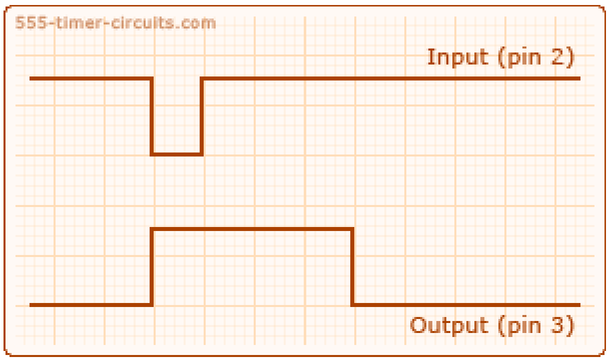


The following picture shows how to build a circuit that blinks a LED ON/OFF at specific time intervals using a breadboard. All the components used in this breadboard view should be familiar to you by now.

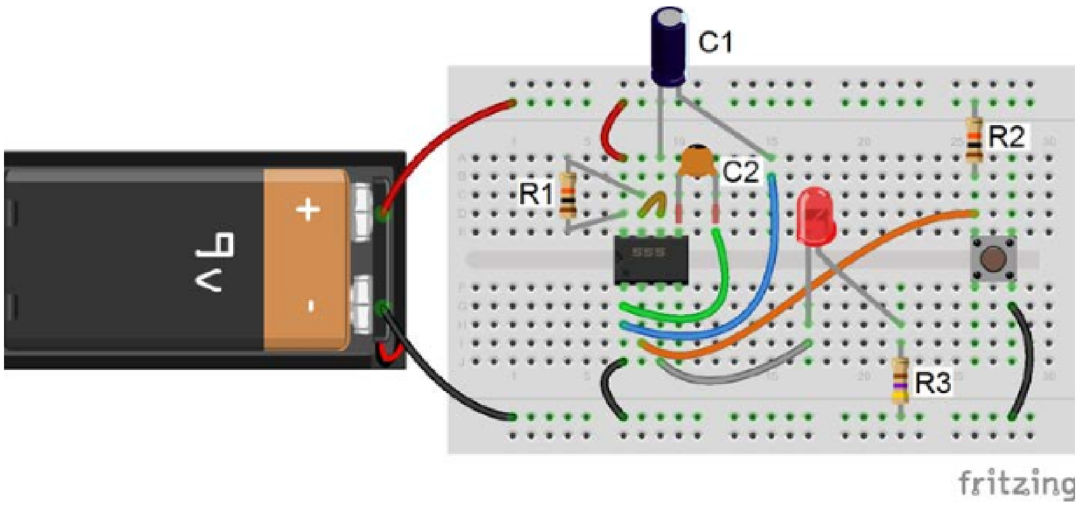


2-Monostable Mode: A Monostable circuit produces one pulse of a set length in response to a trigger input such as a push button. The output of the circuit stays in the LOW state until there is a trigger input, hence the name ‘monostable’ meaning ‘one stable state’. This type of circuit is ideal for use in a “push to operate” system. A user can push a button to start a device’s mechanism moving for example and the mechanism will automatically switch off after a set time. Another example of the use of a monostable is delaying lighting inside a dark building where the user can push a button to light the building as they they navigate it until they reach their destination and the light will switch off automatically after a period of time, giving the user enough time to reach their destination in a place well lit.

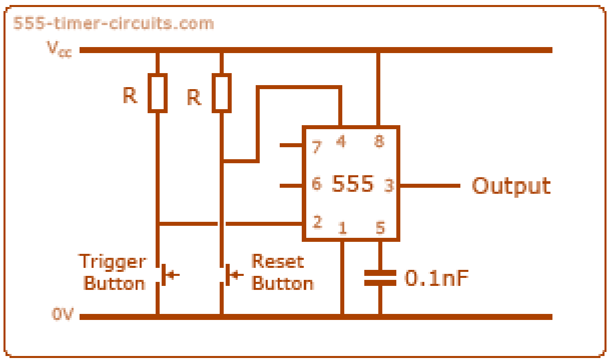


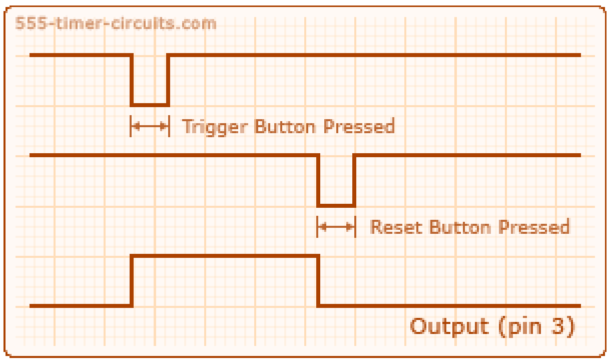


The following is a breadboard view of a circuit build of a one-press LED timer using components you already know



3- Bistable Mode: A Bistable Mode (sometimes called a Schmitt Trigger) has 2 stable states HIGH and LOW. Taking the Trigger input LOW makes the output of the circuit go into the HIGH state. Taking the Reset input LOW makes the output of the circuit go into the LOW state. This type of circuit is ideal for use in an automated model railway system where the train is required to run back and forth over the same track. A push button (or relay on the underside of the train) would be placed at each end of the track so that when one is hit by the train, it will either trigger or reset the bistable. The output of the 555 would control a relay which would be wired as a reversing switch to reverse the direction of current to the track, thereby reversing the direction of the train

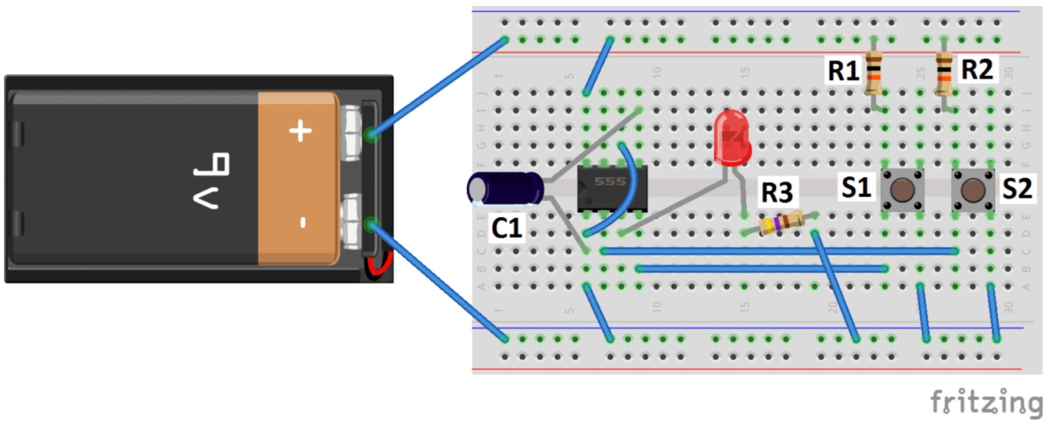




for more information, you can visit this link which is the source of the above information

www.555-timer-circuits.com/operating-modes.html

Below is a breadboard view of a circuit built to make use of the bistable mode of the 555 timer, can you guess how it works ?



(The 3 previous breadboard views of the 3 circuits demonstrating different 555 operating modes were taken from : http://www.circuitbasics.com/)

The output (pin3) of the 555 can be in one of 2 states at any time (HIGH or LOW), which means it's a digital output. It can be connected directly to the inputs of other digital ICs, or it can control other devices with the help of a few extra components. The devices could be anything that can be switched ON and OFF. Make sure your power supply can provide enough current for both the device and the 555, otherwise the timing of the 555 will be affected.

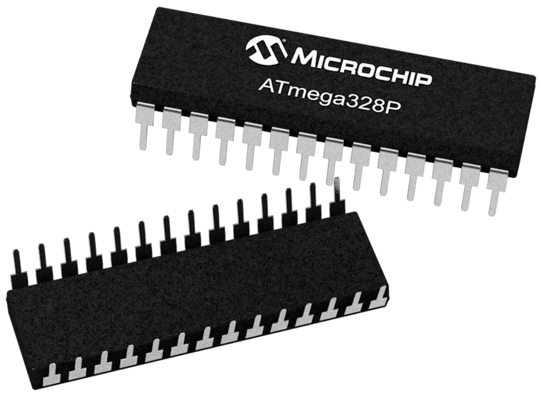
Microcontrollers

So far we have moved from building simple circuits using basic components like resistors and LEDs and capacitors and push-buttons, to a higher level using transistors as automatic switches, to making more complex circuits using ICs. So the question becomes: can we build more sophisticated circuits ? The answer is Yes ! While you can use as many ICs as you can to build much more powerful circuits it will be time consuming and hard to connect and maintain. There is another way which is both time and space efficient that enables you not only to achieve the same results as using many ICs, but also enables you to customize the electronic circuits according to your own needs and benefits, it's called micro-controllers !

A micro-controller (sometimes abbreviated to MC) is a small computer on a single integrated chip programed by the user and used for carrying out specific applications within a larger mechanical or electrical system. So you can use a single micro-controller to do the same tasks that you would be doing using many ICs or even more complex tasks using only a few lines of code !

To explain micro-controllers even further, it's extremely important to differentiate between micro-controllers and micro-processors. Since people often get confused between the Two. Micro-processors are devices that are very small in size that can do a lot of computing and calculations at the same time. While micro-controllers are used to perform a single task repeatedly. Even though both are programed by humans, their functions vary differently. Whenever you find yourself confused between the two think of it like this, the micro-controller is the muscle and the micro-processor is the brain. So micro-processors are used for stuff needing a lot of brain (like doing a lot of calculations), while micro-controllers are used for things needing a lot of muscles (like moving a lot of motors). In other words, micro-processors can perform many tasks at the same time (multi-tasking) while micro-controllers can only do one task at a time.

So a micro-controller is a chip that you can program using a computer, and you can program it to perform certain tasks for you in the circuit. There are many types of micro-controllers out there, the most famous families are PIC and Atmel.



We will be using a micro-controller in our next few circuits, but before that let's learn more about the type of micro-controller we will be using. It's called Arduino !

What’s an Arduino ?

Arduino is a name given to a single-board micro-controller (based on the Atmel family of micro-controllers) that can be used to build digital devices and interactive circuits. Unlike other micro-controller chips that require a special programming kit for you to be able to program the chip by putting it in the kit then ejecting it from the kit then connecting it to your circuit, the Arduino board contains not only the chip but all the peripherals needed for you to program it. All you need to do is just connect the cable from the Arduino board to the USB port on your computer and upload the code you wrote on the Arduino software on your computer to the Arduino board and that’s it ! The board will make sure the Arduino micro-controller functions the way you wanted it to function based on your code. Before we begin building our first circuit using an Arduino micro-controller please note that there are many different boards of Arduino with different shapes and sizes made to suit different functions. The most popular and most widely used board around is the Arduino Uno. However, in this kit we will be using Arduino Nano since it’s much smaller so it can fit in this kit. But don’t worry if you are planning to do these circuits away from this kit using an Arduino Uno or Arduino Mega or any other type of Arduino boards. Since these circuits can be done using any of them and the only difference is that you change the type of the board you are going to upload to (and maybe the number of the pins) on the Arduino IDE software. Again don’t worry if you don’t fully understand yet since all is going to be explained in the next few lines.

The previous image was of the Arduino Uno, the next image show the Arduino Nano board along with its USB cable which has one end that can be connected to a regular USB hub on any computer just like any flash drive. While the other end of this cable should be connected properly to the Arduino Nano which from now on we’ll refer to it simply as Nano.

Circuit 9

Before doing our connection sequence you need to turn on your PC/Laptop and install Arduino IDE Software, to know how check these following links depending on your Operating System:-

To install the Arduino Software (IDE) on Windows PCs:-

https://www.arduino.cc/en/Guide/Windows

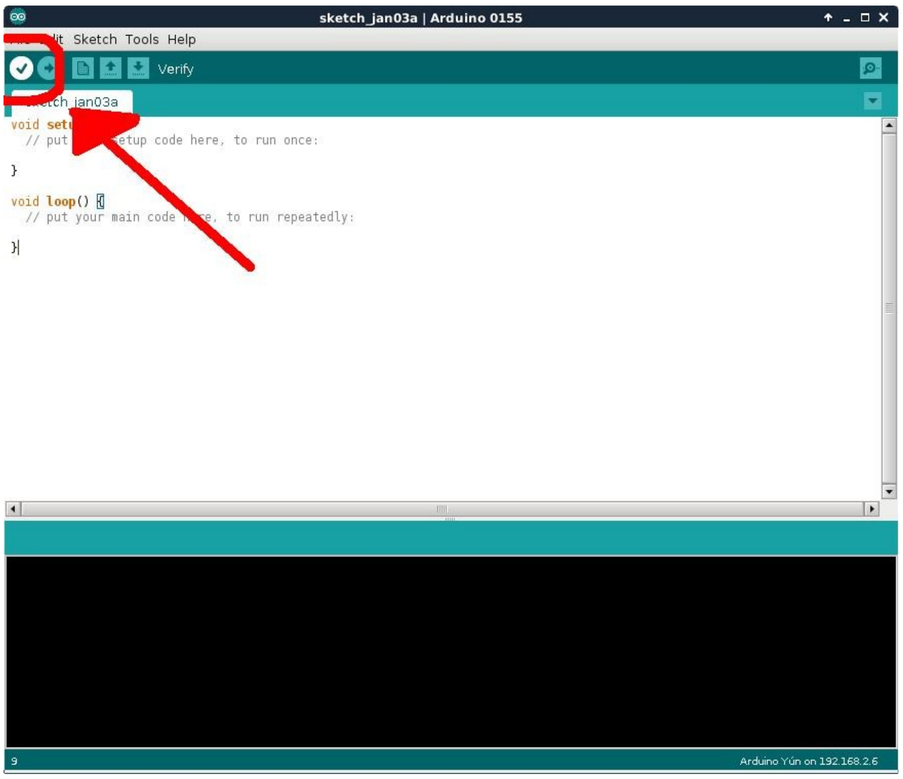
To install the Arduino Software (IDE) on Linux:-

https://www.arduino.cc/en/Guide/Linux

To install the Arduino Software (IDE) on OS X:-

https://www.arduino.cc/en/Guide/MacOSX

After successfully installing and running the Arduino IDE software on your computer connect the USB Cable to the USB hub on your PC , click **File**, then choose **Examples**, then choose **01.Basics**, from it click Blink. A bunch of lines appear on what was previously a big white space. Go to the top of the Arduino IDE window, the tick icon at the very far left of the Arduino IDE window, the tick icon is circled here in case you haven't found it by now.



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

}

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

}
```

Since you haven't entered any code by yourself yet, then there should be no problem so far since we used one of the example codes saved within the Arduino IDE library, such examples are used to teach you how to code with Arduino in case you are just getting started.

With your mouse go back to the top menu of the Arduino IDE software, click on **Tools**=>,then hover over **Board**, from it click on **Arduino Nano w/ Atmega 328** and make sure that there is a tick sign next to this type of board.

Now click on the arrow icon which is next to the tick icon you clicked earlier and wait for a few seconds, if all goes well, you won't see any color on the IDE except the ones you already see. If you happen to see an orange color that means something is wrong when you clicked the arrow icon but don't worry this problem can be easily solved. Just click on **Tools**, from it hover with the mouse over **Serial Port** and then choose another port for the USB cable connecting your Nano to your PC. Depending on your Operating System and how many USB ports you have available on your hardware you may need to try many different ports and then click on the arrow icon again every time until the orange color disappears. If the **Serial Port** is deactivated (grey and you can't choose anything from it since it doesn't show anything when you hover with the mouse over it), then this means that the USB cable from the Nano to your PC isn't connected properly so remove it and connect it again until the Serial Port Option is active for you to choose from.

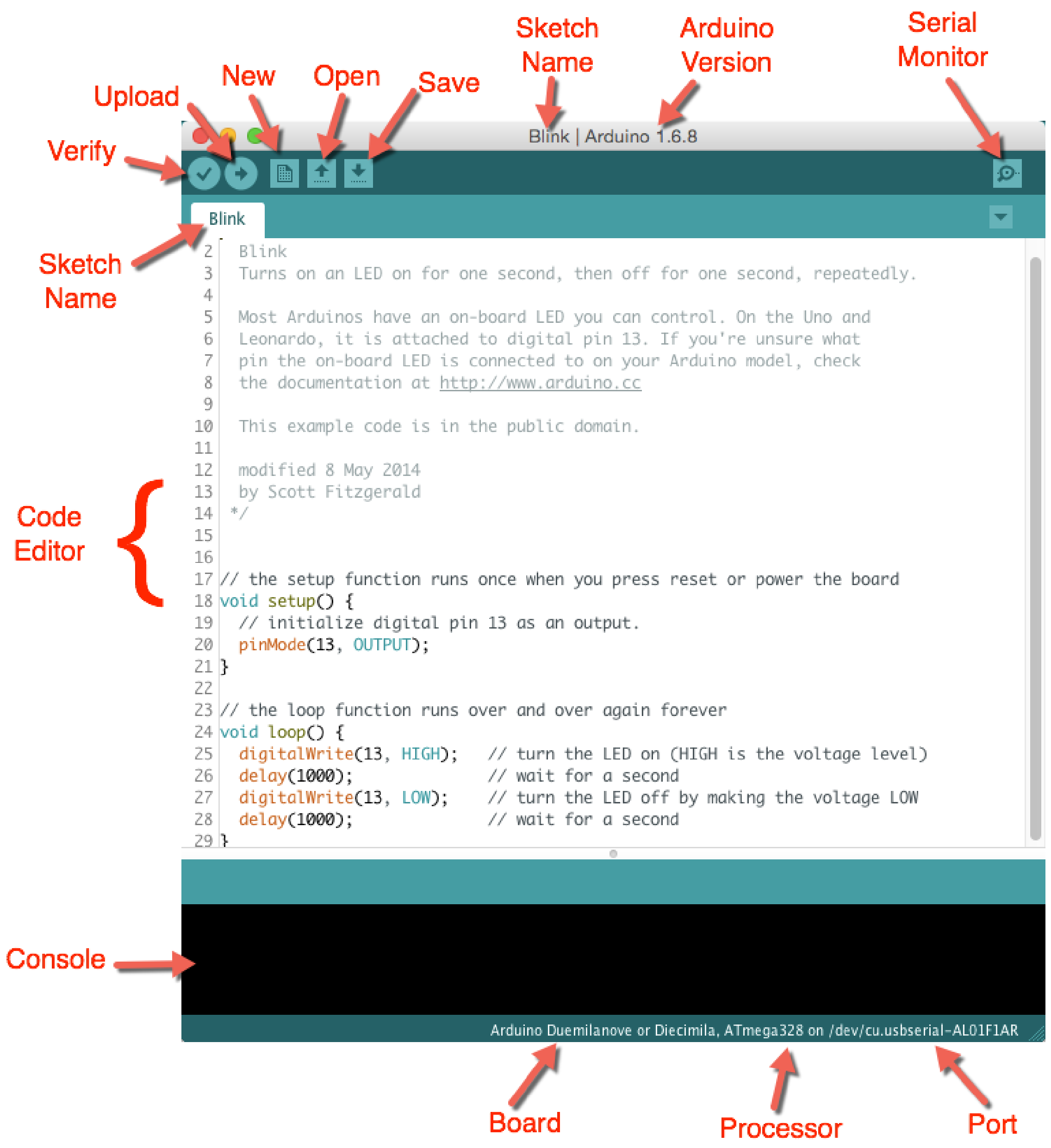
Now go back to the kit for a few minutes but leave your computer on and don't close the Arduino IDE software. Follow this connection sequence:-

44-47, 45-54, Nano-PC OR 46-1, 57-2

Note that OR means you can only use this connection OR the other two. For testing purposes it's advisable to use the Nano-PC and ignore the other option until you finish all the circuits in this booklet for the first time at least.

Now what do you notice when you finish the connection sequence exactly as written above ? If you have followed the previous instructions precisely about the Arduino IDE and the USB cable was properly connected to your PC and read by your computer then by now you should be seeing a Blinking Green LED above the Nano ! If that's the case then congrats ! You have just written your first program in Arduino and you managed to program this little micro-controller to do this simple task, the possibilities for you are endless by now and in the next few lines you will understand everything you need to know about Arduino programming and the two icons you clicked, if the circuit didn't work properly and you are not seeing a blinking LED even though the board itself is lighting then check your connections by starting from the beginning of this circuit and following the previous steps.

Now let's have a closer look at the Arduino IDE screen and know what exactly is there



```
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the Uno and
  Leonardo, it is attached to digital pin 13. If you're unsure what
  pin the on-board LED is connected to on your Arduino model, check
  the documentation at http://www.arduino.cc

  This example code is in the public domain.

  modified 8 May 2014
  by Scott Fitzgerald
 */


// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

Arduino Duemilanove or Diecimila, ATmega328 on /dev/cu.usbserial-AL01F1AR

So now you know that the tick icon is used to **verify** your code (make sure your code doesn't have any errors). If you click the **verify** icon and there are errors in your code. You will find an orange color in your console at the bottom of the Arduino IDE window as shown. The console might also turn orange in case there is a problem when you click the upload (arrow) icon to upload the code to the Arduino board. So always check the console for errors and problems if something goes wrong. By now you should be reading "Arduino Nano w/Atmega328" at the bottom of the console since we already chose it in a previous step, followed by the path of your USB port that you connected the cable to from one end and the Arduino to the other end. The "New" icon is used to open a new Arduino file while the "Open" icon is used to open an existing Arduino file and "Save" is used to save your current file that you are working on. The magnifying glass icon is to see the serial monitor and we will fully understand in a later circuit. Most of our work will be in the Code Editor, that is the big white part in the middle of the Arduino IDE where the code is written.

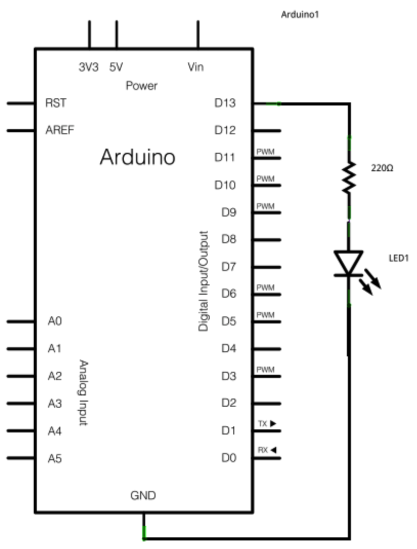
Now that we have understood the interface of the software that we use to write code to program our Arduino Micro-controller, let us understand the code that we took from an existing Arduino library and helped us make this blinking LED circuit. We will look at the code line by line but before that let’s explain a very important concept in every programming language, the comments ! Every programming code in any programming language gives the programmer the ability to write comments next to his code. These comments do not affect the flow of the code written by the programmer since they are ignored by the IDE that you use to write your code. So what are they used for then ? Well, since pretty much anything can be written in these comments in normal human language. They are used to help programmers understand what they wrote in case someone else sees the program or the one who wrote it get back to it after leaving it fro some period of time and these comments help the programmer remember the reason of each line of code was written. It’s advisable that you write comments for your program as much as possible. In the Arduino C (the programming language used to program the Arduino MC boards) to write a comment, simply write “**//**” to make a single line comment or write “**/***” at the beginning of a comment and “***/**” at the end if you need to write a comment that will be more than just a single line. We will be using comments next to every line of code to both help explain the meaning of each line of code and at the same time to make you get used to how actual programs are written. Notice that comments are written in different color than the programming code. The code for the circuit we’ve just built was:-

```
int led = 13;  // We declare the use of the pin number 13 on the Arduino MC to connect it to a LED

void setup()  //here you write your setup settings
{                // This block of code enclosed between 2 braces is used for input/output settings
pinMode(led, OUTPUT);  // Is pin number 13 an input or an output ?
                        // In this case it’s an output.
                       /*The general form of code here is : pinMode(pin number, State) */
                      // where the state can be an input or an output */
            //also note that the state (input or output) MUST BE written in CAPS !
}           // so all input/output settings must in these 2 braces and that’s the end of the setup block

void loop() //this part enclosed between 2 braces is used for writing what you want the MC
            // to actually do for as long as it’s turned on (infinite loop)
{
 digitalWrite(led, HIGH);    // turn the LED on (HIGH is the voltage level)
 delay(1000);                // wait for a second
 digitalWrite(led, LOW);    // then turn the LED off by making the voltage LOW
 delay(1000);               // wait for a second
} // the end of the loop block and the end of our Blinking LED program
```

/* Did you notice the code lines ending with a semi-colon ? This “;” is very important since it tells the Arduino program that this line of code has ended. If you forget to write one where you were supposed to that will result in errors in your console so always make sure you end your lines with a semi-colon where there should be one. */

/* Always remember : Any program consists of 3 main parts; Part A: Defining or Declaring any constants (values that never change) or variables (values that are constantly changing) in the program (for example when we declared the use of pin 13 and named it led we were telling the MC that we have declared a variable named ‘led’ at pin 13 so that it treats it as a pin with the value led which we later changed it between LOW and HIGH in the loop block). Part B (the setup block): Specifying the function of each pin and whether it’s an input or an output and specifying the settings of the pins that are going to be used. Part C (the loop block): the main program is written. Specifically what we want the micro-controller to do. */

Are you confused a little ? No worries, just go back and study the code you have imported from the Arduino Library to make the blinking LED circuit and try to write a similar one in your text editor or even try to write it in the Code Editor of your Arduino IDE but this time write your own comments and try to explain to yourself how each line works. Are you making mistakes ? That’s OK, making mistakes is how you learn so revise the code of this circuit and move on the next which will be a lot similar to this one but hopefully much easier since we have covered already most concepts you need to understand regrading Arduino Programming. But before we go to the next circuit have a look at the schematic of the Arduino micro-controller connected to the blinking LED circuit that we constructed using our kit. Notice that in this schematic a different value for the current limiting resistor is used than our usual 470 Ω.



Circuit 10

This circuit is an improvement to the previous circuit, we are going to add another component to the previous circuit which is a push-button and we will program the Arduino MC to stop blinking the LED only when we press this push-button.

Open the Arduino IDE and in the code editor write the following code:-

```
int ledPin = 13;
int buttonPin = 2;
int val = 0;

void setup ()
{
 pinMode(ledPin, OUTPUT);
 pinMode(buttonPin, INPUT);
}

void loop ()
{
 val = digitalRead(buttonPin);

 if (val == HIGH)
   {
     digitalWrite(ledPin, HIGH);
     delay(1000);
     digitalWrite(ledPin, LOW);
     delay(1000);
   }
 else
 {
        digitalWrite(ledPin, LOW);
 }
}
```

We are going to explain the meaning of this code in details but first let’s do the following connection sequence:-

44-47, 45-54, 58-16, 57-15, Nano-PC OR 46-1, 57-2

After doing this connection sequence, click the verify (tick) icon to make sure there aren’t any errors. If there are check your code. Did you write something wrong (Capital and small letters do matter here!) ? Did you forget a semi-colon or added a one where you shouldn’t perhaps ? Once you’ve verified your code. Connect the USB cable to your PC and click the upload (arrow) icon. If you face any kind of errors in this step then go back to the previous circuit and check the steps we have applied to set the IDE to connect with the Nano board. You should have no problem from here

You’ll notice that the LED keeps blinking as if nothing has changed from the previous circuit. Now press the push-button and see what happens. The LED turns off for as long as you are pressing that button. As soon as you release your finger the LED starts blinking again ! Now let’s understand the code we have written.
Note that in order to keep things simple, I’ll only be explaining the new parts introduced in this code. If you still have problems in understanding the parts that were already explained in the previous circuit no problem. Just go back and read the code thoroughly again until you fully understand it and come back and do this circuit from the start again. As we did with the previous code we will explain what each code line means using comments.

```
int ledPin = 13;                //declaring a variable called ‘ledPin’ at pin 13 of the MC
int buttonPin = 2;              //declaring a variable called ‘buttonPin’ at pin 2 of the MC
int val = 0;                    //declaring a variable ‘val’ with initial value of zero
                                //’val’ is a variable withing the program which will be used
                                //to read an input value we give to the MC through the button

void setup ()
{
 pinMode(ledPin, OUTPUT);           //like the previous code ‘ledPin’ state is output
 pinMode(buttonPin, INPUT);         //’buttonPin’ state is input since we input using the push-button
}

void loop ()
{
 val = digitalRead(buttonPin);       //this is to give a new value for ‘val’ variable
                                    //this value is reading the input coming from the push-button

 if (val == HIGH)                   //this is called an if-statement, here we tell the MC
   {                                //if ‘val’ reads a HIGH value from the push-button
    digitalWrite(ledPin, HIGH);     // then make the LED blink the same way as the previous code
    delay(1000);
    digitalWrite(ledPin, LOW);
    delay(1000);
   }                                //notice how every part is enclosed in braces
 else                               //else (if the condition of the if-statement above isn’t the case
{                                   //which means if ‘val’ reads a LOW value from the button
      digitalWrite(ledPin, LOW);    // then turn off the LED
}
}       /*end of the program*/
```

To elaborate more, we used something called if, else statement to tell the MC to do something **if** this happens, **else** do another thing.

We can also use if-else if…. Like in the English language:-

if (you like science) {you should read books}
else if (you like sports) {you should play football}
else if (you like art) {you should take art lessons}
else {you should search for another hobby}

Notice how the last else in any if-else statement is NOT followed by an if, just like regular human English language.

Also notice that the Arduino C is a case-sensitive programming language, so be careful what is capital and what isn't to avoid unnecessary errors. If you're still wondering why we write things like "digitalWrite" and not "digital Write" or "buttonPin" instead of "button Pin" remember that while the Arduino C is very similar to the regular human English language, IT IS NOT ENGLISH LANGUAGE. So spaces are not allowed in Arduino C so programmers use this trick to make distinctions between different variables which is naming the first name with small letters and the second name with a capital letter in the beginning to make it easier to read. So for example, if we want to write "variable one" in our programming code it will be written like this : "variableOne".

You maybe still a little bit confused, if you are not that's great. If you are that's great too ! In both cases you're learning and realizing some patterns in the programming code and your knowledge base is growing.

Try to reverse the function of this program to make the LED blink when the button is pressed instead of the other way around. Challenge yourself and in case something didn't work no worries you are on your path to learning more and more.

Since we already covered most of the electronic circuit schematics and you now have a very good understanding of them and since we won't be connecting the Arduino MC to a lot of components since it does most of the difficult jobs itself we will only be providing circuit schematics in case the components connected to the MC were complicated and nothing similar to what you have built before in this kit.

Before we go to the next circuit let's dive into the last concept of Electronics that is explained briefly here in this booklet, the Logic Gates !

Introduction To Logic Gates

Remember what we said earlier about the fact that we can use micro-controllers (MCs) to build circuits doing the same functions if we have used integrated circuits (ICs) instead. Well before we go on building our next circuit, let's introduce one final concept in this booklet. That is logic gates.

A Logic gate is an electronic component that can be used to conduct electricity based on a rule. The output of the gate is the result of applying this rule to one or more "inputs". These inputs may be 2 wires or the output of other logic gates.

In the real world, the data may vary a lot. For example, human voices vary greatly since there are very high voices and high voices and regular voices and low voices and very low voices. In the electronics world that's not always the case since many components can only accept 2 values : ON and OFF – 0 Volts or VCC Volts (depending on the value of the power source). We call the real world analog and the electronic components that only accept 2 values with no variation in between we call them digital. There are many devices and components that have been built to convert from analog to digital (like the microphone) or from digital to analog (like speakers).

Logic gates are digital components which means that they work at only 2 levels of voltage, a positive level and a zero level. Or based on 2 states : ON and OFF. In the ON state, voltage is present. In the OFF state, the voltage is zero.

A Logic gate will examine the state at its input(s) to decide what the state at its output should be. A logic gate is ON (or active) when its rules are correctly met. When that happens, electricity flows through the gate and the voltage at its output is at the level of its ON state.

Every Logic gate has something called a truth table. A Truth table will tell you what the output of the gate will be, depending on the input(s) and the rule that the logic gate follows.

Since they are used widely to build many digital devices, there are many logic gates out there. In this booklet we will be looking at the most important and the most basic three logic gates ever. The AND gate, the OR gate and the NOT (inverter) gate.

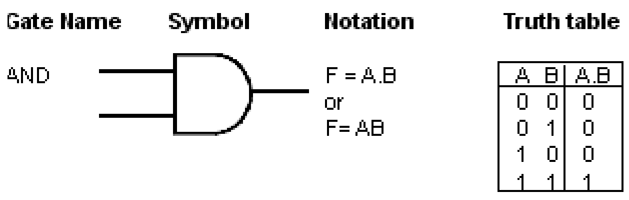
BONUS CIRCUIT 2

Before building the circuit, note that there are 2 ways to use logic gates in an electronic circuit, one of them is using integrated circuits (ICs) and the other is using a micro-controller (MC). Here we are using an Arduino micro-controller to simulate what different logic gates would do.

Since this circuit has a lot of wiring maybe it won't be simple to connect it using the crocodile clips in the kit. So you may want to try connecting it outside the kit using a breadboard and an Arduino Uno board which is the simplest way to test this circuit since all you have to do is put the components and remove them to try other configurations or to correct a mistake you made. The following diagram will include th breadboard view of the connection and the schematic diagram as always. You can still try to challenger yourself by trying to make it using this kit. The most important thing however, is to understand the following ideas.

We mentioned earlier that we can know the output of a logic gate based on a truth table for that specific logic gate that follows certain rules. Well, these rules are called Boolean Algebra. We are not going to dive into them, you just have to know that the rules of regular arithmetic and Algebra do not apply to them even though they might seem similar at times, THEY ARE NOT THE SAME.

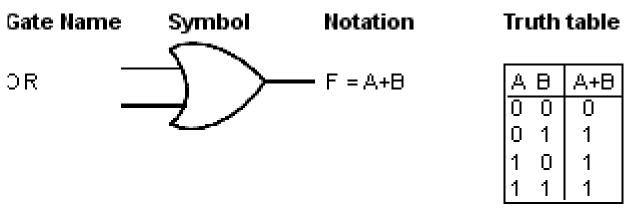
The 1st Logic gate we are going to use is the AND gate, it has 2 inputs (A,B) and one output (F). Its symbol looks like this



A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

As you can see from the truth table, the **AND** gate can only have an output of 1 when BOTH its inputs A **AND** B have 1. That's why it's called an **AND** gate.

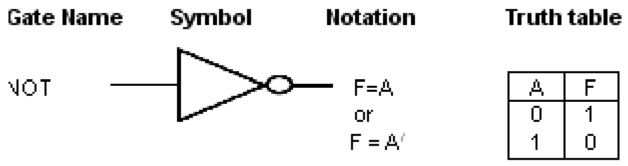
The 2nd Logic gate we're using is the OR gate, it also has 2 inputs (A, B) and one output (F). Its electronic symbol looks like this



A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

From the previous truth table, you can deduce that the OR gate will have and output of 1 when either one of its inputs A OR B at least have 1. That’s why it’s called an OR gate. If you are confused about the output of F when both A and B equal 1 and how does A + B = 1 in this case and not 2, again PLEASE REMEMBER what we said earlier about these operations having nothing to do with regular arithmetic or algebra. It follows the rules of another branch of mathematics called BOOLEAN ALGEBRA. So keep that in mind when dealing with Logic gates.

The 3rd and final Logic gate we will be using here is called the NOT gate, unlike the previous 2 gates it has only 1 input (A) and a single output which we will call A`. It looks like this when drawn in a circuit



A	F
0	1
1	0

There is also another name for the NOT gate, some people call it INVERTER gate. And from looking at the truth table you can see why. It inverts the input state in the output. If the input is 1, the output is 0 and if the input is 0 the output is 1.

Now that you understood the basics of logic gates, open the Arduino IDE and type the following code. As always every new part will be explained by comments:-

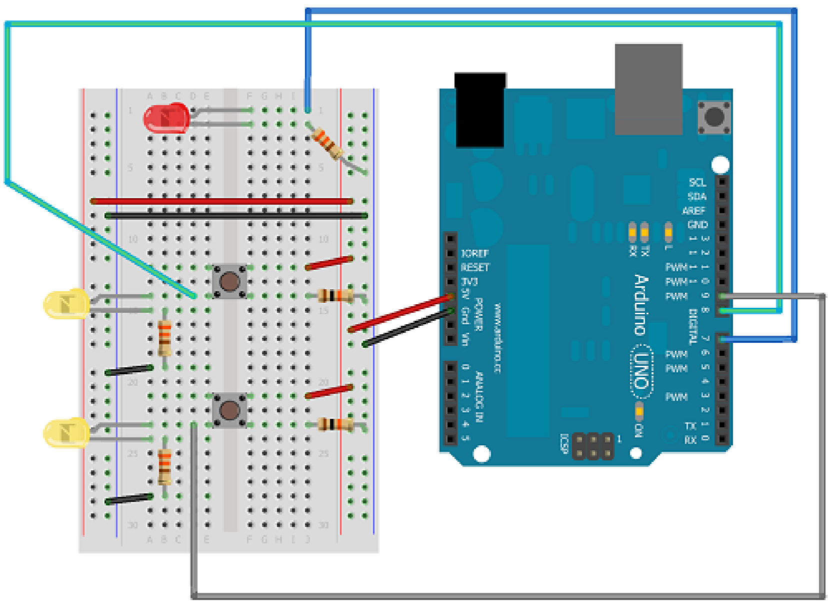
```
int pinOut = 7;          //the output of the logic gate is at pin 7 of the MC
int pinA = 8;            //the 1st input of the logic gate is at pin 8 of the Arduino MC
int pinB = 9;            //the 2nd input of the logic gate is at pin 9 of the Arduino MC

void setup()
{
  pinMode(pinOut, OUTPUT);        //1 output pin and 2 input pins (A,B) for the logic gate
  pinMode(pinA, INPUT);
  pinMode(pinB, INPUT);
}
void loop()
{
  boolean pinAState = digitalRead(pinA);     //boolean is a type of data that can only be 0 or 1
  boolean pinBState = digitalRead(pinB);     //we’re telling the MC to read either 0 or 1
  boolean pinOutState;                       //from the 2 inputs and output a boolean value
  // the next line we tell the MC to follow the rule of the and gate, its symbol in Arduino C is ‘&’
  pinOutState =pinAState & pinBState;
  digitalWrite(pinOut, pinOutState);         //the output at pinOut is the value at pinOutState
}
```

/*This is the basic code to demonstrate the behavior of an AND gate. When you connect the circuit and upload the code to the Arduino, you should be able to see the output LED light **when, and only when, both push-buttons are pressed at the same time.**

If you replace the line, pinOutState =pinAState & pinBState; with one of the following lines, you can simulate the other common logic gates. For the NOT gate, you only have a single input.

```
 pinOutState = pinAState | pinBState;        // or gate
 pinOutState = !pinAState;                   //  not gate
END OF THE CODE */
```

Here is the breadboard view of this circuit showing all the components and the wiring, Arduino Uno board is used but you can use any Arduino board and it will work just make sure that you do the settings properly as discussed before:-



Circuit 11

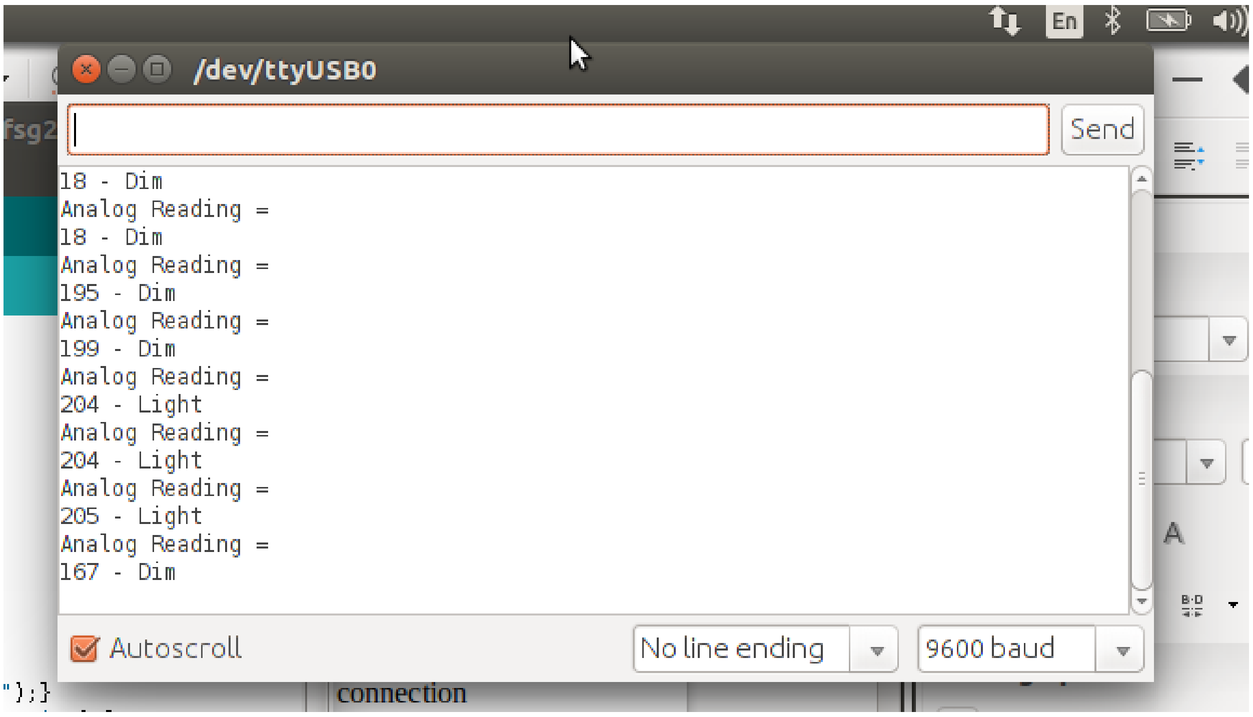
Remember Circuit 8 ? We learned about LDRs. This time we are going to use it with the Arduino and instead of using a LED to show us the output, we are gonna use the serial monitor in the Arduino IDE. This circuit is a demonstration of how micro-controllers can be used to monitor real-life environmental changes and record them in the computer since this same concept can be used to monitor temperature, humidity,...etc. Now open the Arduino IDE and write the following code:-

```
int ldrPin = A0;        //A0 is an analog data pin, we tell the MC that the LDR is at that pin
int ldrReading;         // variable to read the values coming from the LDR

void setup()
{
Serial.begin(9600);     //initialize the serial monitor to read values, 9600 is the rate of data display
}
void loop()
{
ldrReading = analogRead(ldrPin);     //read the values coming to the LDR
Serial.println("Analog Reading = ");          //print new line saying: “Analog Reading = ”
Serial.print(ldrReading);                     //then print the value taken from the LDR

if (ldrReading < 10) {Serial.println(" - Dark");}       //if statement: depending on the value of
else if (ldrReading < 200) {Serial.println(" - Dim");}         //the LDR, a description fitting
else if (ldrReading < 500) {Serial.println(" - Light");}       // the lighting will be written
else if (ldrReading < 800) {Serial.println(" - Bright");}
else {Serial.println(" - Very Bright");}
delay(1000);
}
```

Before Uploading that code connect the crocodile clips according to the following connection sequence:-

19-46, 21-47, 20-22-52, Nano-PC

After finishing the connection sequence correctly, upload the code to the Nano and click on the magnifying glass icon at the far right top of the screen remember it ? This is the serial monitor icon. After you click it a new window should appear. Move your finger over the LDR at variable distances, try covering it completely and moving it quickly away or even try using a flash light. You should see results from your serial monitor that looks something like this:-



```
18 - Dim
Analog Reading =
18 - Dim
Analog Reading =
195 - Dim
Analog Reading =
199 - Dim
Analog Reading =
204 - Light
Analog Reading =
204 - Light
Analog Reading =
205 - Light
Analog Reading =
167 - Dim
```

Circuit 12

At the very beginning of our learning journey we first learned how to power up a circuit that had a LED in it, then as we moved along we learned how to control the LED lighting using a push-button. This time we will do something more advanced in that area. We will use the micro-controller to turn the LED ON or OFF. Like the previous circuit we will be using the serial monitor.

In the Arduino IDE write the following code:-

```
int ledPin = 13;
int value;

void setup()
{
Serial.begin(9600);
pinMode(ledPin,OUTPUT);
}

void loop()
{
value = Serial.read();

if (value == '1') {digitalWrite(ledPin,HIGH);}
else if (value == '0') {digitalWrite(ledPin,LOW);}
}

/*By now this code should be understood completely by you since there is nothing new in it in and you have seen all this code written before in different circuits for different uses but just in case there is some difficulty you still have, this code tells the MC to read from the serial monitor and if it reads the value 1 it will turn the LED ON and if it reads the value 0 it will turn the LED OFF*/
```

Click the verify icon but don’t upload the code until you’ve made the connection sequence which is the following:-

3-54, 6-47, Nano-PC

Now Upload the code you’ve just verified to the Nano board and click the serial monitor icon, you’ll find a text area, write 1 and click send (or press Enter on your keyboard) and observe what happens to the LED, now write 0 and click send (or press Enter on your keyboard) and watch what happens to the LED. So now you can turn the LED ON or OFF using just your keyboard on your computer instead of a push-button through the power of Micro-controllers !

Circuit 13

You have probably heard about motors before since they are used to convert electricity into motion. Motors are used in any device that requires movement. There are many types of motors and in this circuit we will be using one of them : The DC Motor.

In your Arduino IDE Code Editor, write the following:-

```
int motorPin = 9;       //motor connected at pin 9 of the MC, (see the connection sequence)
int onTime = 2500;    //instead of writing the value of the delay every time for each line code
int offTime = 1000;    //create 2 variables with different values for the time the motor is on/off


void setup()
{
pinMode(motorPin,OUTPUT);
}

void loop()
{
analogWrite(motorPin,100);  //analog since the motion of the motor has many values, not just 0 & 1
delay(onTime);        //delay for some time (specified above) so that the motor will rotate for  a while
digitalWrite(motorPin,LOW);          //turn the motor OFF
delay(offTime);                              // leave it OFF for some time (off time specified above)

analogWrite(motorPin,190);  //just like the previous analogWrite above but different rotation angle
delay(onTime);
digitalWrite(motorPin,LOW);
delay(offTime);

analogWrite(motorPin,255); //just like the previous analogWrite above but different rotation angle
delay(onTime);
digitalWrite(motorPin,LOW);
delay(offTime);
}         /*END OF THE PROGRAM*/
```

Now do the following connection sequence:-

62-65, 57-66, Nano-PC OR 46-1, 47-2

Now verify and upload your Arduino Code to the Nano board and wait for a few seconds, put your finger carefully on the top of the metal pole of the DC motor between points 65 and 66, you will feel some movement and you would probably hear the voice of the motor moving as well ! Off course motor motion control using micro-controllers is a complicated subject and you may need a few other components to make the movement more precise for minute applications such as building a robot but now you have an idea of how to control a motor using an Arduino MC. Some applications of controlling a DC motor using an Arduino can be : controlling a fan, robotics,..etc.

Circuit 14

In the previous circuit we mentioned briefly the use of motors controlled by micro-controllers in robotics (making and controlling robots). Aside from the DC motor that we used in our previous circuit there is another type of motors used in robots, it's called the servo motor which we will be using in this circuit. If you look at the kit you will find the servo motor next to the Arduino Nano micro-controller, it looks like this

Of course in the kit the tip of the motor looks a little different but the movement is the same, if you decide to buy your own micro servo motor from an electronics store it should be accompanied by some screws and different plastic tips and/or feathers for you to install to suit your application or the device you want to build. Now open the IDE and type the following:-

```
#include <Servo.h>  //some codes functions need to be imported from their code library
                            //here we have included(imported) the servo library to use its code functions
Servo myservo;  // create servo object to control a servo (code from the imported servo library)
                       // twelve servo objects can be created on most boards

int pos = 0;    // variable to store the servo position

void setup()
{
  myservo.attach(9);  // attaches the servo (YELLOW wire) on pin 9 to the servo object
                             //while attaching RED wire to 5 V pin and attach BLACK wire to GND pin
}

void loop()
{
  for(pos = 0; pos <= 45; pos += 1)    // make the servo move from 0 degrees to 45 degrees
  {                                                     // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                         // waits 15ms for the servo to reach the position
  }
  for(pos = 45; pos>=0; pos-=1)     // goes from 45 degrees to 0 degrees
  {
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                         // waits 15milli seconds for the servo to reach the position
  }
}         /*END OF THE PROGRAM*/
```

As for the connection sequence if you have been reading the comments next to each line of code you’ve probably guessed it since it’s mentioned there the servo connection, if you are a bit confused then it is as follows:-

62-70, 48-71, 57-72, Nano-PC

Note: Due to some voltage variations, the servo motor might not work properly if you use the battery directly as a power source instead of connecting the Nano to the PC and using it as a power source.

So now you know how to program a micro-controller to control the motion of a DC motor and you have just programmed your MC to control a Servo motor in a movement that may look very similar to a waving hand by a robot ! Since you already know how to make a blinking LED using Arduino Code why not try to challenge yourself and use the 2 Green LEDs above the Nano as robot eyes and make them blink while waving the servo motor like a hand ? All you have to do is combine the blinking LED code with the servo control code you’ve just learned and change the position of some pins from which you have plenty ! Sounds Easy and exciting right ?

Circuit 15

This time we will be using Arduino to build a traffic light controller which should be pretty simple. The code to be written in the IDE for this project is:-

```
int red = 9;            //red light LED to be connected at pin 9
int yellow = 8;         //yellow light LED to be connected at pin 8 (or orange LED as a replacement)
int green = 7;          //green light LED to be connected at pin 7

void setup(){
   pinMode(red, OUTPUT);
   pinMode(yellow, OUTPUT);
   pinMode(green, OUTPUT);
}

void loop(){
   changeLights();      //this is a special function we created below
   delay(15000);
}

void changeLights()             //every programming language enables you to create special functions
{                               //it just has to be in the same format as regular language functions

   digitalWrite(green, LOW);          // green off, yellow on for 3 seconds
   digitalWrite(yellow, HIGH);
   delay(3000);


   digitalWrite(yellow, LOW);         // turn off yellow, then turn red on for 5 seconds
   digitalWrite(red, HIGH);
   delay(5000);


   digitalWrite(yellow, HIGH);        // red and yellow on for 2 seconds (red is already on though)
   delay(2000);


   digitalWrite(yellow, LOW);         // turn off red and yellow, then turn on green
   digitalWrite(red, LOW);
   digitalWrite(green, HIGH);
   delay(3000);
}        /*END OF THE PROGRAM*/
```

Connection Sequence:-

3-60, 7-61, 11-62, 6-57, 10-57, 14-57, Nano-PC OR 46-1, 47-2

Now make sure you've followed the connection sequence properly, verify the code in the IDE then upload it to the Arduino and watch the LEDs behave exactly like a street traffic light !

Circuit 16

You have now reached the final circuit in this booklet ! Congratulations ! This time we are going to build a house alarm using a PIR sensor (also called motion sensor) and a buzzer to make the sound of an alarm in case the motion sensor detects a motion. This is probably the hardest circuit to make since it involves a lot of coding. So the code will take more than one page. We are also going to write a special function called “playTone”.

Open the Arduino IDE and write the following code:-

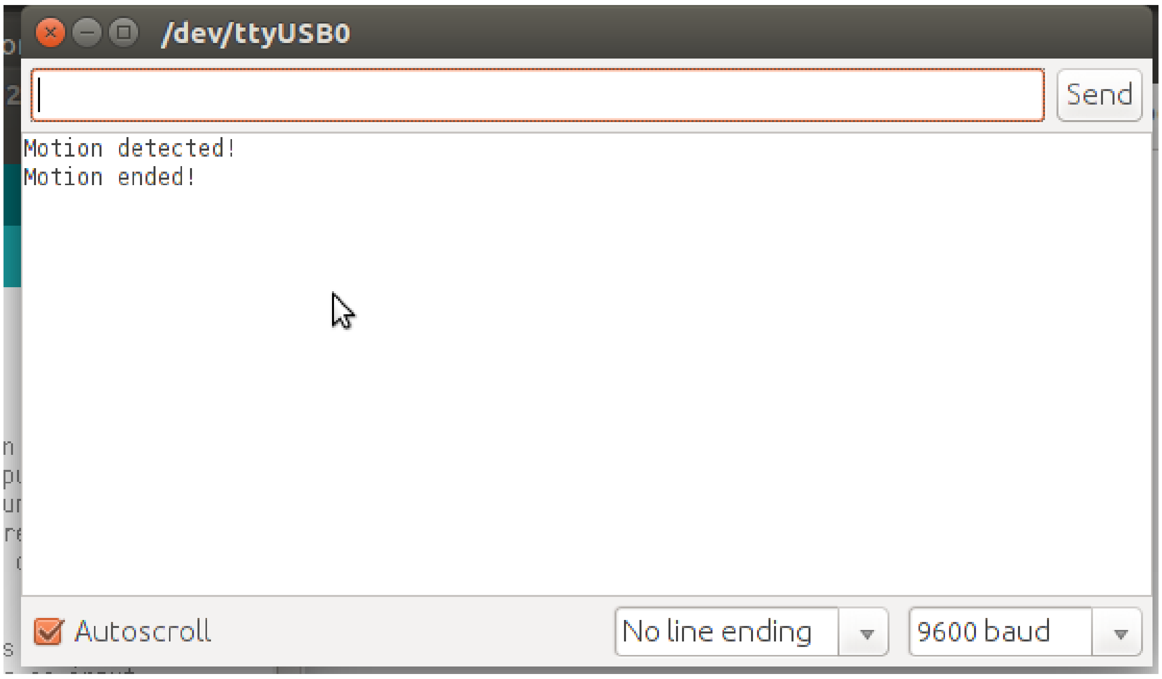
```
int ledPin = 13;                // the pin for the LED
int inputPin = 2;               // the input pin (for PIR sensor)
int pirState = LOW;             // we start, assuming no motion detected
int val = 0;                    // variable for reading the pin status
int pinSpeaker = 9;             //Set up a connect the buzzer on pin 9

void setup() {
  pinMode(ledPin, OUTPUT);          // declare LED as output
  pinMode(inputPin, INPUT);         // declare sensor as input
  pinMode(pinSpeaker, OUTPUT);
  Serial.begin(9600);
}

void loop(){
  val = digitalRead(inputPin);          // read input value
  if (val == HIGH) {                    // check if the input is HIGH
    digitalWrite(ledPin, HIGH);         // turn LED ON
    playTone(300, 160);
    delay(150);


    if (pirState == LOW) {
      // we have just turned on
      Serial.println("Motion detected!");
      // We only want to print on the output change, not state
      pirState = HIGH;
    }
  } else {
      digitalWrite(ledPin, LOW); // turn LED OFF
      playTone(0, 0);
      delay(300);
      if (pirState == HIGH){
      // we have just turned off
      Serial.println("Motion ended!");
      // We only want to print on the output change, not state
      // Inside Of The Serial Port
      pirState = LOW;
    }
  }
}
```

```
// Tell Arduino To Play Buzzer Tone
// duration in mSecs, frequency in hertz

void playTone(long duration, int freq) {
   duration *= 1000;
   int period = (1.0 / freq) * 1000000;
   long elapsed_time = 0;
   while (elapsed_time < duration) {
      digitalWrite(pinSpeaker,HIGH);
      delayMicroseconds(period / 2);
      digitalWrite(pinSpeaker, LOW);
      delayMicroseconds(period / 2);
      elapsed_time += (period);
   }
}        /*END OF THE PROGRAM*/
```

After you finish writing this code and making sure that you wrote it correctly by clicking verify. Do the following connection sequence:-

58-68, 46-69, 11-54, 67-57, 64-57, 14-57, Nano-PC

After you make sure that you connected everything properly and not confused any wire with another (this is a very common cause of errors so if something doesn't work check your wiring first), upload the code to the Arduino Nano board and then click on the serial monitor icon in the Arduino IDE. Now try moving your hand all over the PIR sensor (that ball shaped component next to the DC motor on the kit) or tap over it until you hear an alarm sound from the buzzer, look at the serial monitor, you should see something like this:-



```
Motion detected!
Motion ended!
```

Autoscroll ✓ | No line ending | 9600 baud

So now you have built a fully functional motion-detector alarm ! Congrats !

The End Of The Beginning !

If you have reached this far this means that you have gone from building a simple LED light circuit to making complex projects like building a robot or an alarm system ! Congratulations ! You have learned so much concepts already and acquired a whole lot of skills along the way. You may have reached the end of this booklet and you are probably done with the kit but that is just the beginning my friend. The Beginning of your wonderful and exciting journey into the world of electronics. While I hope that the kit and the booklet were beneficial to you, I regret that they lacked covering certain areas that should have been covered but I decided to not include them to make the beginning of this journey as simple and smooth as possible. But now that you have learned what you have learned I encourage you to explore these topics yourself, and don’t be afraid if something sounds so complicated at first. Ask around for instructors, teachers,..etc. Read books about any subject you don’t understand and I don’t mean text books, look for books that simplify the subject so that you can build your confidence as you learn more about it. Go online to websites and forums and groups where you will find thousands of people who are beginning their journey just like you, you’ll find those who are ahead of you as well who will be more than happy to guide you. If you don’t know where to look I have provided some links in the next section to help get you started. Remember that nobody was born an expert, everybody had to start at nothing and learn first so don’t panic if you get confused and overwhelmed since this happened to everyone, for all I know you could be the next greatest inventor who would build a new electronic circuit that will change the world! You can also check communities in your local area where there are people with similar interests who gather around to build stuff, look for maker spaces or Fab Labs and other creative places near you. Here are a few topic suggestions for you to consider learning as you embark on your next few steps in the path of self learning electronics: Closed-Loop and Open Loop Systems, Resistor Types, Inductors, Transformers, Voltage Divider circuit, The Thevenin Theorem, Amplifiers, Astable Multivibrators, NAND And NOR Logic Gates, Using Logic Gates To Build Adders, Decoders And Multiplexers, How To Use Breadboards, PCBs, PCB Design Software, Soldering Electronic Components, CNC Machines, 3D Printing, Internet Of Things (IOT), Beagle Bone Boards, Raspberry Pi Computers, The Life Of Nikola Tesla (The Father Of Modern Electronics).

References

A- Books

1- Electronics For Kids by Øyvind Nydal Dahl

2- Make: Electronics (Learning by Discovery) by Charles Platt

3- Soldering is Easy by Mitch Altman, Andie Nordgren and Jeff Keyzer (Comic Book)

It can be downloaded from this link

https://mightyohm.com/blog/2011/04/soldering-is-easy-comic-book/

4- Make: More Electronics: Journey Deep Into the World of Logic Chips, Amplifiers, Sensors, and Randomicity by Charles Platt

5- Getting Started with Arduino by Massimo Banzi

6- أردوينو ببساطة : دليلك العملي لتعلم أساسيات الإلكترونيات التفاعلية – تأليف عبد الله علي عبد الله

It can be downloaded from this link

http://simplyarduino.com/%D9%83%D8%AA%D8%A7%D8%A8-%D8%A7%D8%B1%D8%AF%D9%88%D9%8A%D9%86%D9%88-%D8%A8%D8%A8%D8%B3%D8%A7%D8%B7%D8%A9/

7- The Maker's Guide to the Zombie Apocalypse Defend Your Base with Simple Circuits, Arduino, and Raspberry Pi by Simon Monk

B- Websites

1- https://www.build-electronic-circuits.com/

2- https://www.instructables.com/

3- https://www.hackster.io/

4- https://makezine.com/tag/collinslab/

5- https://www.youtube.com/AddOhms (YouTube Channel)

6- https://www.youtube.com/user/jeriellsworth (YouTube Channel)

7- https://www.youtube.com/user/jamesbruton (YouTube Channel)

8- http://www.learningaboutelectronics.com/Articles/How-to-use-a-charged-capacitor-to-light-an-led

9- http://www.learningaboutelectronics.com/Articles/555-timer-pinout.php

10- http://www.circuitbasics.com/555-timer-basics-bistable-mode/

11- http://www.circuitbasics.com/555-timer-basics-astable-mode/

12- https://www.arduino.cc/en/Tutorial/

13- http://simplyarduino.com/

14- http://www.multiwingspan.co.uk/as2.php?page=truth

15- http://www.multiwingspan.co.uk/arduino.php?page=led7

16- http://www.multiwingspan.co.uk/arduino.php?page=nand

17- http://www.makeuseof.com/tag/arduino-traffic-light-controller/

18- http://www.multiwingspan.co.uk/arduino.php?page=buzzer1

19- http://www.instructables.com/id/Arduino-PIR-Sensor-Alarm-SIMPLE-VERSION/

20- https://www.youtube.com/channel/UChtY6O8Ahw2cz05PS2GhUbg (YouTube Channel)

21- http://www.hobbyprojects.com/tutorial.html

22- https://ilt.seas.harvard.edu/students/lectures/printablelecture/?PHPSESSID=&lectureID=8996

23- http://slideplayer.com/slide/7350354/

24- https://learn.sparkfun.com/tutorials/diodes

25- https://www.petervis.com/electronics/diode_symbol/diode_symbol.html

26- https://www.elprocus.com/rectifier-diode-working-applications/

27- http://farhek.com/jd/1192u9a/symbol-tutorial/59z29j/

28- https://www.myledlightingguide.com/blog-what-is-an-led

29- http://www.mainbyte.com/ti99/electronics/led.html

30- https://learn.sparkfun.com/tutorials/light-emitting-diodes-leds

31- https://en.wikipedia.org/wiki/P%E2%80%93n_junction

32- https://startingelectronics.org/beginners/components/LDR-photoresistor/

33- https://eeecommunity.blogspot.com.eg/2015/10/difference-between-electrolytic.html

34- http://www.555-timer-circuits.com/

35- http://learn.robotgeek.com/classroom/259-rgc-arduino-introduction.html

36- http://kkfscs.weebly.com/arduino-basic-sketch.html

37- http://www.instructables.com/id/Arduino-PIR-Sensor-Alarm-SIMPLE-VERSION/

38- http://www.makeuseof.com/tag/arduino-traffic-light-controller/

39- http://www.learningaboutelectronics.com/Articles/How-to-build-a-touch-sensor-circuit

40- https://www.build-electronic-circuits.com/electronics-for-beginners/

41- https://www.build-electronic-circuits.com/pnp-transistor/

42- www.circuitbasics.com/555-timer-basics-bistable-mode/

43- http://www.kkhsou.in/main/EVidya2/computer_science/logic_gates.html

C- Smart Phone Apps

1- ElectroDroid

2-Logic Gates - Electronic Simulator and learning