BASIC EMBEDDED SYSTEMS AND AUTO-TRONICS

For more information or feedback, contact:
Joseph Habiyaremye, Lecturer
Electrical and Electronics Engineering Department
Rwanda Polytechnic, IPRC Kigali
P.O. Box 6579 Kigali-Rwanda
KK 15 Rd, Kigali
Josephu2020@ieee.org/hjoseph@iprckigali.ac.rw
www.iprckigali.ac.rw

Introduction

This document will deal with autotronics and embedded system as a tool for industrialization and promoting modern automotive technology for sustainable development .

The Autotronics is referred to as modern automotive technology and also commonly known as Automotive Mechatronics Autotronics is the combination of automobile and electronics. While Embedded Systems is defined as a combination a hardware and a software for achieving a given application.
Studying this course requires the knowledge about electronics and measurement .It also requires some programming skills.

Objectives: The objective of Automobile Engineering (Autotronics) and embedded systems is to develop and understand the principles of conversion in design, construction and working of mechanical systems and electronic systems in automobiles for Staffs from Mechanical Engineering Department

Outcomes: - Staffs from Mechanical Engineering Department are trained and are able to innovate while maintaining or troubleshooting nowadays cars. The gained skills will later be transferred to students.

A Note from the Author

This document shall be used at Rwanda Polytechnic/IPRC Kigali, for our Mechanical engineering department Staffs and Students. Specifically, it shall be used in automobile program. I encourage others to make use of this manual for their own work and to build upon it. If you do add to this effort, I would appreciate a notification

Table of Contents

List of Figures

Chap 1: Basics of Automotive Electronics/ basic electronics

Electronics components

Figure 1: Different electronics components

Electronic components are basic electronic element or electronic parts usually packaged in a discrete form with two or more connecting leads or metallic pads.

Electronic Components are intended to be connected together, usually by soldering to a printed circuit board (PCB), to create an electronic circuit with a particular function (for example an amplifier, radio receiver, oscillator, wireless). Some of the main Electronic Components are: resistor, capacitor, transistor, diode, operational amplifier, resistor array, logic gate etc.

Types of Electronic Components

Electronic Components are of 2 types*: Active and Passive Electronic Components.*

- **Passive electronic components** are those that do not have gain or directionality. They are also called Electrical elements or electrical components. e.g. resistors, capacitors, diodes, Inductors.
- **Active components** are those that have gain or directionality. e.g. transistors, integrated circuits or ICs, logic gates.

Electronic Components and Their Functions

1. **Terminals and Connectors**: Components to make electrical connection.
2. **Resistors**: Components used to resist current.
3. **Switches**: Components that may be made to either conduct (closed) or not (open).
4. **Capacitors**: Components that store electrical charge in an electrical field.
5. **Magnetic or Inductive Components**: These are Electrical components that use magnetism.
6. **Network Components**: Components that use more than 1 type of Passive Component.
7. **Piezoelectric devices, crystals, resonators**: Passive components that use piezoelectric. effect.
8. **Semiconductors**: Electronic control components with no moving parts.

9. **Diodes**: Components that conduct electricity in only one direction.
10. **Transistors**: A semiconductor device capable of amplification.
11. **Integrated Circuits or ICs**: A microelectronic computer electronic circuit incorporated into a chip or semiconductor; a whole system rather than a single component

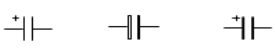
Electronic Components Abbreviations

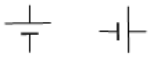
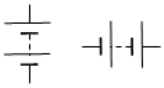
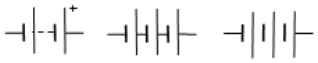
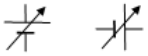
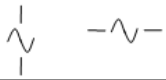
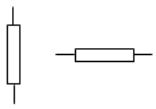
Here is a list of Electronic Component name abbreviations widely used in the electronics industry:

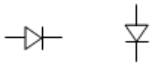
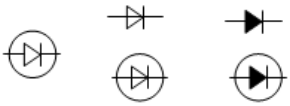
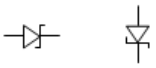
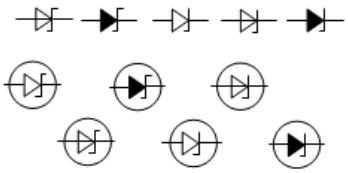
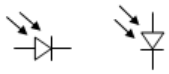
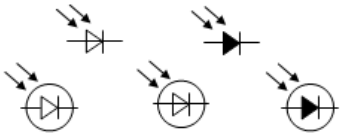
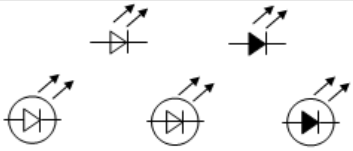
- **AE**: aerial, antenna
- **B**: battery
- **BR**: bridge rectifier
- **C**: capacitor
- **CRT**: cathode ray tube
- **D or CR**: diode
- **F**: fuse
- **GDT**: gas discharge tube
- **IC**: integrated circuit
- **J**: wire link
- **JFET**: junction gate field-effect transistor
- **L**: inductor
- **LCD**: Liquid crystal display
- **LDR**: light dependent resistor
- **LED**: light emitting diode
- **LS**: speaker
- **M**: motor
- **MCB**: circuit breaker
- **Mic**: microphone
- **Ne**: neon lamp
- **OP**: Operational Amplifier
- **PCB**: printed circuit board
- **PU**: pickup
- **Q**: transistor
- **R**: resistor
- **RLA**: RY: relay
- **SCR**: silicon controlled rectifier
- **FET**: field effect transistor
- **MOSFET**: Metal oxide semiconductor field effect transistor
- **TFT**: thin film transistor(display)
- **VLSI**: very large scale integration
- **DSP**: digital signal processor
- **SW**: switch
- **T**: transformer
- **TH**: thermistor
- **TP**: test point
- **Tr**: transistor
- **U**:integrated circuit
- **V**: valve (tube)
- **VC**: variable capacitor
- **VFD**: vacuum fluorescent display
- **VR**: variable resistor

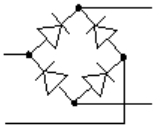
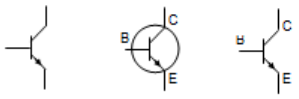
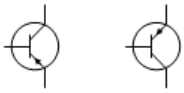
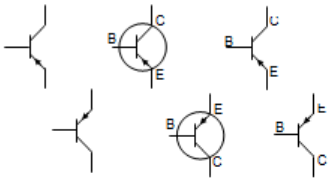
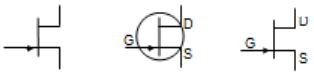
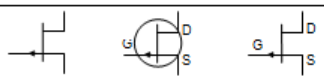
- **X**: crystal, ceramic resonator
- **XMER**: transformer
- **XTAL**: crystal
- **Z**: zener diode

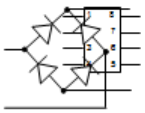
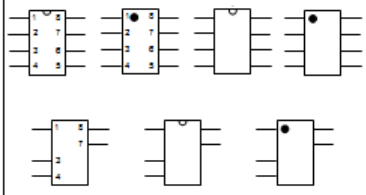
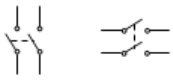
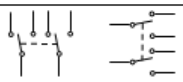
Some symbols used in electronics

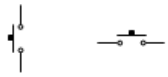
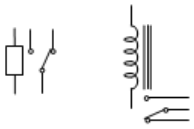
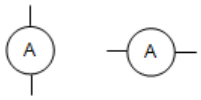
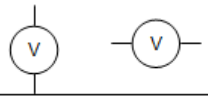
Component group	Component	Recommended symbol	Alternative symbols	Notes
capacitors	(non-polarised) capacitor			
	variable capacitor			
	polarised capacitor, electrolytic capacitor			
transformers	iron-cored transformer (one secondary winding)			
	iron-cored transformer (one secondary winding – centre-tapped)			
	iron-cored transformer (two secondary windings)			

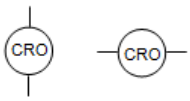
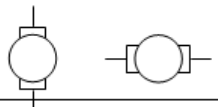
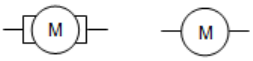
Component group	Component	Recommended symbol	Alternative symbols	Notes
sources	cell			
	battery, DC power supply			
	variable DC power supply			
	AC power supply			
resistors	resistor			Alternative symbol now regarded as outdated.

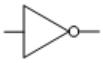
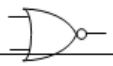
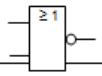
Component group	Component	Recommended symbol	Alternative symbols	Notes
diodes	junction diode			Additional alternative symbols for **all diodes** include those where the single straight-line section of the symbol is shown as a heavier line.
	Zener diode			
	photo diode			
	light-emitting diode (LED)			

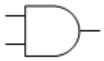
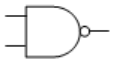
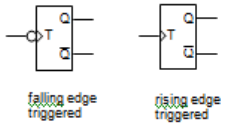
Component group	Component	Recommended symbol	Alternative symbols	Notes
	diode bridge	 (Note that in an earlier version of this table, this symbol was incorrectly represented.)		There are alternative forms using the other alternative symbols for the **junction diode** given above.
transistors	npn transistor			Additional alternative symbols for **all transistors** include those where the vertical straight-line section of the symbol is shown as a heavier line.
	pnp transistor			Both orientations of the recommended symbol are used depending on whether the *emitter* (E) is at the bottom or top.
	n-type junction field effect transistor (NJFET)			
	p-type junction field effect transistor (PJFET)			

Component group	Component	Recommended symbol	Alternative symbols	Notes
integrated circuits	integrated circuit (IC)			This example shows an IC with 8 connections. Others occur, e.g. 14 and 16. Under alternative symbols identifying marks are shown to indicate the numbering of the connections. Other conventions show only the connections used in a given circuit, here 1, 3, 4, 7 and 8.
switches	single pole, single throw (SPST) switch			
	single pole, double throw (SPDT) switch			
	double pole, single throw (DPST) switch			
	double pole, double throw (DPDT) switch			

Component group	Component	Recommended symbol	Alternative symbols	Notes
	normally open (NO) switch			
	normally closed (NC) switch			
	relay			
meters	ammeter			
	voltmeter			

Component group	Component	Recommended symbol	Alternative symbols	Notes
	galvanometer			The alternative symbol is not recommended as it is sometimes used to indicate a *generator*.
	cathode ray oscilloscope (CRO)			Neither of these encountered symbols is widely used.
amplifiers	voltage amplifier			
	operational amplifier (op amp)			On the alternative symbol the two vertical connections are shown when it is connected to the power supply.
transducers	motor			

Component group	Component	Recommended symbol	Alternative symbols	Notes
	microphone			
	loudspeaker			
logic gates	NOT or invert gate			
	OR gate			
	XOR (exclusive OR) gate			
	NOR gate			

Component group	Component	Recommended symbol	Alternative symbols	Notes
	AND gate			
	NAND gate			
flip-flop	T (toggle) flip-flop			
external connections	earth			
	aerial			
circuit connections	non-connected leads			Alternative symbol regarded as outdated.

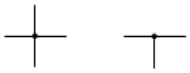
Component group	Component	Recommended symbol	Alternative symbols	Notes
	connected leads			
	dot for junction of connected leads			

Figure 2: Symbol of some electronics/electrical components

RESISTOR

A *resistor* is a component that resists the flow of current. It's one of the most basic components used in electronic circuits.

Figure 3: Resistor

Where to locally buy this?: http://naradaelectronics.rw/

CAPACITOR

Next to resistors, capacitors are probably the second most commonly used component in electronic circuits. A *capacitor* is a device that can temporarily store an electric charge.

Figure 4: capacitor

Where to locally buy this?: http://naradaelectronics.rw/

DIODES

A *diode* is a device that lets current flow in only one direction. A diode has two terminals, called the *anode* and the *cathode*. Current will flow through the diode only when positive voltage is applied to the anode and negative voltage to the cathode. If these voltages are reversed, current will not flow.

Figure 5: Diodes

Where to locally buy this?: http://naradaelectronics.rw/

TRANSISTORS

A *transistor* is a three-terminal device in which a voltage applied to one of the terminals (called the *base*) can control current that flows across the other two terminals (called the *collector* and the *emitter*). The transistor is one of the most important devices in electronics.

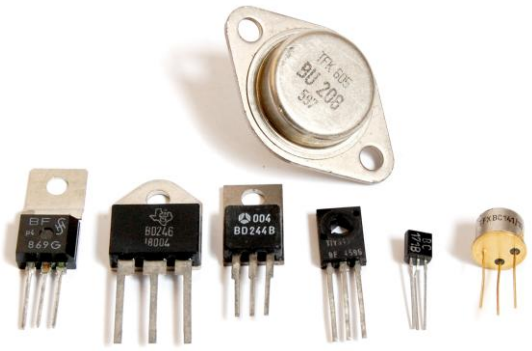


Figure 6: Transistor
Where to locally buy this?: http://naradaelectronics.rw/

INTEGRATED CIRCUITS

An *integrated circuit* is a special component that contains an entire electronic circuit, complete with transistors, diodes, and other elements, all photographically etched onto a tiny piece of silicon. Integrated circuits are the building blocks of modern electronic devices such as computers and cell phones.

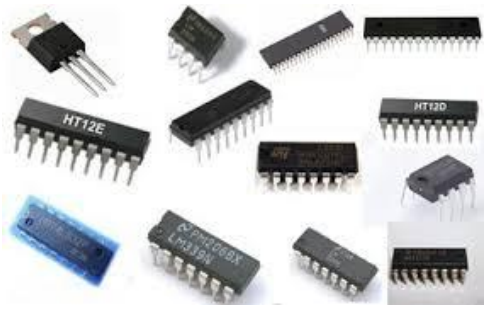


Figure 7: Integrated Circuits
Where to locally buy this?: http://naradaelectronics.rw/

INDUCTORS

An inductor is also referred as AC resistor which stores electrical energy in the form of magnetic energy. It resists the changes in the current and the standard unit of inductance is Henry. Capability of producing magnetic lines is referred as inductance.

Figure 8: Inductor
Where to locally buy this?: http://naradaelectronics.rw/
Hands on: all types of electronics components will shown to trainees .

Chap 2.Electronics components types, identification ,fault finding and repair

Electronic devices and components of modern electrical circuits take many appearances. Looking into a radio or calculator,… to see the working parts will baffle most people. For you to have a better understanding of electronic devices and components, you need to be able to recognize some of the more common parts. Because of their great variety of colors, shape and sizes it is possible and easy to identify them.

Resistor

Figure 9: Resistors identification

Standard resistors are NOT polarised and can be placed in the PCB either way round.

The most common have four bands. Bands 1 and 2 identify the first two digits, band 3 is the Multiplier and band 4 is the tolerance (normally Gold 5%)

The Multipliers are:

R = x 0

K = x 1,000

M = x 1,000,000

The kit parts list might contain for example, a 4K7 resistor. The K refers to the number multiplier (K = x 1,000). The position of the K between the 4 and 7 identifies the position of the decimal point so we have

4.7 x 1,000 = 4,700 Ohms (with a colour code of Yellow, Violet, Red)

Another example is say 56R. This gives us 56 x 0 = 56 Ohm resistor with colour code Green, Blue, Black.

You will find some five band resistors in our kits and this colour code is likely to be associated with the more precision 1% and 2% types. The work in a similar way to the four band but the first three bands are used to determine the numbers, band 4 is the multiplier and band 5 is the tolerance - see table below for details. Your "garden variety" 5% general purpose types will be four band resistance codes.

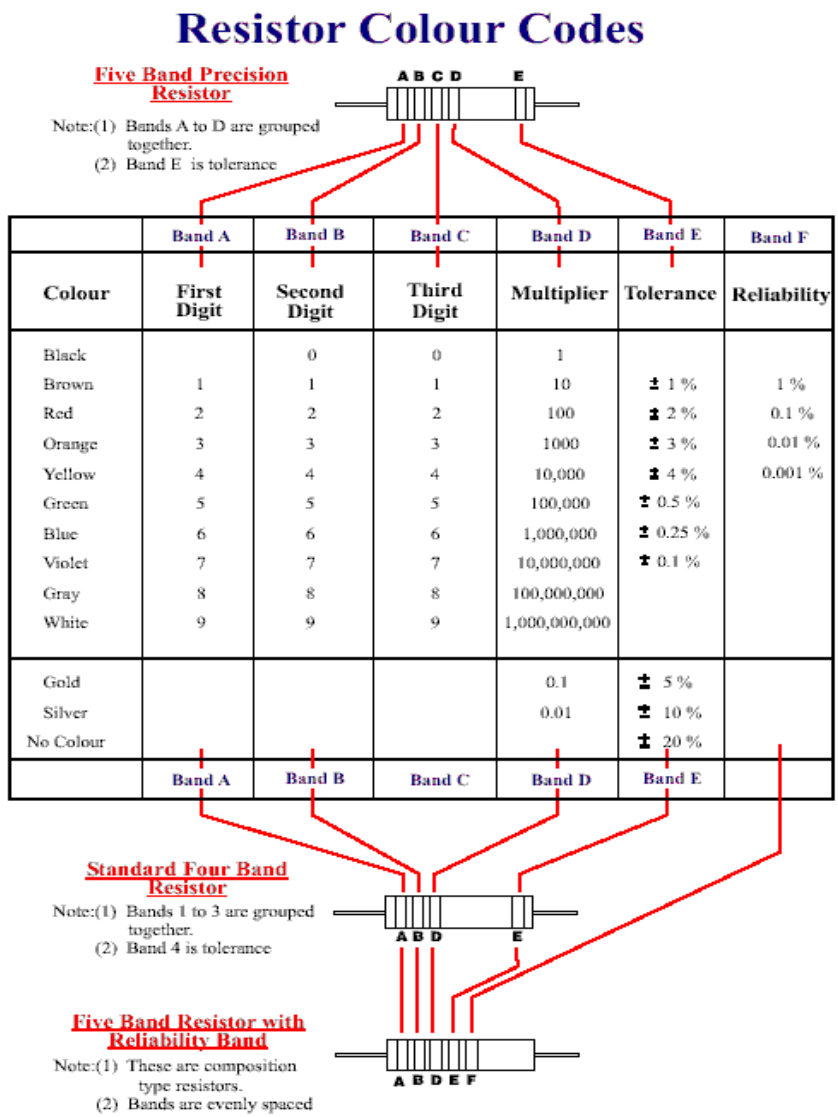


Figure 10: Resistor Color Code

Resistor Networks

Many resistor networks have an orientation marked with a dot at the end of one side. This dot must be marks pin 1. This pin is marked on the PCB by a square box at one end of the main b

Identification

Resistors have 2 leads and very commonly their resistance is denoted by a number of coloured bands

Surface mount resistors are usually black, rectangular, and with a silvery solder pad on each end. They range in size from a few millimetres down to a fraction of a millimetre.

A power resistor is larger than a regular one allowing it to dissipate the required amount of heat. Often, its value will be printed on it instead of being colour coded.

Fault-finding and Repair

Resistors are usually very reliable. When they fail, generally through overheating, it's almost always as a result of some other component failing and causing too much current to flow. In

poorly designed equipment having inadequate provision for heat to escape, moderate overheating over a long period of time may cause a failure.

Potentiometer

A potentiometer (or pot, for short) is simply a resistor with a 3rd connection that can be moved to any point along its length, so as to tap off any desired portion of the total resistance.

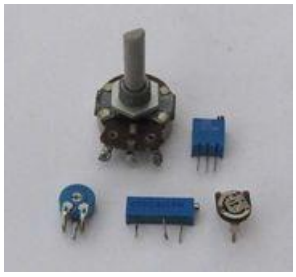


Figure 11: Potentiometer

Identification

Potentiometers are very commonly used for the volume control in audio equipment (though being superseded by digital controls). These have a spindle with a front panel knob attached, or sometimes a knurled wheel with the edge exposed for adjustment. Twin-gang potentiometers consisting of two mounted on one spindle are often found in stereo equipment for controlling the volume of both stereo channels.

Small potentiometers with a slotted screw head are often found inside equipment for one-time adjustment during manufacture and testing.

Fault-finding and Repair

Potentiometers are much less reliable than fixed resistors. The track can get worn out or crack, or the pressure of the slider on the track can weaken. Sometimes the pressure can be increased by bending the metal slider, though pots are not normally designed to be taken apart and are usually best replaced for a lasting fix.

A quick fix which sometimes works is to squirt switch cleaning fluid into the casing through any gaps or slots you can see, such as beneath the terminals, then repeatedly turn the knob from one end of its travel to the other.

If you find a slotted screw head pot inside a piece of equipment, never adjust it unless you know what it's for and how to find the correct position. Even then, it's worth marking the original position before starting so you can always return to it

Capacitors

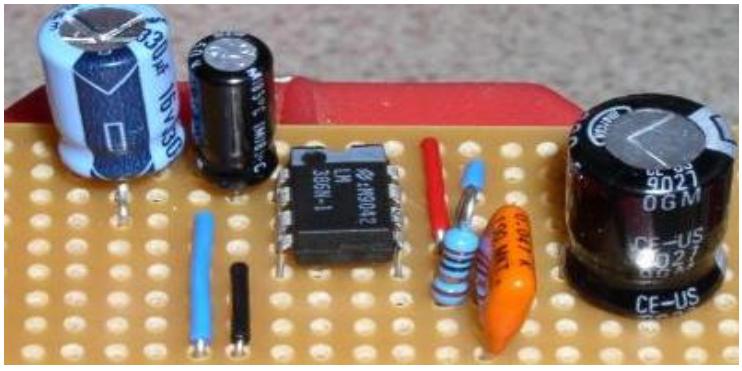


Figure 12: Capacitors

Capacitors values may be designated in micro-farad (uF), nano-farad (nF) or pico-farad (pF) and there is a certain amount of overlap. There are many instances where two components of the same value may be quoted in different ways: one may be quoted in terms of pico-Farads and another in terms of nano-Farads. For example 100 nF is the same as 0.1 micro Farads. The chart below quickly helps show what which values are the same and how many nano-Farads makes a micro-Farad and so forth. It can be used as a quick reference guide for capacitors, or any other electronic component when looking at different items from different manufacturers.

Capacitor conversion table for pico-Farads, nano-Farads, and micro-Farads

Table 1: Capacitor unit conversion

micro-Farads (µF)	Nano-Farads (nF)	Pico-Farads (pF)
0.000001	0.001	1
0.00001	0.01	10
0.0001	0.1	100
0.001	1	1000
0.01	10	10000
0.1	100	100000
1	1000	1000000
10	10000	10000000
100	100000	100000000

Using the capacitor conversion table it is possible to quickly check the relationship between two capacitors with different markings. In this way it is possible to see whether they capacitor values are the same or not.

Electrolytic & Tantalum Capacitors

Electrolytic and tantalum capacitors are POLARISED devices which means they must be inserted with the correct polarity.

Electrolytics have one lead longer than the other and this denotes the POSITIVE lead and on the opposite side of body is a black band with a minus sign.

Tantaulm beads have a + sign next to the positive lead.

Ceramic Capacitor Codes

How to read a Ceramic capacitor with Numeric coding.

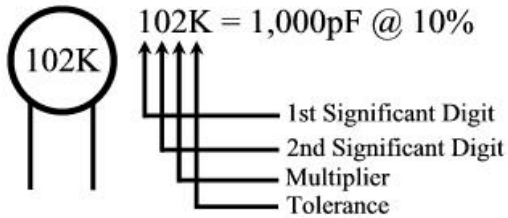


Figure 13: Ceramic capacitor coding

Multiplier Table (Ceramic)

Table 2: Multiplier for Ceramic capacitor

Number	Multiply By (Additional # of Zeros)
0	None (0)
1	10 (1)
2	100 (2)
3	1,000 (3)
4	10,000 (4)
5	100,000 (5)
6	1,000,000 (6)

Common Temperature Coefficient Codes (Ceramic)

Table 3: Temperature Coefficient Code for Ceramic capacitor

Code	Tolerance
C	±0.25pF
J	±5%
K	±10%
M	±20%
D	±0.5pF
Z	+80% / -20%

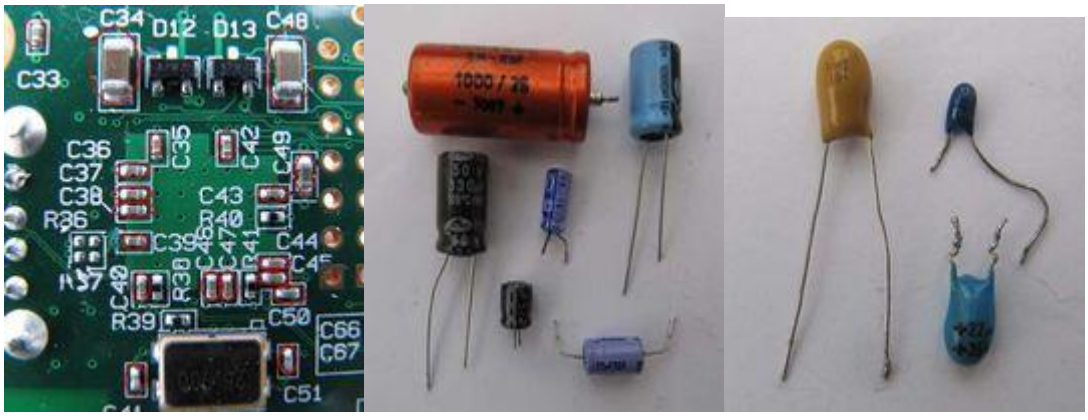


Figure 14:chip, Tatantalum, Ceramic Capacitors

Capacitors are used whenever the circuit designer needs to smooth out fluctuations, or when it is required to allow fluctuations (e.g. an audio signal) to flow from one part of a circuit to another whilst blocking any net flow.

Identification

Like resistors, capacitors have just two connections, but they come in a wide range of shapes and sizes. They usually have their capacitance and voltage rating printed on them, and for some types, a maximum temperature.

Low value surface mount capacitors are usually grey or buff in colour, rectangular, and with a silvery solder pad on each end. They are typically a few millimetres in length.

Electrolytic capacitors are very frequently used where a high value of capacitance is required. Much the most common are aluminium types, which can be recognised by the cylindrical aluminium case, usually with a plastic film cover. One lead is marked negative ("-") on the adjacent side or end of the case.

Tantalum capacitors are a higher quality (and more expensive) type of electrolytic capacitor using tantalum instead of aluminium. They come as a resin coated bead. Usually, the positive lead is marked "+".

Capacitors connected directly to the mains require a special safety rating. This is Class X for those connected across the mains supply, where a failure could present a fire hazard, and Class Y for those connected between the mains and ground, where the failure could result in an electric shock hazard. Such capacitors must always be replaced with ones of the same class and at least the same voltage rating.

Fault-finding and Repair

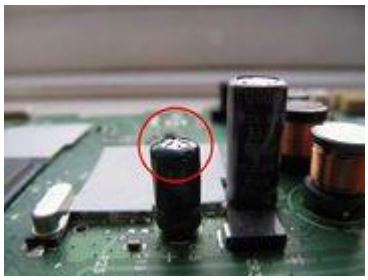


Figure 15: Electrolytic capacitor

Capacitors are usually very reliable except for electrolytic types, which are one of the commonest causes of failure in electronic equipment.

In an electrolytic capacitor, the insulating layer consists of an electrochemically formed film of aluminium oxide with a thickness of only a matter of millionths of a millimetre. This can deteriorate after a long period of disuse (many years) or a shorter period close to or beyond its maximum voltage and/or temperature rating. Also, the liquid used to form the insulating layer can dry out. Poor quality electrolytic capacitors are quite often seen which have failed within their ratings.

A failing electrolytic capacitor can often be recognised as a build up of internal pressure may cause the top to bulge, or the capacitor no longer to sit flush with the board, or electrolyte to leak from the bottom. At this stage it probably won't be performing well, causing the equipment to malfunction. If not replaced, it may even explode. However the fact that an electrolytic capacitor shows no visible sign of deterioration is by no means a reliable indicator that it's good.

Figure 16: Capacitor Tester

The simplest reliable method of testing an electrolytic capacitor is with an ESR (Equivalent Series Resistance) tester. A basic one with a graphical display but without a case is available very cheaply from Far Eastern sellers. This is an excellent investment as it will also identify and test many other types of component. A good electrolytic capacitor should show an ESR of a fraction of an ohm, and a vloss (another measure given by these testers) of a fraction of a percent.

If an electrolytic capacitor has to be replaced it's very important to fit the replacement the right way round (the "+" or the "-" marking on the same side) as otherwise the electrolytic forming process will be reversed and it will very rapidly fail.

Also, it's always a good idea to replace it with one with a higher voltage and/or temperature rating as the original may have been under-rated. Never ever use a lower rated replacement. If a replacement with the same capacity is not available, a higher value up to twice the original will almost invariably work well, or possibly even better, as there is in any case considerable variation in the capacitance of identically marked electrolytic capacitors.

Diodes

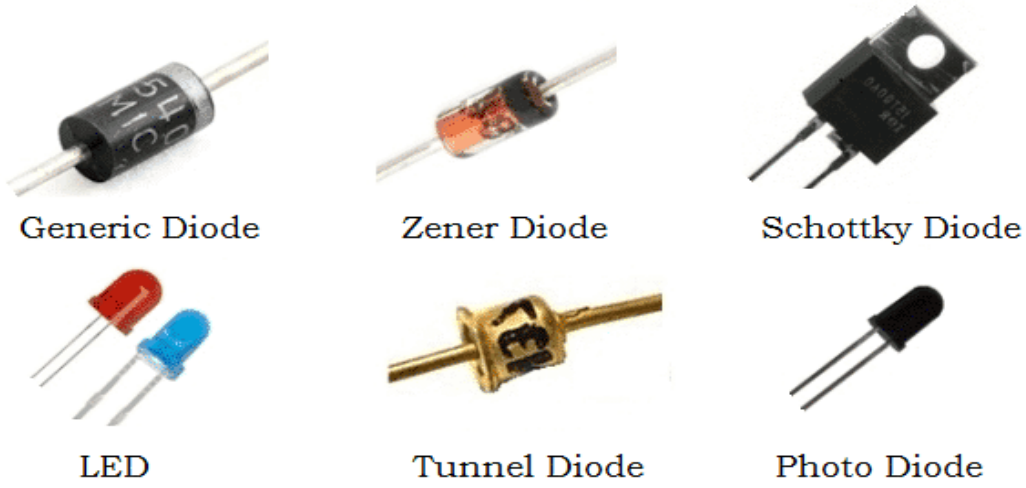


Figure 17: Different Types of Diodes

Where to locally buy this?: http://naradaelectronics.rw/

All diodes are polarised and must be inserted with the correct orientation to avoid damage to the circuit.

Signal &Y Power Diodes

Diodes are polarised and need to be inserted with the band marked at one end of the body (the cathode) to the band marked on the PCB.

Diodes come all all shapes and sizes but the ones used most often in our kits are 1N4148 signal diodes (orange body with blank band) and 1N4004 rectifier diodes (black with silver band). Both are marked on the body with their part number.

Zener Diodes

You may also find Zener diodes. These look similar to the 1N4148s and are easily muddled up! The voltage is marked on the body e.g. a 5.6V Zener will have 5V6 on the body.

Light Emitting Diodes (LEDs)

Light Emitting Diodes (LEDs) are found in many kits. They are normally red in colour but green and yellow are also sometimes used. LEDs are polarised and must be inserted the correct way.

The negative (cathode) lead is indicated in most cases by a "flat" on that side of the body. This flat is also indicated on the PCB legend for orientation purposes. You may also find that the positive (anode) lead is indicated on the board, corresponding to the long leg of the LED.

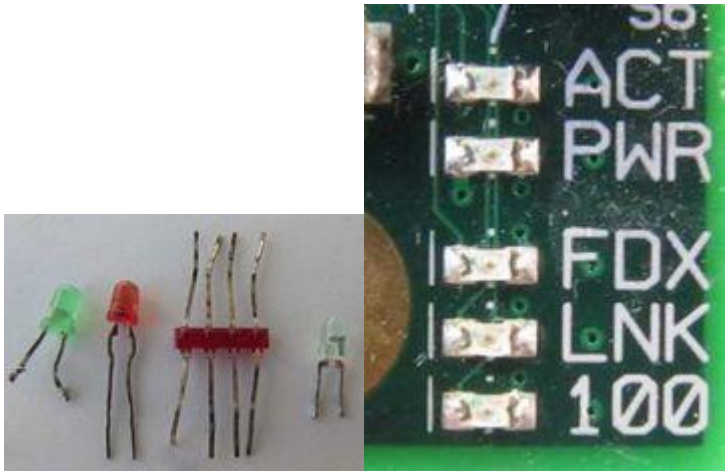


Figure 18: LED, SMD LED

Identification

Diodes have 2 leads like resistors, but generally have a painted band around one end. This is the end that positive current flows out of. A very common type of rectifier diode has designation 1N400n, or a higher power version, 1N540n, where "n" is a digit indicating the maximum voltage it can be used for. A clutch of 4 of them or a bridge rectifier in a single 4-legged package will often be seen close to an iron-cored mains transformer, or close to the mains input.

Signal diodes and zener diodes often come in a very small glass package.

LEDs are easily recognised by their clear or coloured transparent package. Sometimes (for example in toys) a LED will have a silicon chip incorporated in the package to make it flash or flicker, or to eliminate the need for an external resistor. Two or more LEDs are sometimes incorporated in a single package to give multiple colours.

Fault-finding and Repair

Very many digital test meters, even the cheapest, will have a diode test function. Connect the red test lead to the end of the diode with the painted band and the black lead to the other end. Most diodes should give a reading of around 0.7V, except for Shottky diodes for which it will be more like 0.4V. Anything much closer to zero or a low reading with the leads reversed indicates that the diode has failed. A LED on the other hand would give a much higher reading of between 1.8V and 4V, in many cases off the range of a test meter.

LEDs rarely fail unless seriously abused. They can be tested by connecting them across a battery of between 3V and 9V in series with a resistor between 330Ω an 1kΩ. The negative lead is often designated by a flat on the side of the package.

Inductor

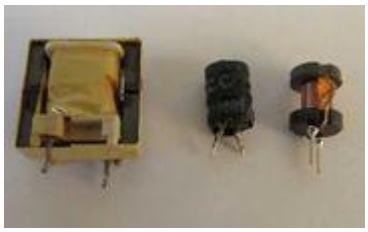


Figure 19:Inductors.

An inductor simply consists of a coil of wire. When a current flows it creates a magnetic field, which stores energy. By winding it around a magnetic material such as iron or ferrite this gets magnetised, greatly increasing the amount of stored energy.

Whereas a capacitor stores energy in as electrical charge and can be used to smooth out variations in voltage, an inductor stores energy as magnetic flux and tends to smooth out variations in current flow. In fact, there is a beautiful symmetry between the mathematical equations describing capacitors and inductors.

If you combine an inductor and a capacitor in a circuit, the mathematical symmetry blossoms and something rather special happens. A voltage on the capacitor tries to drive a current through the inductor, but once the current gets going the inductor tries to keep it going, and ends up driving the charge onto the other side of the capacitor. So it flows backwards and forwards at a very regular rate, exactly like a child swinging back and forth on a swing. By using a variable capacitor (or a variable inductor), the rate can be altered. This is how nearly all older AM and FM radios tune in the station you want.

Inductance is measured in Henrys (H), milliHenrys (mH - thousandths of a Henry), or microHenrys (μH - millionths of a Henry).

Identification

The smallest value inductors consist of no more than a coil of thick wire standing up from the circuit board. Some small inductors consist of a toroid of ferrite with the coil of wire wound around it, and are easily spotted. In others, the coil is would around a ferrite core shaped like a cotton reel, which may be fitted snugly inside a hollow cylinder of ferrite. For large values of inductance a laminated iron core is used. This is rarely seen in reasonably modern equipment, but vintage valve radios often used two large capacitors and an iron-cored inductor to smooth the rectified mains.

An inductor frequently has no markings on it.

Fault-finding and Repair

There is very little to go wrong in an inductor apart from possibly a badly soldered joint. A very heavy current could cause an inductor to overheat or burn out, but probably not before much damage had been done elsewhere in the circuit.

Transformer

Figure 20:small toroidal ferrite transformer with 3 windings.

Figure 21:A mains transformer.

A transformer is simply an inductor with two (or more) coils of wire.

An electric current always creates a magnetic field which loops around the current, and a change in the magnetism looping through a circuit generates a voltage in that circuit. So in a transformer, we apply power to one coil of wire, the primary, and the magnetic flux which this creates induces a voltage in the other coil(s), the secondary(s). But it only works while the magnetic field is changing, and so a transformer can only be used for AC, not DC.

Transformers are very useful for two reasons:

- If the secondary coil has more or fewer turns than the primary, the voltage induced in it will be greater or less than that applied to the primary, in proportion.
- Since the only connection between the primary and the secondary is magnetic, they are electrically isolated from each other. This can be useful for safety reasons, or where the circuit designer needs to block a net flow of current from one part to another.

Identification

Figure 22:A toroidal mains transformer.

If you know how to identify an inductor, then a transformer looks exactly the same except that it has at least 3 wires coming out of it, and nearly always 4 or more.

Older mains powered electronic equipment almost always contains an iron cored mains transformer, which is easy to spot. Good quality audio equipment sometimes uses a toroidal mains transformer as this type produces less stray magnetic field and hence less background hum in the audio output. Newer equipment tends to use a much smaller transformer with a ferrite core.

Fault-finding and Repair

Mains transformers may be required to handle a substantial amount of power, and so in fault conditions they can become very hot. If this results in a breakdown of the insulation between two adjacent turns of either the primary or the secondary, these turns will act like a short-circuited secondary and become very hot indeed.

Rewinding a burnt out mains transformer is not difficult, but rarely would be worth the considerable time and patience required.

Transistors

Figure 23:Transistors

Transistors have 3 connections. They are used for amplifying a weak signal, or for switching current on and off. In this section we will also cover thyristors and triacs, which are related.

There are 2 types of transistor:

- **Bipolar or Junction transistors**. The 3 terminals are known as *emitter*, *base* and *collector*. Current flows from the emitter to the collector, but is only able to in proportion to a much smaller current that you feed into the base.
- **Field effect transistors (FETs).** In these, the 3 terminals are known as *source*, *gate* and *drain*. Current flows from the source to the drain, but is controlled by the voltage you apply to the gate.

Each type comes in two complementary varieties or *polarities*: bipolar transistors are either NPN or PNP, and FETs are either N-channel or P-channel. The direction of current flow and the voltages are reversed between the two varieties.

Additionally, FETs come in several types, such as enhancement versus depletion mode, and junction (jFET) versus insulated gate FET.

A thyristor is essentially a PNP and NPN transistor merged into one, and acts like a switch that latches on. A triac does the same job but is designed to work on AC. Dimmer switches contain a triac.

Identification

Transistors and their cousins are quite easily recognisable as almost the only components having just 3 legs.

Practically all are marked with their type code, not to be confused with a manufacturing date code or other markings. Looking up the type code with a web search engine and consulting the datasheet is the only visual means of distinguishing bipolar and field effect transistors and determining their polarity, or identifying a thyristor or a triac, and identifying the 3 leads.

A very common family of voltage regulator is only visually distinguishable from a transistor by its type number. This is usually 78 or 79, optionally a letter, and followed by 2 digits, e.g 78L05. These are integrated circuits, discussed later.

Fault-finding and Repair

Many digital test meters, even some of the cheapest, have a transistor test function for bipolar transistors, often marked h_{FE}, and with separate sockets or switch positions for PNP and NPN. Identify the emitter, base and collector, and push them into the appropriate holes in the test socket. A reading can be expected of anything between around 20 and 500.

Alternatively, if your test meter only has a diode test facility, a bipolar transistor can be tested as two diodes. Connect the positive test lead to the base and the negative to the emitter and collector in turn. Both should give a reading of around 0.7V for an NPN transistor or an out of range indication for a PNP. Now connect the negative test lead to the base. This time you should get 0.7V for a PNP or out of range for an NPN. In the case of older germanium transistors from a vintage radio, expect a reading of around 0.3V instead of 0.7V. If any of the readings is much lower than expected, or you get anything but out of range with the test leads between the emitter and collector (whilst not touching the base) then the transistor is dead.

When replacing a transistor it's very important to install the replacement exactly like the original.

An audio amplifier often uses a complementary pair of PNP and NPN transistors in the output stage. These sometimes develop a short circuit between the emitter and collector resulting in a blown fuse. In replacing the transistors, make sure you clean the heat sink and replace any heat conductive paste or insulating washer. If replacing a complementary pair, take great care not to get them muddled up.

Field effect transistors tend to be more robust, but a simple test is to connect a test meter between the gate and the source and drain in turn. An infinite resistance should be indicated, except for a jFET where it will behave like a diode.

Integrated Circuits

Figure 24:Integrated circuits.

The techniques for fabricating multiple electronic components and their interconnections on a silicon chip to form a "integrated circuit" or IC were first developed in the 1960s. They underpin the whole of the modern electronics and information technology revolution and earned the key players a Nobel prize for physics in the year 2000.

There are many thousands of different types of integrated circuits (or ICs) ranging from the simplest types, a few of which have been available for up to 30 - 40 years, up to the latest and most complex, containing billions of circuit elements.

Identification

A few ICs have just 2 or 3 connections. The commonest in this class are voltage regulators, visually very similar to power transistors. They generally have a type designation printed on them consisting of 2 digits "78" or "79", optionally a letter, then 2 more digits indicating the voltage, e.g. 78M05. The 78 types produce a positive regulated voltage, and the 79 types a negative.

Most ICs have anything from 8 to many hundreds of connections (generally known as pins), usually in a black plastic case. A notch at one end of the case or a dot in the corner indicates which is Pin 1 (other pins are numbered in sequence). All ICs have a type designation printed on the top.

Simpler and commoner ICs come in a "dual in-line" (DIL) package, with a row of pins down each side at 0.1" spacing. In modern equipment, very fine pin spacing is used in surface mounted packages for all the more complex ICs and even many of the simpler ones.

Fault-finding and Repair

ICs can be permanently damaged by static electricity. Whilst they normally contain protection against static up to a certain level, static precautions should be taken for the more expensive and complicated types such as those used in computers, particularly in conditions of low humidity or if nylon carpets or clothing may produce static. These precautions consist of frequently earthing oneself by touching a radiator or an appliance with an earthed metal case, and use of an antistatic wrist strap and mat.

Most ICs are reliable, but failures can result from a fault elsewhere in the circuit, a battery or power supply connected the wrong way around, static, a nearby lightning strike, or a nuclear explosion (but in the latter case you will have other things to worry about). An audio power amplifier IC might fail due to overload or a shorted output.

Voltage regulators and other low pin count devices are easily replaced by another of the same type.

DIL ICs can be removed and replaced with a little care and patience, using a solder sucker and/or solder wick to unsolder the pins. Watch out for any sign of the copper printed circuit tracks lifting from the board and then cracking due to too much heat being applied. This can significantly complicate the repair.

Replacing surface mount ICs is very much more challenging and generally requires professional equipment and considerable skill, though it may be possible in some cases where the pin density isn't too high.

Hands on: Active and passive components will be identified and instructors will demonstrate how to know the values of coded components such as resistor, inductor and capacitor .

Series and Parallel Circuits

Simple circuits (ones with only a few components) are usually fairly straightforward for beginners to understand. But, things can get sticky when other components come to the party. Where's the current going? What's the voltage doing? Can this be simplified for easier understanding? Fear not, intrepid reader. Valuable information follows.

we'll first discuss the difference between series circuits and parallel circuits, using circuits containing the most basic of components – resistors and batteries – to show the difference between the two configurations. We'll then explore what happens in series and parallel circuits when you combine different types of components, such as capacitors and inductors.

- **Series Circuits**

Nodes and Current Flow

Before we get too deep into this, we need to mention what a **node** is. It's nothing fancy, just the electrical junction between two or more components. When a circuit is modeled on a schematic, the nodes are the wires between components.

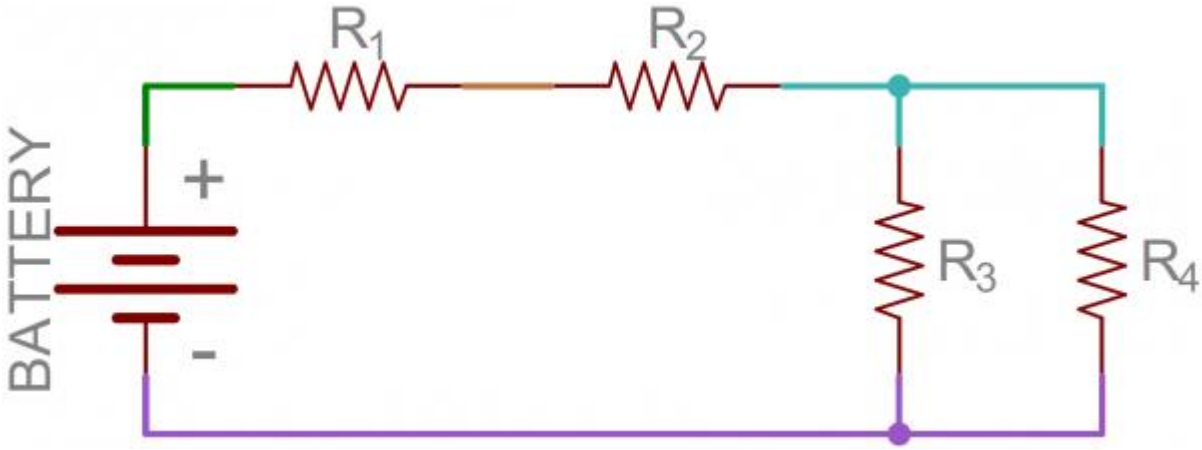


Figure 25:Example schematic with four uniquely colored nodes.

That's half the battle towards understanding the difference between series and parallel. We also need to understand **how current flows** through a circuit. Current flows from a high voltage to a lower voltage in a circuit. Some amount of current will flow through every path it can take to get to the point of lowest voltage (usually called ground). Using the above circuit as an example, here's how current would flow as it runs from the battery's positive terminal to the negative:

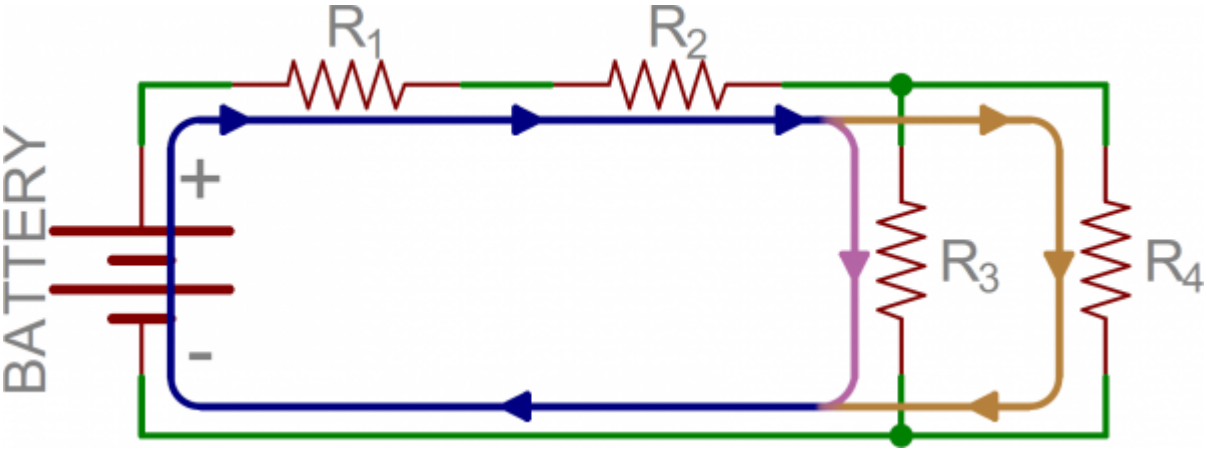


Figure 26: Current Flow

Current (indicated by the blue, orange, and pink lines) flowing through the same example circuit as above. Different currents are indicated by different colors.

Notice that in some nodes (like between R_1 and R_2) the current is the same going in as at is coming out. At other nodes (specifically the three-way junction between R_2, R_3, and R_4) the main (blue) current splits into two different ones. *That's*the key difference between series and parallel!

Series Circuits Defined

Two components are in series if they share a common node and if the **same current** flows through them. Here's an example circuit with three series resistors:

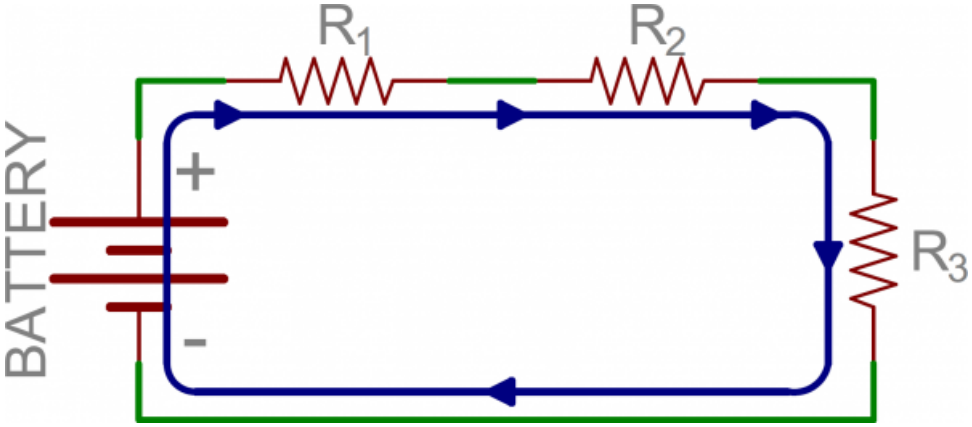


Figure 27: flow of current

There's only one way for the current to flow in the above circuit. Starting from the positive terminal of the battery, current flow will first encounter R_1. From there the current will flow straight to R_2, then to R_3, and finally back to the negative terminal of the battery. Note that there is only one path for current to follow. These components are in series.

- **Parallel Circuits**

Parallel Circuits Defined

If components share *two* common nodes, they are in parallel. Here's an example schematic of three resistors in parallel with a battery:

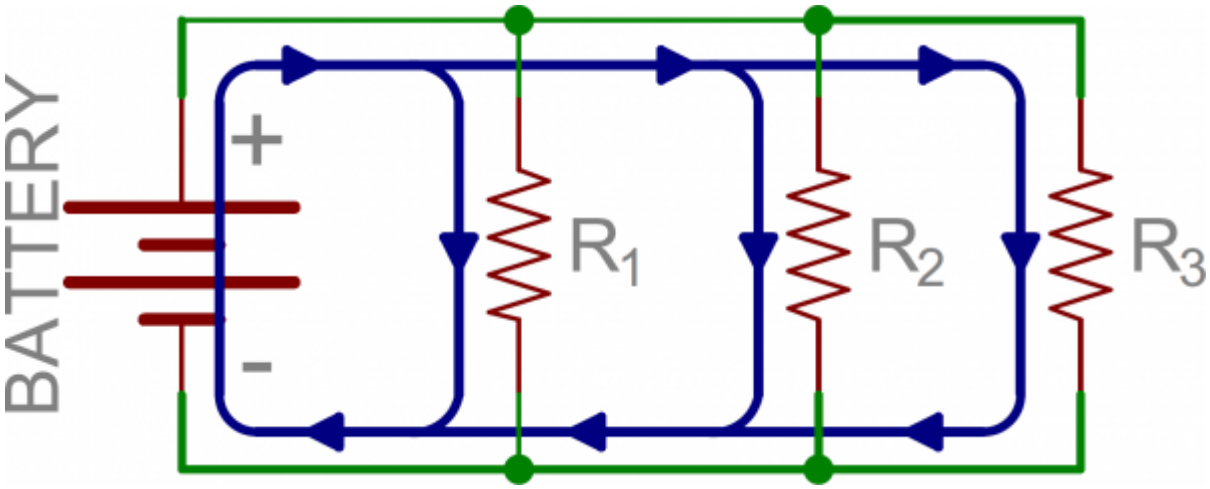


Figure 28: Parallel circuit

From the positive battery terminal, current flows to R_1… and R_2, and R_3. The node that connects the battery to R_1 is also connected to the other resistors. The other ends of these

resistors are similarly tied together, and then tied back to the negative terminal of the battery. There are three distinct paths that current can take before returning to the battery, and the associated resistors are said to be in parallel.

Where series components all have equal currents running through them, parallel components all have the same voltage drop across them – series:current::parallel:voltage.

- **Series and Parallel Circuits Working Together**

From there we can mix and match. In the next picture, we again see three resistors and a battery. From the positive battery terminal, current first encounters R_1. But, at the other side of R_1 the node splits, and current can go to both $R_2$and R_3. The current paths through R_2 and R_3 are then tied together again, and current goes back to the negative terminal of the battery.

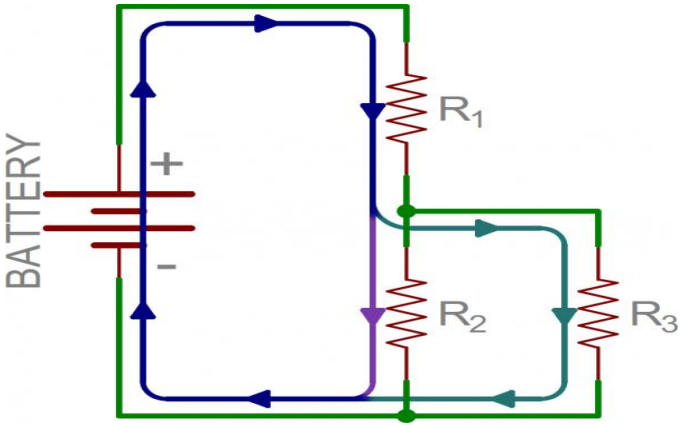


Figure 29: Series and Pararallel Circuit

In this example, R_2 and R_3 are in parallel with each other, and R_1 is in series with the parallel combination of R_2 and R_3.

- **Calculating Equivalent Resistances in Series Circuits**

Here's some information that may be of some more practical use to you. When we put resistors together like this, in series and parallel, we change the way current flows through them. For example, if we have a 10V supply across a 10kΩ resistor, Ohm's law says we've got 1mA of current flowing.

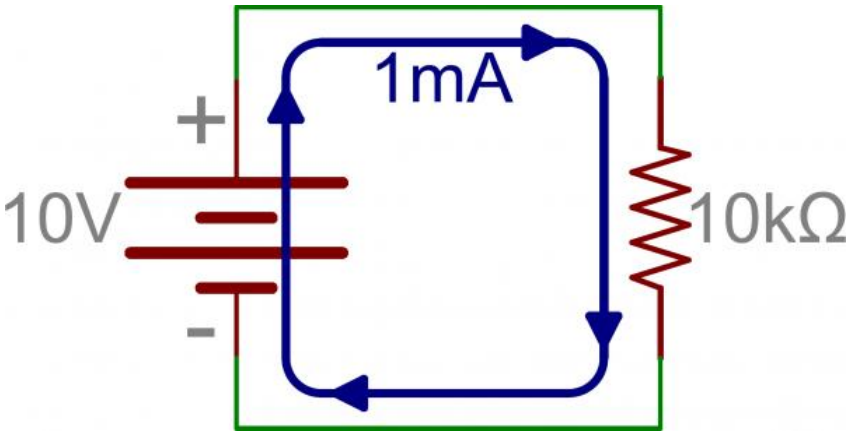


Figure 30: Simple circuit

If we then put another 10kΩ resistor in series with the first and leave the supply unchanged, we've cut the current in half because the resistance is doubled.

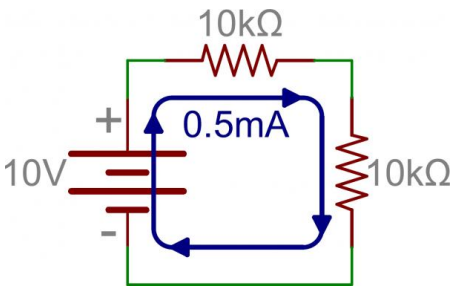


Figure 31: Current Reduction

In other words, there's still only one path for current to take and we just made it even harder for current to flow. How much harder? 10kΩ + 10kΩ = 20kΩ. And, that's how we calculate resistors in series – just **add their values**.

To put this equation more generally: the total resistance of *N* – some arbitrary number of – resistors is their total sum.

Figure 32: Resistor in Series

Equation 1: Equivalent Series resistors

$$R_{tot} = R_1 + R_2 + \ldots + R_{N-1} + R_N$$

Calculating Equivalent Resistances in Parallel Circuits

What about parallel resistors? That's a bit more complicated, but not by much. Consider the last example where we started with a 10V supply and a 10kΩ resistor, but this time we add another 10kΩ in parallel instead of series. Now there are two paths for current to take. Since the supply voltage didn't change, Ohm's Law says the first resistor is still going to draw 1mA. But, so is the second resistor, and we now have a total of 2mA coming from the supply, doubling the original 1mA. This implies that we've cut the total resistance in half.

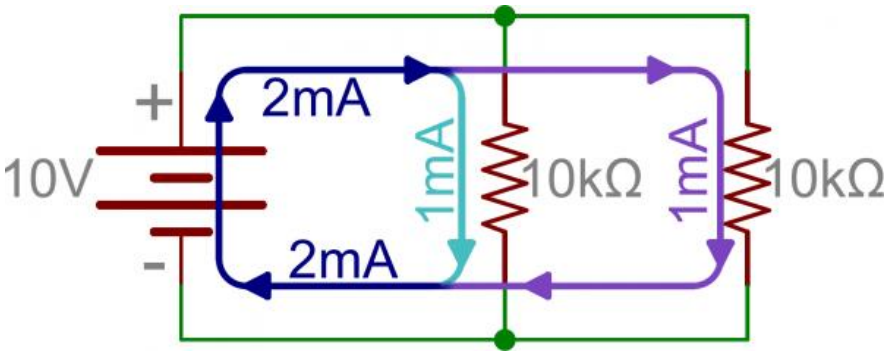


Figure 33: Resistor In Pararallel

While we can say that 10kΩ || 10kΩ = 5kΩ ("||" roughly translates to "in parallel with"), we're not always going to have 2 identical resistors. What then?

The equation for adding an arbitrary number of resistors in parallel is:

Equation 2: Equivalent parallel resistance calculation

$$\frac{1}{R_{tot}} = \frac{1}{R_1} + \frac{1}{R_2} + \ldots + \frac{1}{R_{N-1}} + \frac{1}{R_N}$$

If reciprocals aren't your thing, we can also use a method called "product over sum" when we have two resistors in parallel:

Equation 3: Equivalent Resistance in parallel

$$R_{tot} = \frac{R_1 \cdot R_2}{R_1 + R_2}$$

However, this method is only good for two resistors in one calculation. We can combine more than 2 resistors with this method by taking the result of R1 || R2 and calculating that value in parallel with a third resistor (again as product over sum), but the reciprocal method may be less work.

Experiment Time - Part 1(this will be covered after knowing DMM& Breadboard)

What you'll need:

- A handful of 10kΩ resistors
- A multimeter
- A breadboard

Let's try a simple experiment just to prove that these things work the way we're saying they do.

First, we're going to hook up some 10kΩ resistors in series and watch them add in a most un-mysterious way. Using a breadboard, place one 10kΩ resistor as indicated in the figure and measure with a multimeter. Yes, we already know it's going to say it's 10kΩ, but this is what we in the biz call a "sanity check". Once we've convinced ourselves that the world hasn't changed significantly since we last looked at it, place another one in similar fashion but with a lead from each resistor connecting electrically through the breadboard and measure again. The meter should now say something close to 20kΩ.

You may notice that the resistance you measure might not be exactly what the resistor says it should be. Resistors have a certain amount of **tolerance**, which means they can be off by a certain percentage in either direction. Thus, you may read 9.99kΩ or 10.01kΩ. As long as it's close to the correct value, everything should work fine.

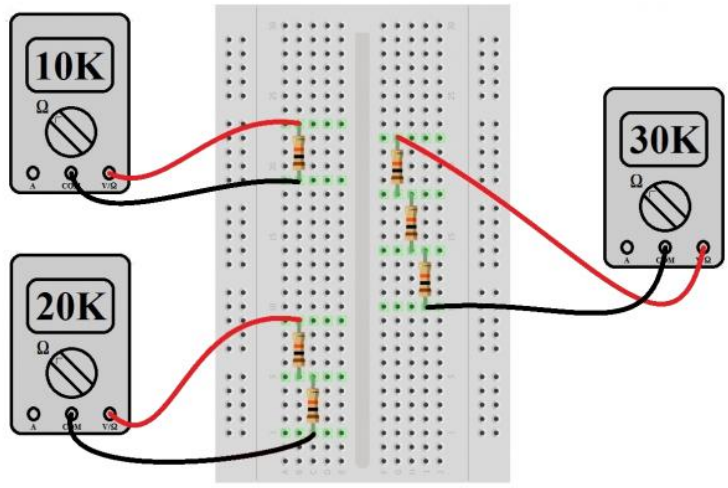


Figure 34: Series Resistors on A breadboard

The reader should continue this exercise until convincing themselves that they know what the outcome will be before doing it again, or they run out of resistors to stick in the breadboard, whichever comes first.

Experiment Time - Part 2

Now let's try it with resistors in a **parallel** configuration. Place one 10kΩ resistor in the breadboard as before (we'll trust that the reader already believes that a single 10kΩ resistor is going to measure something close to 10kΩ on the multimeter). Now place a second 10kΩ resistor next to the first, taking care that the leads of each resistor are in electrically connected rows. But before measuring the combination, calculate by either product-over-sum or reciprocal methods what the new value should be (hint: it's going to be 5kΩ). Then measure. Is it something close to 5kΩ? If it's not, double check the holes into which the resistors are plugged.

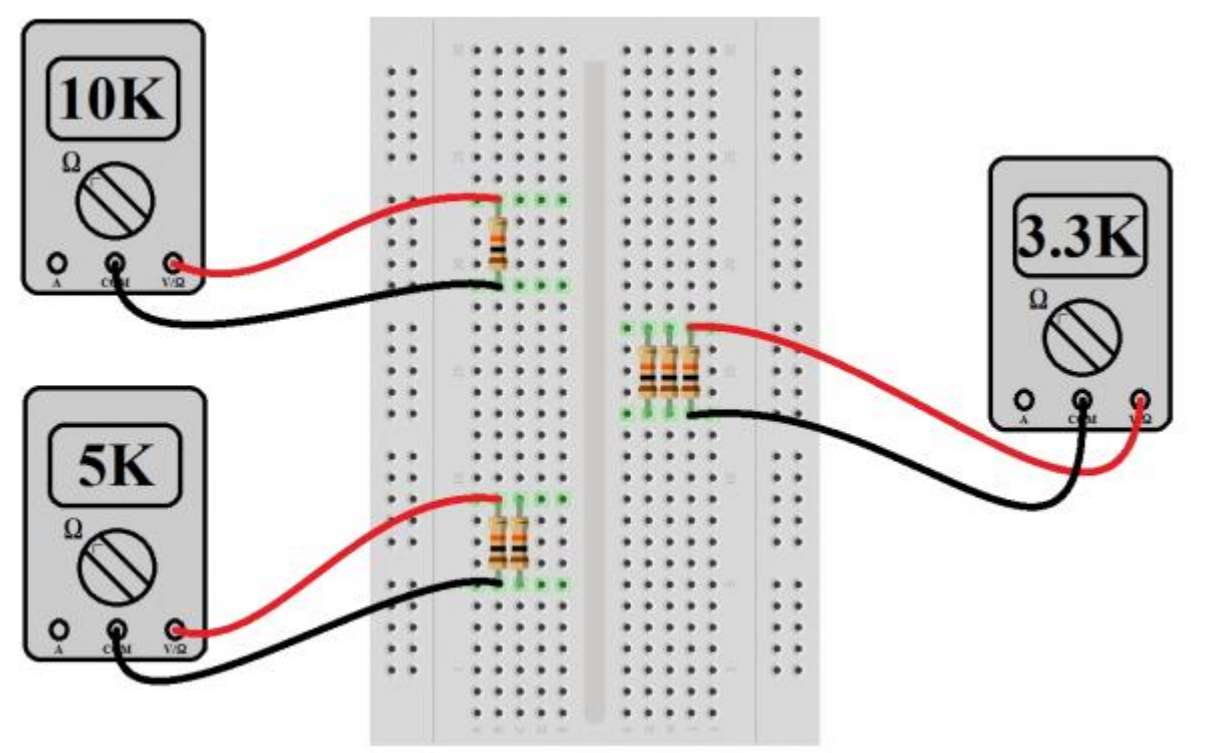


Figure 35: Parallel Restistors on Breadboad

Repeat the exercise now with 3, 4 and 5 resistors. The calculated/measured values should be 3.33kΩ, 2.5kΩ and 2kΩ, respectively. Did everything come out as planned? If not, go back and check your connections. If it did, EXCELSIOR! Go have a milkshake before we continue. You've earned it.

Rules of Thumb for Series and Parallel Resistors

There are a few situations that may call for some creative resistor combinations. For example, if we're trying to set up a very specific reference voltage you'll almost always need a very specific ratio of resistors whose values are unlikely to be "standard" values. And while we can get a very high degree of precision in resistor values, we may not want to wait the X number of days it takes to ship something, or pay the price for non-stocked, non-standard values. So in a pinch, we can always build our own resistor values.

Tip #1: Equal Resistors in Parallel

Adding *N* like-valued resistors *R* in parallel gives us *R/N* ohms. Let's say we need a 2.5kΩ resistor, but all we've got is a drawer full of 10kΩ's. Combining four of them in parallel gives us 10kΩ/4 = 2.5kΩ.

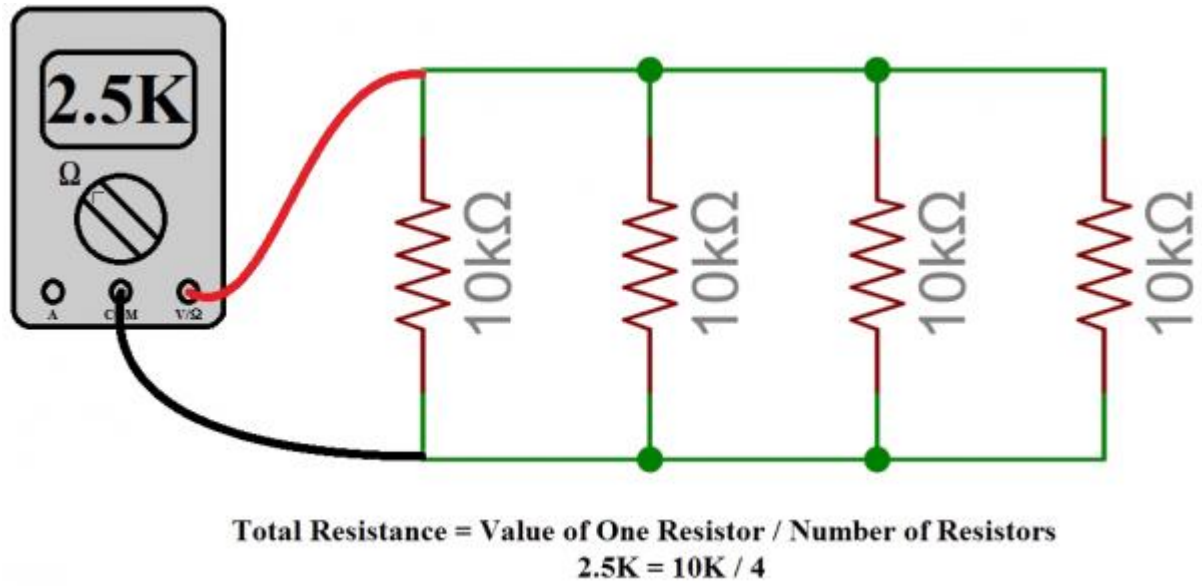


Figure 36:Identical resistor in Parallel

Tip #2: Tolerance

Know what kind of tolerance you can tolerate. For example, if you needed a 3.2kΩ resistor, you could put 3 10kΩ resistors in parallel. That would give you 3.3kΩ, which is about a 4% tolerance from the value you need. But, if the circuit you're building needs to be closer than 4% tolerance, we can measure our stash of 10kΩ's to see which are lowest values because they have a tolerance, too. In theory, if the stash of 10kΩ resistors are all 1% tolerance, we can only get to 3.3kΩ. But part manufacturers are known to make just these sorts of mistakes, so it pays to poke around a bit.

Tip #3: Power Ratings in Series/Parallel

This sort of series and parallel combination of resistors works for power ratings, too. Let's say that we need a 100Ω resistor rated for 2 watts (W), but all we've got is a bunch of 1kΩ quarter-watt (¼W) resistors (and it's 3am, all the Mountain Dew is gone, and the coffee's cold). You can combine 10 of the 1kΩ's to get 100Ω (1kΩ/10 = 100Ω), and the power rating will be 10x0.25W, or 2.5W. Not pretty, but it will get us through a final project, and might even get us extra points for being able to think on our feet.

We need to be a little more careful when we combine resistors of dissimilar values in parallel where total equivalent resistance and power ratings are concerned. It should be completely obvious to the reader, but…

Tip #4: Different Resistors in Parallel

The combined resistance of two resistors of different values is always less than the smallest value resistor. The reader would be amazed at how many times someone combines values in their head and arrives at a value that's halfway between the two resistors (1kΩ || 10kΩ does NOT equal anything around 5kΩ!). The total parallel resistance will always be dragged closer to the lowest value resistor. Do yourself a favor and read tip #4 10 times over.

Tip #5: Power Dissipation in Parallel

The power dissipated in a parallel combination of dissimilar resistor values is not split evenly between the resistors because the currents are not equal. Using the previous example of (1kΩ || 10kΩ), we can see that the 1kΩ will be drawing 10X the current of the 10kΩ. Since Ohm's

Law says power = voltage x current, it follows that the 1kΩ resistor will dissipate 10X the power of the 10kΩ.

Ultimately, the lessons of tips 4 and 5 are that we have to pay closer attention to what we're doing when combining resistors of dissimilar values in parallel. But tips 1 and 3 offer some handy shortcuts when the values are the same.

- **Series and Parallel Capacitors**

Combining capacitors is just like combining resistors…only the opposite. As odd as that sounds, it's absolutely true. Why would this be?

A capacitor is just two plates spaced very close together, and it's basic function is to hold a whole bunch of electrons. The greater the value of capacitance, the more electrons it can hold. If the size of the plates is increased, the capacitance goes up because there's physically more space for electrons to hang out. And if the plates are moved farther apart, the capacitance goes down, because the electric field strength between them goes down as the distance goes up.

Now let's say we've got two 10µF capacitors wired together in series, and let's say they're both charged up and ready discharge into the friend sitting next to you.

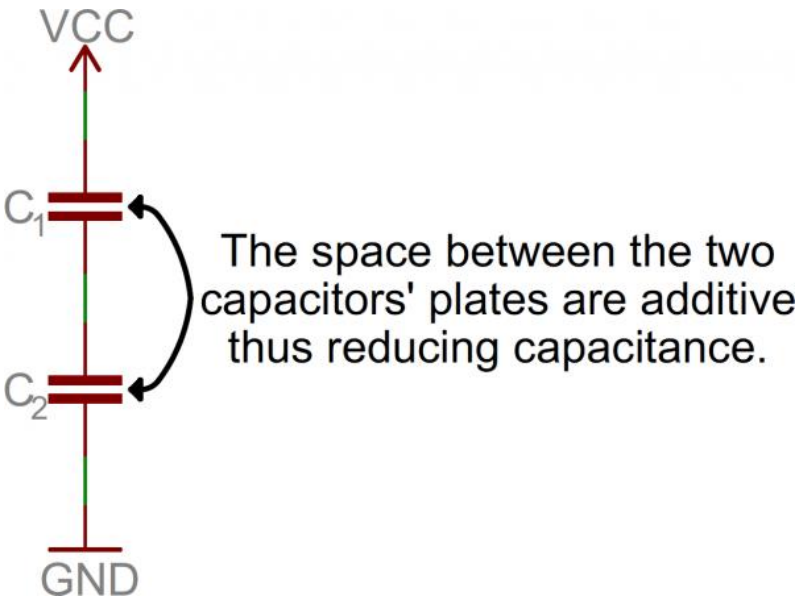
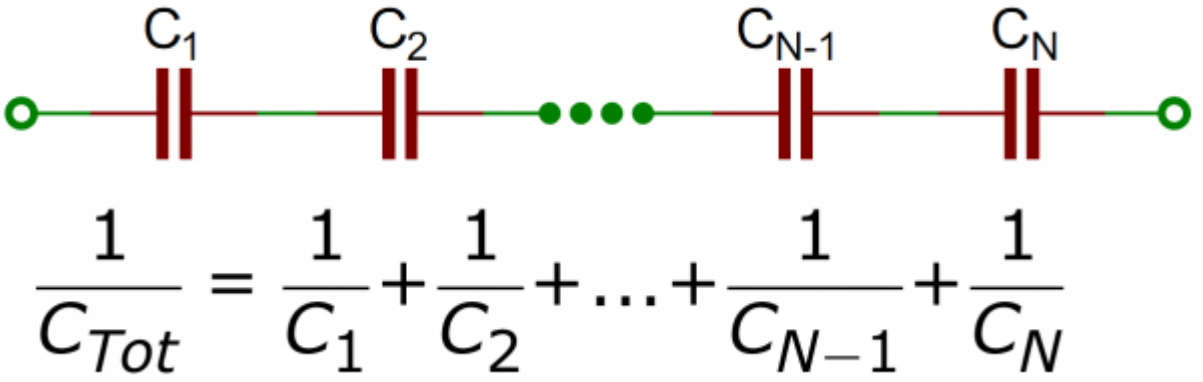


Figure 37: Capacitors In Series

Remember that in a series circuit there's only one path for current to flow. It follows that the number of electrons that are discharging from the cap on the bottom is going to be the same number of electrons coming out of the cap on the top. So the capacitance hasn't increased, has it?

In fact, it's even worse than that. By placing the capacitors in series, we've effectively spaced the plates farther apart because the spacing between the plates of the two capacitors adds together. So we don't have 20µF, or even 10µF. We've got 5µF. The upshot of this is that we add series capacitor values the same way we add parallel resistor values. Both the product-over-sum and reciprocal methods are valid for adding capacitors in series.

Equation 4: Capacitors in Series

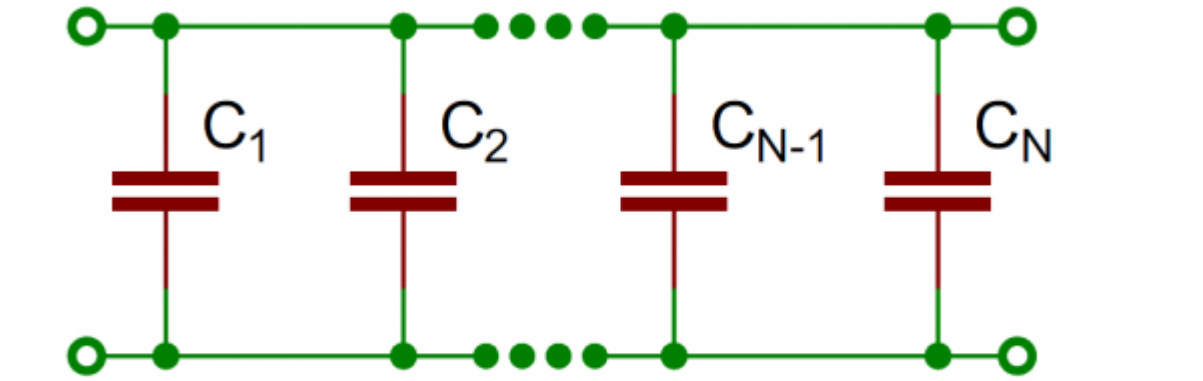


$$\frac{1}{C_{Tot}} = \frac{1}{C_1} + \frac{1}{C_2} + \ldots + \frac{1}{C_{N-1}} + \frac{1}{C_N}$$

It may seem that there's no point to adding capacitors in series. But it should be pointed out that one thing we did get is twice as much voltage (or voltage ratings). Just like batteries, when we put capacitors together in series the voltages add up.

Adding **capacitors in parallel** is like adding resistors in series: the values just add up, no tricks. Why is this? Putting them in parallel effectively increases the size of the plates without increasing the distance between them. More area equals more capacitance. Simple.

Equation 5: Capacitors in parallel



$$C_{Tot} = C_1 + C_2 + \ldots + C_{N-1} + C_N$$

Experiment Time - Part 3(this will be covered after knowing DMM& Breadboard)

What you'll need:

- One 10kΩ resistor
- Three 100µF caps
- A 3-cell AA battery holder
- Three AA cells
- A breadboard
- A multimeter
- Clip-leads

Let's see some series and parallel connected capacitors in action. This will be a little trickier than the resistor examples, because it's harder to measure capacitance directly with a multimeter.

Let's first talk about what happens when a capacitor charges up from zero volts. When current starts to go in one of the leads, an equal amount of current comes out the other. And if there's no resistance in series with the capacitor, it can be quite a lot of current. In any case, the current flows until the capacitor starts to charge up to the value of the applied voltage, more slowly trickling off until the voltages are equal, when the current flow stops entirely.

As stated above, the current draw can be quite large if there's no resistance in series with the capacitor, and the time to charge can be very short (like milliseconds or less). For this experiment, we want to be able to watch a capacitor charge up, so we're going to use a 10kΩ resistor in series to slow the action down to a point where we can see it easily. But first we need to talk about what an RC time constant is.

Equation 6: Time Constant

$$\tau = R * C$$

What the above equation says is that one time constant in seconds (called tau) is equal to the resistance in ohms times the capacitance in farads. Simple? No? We shall demonstrate on the next page.

Experiment Time - Part 3, Continued...

For the first part of this experiment, we're going to use one 10K resistor and one 100µF (which equals 0.0001 farads). These two parts create a time constant of 1 second:

Equation 7: Example of Time Constant

$$\tau = (10,000\ \Omega) * (0.0001\ F) = 1\ second$$

When charging our 100µF capacitor through a 10kΩ resistor, we can expect the voltage on the cap to rise to about 63% of the supply voltage in 1 time constant, which is 1 second. After 5 time constants (5 seconds in this case) the cap is about 99% charged up to the supply voltage, and it will follow a charge curve something like the plot below.

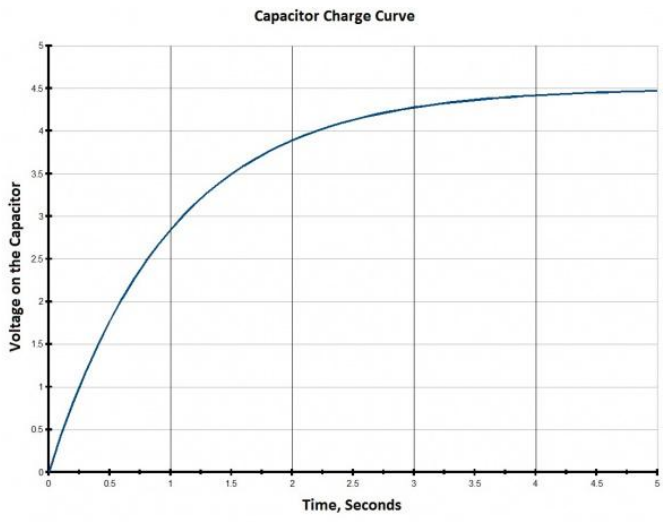


Figure 38: Capacitor Charging Curve

Now that we know that stuff, we're going to connect the circuit in the diagram (make sure to get the polarity right on that capacitor!).

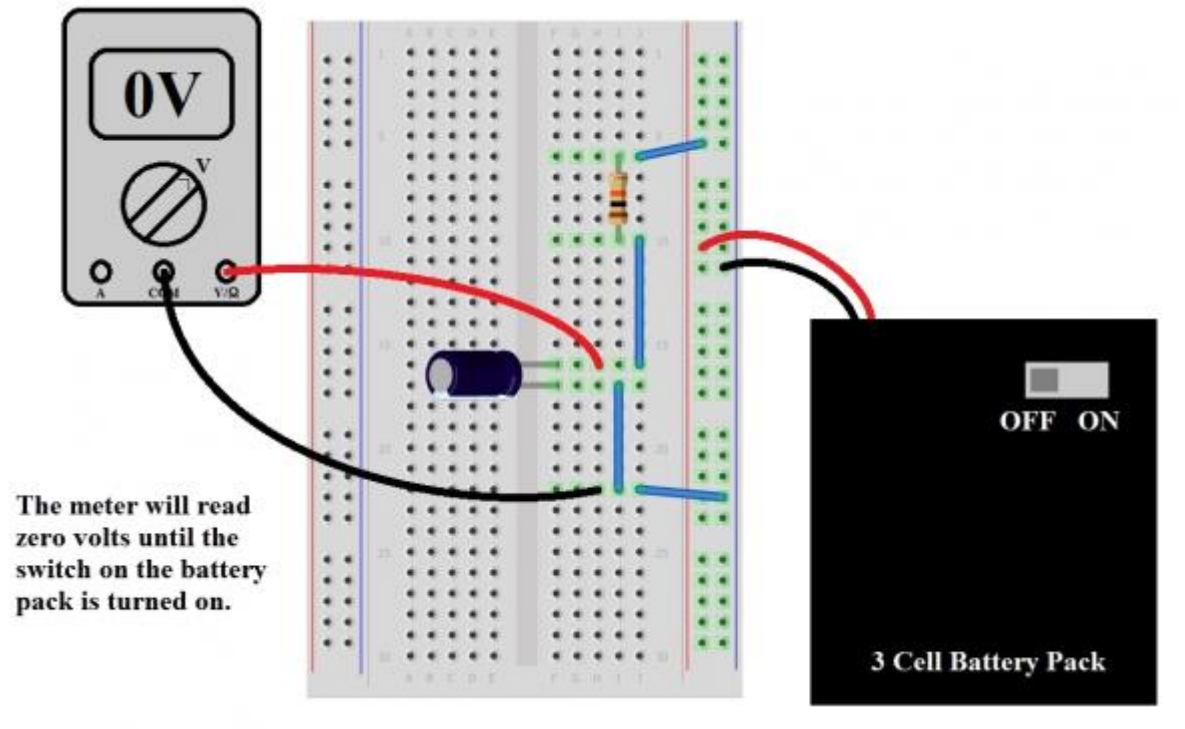


Figure 39: Charging Capacitor

With our multimeter set to measure volts, check the output voltage of the pack with the switch turned on. That's our supply voltage, and it should be something around 4.5V (it'll be a bit more if the batteries are new). Now connect the circuit, taking care that the switch on the battery pack is in the "OFF" position before plugging it into the breadboard. Also, take care that the red and black leads are going to the right places. If it's more convenient, you can use alligator clips to attach the meter probes to the legs of the capacitor for measurement (you can also spread those legs out a bit to make it easier).

Once we're satisfied that the circuit looks right and our meter's on and set to read volts, flip the switch on the battery pack to "ON". After about 5 seconds, the meter should read pretty close to the battery pack voltage, which demonstrates that the equation is right and we know what we're doing. Now turn the switch off. It's still holding that voltage pretty well, isn't it? That's because there's no path for current to discharge the capacitor; we've got an open circuit. To discharge the cap, you can use another 10K resistor in parallel. After about 5 seconds, it will be back to pretty close to zero.

Experiment Time - Part 3, Even More...

Now we're on to the interesting parts, starting with connecting two capacitors in series. Remember that we said the result of which would be similar to connecting two resistors in parallel. If this is true, we can expect (using product-over-sum)

Equation 8: Capacitance

$$C = \frac{100\ \mu F \cdot 100\ \mu F}{100\ \mu F + 100\ \mu F} = 50\ \mu F$$

What's that going to do to our time constant?

Equation 9: Time Constant

$$\tau = (10,000\ \Omega) * (0.00005\ F) = 0.5\ seconds$$

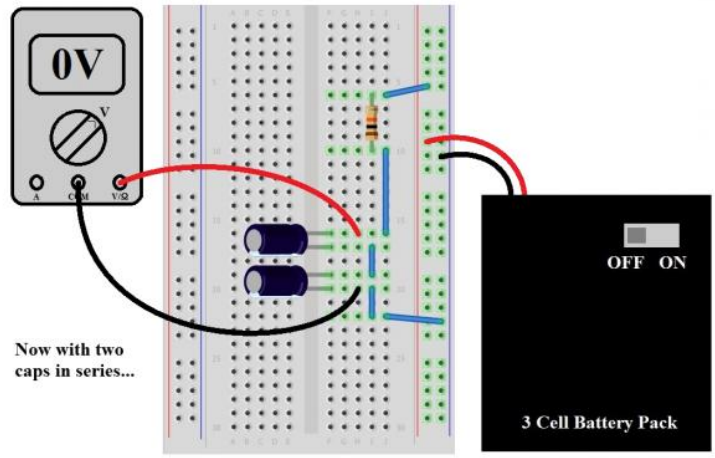


Figure 40: Two capacitors in Series

With that in mind, plug in another capacitor in series with the first, make sure the meter is reading zero volts (or there-abouts) and flip the switch to "ON". Did it take about half as much time to charge up to the battery pack voltage? That's because there's half as much capacitance. The electron gas tank got smaller, so it takes less time to charge it up. A third capacitor is suggested for this experiment just to prove the point, but we're betting the reader can see the writing on the wall.

Now we'll try capacitors in parallel, remembering that we said earlier that this would be like adding resistors in series. If that's true, then we can expect 200µF, right? Then our time constant becomes

Equation 10: Time Constant

$$\tau = (10,000\ \Omega) * (0.0002\ F) = 2\ second$$

This means that it will now take about 10 seconds to see the parallel capacitors charge up to the supply voltage of 4.5V.

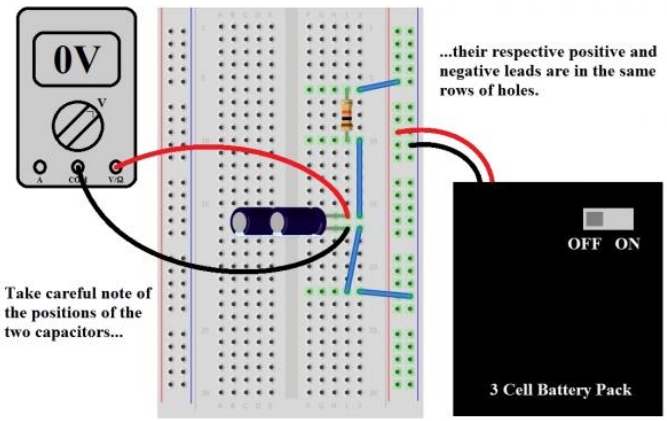


Figure 41: Capacitors on Breadboard

For the proof, start with our original circuit of one 10kΩ resistor and one 100µF capacitor in series, as hooked up in the first diagram for this experiment. We already know that the

capacitor is going to charge up in about 5 seconds. Now add a second capacitor in parallel. Make sure the meter is reading close to zero volts (discharge through a resistor if it isn't reading zero), and flip the switch on the battery pack to "ON". Takes a long time, doesn't it? Sure enough, we made the electron gas tank bigger and now it takes longer to fill it up. To prove it to yourself, try adding the third 100µF capacitor, and watch it charge for a good, long time.

- **Series and Parallel Inductors**

Cases where inductors need to be added either in series or in parallel are rather rare, but not unheard of. In any case, let's address them just to be complete.

In a nutshell they add just like resistors do, which is to say they add with a plus sign when in series, and with product-over-sum when in parallel. The tricky part comes when they are placed close together so as to have interacting magnetic fields, whether intentionally or not. For this reason, it is preferable to have a single component rather than two or more, though most inductors are shielded to prevent interacting magnetic fields.

In any case, suffice it to say that they add like resistors do. More information than that regarding inductors is well beyond the scope of this section.

Hands on: Different resistors and capacitors will be provided to finally achieve different experiments as indicated above .

Chap 3: Basics of Measurement Techniques

Electronic Testing Equipment and Their Types

The testing equipment used to detect faults in the operation of electronic devices by creating stimulus signals and capture responses from electronic devices under test is known as electronic test equipment. If any faults are detected, then identified faults can be traced an rectified using electronic testing equipment. Most often all electrical and electronic circuits are tested and troubleshooted to detect faults or abnormal functioning if any.

Figure 42: Some Measuring Instruments

Therefore, testing equipment is necessary to find and analyze the circuit conditions, for checking electronic test equipment and maintenance in various industries. Many industries utilize different types of electronic test equipment ranging from the very simple and inexpensive to complex and sophisticated ones.

- **Voltmeter**

A basic electronics device or instrument used to measure voltage or electrical potential difference between two points in electrical circuits is known as voltmeter. There are two types of voltmeters: analog and digital. An analog voltmeter moves a pointer across a scale in proportional to the voltage of the electrical circuit. A digital voltmeter measures an unknown input voltage by converting the voltage to a digital value by using a converter and then displays the voltage in numeric form.

Figure 43: Voltmeter

- **Ohmmeter**

An electrical instrument that measures electrical resistance is known as an ohmmeter. The instrument used to measure small value of resistance are micro-ohmmeters. Similarly meg-ohmmeters is used to make large resistance measurements. Resistance values are measured in ohms (Ω). Originally, ohmmeter is designed with a small battery to apply a voltage to a resistance.

Figure 44:Ommeter

It uses a galvanometer to measure the electric current through the resistance. The scale of the galvanometer was marked in ohms (Ω), because the fixed voltage from the battery assures that the resistance decreases and the current through the meter increases.

- **Ammeter**

A measuring instrument which is used to measure the electric current in a circuit is known as an ammeter. The units of measurement for electric current is amperes (A) Earlier ammeters were laboratory instruments which depend on the earth's magnetic field for operation. In an era of the 19th century, improved instruments were designed which could be placed in any position and allows accurate measurements in electric power systems.

Figure 45: ammeter

The smaller currents can be measured by using milliammeters or micro ammeters, units of measuring the smaller current are in the milliampere or micro-ampere range. There are different types of ammeters such as moving-coil, moving magnet and moving-iron, etc.

- **Multimeter**

A multimeter is an electronic instrument used to measure the three basic electrical characteristics: voltage, current and resistance. It has multiple functions and acts like ohmmeter, voltmeter and ammeter and also used for household wiring, electric motors, testing batteries and power supplies. The multimeter is a handheld device with a needle over a numeric LCD digital display for indication purpose. It is also used to test continuity between two points in an electrical circuit. There are three types of multimeters made available in the market such as: digital multimeter, analog multimeter and fluke multimeter.

Figure 46:Multimeter

The Following are used for Testing Stimulus Signals of the Circuit Under Test

- **Power Supplies**

A power supply is an electronic instrument that supplies electric energy to an electric load. Regulated power supplies refers to a power supply which supplies a variety of output voltages used for bench testing of electronic circuits, with the variation of output voltages or some preset voltages. Almost all the electronic circuits make use of a DC source of power for operation. A regulated power supply consists of various blocks such as an ordinary power supply and a voltage regulating device. The output generated from ordinary power supply is fed to the voltage regulating device that provides the final output. The main function of a power supply is to convert one form of electrical energy into another.

Figure 47: Power Supply

Where to locally buy this?: http://naradaelectronics.rw/

- **Signal Generator**

A signal generator is also named as pitch generator, function generator or frequency generator is an electronic device used for generating electronic signals either in the analog or digital domains (repeating or non-repeating signals). Signal generators are used in testing, designing and repairing electro acoustic or electronic devices.

Figure 48: Signal Generator

Where to locally buy this?: http://naradaelectronics.rw/

In general no electronic device is suitable for all applications. There are various types of signal generators with different applications and purposes. During the development in

technology, compared to signal generators there are flexible and programmable software tone generators with embedded hardware units are made available in the market.

- **Pulse Generator**

A pulse generator is either an electronic circuit or a piece of electronic test equipment used to generate electrical pulses in different shapes: mostly used for tests at analog or electrical level. Pulse generators are used to control the width, frequency,delay based on the low and high voltage levels of the pulses and with respect to an internal and external triggering. There are three types of pulse generators namely optical pulse generator, bench pulse generators and microwave pulsers.

Figure 49:Pulse Generator

- **Digital Pattern Generator**

A digital generator is an electronic testing equipment or software used to generate digital electronics stimuli. Digital electronics stimuli are a specific type of electrical waveform varying between two conventional voltages corresponding to two logic gates (either 1 or 0, low or high). The function of the digital pattern generator is to stimulate the inputs of a electronic device. For that purpose, the voltage levels are generated by a digital pattern generator are compared to I/O standards of digital electronics: TTL, LVTTL and LVDS. It is also known as a logic source because it is a source of synchronous digital stimulus.

Figure 50:Digital Pattern Generator

It generates a signal for testing digital electronics at logic level. This generator also produces a single shot or repetitive signals in which some sort of triggering source takes place (internally or externally)

The Following Equipments Analyze the Response of the Circuit Under Test

- **Oscilloscope**

The oscilloscope is an electronic test instrument that constantly obverses varying voltage signals as a two dimensional plot of one or more signals as a function of time. The other names for oscilloscope are oscillograph, cathode ray oscilloscope or digital storage oscilloscope. It is also used for converting non electrical signals such as vibration or sound into voltages and then displays the result.

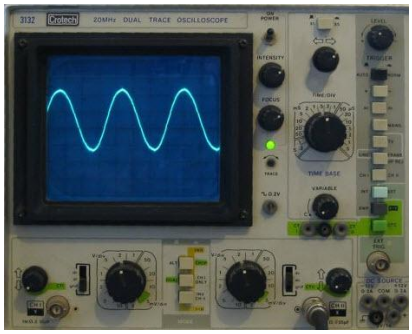


Figure 51:Cathode Ray Oscilloscope

Oscilloscopes are used to observe the change of an electrical signal based on time such that voltage and time describe a shape of the signals and graphed continuously compared with a calibrated scale.The obtained waveforms can be considered for following properties such as frequency, amplitude, time interval, rise time and others. Modern digital instruments may calculate these properties directly and displays them.

- **Frequency counter**

Digital frequency counter is an electrical test equipment used for measuring the frequency of repetitive signals and elapsed time between events. Digital frequency counters are also used to measure the radio frequency where it is important to measure the precise frequency of a particular signal.

Figure 52:Frequency Counter

There is a slight difference between the timers and frequency counters in the electronic industry. It is often possible to use both timers and frequency counters to perform the both functions: to measure the time and frequency. Frequency counters are mostly used as general purpose laboratory test equipment to measure higher frequencies.

Advanced or Less Commonly used Testing Equipment

LCR Meter

LCR Meter name itself indicates that it is used to measure the inductance, capacitance and resistance of electronics components. The inductance, capacitance and resistance are denoted by the letters L, C, and R so it is named as LCR Meter. A variety of meters are made available in the market, but simple versions of LCR meters indicates impedance only for converting the values to capacitance or inductance.

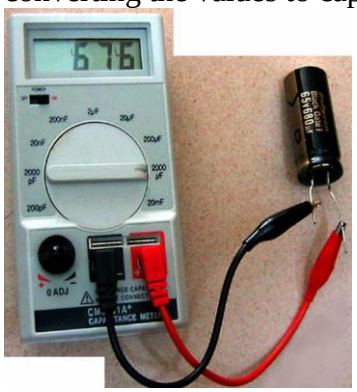


Figure 53:LCR Meter

More designs are available and used to measure the capacitance or inductance, and also the equivalent series resistance of capacitors and the Q factor of inductive components. These conditions make the LCR meters valuable for measuring the quality and overall performance of the component.

Hands on: Different types of measuring instruments will be shown to trainees .

- **How to Use a Multimeter**

A digital multimeter (DMM), is an indispensable tool that you can use to diagnose circuits, learn about other people's electronic designs, and even test a battery. Hence the 'multi'-'meter' (multiple measurement) name.

The most basic things we measure are voltage and current. A multimeter is also great for some basic sanity checks and troubleshooting. Is your circuit not working? Does the switch work? Put a meter on it! The multimeter is your first defense when troubleshooting a system. In this section we will cover measuring voltage, current, resistance and continuity.

Figure 54: Multimeter

Where to locally buy this?: http://naradaelectronics.rw/

Parts of a Multimeter

Figure 55:multimeter

A multimeter is has three parts:

- Display
- Selection Knob
- Ports

The **display** usually has four digits and the ability to display a negative sign. A few multimeters have illuminated displays for better viewing in low light situations.

The **selection knob** allows the user to set the multimeter to read different things such as milliamps (mA) of current, voltage (V) and resistance (Ω).

Two probes are plugged into two of the **ports** on the front of the unit. **COM** stands for common and is almost always connected to Ground or '-' of a circuit. The **COM** probe is conventionally black but there is no difference between the red probe and black probe other than color. **10A** is the special port used when measuring large currents (greater than 200mA). **mAVΩ** is the port that the red probe is conventionally plugged in to. This port allows the measurement of current (up to 200mA), voltage (V), and resistance (Ω). The probes have a *banana* type connector on the end that plugs into the multimeter. Any probe with a banana plug will work with this meter. This allows for different types of probes to be used.

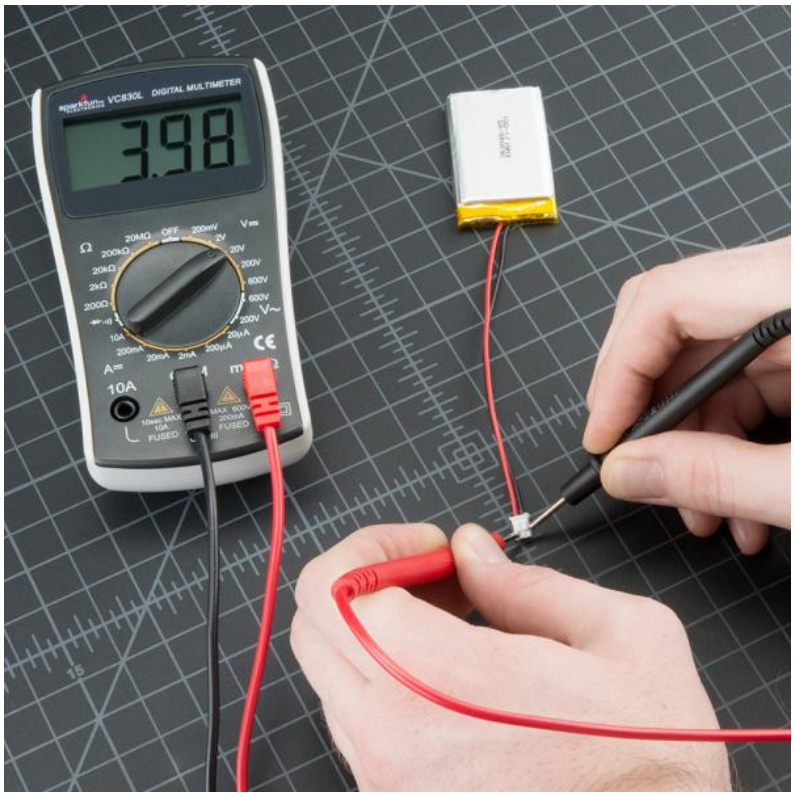


Figure 56:Using a Multimeter to test the voltage on a LiPo Battery.

Probe Types

There are many different types of probes available for multimeters. Here are a few of our favorites:

- Banana to Alligator Clips : These are great cables for connecting to large wires or pins on a breadboard. Good for performing longer term tests where you don't have to have to hold the probes in place while you manipulate a circuit.
- Banana to IC Hook : IC hooks work well on smaller ICs and legs of ICs.
- Banana to Tweezers : Tweezers are handy if you are needing to test SMD components.
- Banana to Test Probes : If you ever break a probe, they are cheap to replace!

Measuring Voltage

To start, let's measure voltage on a AA battery: Plug the black probe into **COM** and the red probe into **mAVΩ**. Set the multimeter to "2V" in the DC (direct current) range. Almost all

portable electronics use direct current), not alternating current. Connect the black probe to the battery's ground or '-' and the red probe to power or '+'. Squeeze the probes with a little pressure against the positive and negative terminals of the AA battery. If you've got a fresh battery, you should see around 1.5V on the display (this battery is brand new, so its voltage is slightly higher than 1.5V).

If you're measuring DC voltage (such as a battery or a sensor hooked up to an Arduino) you want to set the knob where the V has a straight line. AC voltage (like what comes out of the wall) can be dangerous, so we rarely need to use the AC voltage setting (the V with a wavy line next to it). If you're messing with AC, we recommend you get a non-contact testerrather than use a digital multimeter.

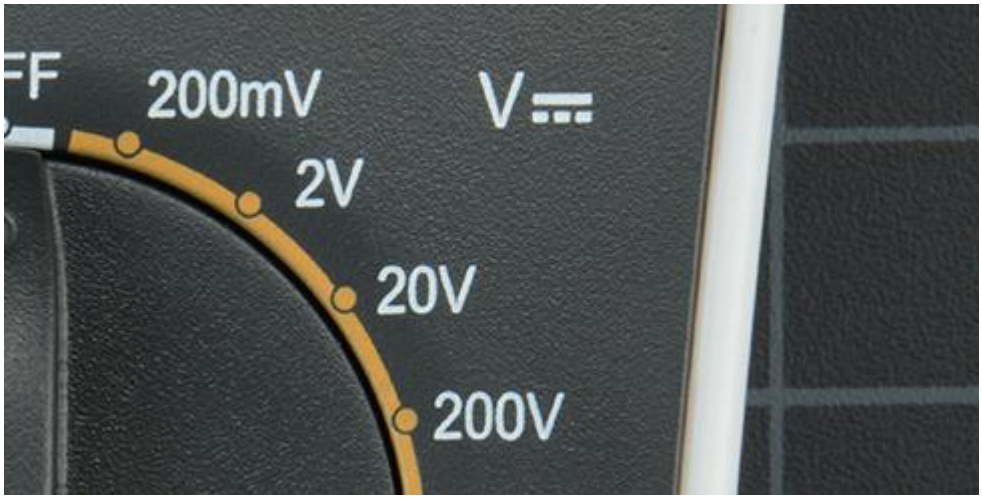


Figure 57:DC Voltage range
Use the V with a straight line to measure DC Voltage

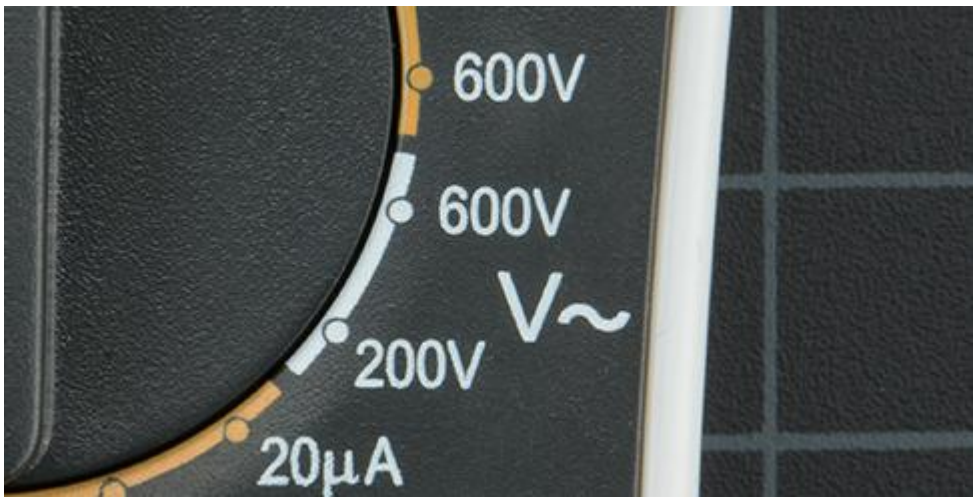


Figure 58: AC Voltage range
Use the V with a wavy line to measure AC Voltage

What happens if you switch the red and black probes? The reading on the multimeter is simply negative. Nothing bad happens! The multimeter measures voltage in relation to the common probe. How much voltage is there on the '+' of the battery compared to common or the negative pin? 1.5V. If we switch the probes, we define '+' as the common or zero point. How much voltage is there on the '-' of the battery compared to our new zero? -1.5V!

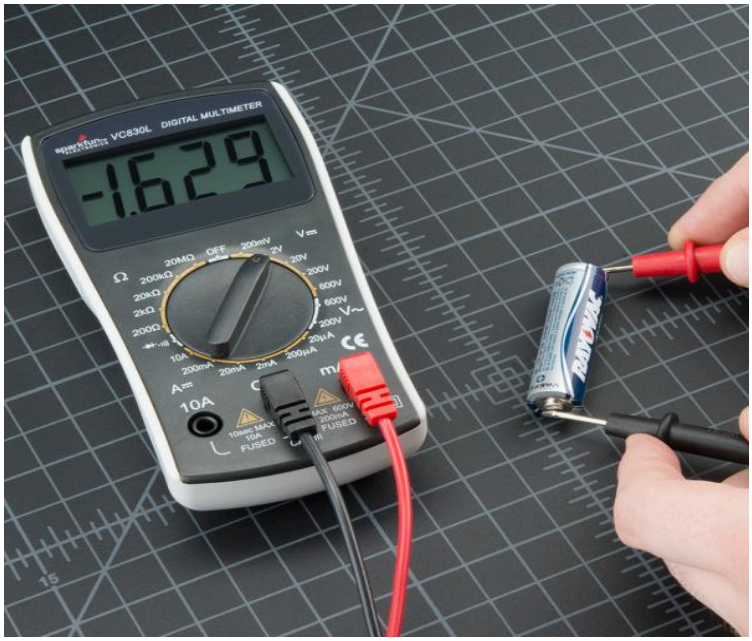


Figure 59: Measuring a battery

Now let's construct a simple circuit to demonstrate how to measure voltage in a real world scenario. The circuit is simply a 1kΩ and a Blue super bright LED powered with a Breadboard Power Supply Stick. To begin, let's make sure the circuit you are working on is powered up correctly. If your project should be at 5V but is less than 4.5V or greater than 5.5V, this would quickly give you an indication that something is wrong and you may need to check your power connections or the wiring of your circuit.

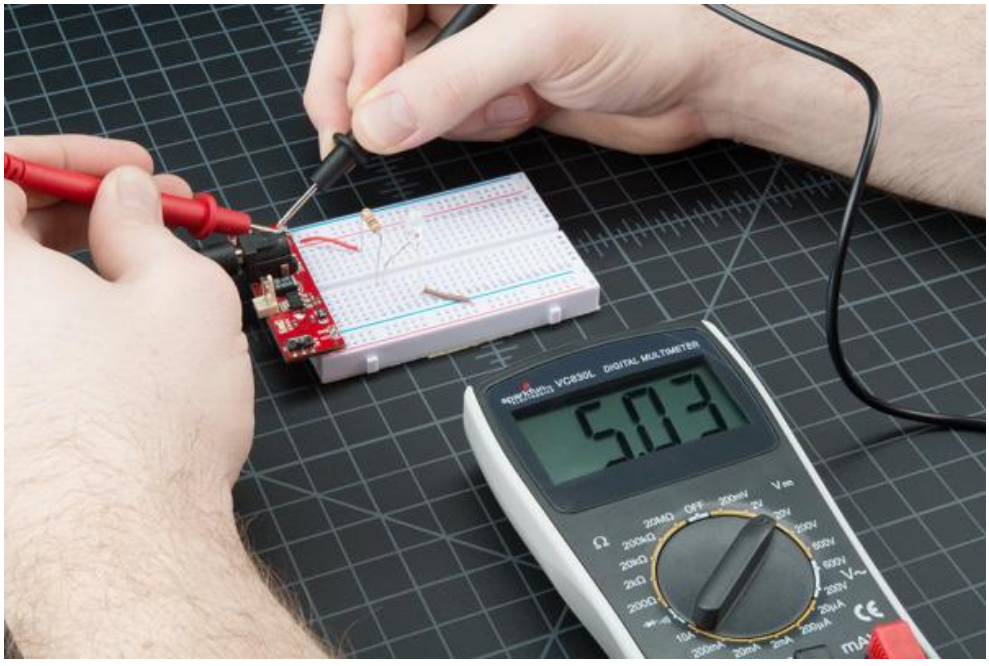


Figure 60:Measuring the voltage coming off of a Power Supply Stick.

Set the knob to "20V" in the DC range (the DC Voltage range has a V with a straight line next to it). Multimeters are generally not autoranging. You have to set the multimeter to a range that it can measure. For example, **2V** measures voltages **up to 2 volts**, and **20V** measures voltages **up to 20 volts**. So if you've measuring a 12V battery, use the 20V setting. 5V system? Use the 20V setting. If you set it incorrectly, you will probably see the meter screen change and then read '1'.

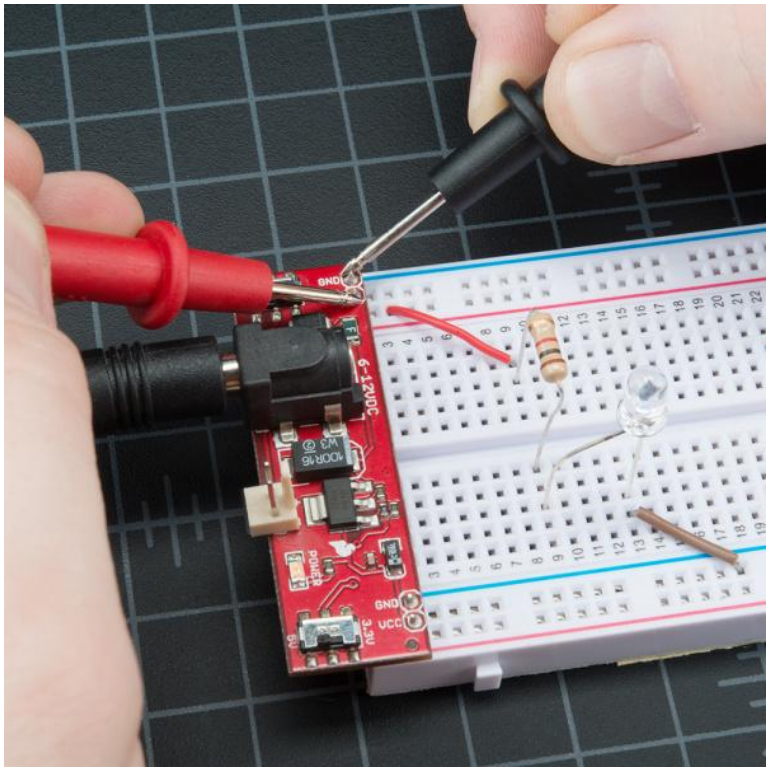


Figure 61:Measuring the voltage coming off of a Power Supply Stick.
With some force (imagine poking a fork into a piece of cooked meat), push the probes onto two exposed pieces of metal. One probe should contact a GND connection. One probe to the VCC or 5V connection.

We can test different parts of the circuit as well. This practice is called nodal analysis, and it is a basic building block in circuit analysis. By measuring the voltage across the circuit we can see how much voltage each component requires. Let's measure the whole circuit first. Measuring from where the voltage is going in to the resistor and then where ground is on the LED, we should see the full voltage of the circuit, expected to be around 5V.

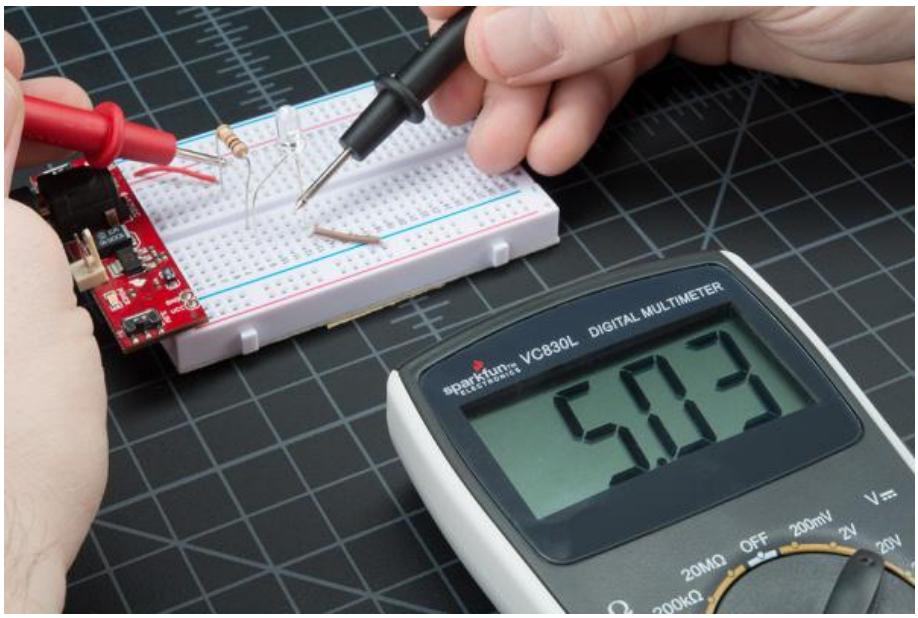


Figure 62: Checking the voltage accross a LED and Resistor
We can then see how much voltage the LED is using. This is what is referred to as the **voltage drop** across the LED. If that doesn't make sense now, fear not. It will as you

explore the world of electronics more. The important thing to take away is that different parts of a circuit can be measured to analyze the circuit as a whole.

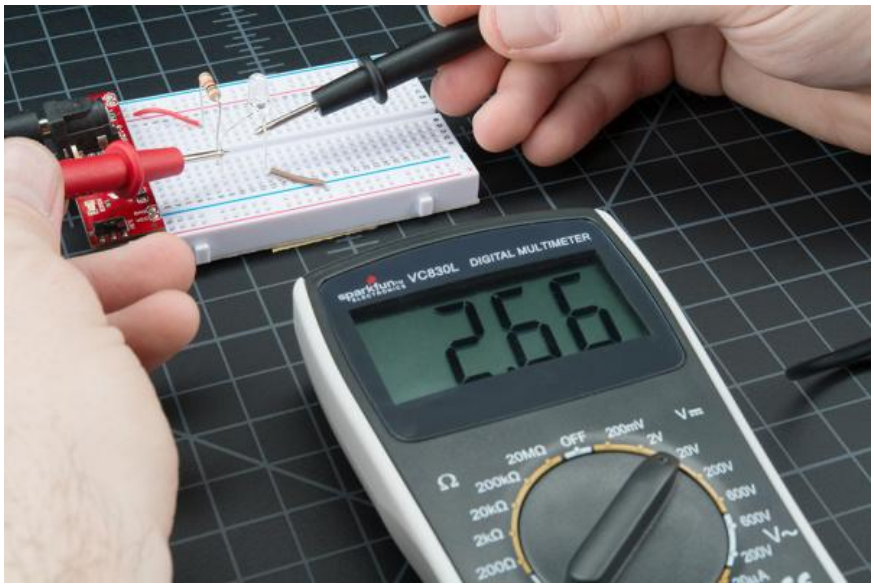


Figure 63:Checking the voltage accross a LED
This LED is using 2.66V of the available 5V supply to illuminate. This is lower than the forward voltage stated in the datasheet on account of the circuit only having small amount of current running though it, but more on that in a bit.

Overload

What happens if you select a voltage setting that is too low for the voltage you're trying to measure? Nothing bad. The meter will simply display a 1. This is the meter trying to tell you that it is overloaded or out-of-range. Whatever you're trying to read is too much for that particular setting. Try changing the multimeter knob to a the next highest setting.

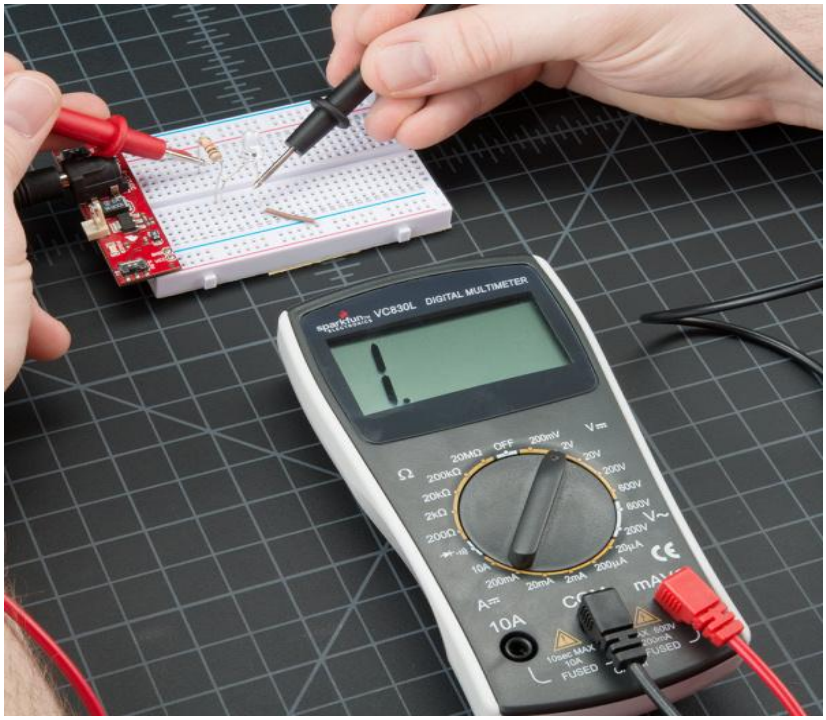


Figure 64:Reading the 5V across this circuit is too much for the 2V setting on the multimeter.

Selection Knob

Figure 65:Selection Knob

Why does the meter knob read 20V and not 10V? If you're looking to measure a voltage less than 20V, you turn to the 20V setting. This will allow you to read from **2.00** to **19.99**.

The first digit on many multimeters is only able to display a '1' so the ranges are limited to **1**9.99 instead of **9**9.99. Hence the 20V max range instead of 99V max range.

Warning! In general, stick to DC circuits (the settings on the multimeter with straight lines, not curvy lines). Most multimeters can measure AC (alternating current) systems, but AC circuits can be dangerous. A wall outlet with AC or 'main voltage' is the stuff that can zap you pretty good. VERY carefully respect AC. If you need to check to see if an outlet is 'on' then use a AC tester. Really the only times we've needed to measure AC are when we've got an outlet that is acting funny (is it really at 110V?), or if we're trying to control a heater (such as a hot plate). Go slow and double check everything before you test an AC circuit.

Measuring Resistance

Normal resistors have color codes on them. If you don't know what they mean, that's ok! There are plenty of online calculators that are easy to use. However, if you ever find yourself without internet access, a multimeter is very handy at measuring resistance.

Pick out a random resistor and set the multimeter to the 20kΩ setting. Then hold the probes against the resistor legs with the same amount of pressure you when pressing a key on a keyboard.

Figure 66: Resistor Measurement
The meter will read one of three things, **0.00**, **1**, or the **actual resistor value**.

- In this case, the meter reads 0.97, meaning this resistor has a value of 970Ω, or about 1kΩ (remember you are in the 20kΩ or 20,000 Ohm mode so you need to move the decimal three places to the right or 970 Ohms).
- If the multimeter **reads 1** or displays **OL**, it's overloaded. You will need to try a higher mode such as **200kΩ** mode or **2MΩ** (megaohm) mode. There is no harm if this happen, it simply means the range knob needs to be adjusted.
- If the multimeter reads **0.00** or nearly zero, then you need to lower the mode to **2kΩ** or **200Ω**.

Remember that many resistors have a 5% tolerance. This means that the color codes may indicate 10,000 Ohms (10kΩ), but because of discrepancies in the manufacturing process a 10kΩ resistor could be as low as 9.5kΩ or as high as 10.5kΩ. Don't worry, it'll work just fine as a pull-up or general resistor.

Let's drop the meter down to the next lowest setting, 2KΩ. What happens?

Figure 67:Resistor Measurement
Not a whole lot changed. Because this resistor (a 1KΩ) is less than 2KΩ, it still shows up on the display. However, you'll notice that there is one more digit after the decimal point giving us a slightly higher resolution in our reading. What about the next lowest setting?

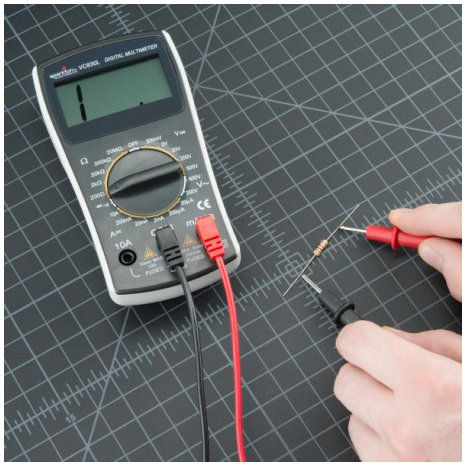


Figure 68:Resistor Measurement
Now, since 1kΩ is greater than 200Ω, we've maxed out the meter, and it is telling you that it is overloaded and that you need to try a higher value setting.

As a rule of thumb, it's rare to see a resistor less than 1 Ohm. Remember that measuring resistance is not perfect. Temperature can affect the reading a lot. Also, measuring resistance of a device while it is physically installed in a circuit can be very tricky. The surrounding components on a circuit board can greatly affect the reading.

Measuring Current

Reading current is one of the trickiest and most insightful readings in the world of embedded electronics. It's tricky because you have to measure current in series. Where voltage is measure by poking at VCC and GND (in parallel), to measure current you have to physically interrupt the flow of current and put the meter in-line. To demonstrate this, we'll use the same circuit we used in the measuring voltage section.

The first thing we'll need is an extra piece of wire. As mentioned, we'll need to physically interrupt the circuit to measure the current. Said another way, pull out the VCC wire going to the resistor, add a wire where that wire was connected, and then probe from the power pin on the power supply to the resistor. This effectively "breaks" power to the circuit. We then insert the multimeter in-line so that it can measure the current as it "flows" through to the multimeter into the bread board.

For these pictures, we cheated and used alligator clips. When measuring current, it's often good to watch what your system does over time, for a few seconds or minutes. While you might want to stand there and hold the probes to the system, sometimes it's easier to free up your hands. These alligator clip probes can come in handy. Note that almost all multimeters have the same sized jacks (they're called "banana plugs") so if you're in a pinch, you can use your friend's probes.

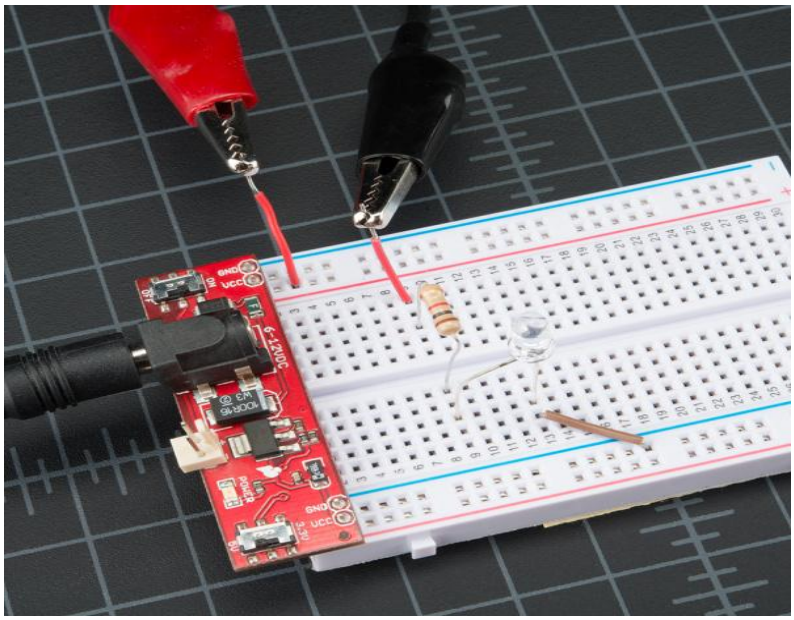


Figure 69: Current Measurement

With the multimeter connected, we can now set the dial to the proper setting and measure some current. Measuring current works the same as voltage and resistance – you have to get the correct range. Set the multimeter to 200mA, and work from there. The current consumption for many breadboard projects is usually under 200mA. Make sure the red probe is plugged into the 200mA fused port. On our favorite multimeter, the 200mA hole is the same port/hole as voltage and resistance reading (the port is labeled **mAVΩ**). This means you can keep the red probe in the same port to measure current, voltage, or resistance. However, if you suspect that your circuit will be using close to or more than 200mA, switch your probe

to the 10A side, just to be safe. Overloading the current can result in a blown fuse rather than just an overload display. More on that in a bit.

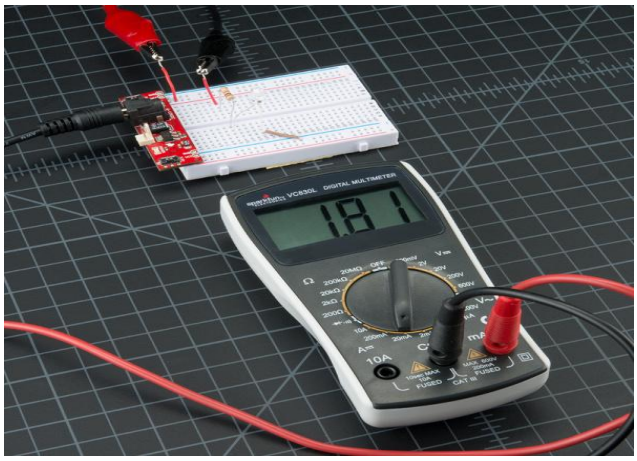


Figure 70:Current Measurement
This circuit was only pulling 1.8mA at the time of measurement, not a lot of current. The average reading was closer to 2.1mA.

Realize that the multimeter is acting as a piece of wire – you've now completed the circuit, and the circuit will power on. This is important because as time goes on the LED, microcontroller, sensor, or whatever device being measured may change its power consumption (such as turning on an LED can resulting in a 20mA increase for a second, then decrease for a second when it turns off). On the multimeter display you should see the instantaneous current reading. All multimeters take readings over time and then give you the **average**, so expect the reading to fluctuate. In general, cheaper meters will average more harshly and respond more slowly, so take each reading with a grain of salt. In your head, take an average range such as 7 to 8mA under normal 5V conditions (not 7.48mA).

Similar to the other measurements, when measuring current, the color of the probes does not matter. What happens if we switch probes? Nothing bad happens! It simply causes the current reading to become negative:

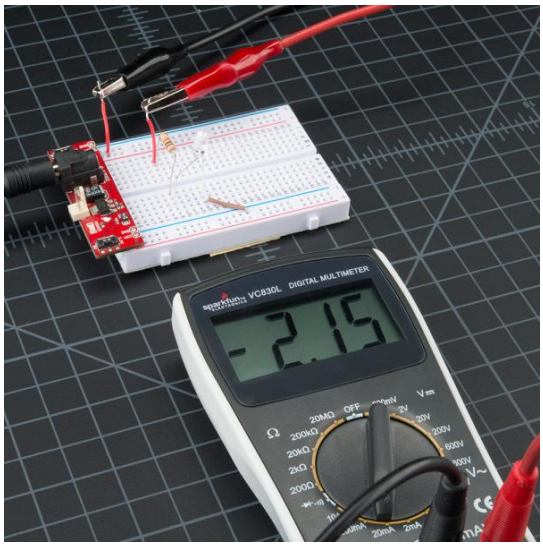


Figure 71:Current Measurement
Current is still flowing through the system, you've just changed your perspective and now the meter reads negative.

Remember! When you're done using the meter, always return the meter to read voltage (return the probes to the voltage port, set the meter to read the DC voltage range if necessary). It's common to grab a meter and begin to quickly measure the voltage between two pins. If you have left your meter in 'current' mode, you won't see the voltage on the display. Instead you'll see '0.000' indicating that there is no current between VCC and GND. Within that split second you will have connected VCC to GND through your meter and the 200mA fuse will blow = not good. So before you put the meter down for the night, remember to leave your meter in a friendly state.

Measuring current can be tricky the first couple of times. Don't worry if you blow the fuse - we've done it dozens of times! We'll show you how to replace the fuse in a later section.

Continuity

Continuity testing is the act of testing the resistance between two points. If there is very low resistance (less than a few Ωs), the two points are connected electrically, and a tone is emitted. If there is more than a few Ωs of resistance, than the circuit is open, and no tone is emitted. This test helps insure that connections are made correctly between two points. This test also helps us detect if two points are connected that should not be.

Continuity is quite possibly the single most important function for embedded hardware gurus. This feature allows us to test for conductivity of materials and to trace where electrical connections have been made or not made.

Set the multimeter to 'Continuity' mode. It may vary among DMMs, but look for a diode symbol with propagation waves around it (like sound coming from a speaker).

Figure 72:Multimeter is set to continuity mode.

Now touch the probes together. The multimeter should emit a tone (Note: Not all multimeters have a continuity setting, but most should). This shows that a very small amount of current is allowed to flow without resistance (or at least a very very small resistance) between probes.

Warning! In general, turn OFF the system before checking for continuity.

On a breadboard that is *not* powered, use the probes to poke at two separate ground pins. You should hear a tone indicating that they are connected. Poke the probes from the VCC pin on a

microcontroller to VCC on your power supply. It should emit a tone indicating that power is free to flow from the VCC pin to the micro. If it does not emit a tone, then you can begin to follow the route that copper trace takes and tell if there are breaks in the line, wire, breadboard, or PCB.

Continuity is a great way to test if two SMD pins are touching. If your eyes can't see it, the multimeter is usually a great second testing resource.

When a system is not working, continuity is one more thing to help troubleshoot the system. Here are the steps to take:

1. If the system is on, carefully check VCC and GND with the voltage setting to make sure the voltage is the correct level. If the 5V system is running at 4.2V check your regulator carefully, it could be very hot indicating the system is pulling too much current.
2. Power the system down and check continuity between VCC and GND. If there is continuity (if you hear a beep), then you've got a short somewhere.
3. Power the system down. With continuity, check that VCC and GND are correctly wired to the pins on the microcontroller and other devices. The system may be powering up, but the individual ICs may be wired wrong.
4. Assuming you can get the microcontroller running, set the multimeter aside, and move on to serial debugging or use a logic analyzer to inspect the digital signals.

Continuity and large capacitors: During normal troubleshooting. you will be probing for continuity between ground and the VCC rail. This is a good sanity check before powering up a prototype to make sure there is not a short on the power system. But don't be surprised if you hear a short 'beep!' when probing. This is because there is often significant amounts of capacitance on the power system. The multimeter is looking for very low resistance to see if two points are connected. Capacitors will act like a short for a split second until they fill up with energy, and then act like an open connection. Therefore, you will hear a short beep and then nothing. That's ok, it's just the caps charging up.

Changing the Fuse

One of the most common mistakes with a new multimeter is to measure current on a bread board by probing from VCC to GND (bad!). This will immediately short power to ground through the multimeter causing the bread board power supply to brown out. As the current rushes through the multimeter, the internal fuse will heat up and then burn out as 200mA flows through it. It will happen in a split second and without any real audible or physical indication that something is wrong.

Wow, that was neat. Now what? Well first, remember that measuring current is done in series (interrupt the VCC line to the breadboard or microcontroller to measure current). If you try to measure the current with a blown fuse, you'll probably notice that the meter reads '0.00' and that the system doesn't turn on like it should when you attach the multimeter. This is because the internal fuse is broken and acts as a broken wire or open. Don't worry, this happens all the time, and it costs about $1 to fix.

To change the fuse, find your handy dandy mini screw driver, and start taking out screws. The Electronicians DMM is pretty easy to pull apart. Start by removing the battery plate and the battery.

Figure 73:Changing the Fuse
Next, remove the two screws hiding behind the battery plate.

Figure 74:Changing the Fuse
Lift the face of the multimeter slightly.

Figure 75:Changing the Fuse

Now notice the hooks on the bottom edge of the face. You will need to slide the face sideways with a little force to disengage these hooks.

Figure 76:Changing the Fuse

Once the face is unhooked, it should come out easily. Now you can see inside the multimeter!

Figure 77:Changing the Fuse
Gently lift up on the fuse, and it will pop out.

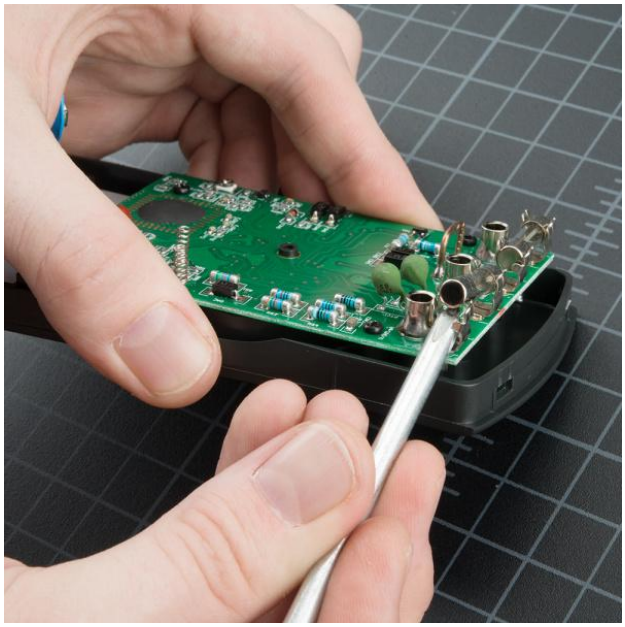


Figure 78:Changing the Fuse
Hands on: Demonstrate how to replace a fuse .

Make sure to **replace the correct fuse with the correct type**. In other words, replace the 200mA fuse with a 200mA fuse.

Warning! DO NOT put a 10A fuse where a 200mA fuse should go. The placement of the fuses may not match the placement of the probe ports. Read the metal cap on either end of the fuse to double check which is which.
The components and PCB traces inside the multimeter are designed to take different amounts of current. You will damage and possibly ruin your multimeter if you accidentally push 5A through the 200mA port.

There are times where you need to measure high current devices like a motor or heating element. Do you see the two places to put the red probe on the front of the

multimeter? **10A** on the left and **mAVΩ** on the right? If you try to measure more than 200mA on the **mAVΩ** port you run the risk of blowing the fuse. But if you use the 10A port to measure current, you run a much lower risk of blowing the fuse. The trade-off is sensitivity. As we talked about above, by using the 10A port and knob setting, you will only be able to read down to 0.01A or 10mA. Most of my systems use more than 10mA so the 10A setting and port works well enough. If you're trying to measure very low power (micro or nano amps) the 200mA port with the 2mA, 200uA, or 20uA could be what you need.

Figure 79: Current Induction

Remember: If your system has the potential to use more than 100mA you should start with the red probe plugged into the **10A** port and **10A** knob setting.

Hands on: at least two trainees will be given a multi-meter and perform different measurements .

- **How to Use a Breadboard**

Breadboards are one of the most fundamental pieces when learning how to build circuits. In this section, you will learn a little bit about what breadboards are, why they are called breadboards, and how to use one. Once you are done you should have a basic understanding of how breadboards work and be able to build a basic circuit on a breadboard.

Figure 80: Breadboard

Where to locally buy this?: http://naradaelectronics.rw/

History

If you wanted to build a circuit prior to the 1960s, chances are you would have used a technique called wire-wrap. Wire wrap is a process that involves wrapping wires around conductive posts attached to a perfboard (a.k.a. a protoboard). As you can see, the process can get rather complex very quickly.Although this method is still used today, there is something that makes prototyping much easier, breadboards!

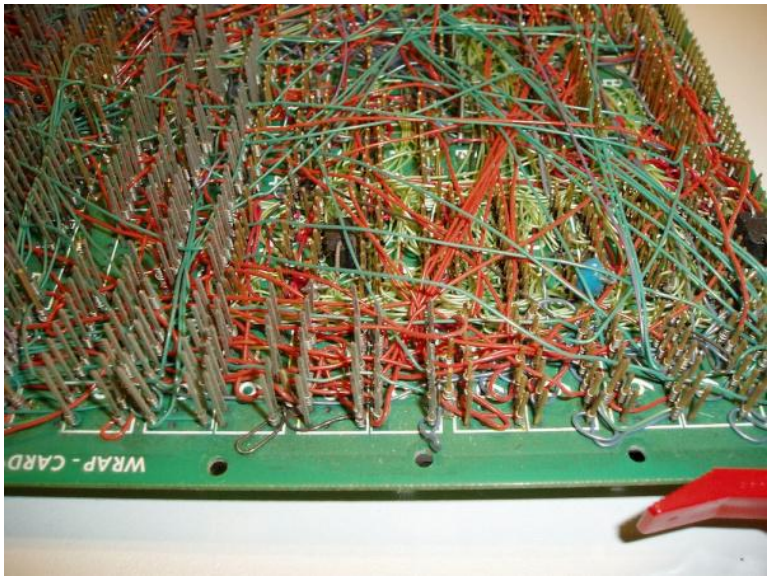


Figure 81:A wire-wrap circuit (image courtesy of Wikipedia user Wikinaut)

What's in a Name?

When you picture a breadboard in your head, you may envision a big piece of wood and a large loaf of freshly baked bread. You wouldn't be too far off either.

Figure 82:Bread on a breadboard

So why do we call this electronic "circuit builder" a breadboard? Many years ago, when electronics were big and bulky, people would grab their mom's breadboard, a few nails or thumbtacks, and start connecting wires onto the board to give themselves a platform on which to build their circuits.

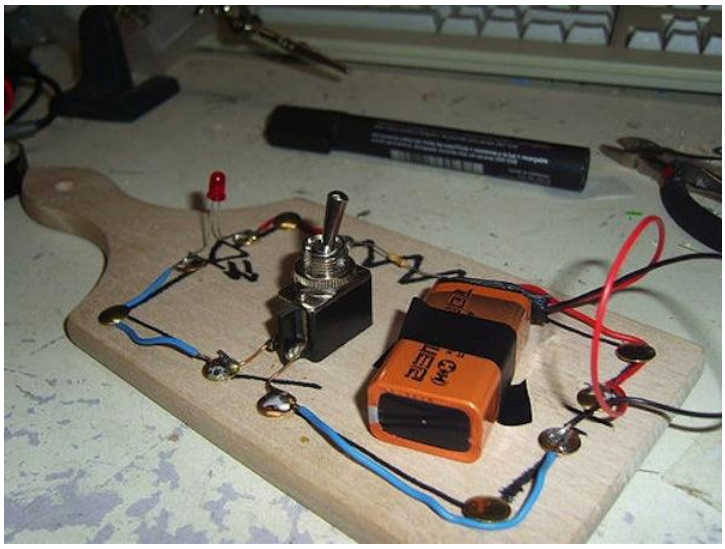


Figure 83:Circuit on an "original" breadboard (image courtesy of mischka and their awesome literal breadboard section)

Since then, electronic components have gotten a lot smaller, and we've come up with better ways to connect circuits, making moms all over the world happy to have their breadboards back. However, we are stuck with the confusing name. Technically, these are still breadboards, but this discussion is going to be on modern, "solderless" breadboards.

Why Use Breadboards?

An electronics breadboard (as opposed to the type on which sandwiches are made) is actually referring to a **solderless breadboard**. These are great units for making temporary circuits and prototyping, and they require absolutely no soldering.

Prototyping is the process of testing out an idea by creating a preliminary model from which other forms are developed or copied, and it is one of the most common uses for breadboards. If you aren't sure how a circuit will react under a given set of parameters, it's best to build a prototype and test it out.

For those new to electronics and circuits, breadboards are often the best place to start. That is the real beauty of breadboards–they can house both the simplest circuit as well as very complex circuits. As you'll see later in this section, if your circuit outgrows its current breadboard, others can be be attached to accommodate circuits of all sizes and complexities.

Another common use of breadboards is testing out new parts, such as Integrated circuits (ICs). When you are trying to figure out how a part works and constantly rewiring things, you don't want to have to solder your connections each time.

As mentioned, you don't always want the circuit you build to be permanent. When trying to duplicate a customer's problem, will often use breadboards to build, test, and analyze the circuit. We can connect the parts the customer has, and once they've gotten the circuit setup and figured out the problem, they can take everything apart and put it aside for the next time they need to do some troubleshooting.

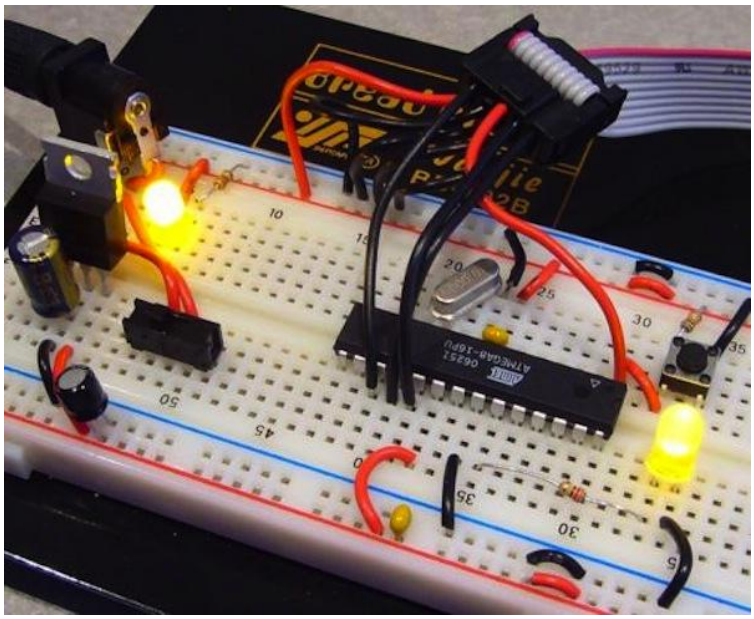


Figure 84:A circuit built on a solderless breadboard

Anatomy of a Breadboard

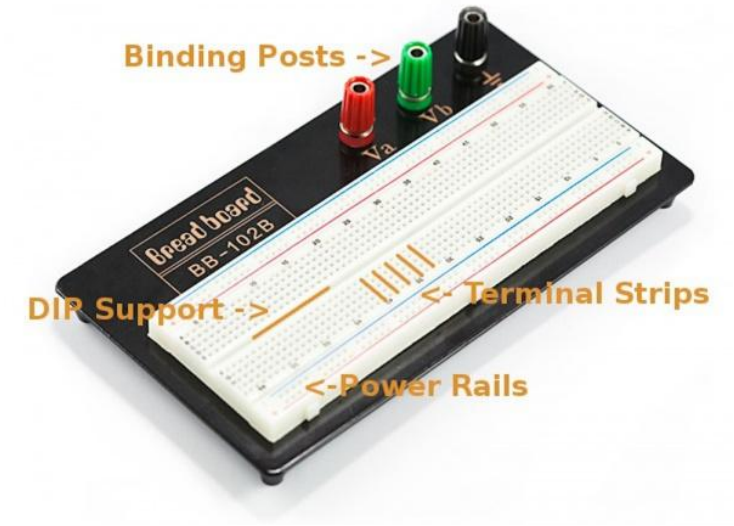


Figure 85:The major features of a Breadboard

The best way to explain how a breadboard works is to take it apart and see what's inside. Using a smaller breadboard it's easier to see just how they function.

Terminal Strips

Here we have a breadboard where the adhesive backing has been removed. You can see lots of horizontal rows of metal strips on the bottom of the breadboard.

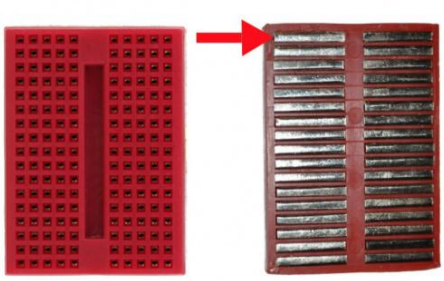


Figure 86:A Electronicians Mini Breadboard from the top (left) and the same breadboard flipped over with the adhesive back removed (right).
The tops of the metal rows have little clips that hide under the plastic holes. These clips allow you to stick a wire or the leg of a component into the exposed holes on a breadboard, which then hold it in place.

Figure 87:A single strip of conductive metal removed from the above breadboard.
Once inserted that component will be electrically connected to anything else placed in that row. This is because the metal rows are conductive and allow current to flow from any point in that strip.

Notice that there are only five clips on this strip. This is typical on almost all breadboards. Thus, you can only have up to five components connected in one particular section of the breadboard. The row has ten holes, so why can you only connect five components? You'll also notice that each horizontal row is separated by a ravine, or crevasse, in the middle of the breadboard. This ravine isolates both sides of a given row from one another, and they are not electrically connected. We'll discuss the purpose of this in just a bit, but, for now, just know that each side of a given row is disconnected from the other, leaving you with five spots for components on either side.

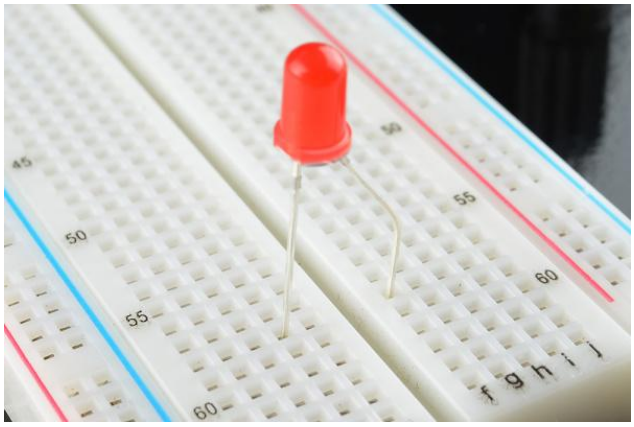


Figure 88:An LED inserted into a breadboard. Notice how each leg of the LED is placed on either side of the ravine. This prevents the connections to the LED from being shorted

.

Power Rails

Now that we've seen how the connections in a breadboard are made, let's look at a larger, more typical breadboard. Aside from horizontal rows, breadboards usually have what are called power rails that run vertically along the sides.

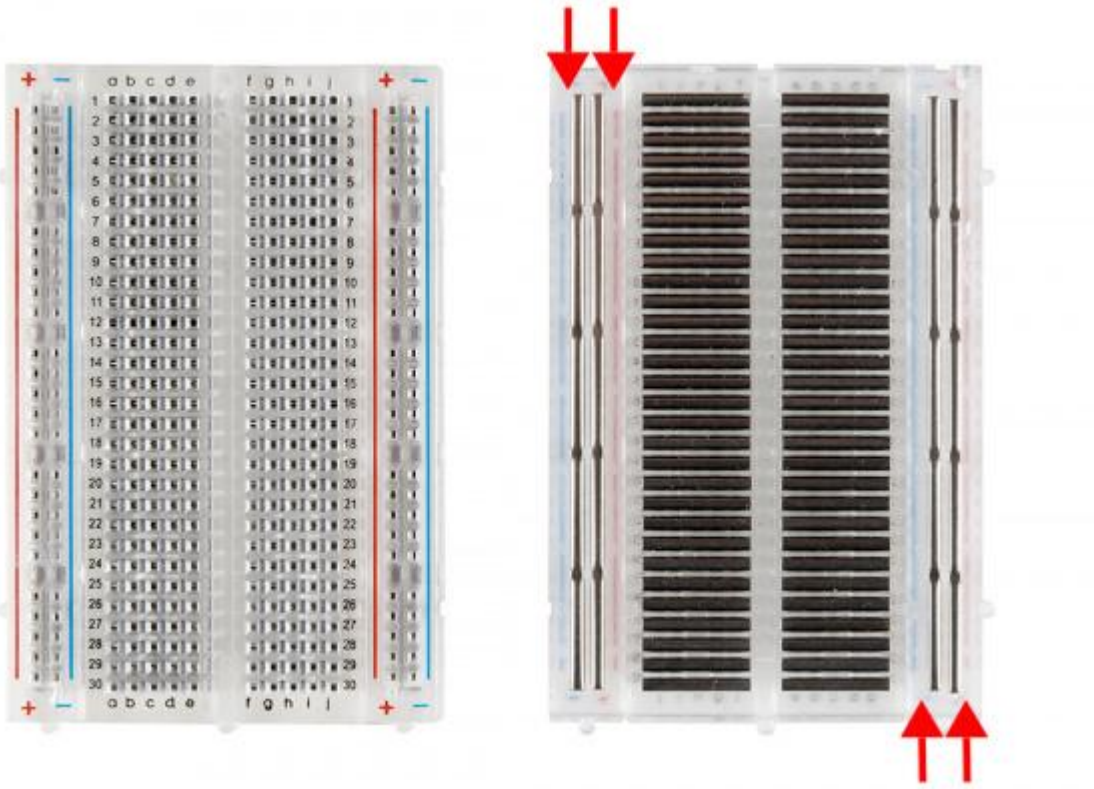


Figure 89:A medium-size breadboard with the adhesive back removed to expose the power rails.

These power rails are metal strips that are identical to the ones that run horizontally, except they are, typically*, all connected. When building a circuit, you tend to need power in lots of different places. The power rails give you lots of easy access to power wherever you need it in your circuit. Usually they will be labeled with a '+' and a '-' and have a redand blue or black stripe, to indicate the positive and negative side.

It is important to be aware that the power rails on either side are not connected, so if you want the same power source on both sides, you will need to connect the two sides with some jumper wires. Keep in mind that the markings are there just as a reference. There is no rule that says you have to plug power into the '+' rail and ground into the '-'rail, though it's good practice to keep everything in order.

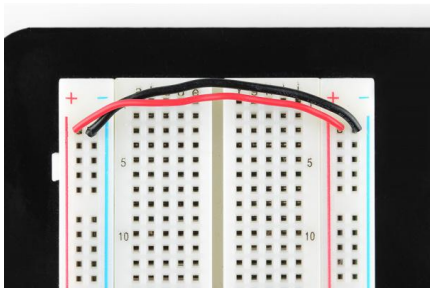


Figure 90:Two jumper wires used to connect the power rails on both sides. Always attach the '+' to '+' and the '-' to '-'.

DIP Support

Earlier we mentioned the ravine that isolates the two sides of a breadboard. This ravine serves a very important purpose. Many integrated circuits, often referred to as ICs or, simply, chips, are manufactured specifically to fit onto breadboards. In order to minimize the amount of space they take up on the breadboard, they come in what is known as a Dual in-line Package, or DIP.

These DIP chips (salsa anyone?) have legs that come out of both sides and fit perfectly over that ravine. Since each leg on the IC is unique, we don't want both sides to be connected to each other. That is where the separation in the middle of the board comes in handy. Thus, we can connect components to each side of the IC without interfering with the functionality of the leg on the opposite side.

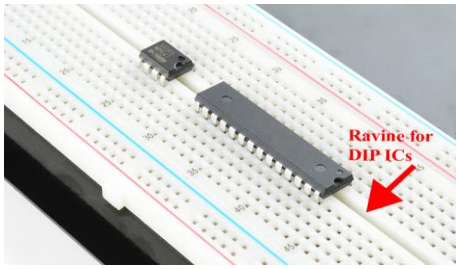


Figure 91:Two DIP ICs, the LM358 (top), a very common op-amp, and the ever-popular ATmega328 microcontroller (bottom).

Rows and Columns

You may have noticed that many breadboards have **numbers** and **letters** marked on various rows and columns. These don't serve any purpose other than to help guide you when building your circuit. Circuits can get complicated quickly, and all it takes is one misplaced leg of a component to make the entire circuit malfunction or not work at all. If you know the row number of the connection you are trying to make, it makes it much simpler to plug a wire into that number rather than eyeballing it.

Breadboard Power Supplies

Yet another method for powering your breadboard is to use one of the many breadboard power supplies available. Electronicians carries a number of kits and boards that you can use to plug power directly into your breadboard. Some allow you to plug a wall wart directly into the breadboard. Others allow you to pull power directly from your computer via the USB connections. And, almost all of them have the capability to adjust the voltage, giving you a full range of the common voltages needed when building circuits.

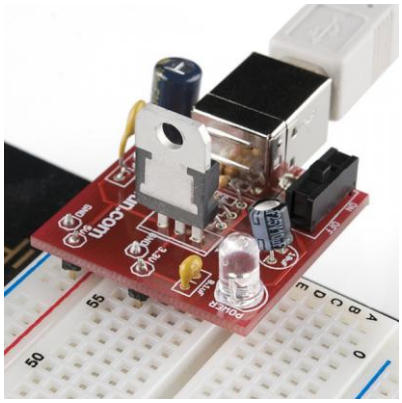


Figure 92: A Electronicians USB Breadboard Power Supply that pulls power from your computer's USB and has the option to choose between 3.3V and 5V.

Building Your First Breadboard Circuit

Now that we're familiar with the internals of a breadboard and how to provide power to them, what do we do with them? We are going to start with a simple circuit.

Here is a small circuit on a breadboard.

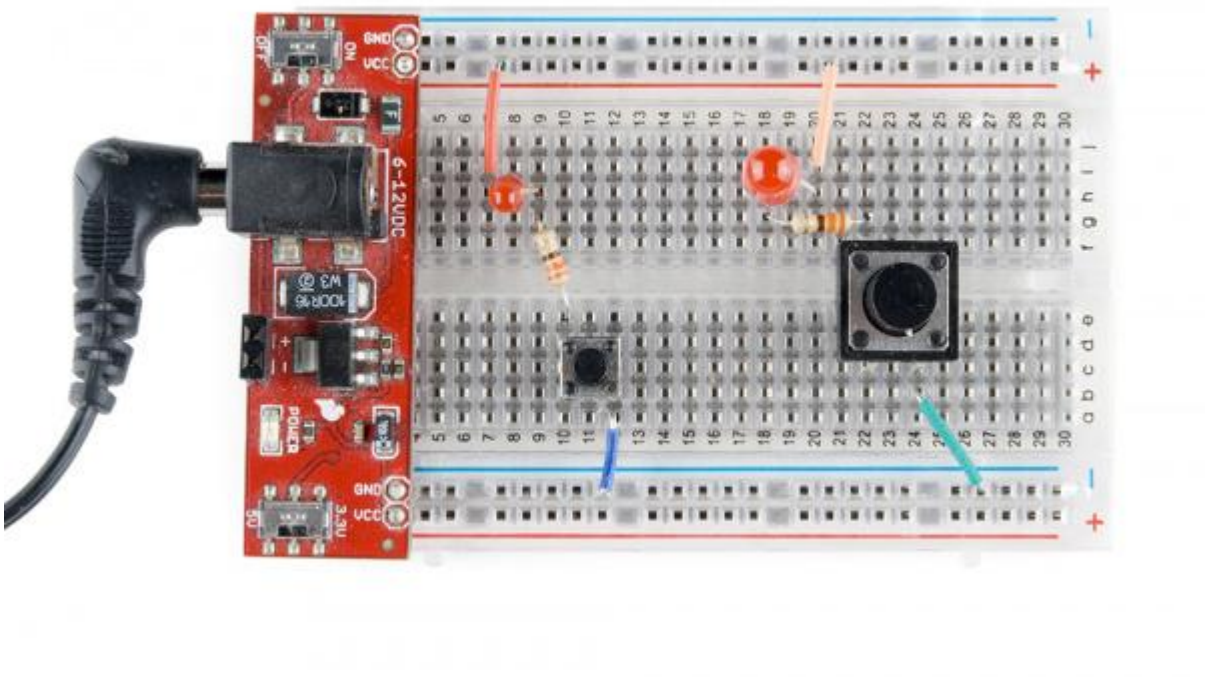


Figure 93:A simple circuit, involving a button, an LED, and a resistor, built two different ways.

The red board you see a Breadboard Power Supply, which supplies 5V to the power rails when it is connected to a 9V wall wart.

The circuit goes as follows:

- There is a wire connecting the 5V power rail to the positive, anode leg of an LED.
- The negative, cathode leg of the LED is connected to a 330Ω resistor.
- The resistor is then connected to a button.
- When the button is pushed, it connects the circuit to ground completing the circuit and turning on the LED.

Circuit Schematics

Schematics are universal pictograms that allow people all over the world to understand and build electronics. Every electronic component has a very unique schematic symbol. These symbols are then assembled into circuits using a variety of programs. You could also draw them out by hand. If you want to dive deeper in the world of electronics and circuit building, learning to read schematics is a very important step in doing so.

Here we have a schematic for the above circuit. Power (5V) is represented by the arrow at the top. It then goes to the LED (the triangle and line with arrows emitting out of it). The LED is then connected to the resistor (the squiggly line). That is connected to the button (the latch-looking symbol). Last the button is connect to ground (the horizontal line at the bottom).

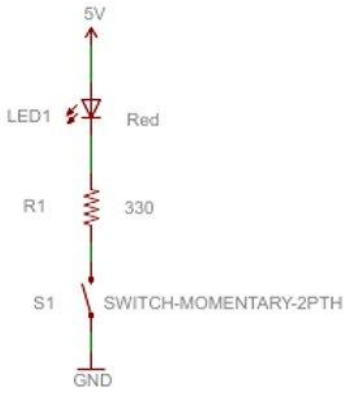


Figure 94: Simple Circuit

This may seem like a funny way to draw a circuit, but it is a fundamental process that has been around for decades. Schematics allow people from different nationalities and languages to build and collaborate on circuits designed by anyone. As mentioned, you can *build* a circuit in many different ways, but, as this schematic shows, there are certain connections that must be made. Diverging from this schematic will give you an entirely different circuit.

Practice Makes Perfect

The last bit of knowledge to leave you with is that there are tons of resources and programs you can use to build circuits without having to actually use your breadboard. One very common program used by Electronicians is Fritzing. Fritzing is a free program that allows you to build your own circuits on a virtual breadboard. It also provides schematic views for all the circuits you build. Here we can see the same circuits as above built using Fritzing.

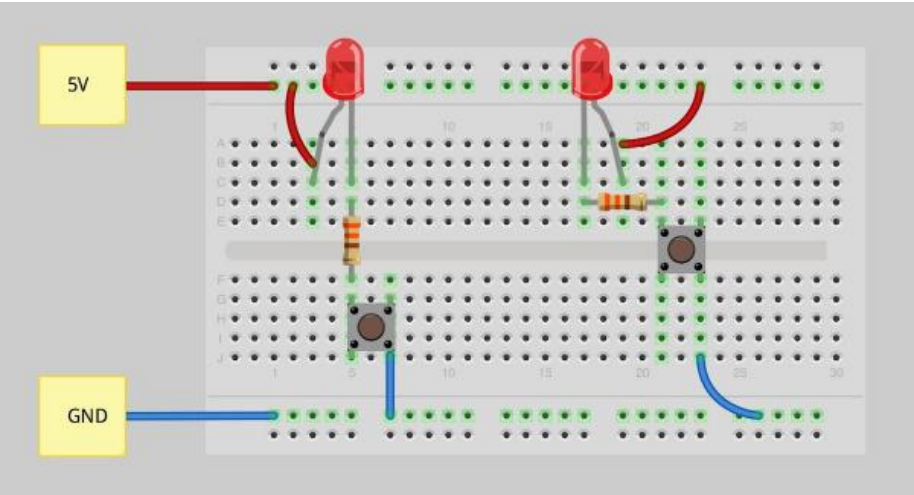


Figure 95:Notice that the green lines indicate to which rows and columns each component is connected.

There are many other programs like Fritzing. Some are free, and some are paid. Some will even allow you to build a circuit and test its functionality through simulations. Go explore the internet, and find the tools that work best for you.

Hands on: Breadboard , jumper wires and some components will be available for making same simple circuis .

- **practicing Ohm's and Kirchhoff's laws**

Ohm's Law is your golden ticket for calculating the voltage, current, or resistance in a simple series or parallel circuit, but what happens when your circuit is more complicated? You might be designing electronics that have both parallel and series resistance, and Ohm's Law starts to fall down. Or what if you don't have a constant current source? In these situations, when you can't only use V = IR, then it's time to stand on the shoulders of Ohm and use Kirchhoff's Circuit Law. Here we'll be looking at what Kirchhoff's Circuit Law is, and how to use it to analyze the voltage and current of complex electrical circuits.

What is Kirchhoff's Circuit Law?

When you're building a complex circuit that includes bridges or T networks, then you can't solely rely on Ohm's Law to find the voltage or current. This is where Kirchhoff's Circuit Law comes in handy, which allows you to calculate both the current and voltage for complex circuits with a system of linear equations. There are two variations of Kirchhoff's Law, including:

- **Kirchhoff's Current Law:** To analyze the total current for a complex circuit
- **Kirchhoff's Voltage Law**: To analyze the total voltage for a complex circuit
- When you combine these two laws, you get **Kirchhoff's Circuit Law**

Like any other scientific or mathematical law named after their creator, Kirchhoff's Circuit Law was invented by German Physicist Gustav Kirchhoff. Gustav was known for many achievements in his lifetime, including the theory of spectrum analysis which proved that elements give off a unique light pattern when heated. When Kirchhoff and chemist Robert Bunsen analyzed these light patterns through a prism, they discovered that each element in the periodic table has its own unique wavelength. The discovery of this pattern allowed the duo to uncover two new elements, cesium and rubidium.

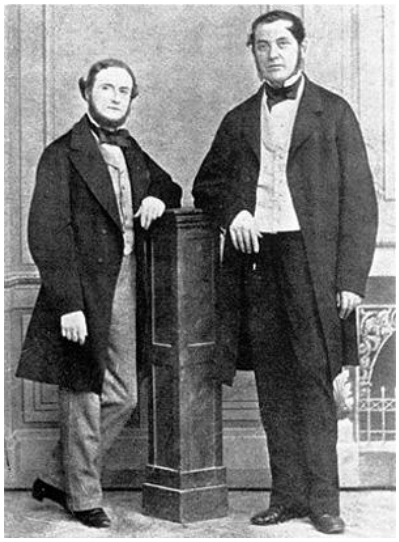


Figure 96:Gustav Kirchhoff (left) and Robert Bunsen (right)

Kirchhoff later went on to apply his spectrum analysis theory to study the composition of the sun, where he discovered many dark lines in the sun's wavelength spectrum. This was caused by gas from the sun absorbing specific wavelengths of light, and this discovery marked the beginning of a new age of research and exploration in the field of astronomy.

A bit closer to home in the world of electronics, Kirchhoff announced his set of laws for analyzing the current and voltage for electrical circuits in 1845, known today as Kirchhoff's Circuit Law. This work builds upon the foundation outlined in Ohm's Law and has helped paved the way for the complex circuit analysis that we rely on today.

First Law – Kirchhoff's Current Law

Kirchhoff's Current Law states that the amount of current that enters a node is equal to the amount of current leaving a node. Why? Because when current enters a node, it has no other place to go except to exit. What goes in must come out. You can identify a node where two or more paths are connected via a common point. In a schematic, this will be the junction dot connecting two intersecting net connections.

Take a look at the image below to understand this Law visually. Here we have two currents entering a node, and three currents leaving the node. According to Kirchhoff's Current Law, the relationship between these currents entering and exiting the node can be represented as **I1 + I2 = I3 + I4 + I5**.

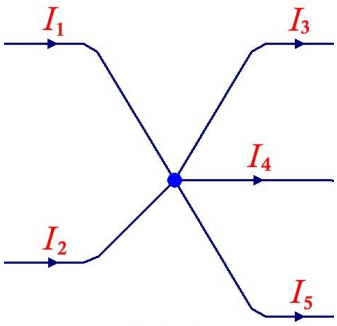


Figure 97:Kirchhoff's Current Law, current in must equal current out

Kirchhoff's Current Law, current in must equal current out

When you balance this equation as an algebraic expression, then you conclude that current entering and exiting a node will always equal 0, or **I1 + I2 + (-I3 + -I4 + -I5) = 0** Everything has to balance out, and Kirchhoff called this principle the **Conservation of Charge**.

Let's look at an example circuit to see how this works. Below we have a circuit with four nodes: A, C, E, and F. Current first flows from its voltage source and separates at Node A, which then flows through resistors R1 and R2. From there, the current recombines at Node C and splits again to flow through resistors R3, R4, and R5 where it meets Node E and Node F.

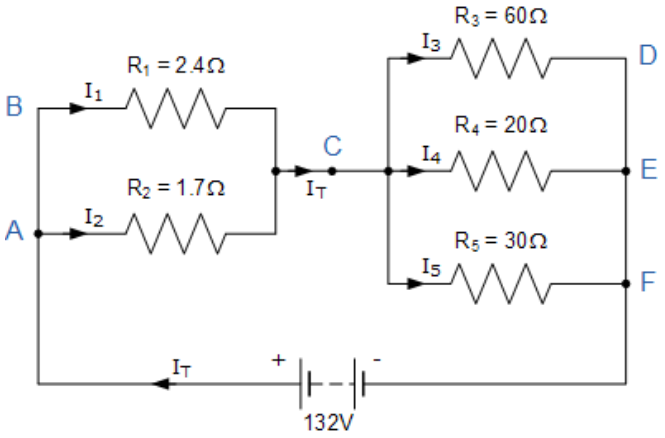


Figure 98: Resistor Network

To validate Kirchoff's Current Law in this circuit, we need to take the following steps:

1. Calculate the total current of the circuit
2. Calculate the current flowing through each node
3. Compare input and output currents at specific nodes to validate Kirchoff's Current Law.

1. Calculate total current

Here we use Ohm's Law to get the total current of our circuit with **I = V/R**. We already have our total voltage of 132V, and now we just need to find the total resistance in all of our nodes. This requires the simple method of calculating the total resistance of resistors wired in parallel, which is:

Equation 11: Resistor in Parallel

$$\frac{1}{R_{EQ}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \ldots + \frac{1}{R_N}$$

When you have only two resistors in parallel: $R_{EQ} = \frac{R_1 \times R_2}{R_1 + R_2}$

Starting at Node AC, we get the following resistance for parallel resistors R1 and R2:

Equation 12:RAC

$$R_{AC} = \frac{2.4\ x\ 1.7}{2.4 + 1.7}$$

$$R_{AC} = 1\Omega$$

And moving on to Node CEF, we get the following resistance for parallel resistors R3, R4, and R5:

Equation 13:RCEF

$$R_{CEF} = \frac{1}{60} + \frac{1}{20} + \frac{1}{30}$$

$$R_{CEF} = 10\Omega$$

We now have our total resistance of 11 Ohms for the entire circuit, which we can then plug into Ohm's Law **I = V/R** to get the total current in our circuit:

Equation 14:Total Current

$$I_T = \frac{132}{11} = 12\ Amps$$

2. Calculate node currents

Now that we know we have 12 amps flowing out of our circuit, we can calculate the current at each set of nodes. We'll again enlist the help of Ohm's Law in the form of **I = V/R** to get the current for each node branch.

First, we need the voltages for node branches AC and CF:

Equation 15: VAC and VCEF

$$V_{AC} = I_T \, x \, R_{AC} = 12 \, x \, 1 \; = \; 12 \; volts$$
$$V_{CEF} = I_T \, x \, R_{CEF} = 12 \, x \, 10 \; = \; 120 \; volts$$

Then we can calculate the current for each node branch:

Equation 16: Branch Currents

$$I_1 = \frac{12}{2.4} = 5 \; Amps$$
$$I_2 = \frac{12}{1.7} = 7 \; Amps$$
$$I_3 = \frac{120}{60} = 2 \; Amps$$
$$I_4 = \frac{120}{20} = 6 \; Amps$$
$$I_5 = \frac{120}{30} = 4 \; Amps$$

3. Validate Kirchhoff's Current Law

With the current for each node branch calculated, we now have two distinct reference points that we can use to compare our input and output currents. This will allow us to analyze our circuit and validate Kirchhoff's Current Law like so:

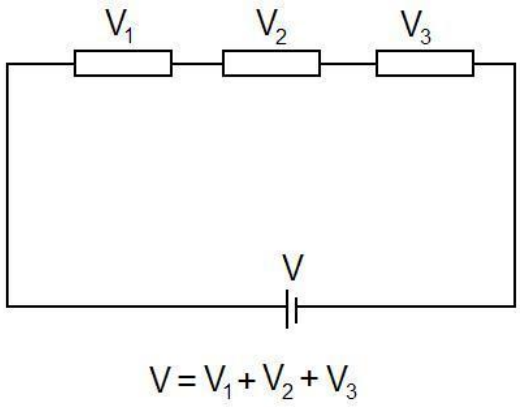
Equation 17: Verifications

$$I_T = I_1 + I_2 = I_3 \; + I_4 + I_5$$
$$12 = (5 + 7) = (2 + 6 + 4)$$

Second Law – Kirchhoff's Voltage Law

Kirchhoff's Voltage Law states that in any closed loop circuit the total voltage will always equal the sum of all the voltage drops within the loop. You'll find voltage drops occurring whenever current flows through a passive component like a resistor, and Kirchhoff referred to this law as the **Conservation of Energy**. Again, what goes in must come out.

Take a look at the image below to understand this visually. In this circuit, we have a voltage source, and four areas in the circuit where the voltage will encounter a passive component, which will cause a distinct voltage drop.



Since these passive components are connected in series, you can simply add the total voltage drops together, and compare it to the total voltage to get a relationship that looks like this:

$$V_s = V_1 + V_2 + V_3$$

Let's start with a straightforward circuit to demonstrate how this works. In the example below, we have two known variables, the total voltage and the voltage drop across R1.

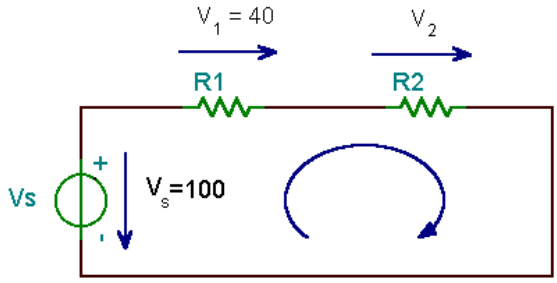


Figure 99: Current Flow

What we need to figure out is the voltage drop across R2, and we can use Kirchoff's Voltage Law to figure this out with the following relationship:

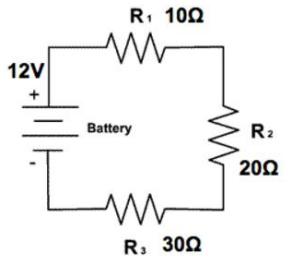
$$V_2 = V_s - V_1 = 100\ volts - 40\ volts = 60\ volts$$

Since the total voltage drop in the circuit has to equal the total voltage source, this provides an easy way to calculate our missing variable. If you wanted to express this relationship as a

proper algebraic expression, you'd get the sum of all voltage drops and the total voltage equalling zero as shown here:

$$-V_s + V_1 + V_2 = 0$$

Let's look at another example. In the circuit below we have three resistors connected in series with a 12 volt battery.



To validate Kirchoff's Voltage Law in this circuit, we need to take the following steps:

1. Calculate the **total resistance** of the circuit
2. Calculate the **total current** of the circuit
3. Calculate the **current through** each resistor
4. Calculate the **voltage drop across** each resistor

Compare the **voltage source to total voltage drop** to validate Kirchoff's Voltage Law

1. Calculate total resistance

Since all of our resistors are wired in series, we can easily find the total resistance by just adding all of the resistance values together as so:

$$R_T = R_1 + R_2 + R_3 = 10\Omega + 20\Omega + 30\Omega = 60\Omega$$

2. Calculate the total current

Now that we know our total resistance, we can again use Ohm's Law to get the total current of our circuit in the form of **I = V/R,** which looks like this:

$$I = \frac{V_s}{R_T} = \frac{12}{60} = 0.2\ amps$$

3. Calculate current through each resistor

Since all of our resistors are wired in series they will all have the same amount of current flowing through them, which we can express as:

$$I_{R1} = I_{R2} = I_{R3} = I_{SERIES} = 0.2\ amps$$

4. Calculate the voltage drop across each resistor

Our final calculation will again use Ohm's Law to give us the total voltage drop for each resistor in the form of **V = IR**, which looks like this:

$$V_{R1} = I\ x\ R_1 = 0.2\ x\ 10 = 2\ volts$$
$$V_{R2} = I\ x\ R_2 = 0.2\ x\ 20 = 4\ volts$$
$$V_{R3} = I\ x\ R_3 = 0.2\ x\ 30 = 6\ volts$$

5. Validate Kirchoff's Voltage Law

Now we have all of the data we need, including the total voltage of our circuit, along with each voltage drop across each of our resistors. When putting all of this together, we can easily validate Kirchhoff's Voltage Law with this relationship:

$$V_s = V_{R1} + V_{R2} + V_{R3}$$
$$12 = 2 + 4 + 6$$

This can also be expressed as:

$$V_s + (-V_{R1}) + (-V_{R2}) + (-V_{R3}) = 0$$
$$12 + (-2) + (-4) + (-6) = 0$$

As you can see, the total voltage equals the total voltage drop in our circuit. What goes in must come out, and Kirchhoff's Law works yet again!

Process for Using Kirchhoff's Circuit Law

With an understanding of how Kirchhoff's Circuit Law works, you now have a new tool in your toolbox for analyzing voltage and current in complete circuits. When using these Laws out in the wild consider using the following step-by-step process:

1. First, begin by labeling all of the known voltages and resistances on your circuit.
2. Then name each branch on your circuit with a current label, such as I1, I2, I3,etc. A branch is a single or group of components connected between two nodes.
3. Next, find Kirchhoff's Current Law for each node in your circuit.
4. Then find Kirchhoff's Voltage Law for each of the independent loops in your circuit.

Once you have Kirchoff's Current and Voltage Laws calculated, ou can then use your equations to find any missing currents. Ready to try it on your own? Take a look at the circuit below and see if you can validate Kirchoff's Current Law and Voltage Law with a little bit of help from Ohm!

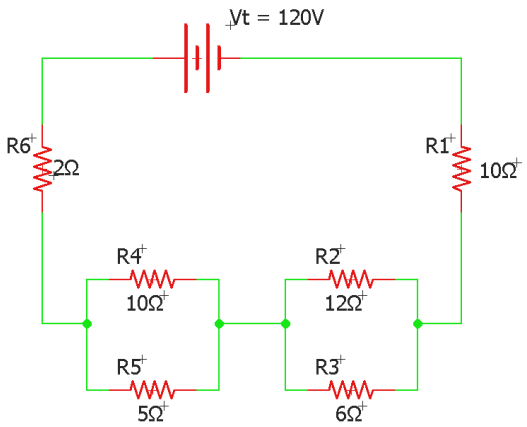


Figure 100: Series and Parallel Circuit

Standing On the Shoulders of Ohm

With Kirchhoff's Circuit Law in hand, you now have all the tools you need to analyze the voltage and current for complex circuits. Like many other scientific and mathematical principles, Kirchhoff's Law stands on the shoulders of what came before it – Ohm's Law. You'll find yourself using Ohm's Law to calculate individual resistances, voltages, or currents, and then building upon these calculations with Kirchhoff's Law to see if your circuit holds true to these Current and Voltage principles.

Hands on: A Resistor combination circuit will be made so that ohm and kirchoofs laws can be practically verified .

- **How to Use an Oscilloscope**

Have you ever found yourself troubleshooting a circuit, needing more information than a simple multimeter can provide? If you need to uncover information like frequency, noise, amplitude, or any other characteristic that might change over time, you need an oscilloscope!

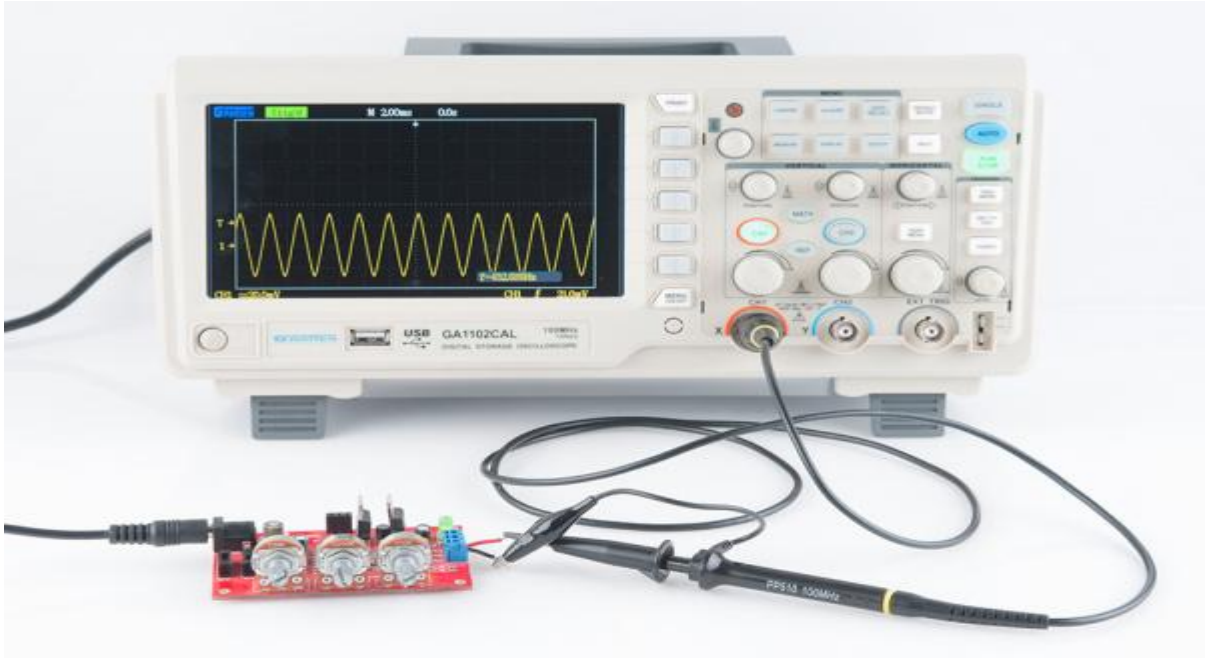


Figure 101: Digital Oscilloscope

O-scopes are an important tool in any electrical engineer's lab. They allow you to *see* electric signals as they vary over time, which can be critical in diagnosing why your 555 timer circuit isn't blinking correctly, or why your noise maker isn't reaching maximum annoyance levels.

Covered in This Section

This Section aims to introduce the concepts, terminology, and control systems of oscilloscopes. It's broken down into the following sections:

- Basics of O-Scopes – An introduction to what, exactly, oscilloscopes are, what they measure, and why we use them.
- Oscilloscope Lexicon – A glossary covering some of the more common oscilloscope characteristics.
- Anatomy of an O-Scope – An overview of the most critical systems on an oscilloscope – the screen, horizontal and vertical controls, triggers, and probes.
- Using an Oscilloscope – Tips and tricks for someone using an oscilloscope for the first time.

We'll be using the Gratten GA1102CAL – a handy, mid-level, digital oscilloscope – as the basis for our scope discussion. Other o-scopes may look different, but they should all share a similar set of control and interface mechanisms.

Basics of O-Scopes

The main purpose of an oscilloscope is to **graph an electrical signal as it varies over time**. Most scopes produce a two-dimensional graph with **time on the x-axis** and **voltage on the y-axis**.

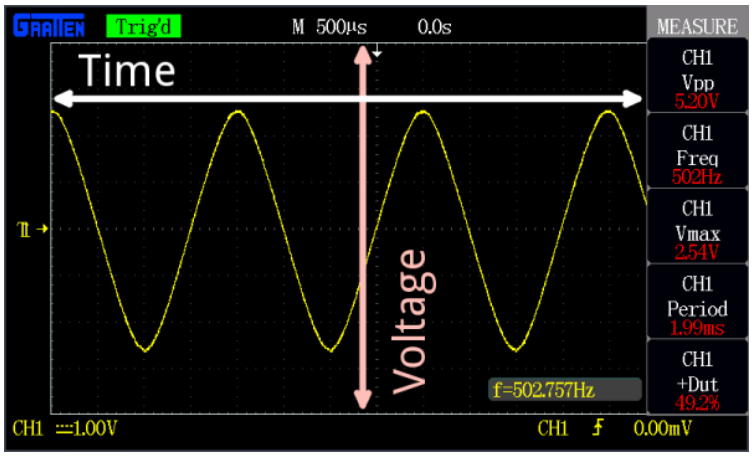


Figure 102: An example of an oscilloscope display. A signal (the yellow sine wave in this case) is graphed on a horizontal time axis and a vertical voltage axis.

Controls surrounding the scope's screen allow you to **adjust the scale** of the graph, both vertically and horizontally – allowing you to zoom in and out on a signal. There are also controls to set the **trigger** on the scope, which helps focus and stabilize the display.

What Can Scopes Measure

In addition to those fundamental features, many scopes have measurement tools, which help to quickly quantify frequency, amplitude, and other waveform characteristics. In general a scope can measure both time-based and voltage-based characteristics:

Timing characteristics:

- **Frequency and period** – Frequency is defined as the number of times per second a waveform repeats. And the period is the reciprocal of that (number of seconds each repeating waveform takes). The maximum frequency a scope can measure varies, but it's often in the 100's of MHz (1E6 Hz) range.
- **Duty cycle** – The percentage of a period that a wave is either positive or negative (there are both positive and negative duty cycles). The duty cycle is a ratio that tells you how long a signal is "on" versus how long it's "off" each period.
- **Rise and fall time** – Signals can't instantaneously go from 0V to 5V, they have to smoothly rise. The duration of a wave going from a low point to a high point is called the rise time, and fall time measures the opposite. These characteristics are important when considering how fast a circuit can respond to signals.

Voltage characteristics:

- **Amplitude** – Amplitude is a measure of the magnitude of a signal. There are a variety of amplitude measurements including peak-to-peak amplitude, which measures the absolute difference between a high and low voltage point of a signal. Peak amplitude, on the other hand, only measures how high or low a signal is past 0V.
- **Maximum and minimum voltages** – The scope can tell you exactly how high and low the voltage of your signal gets.
- **Mean and average voltages** – Oscilloscopes can calculate the average or mean of your signal, and it can also tell you the average of your signal's minimum and maximum voltage.

When to Use an O-Scope

The o-scope is useful in a variety of troubleshooting and research situations, including:

- Determining the **frequency and amplitude** of a signal, which can be critical in debugging a circuit's input, output, or internal systems. From this, you can tell if a component in your circuit has malfunctioned.
- Identifying how much **noise** is in your circuit.
- Identifying the **shape** of a wave – sine, square, triangle, sawtooth, complex, etc.
- Quantifying phase differences between two different signals.

Oscilloscope Lexicon

Learning how to use an oscilloscope means being introduced to an entire lexicon of terms. On this page we'll introduce some of the important o-scope buzzwords you should be familiar with before turning one on.

Key Oscilloscope Specifications

Some scopes are better than others. These characteristics help define how well you might expect a scope to perform:

- **Bandwidth** – Oscilloscopes are most commonly used to measure waveforms which have a defined frequency. No scope is perfect though: they all have limits as to how fast they can see a signal change. The bandwidth of a scope specifies the **range of frequencies** it can reliably measure.
- **Digital vs. Analog** – As with most everything electronic, o-scopes can either be analog or digital. Analog scopes use an electron beam to directly map the input voltage to a display. Digital scopes incorporate microcontrollers, which sample the input signal with an analog-to-digital converter and map that reading to the display. Generally analog scopes are older, have a lower bandwidth, and less features, but they may have a faster response (and look much cooler).
- **Channel Amount** – Many scopes can read more than one signal at a time, displaying them all on the screen simultaneously. Each signal read by a scope is fed into a separate channel. Two to four channel scopes are very common.
- **Sampling Rate** – This characteristic is unique to digital scopes, it defines how many times per second a signal is read. For scopes that have more than one channel, this value may decrease if multiple channels are in use.
- **Rise Time** – The specified rise time of a scope defines the fastest rising pulse it can measure. The rise time of a scope is very closely related to the bandwidth. It can be calculated as $\text{Rise Time} = 0.35 / \text{Bandwidth}$.
- **Maximum Input Voltage** – Every piece of electronics has its limits when it comes to high voltage. Scopes should all be rated with a maximum input voltage. If your signal exceeds that voltage, there's a good chance the scope will be damaged.
- **Resolution** – The resolution of a scope represents how precisely it can measure the input voltage. This value can change as the vertical scale is adjusted.
- **Vertical Sensitivity** – This value represents the minimum and maximum values of your vertical, voltage scale. This value is listed in volts per div.
- **Time Base** – Time base usually indicates the range of sensitivities on the horizontal, time axis. This value is listed in seconds per div.

- **Input Impedance** – When signal frequencies get very high, even a small impedance (resistance, capacitance, or inductance) added to a circuit can affect the signal. Every oscilloscope will add a certain impedance to a circuit it's reading, called the input impedance. Input impedances are generally represented as a large resistive impedance (>1 MΩ) in parallel (||) with small capacitance (in the pF range). The impact of input impedance is more apparent when measuring very high frequency signals, and the probe you use may have to help compensate for it.

Using the GA1102CAL as an example, here are specifications you might expect from a mid-range scope:

Table 4: Some Specifications

Characteristic	Value
Bandwidth	100 MHz
Sampling Rate	1 GSa/s (1E9 samples per second)
Rise Time	<3.5ns
Channel Count	2
Maximum Input Voltage	400V
Resolution	8-bit
Vertical sensitivity	2mV/div - 5V/div
Time base	2ns/div - 50s/div
Input Impedance	1 MΩ ±3% || 16pF ±3pF

Understanding these characteristics, you should be able to pick out an oscilloscope that'll best fit your needs. But you still have to know how to use it…onto the next page!

Anatomy of An O-Scope

While no scopes are created exactly equal, they should all share a few similarities that make them function similarly. On this page we'll discuss a few of the more common systems of an oscilloscope: the display, horizontal, vertical, trigger, and inputs.

Figure 103: Scope

The Display

An oscilloscope isn't any good unless it can display the information you're trying to test, which makes the display one of the more important sections on the scope.

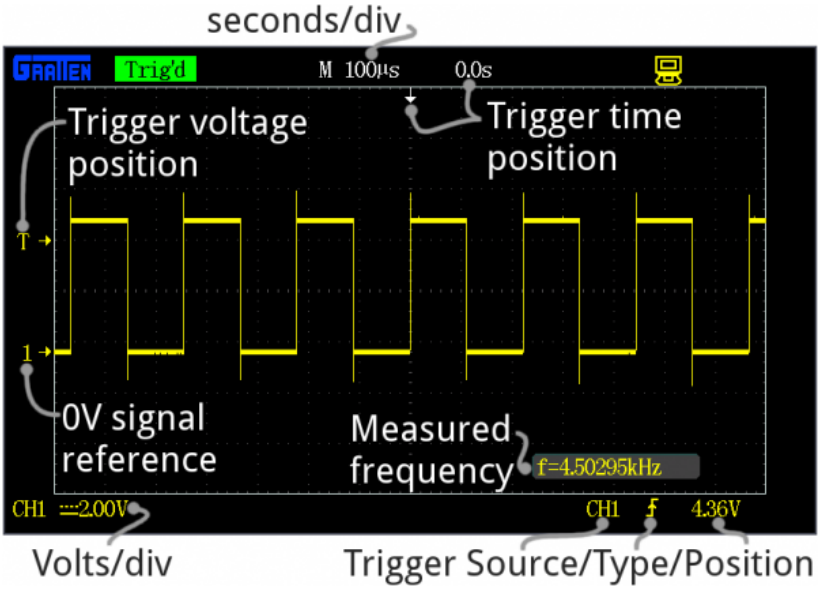


Figure 104:The Display

Every oscilloscope display should be criss-crossed with horizontal and vertical lines called **divisions**. The scale of those divisions are modified with the horizontal and vertical systems. The vertical system is measured in "volts per division" and the horizontal is "seconds per division". Generally, scopes will feature around 8-10 vertical (voltage) divisions, and 10-14 horizontal (seconds) divisions.

Older scopes (especially those of the analog variety) usually feature a simple, monochrome display, though the intensity of the wave may vary. More modern scopes feature multicolor LCD screens, which are a great help in showing more than one waveform at a time.

Many scope displays are situated next to a set of about five buttons – either to the side or below the display. These buttons can be used to navigate menus and control settings of the scope.

Vertical System

The **vertical** section of the scope controls the **voltage scale** on the display. There are traditionally two knobs in this section, which allow you to individually control the vertical position and volts/div.

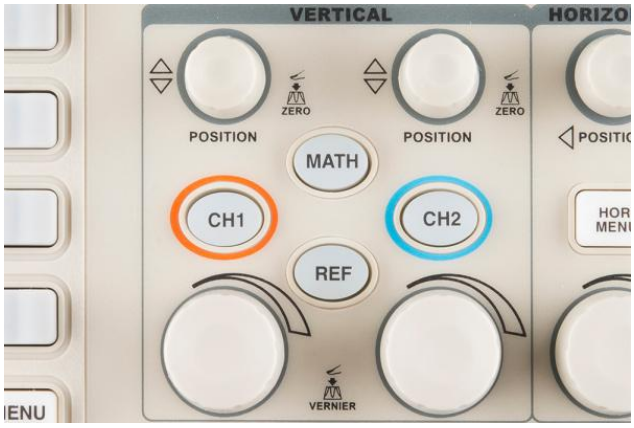


Figure 105: Vertical System

The more critical **volts per division** knob allows you to set the vertical scale on the screen. Rotating the knob clockwise will decrease the scale, and counter-clockwise will increase. A smaller scale – fewer volts per division on the screen – means you're more "zoomed in" to the waveform.

The display on the GA1102, for example, has 8 vertical divisions, and the volts/div knob can select a scale between 2mV/div and 5V/div. So, zoomed all the way in to 2mV/div, the display can show waveform that is 16mV from top to bottom. Fully "zoomed out", the scope can show a waveform ranging over 40V. (The probe, as we'll discuss below, can further increase this range.)

The **position** knob controls the vertical offset of the waveform on the screen. Rotate the knob clockwise, and the wave will move down, counter-clockwise will move it up the display. You can use the position knob to offset part of a waveform off the screen.

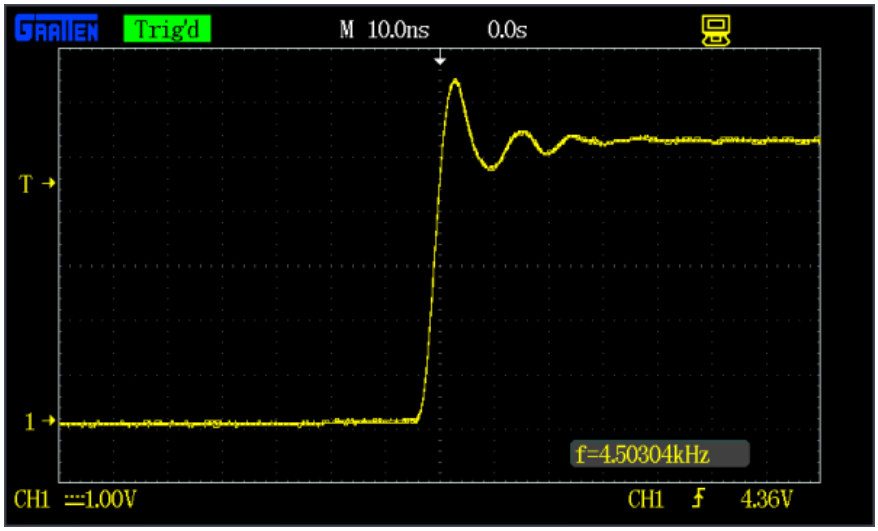


Figure 106: Output

Using both the position and volts/div knobs in conjunction, you can zoom in on just a tiny part of the waveform that you care about the most. If you had a 5V square wave, but only cared about how much it was ringing on the edges, you could zoom in on the rising edge using both knobs.

Horizontal System

The horizontal section of the scope controls the **time scale** on the screen. Like the vertical system, the horizontal control gives you two knobs: position and seconds/div.

Figure 107: Horizontal System

The **seconds per division (s/div)** knob rotates to increase or decrease the horizontal scale. If you rotate the s/div knob clockwise, the number of seconds each division represents will decrease – you'll be "zooming in" on the time scale. Rotate counter-clockwise to increase the time scale, and show a longer amount of time on the screen.

Using the GA1102 as an example again, the display has 14 horizontal divisions, and can show anywhere between 2nS and 50s per division. So zoomed all the way in on the

horizontal scale, the scope can show 28nS of a waveform, and zoomed way out it can show a signal as it changes over 700 seconds.

The **position** knob can move your waveform to the right or left of the display, adjusting the horizontal **offset**.

Using the horizontal system, you can adjust **how many periods** of a waveform you want to see. You can zoom out, and show multiple peaks and troughs of a signal:

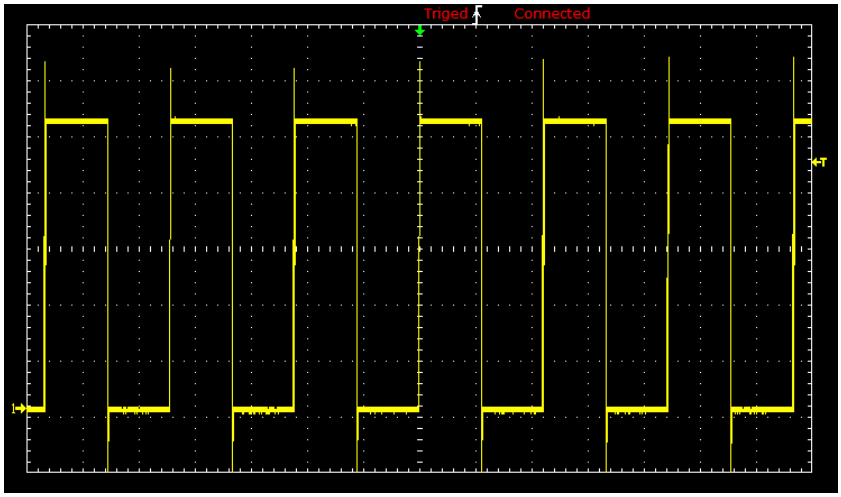


Figure 108: Ouput

Or you can zoom way in, and use the position knob to show just a tiny part of a wave:

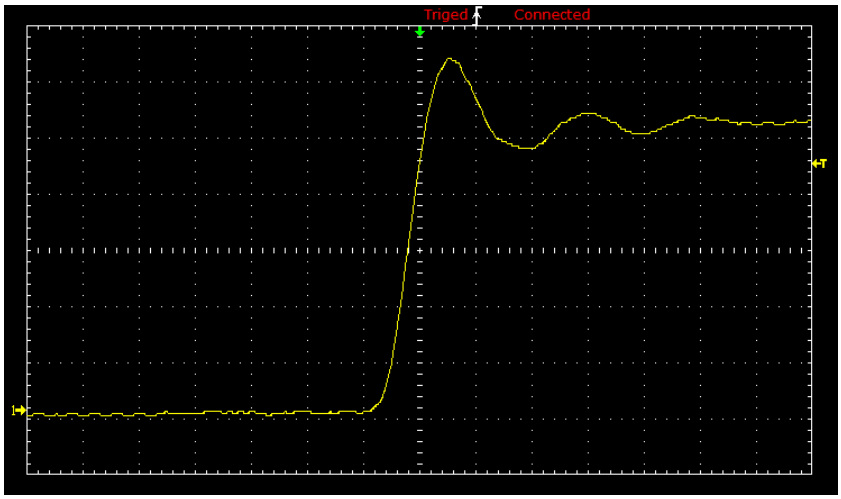


Figure 109:Output

Trigger System

The trigger section is devoted to **stabilizing** and focusing the oscilloscope. The trigger tells the scope what parts of the signal to "trigger" on and start measuring. If your waveform is **periodic**, the trigger can be manipulated to keep the display **static** and unflinching. A poorly triggered wave will produce seizure-inducing sweeping waves like this:

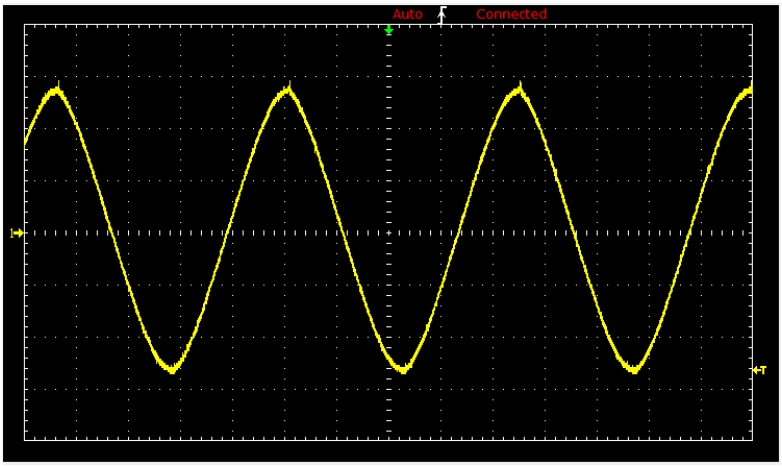


Figure 110: Trigger System

The trigger section of a scope is usually comprised of a level knob and a set of buttons to select the source and type of the trigger. The **level knob** can be twisted to set a trigger to a specific voltage point.

Figure 111: Trigger Nob

A series of buttons and screen menus make up the rest of the trigger system. Their main purpose is to select the trigger source and mode. There are a variety of **trigger types**, which manipulate how the trigger is activated:

- An **edge** trigger is the most basic form of the trigger. It will key the oscilloscope to start measuring when the signal voltage passes a certain level. An edge trigger can be set to catch on a rising or falling edge (or both).
- A **pulse** trigger tells the scope to key in on a specified "pulse" of voltage. You can specify the duration and direction of the pulse. For example, it can be a tiny blip of 0V -> 5V -> 0V, or it can be a seconds-long dip from 5V to 0V, back to 5V.
- A **slope** trigger can be set to trigger the scope on a positive or negative slope over a specified amount of time.
- More complicated triggers exist to focus on standardized waveforms that carry video data, like **NTSC** or **PAL**. These waves use a unique synchronizing pattern at the beginning of every frame.

You can also usually select a **triggering mode**, which, in effect, tells the scope how strongly you feel about your trigger. In automatic trigger mode, the scope can attempt to draw your waveform even if it doesn't trigger. **Normal mode** will only draw your wave if it sees the specified trigger. And **single mode** looks for your specified trigger, when it sees it it will draw your wave then stop.

The Probes

An oscilloscope is only good if you can actually connect it to a signal, and for that you need probes. Probes are single-input devices that route a signal from your circuit to the scope. They have a sharp **tip** which probes into a point on your circuit. The tip can also be equipped with hooks, tweezers or clips to make latching onto a circuit easier. Every probe also includes a **ground clip**, which should be secured safely to a common ground point on the circuit under test.

Figure 112: probes

While probes may seem like simple devices that just latch onto your circuit and carry a signal to the scope, there's actually a lot that goes into probe design and selection.

Optimally, what a probe needs to be is invisible – it shouldn't have any effect on your signal under test. Unfortunately, long wires all have intrinsic inductance, capacitance, and resistance, so, no matter what, they'll affect scope readings (especially at high frequencies).

There are a variety of probe types out there, the most common of which is the **passive probe**, included with most scopes. Most of the "stock" passive probes are **attenuated**. Attenuating probes have a large resistance intentionally built-in and shunted by a small capacitor, which helps to minimize the effect that a long cable might have on loading your circuit. In series with the **input impedance** of a scope, this attenuated probe will create a voltage divider between your signal and the scope input.

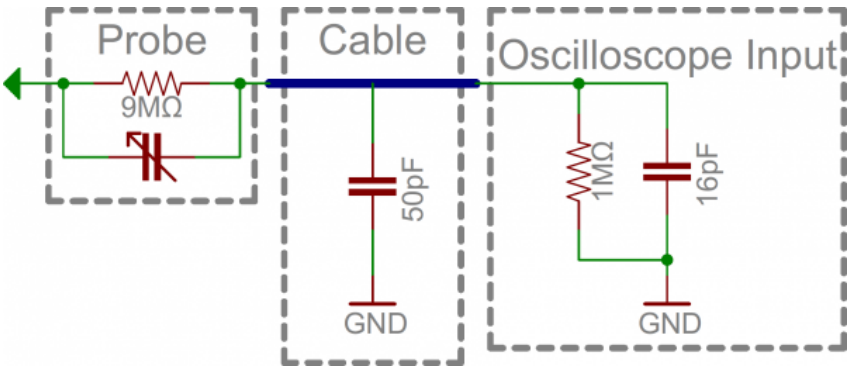


Figure 113: Schematic Diagram

Most probes have a 9MΩ resistor for attenuating, which, when combined with a standard 1MΩ input impedance on a scope, creates a 1/10 voltage divider. These probes are commonly called **10X attenuated probes**. Many probes include a switch to select between 10X and 1X (no attenuation).

Figure 114: Attenuated probe

Attenuated probes are great for improving accuracy at high frequencies, but they will also **reduce the amplitude** of your signal. If you're trying to measure a very low-voltage signal, you may have to go with a 1X probe. You may also need to select a setting on your scope to tell it you're using an attenuated probe, although many scopes can automatically detect this.

Beyond the passive attenuated probe, there are a variety of other probes out there. **Active probes** are powered probes (they require a separate power source), which can amplify your signal or even pre-process it before it get to your scope. While most probes are designed to measure voltage, there are probes designed to measure AC or DC current. **Current probes** are unique because they often clamp around a wire, never actually making contact with the circuit.

- **Using an Oscilloscope**

The infinite variety of signals out there means you'll never operate an oscilloscope the same way twice. But there are some steps you can count on performing just about every time you test a circuit. On this page we'll show an example signal, and the steps required to measure it.

Probe Selection and Setup

First off, you'll need to *select* a probe. For most signals, the simple **passive probe** included with your scope will work perfectly fine.

Next, before connecting it to your scope, **set the attenuation** on your probe. 10X – the most common attenuation factor – is usually the most well-rounded choice. If you're trying to measure a very low-voltage signal though, you may need to use 1X.

Connect the Probe and Turn the Scope On

Connect your probe to the first channel on your scope, and turn it on. Have some patience here, some scopes take as long to boot up as an old PC.

When the scope boots up you should see the divisions, scale, and a noisy, flat line of a waveform.

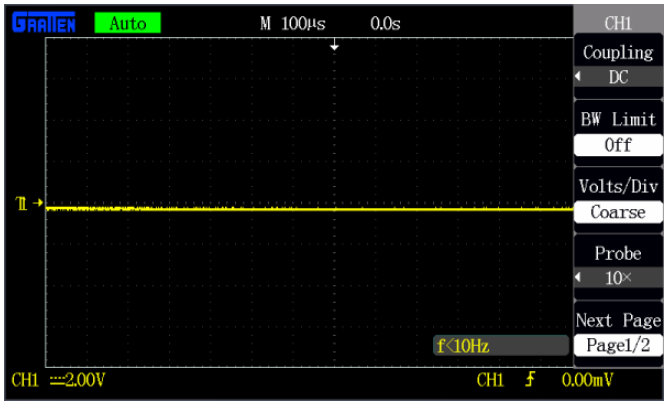


Figure 115: Screen output

The screen should also show previously set values for time and volts per div. Ignoring those scales for now, make these adjustments to put your scope into a **standard setup**:

- Turn **channel 1 on** and channel 2 off.
- Set channel 1 to **DC coupling**.
- Set the **trigger source** to channel 1 – no external source or alternate channel triggering.
- Set the **trigger type** to rising edge, and the **trigger mode** to auto (as opposed to single).
- Make sure the **scope probe attenuation** on your scope matches the setting on your probe (e.g. 1X, 10X).

Testing the Probe

Let's connect that channel up to a meaningful signal. Most scopes will have a **built-in frequency generator** that emits a reliable, set-frequency wave – on the GA1102CAL there is a 1kHz square wave output at the bottom-right of the front panel. The frequency generator output has two separate conductors – one for the signal and one for ground. Connect your probe's **ground clip** to the ground, and the **probe tip** to the signal output.

Figure 116: Testing the Probe

As soon as you connect both parts of the probe, you should see a signal begin to dance around your screen. Try fiddling with the **horizontal and vertical system knobs** to

maneuver the waveform around the screen. Rotating the scale knobs clockwise will "zoom into" your waveform, and counter-clockwise zooms out. You can also use the position knob to further locate your waveform.

If your wave is still unstable, try rotating the **trigger position** knob. **Make sure the trigger isn't higher than the tallest peak of your waveform**. By default, the trigger type should be set to edge, which is usually a good choice for square waves like this.

Try fiddling with those knobs enough to display a single period of your wave on the screen.

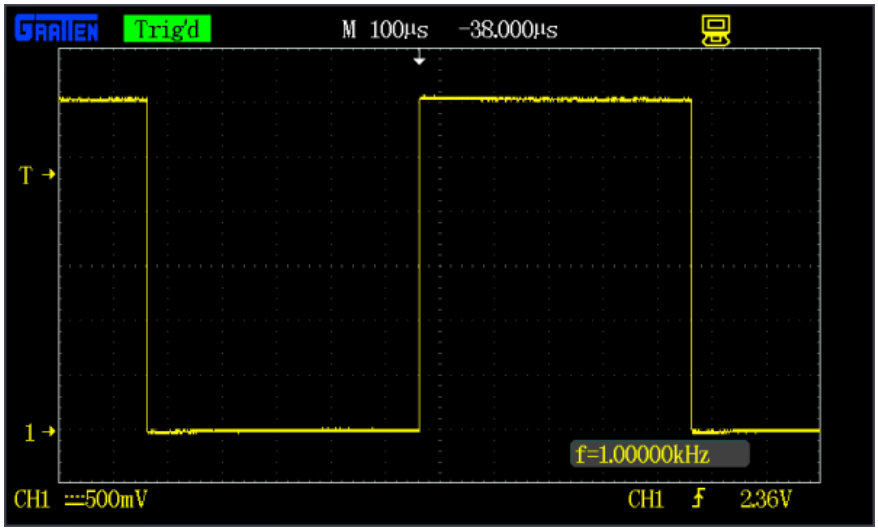


Figure 117: OutPut Signal

Or try zooming way out on the time scale to show dozens of squares.

Compensating an Attenuated Probe

If your probe is set to 10X, and you don't have a perfectly square waveform as shown above, you may need to **compensate your probe**. Most probes have a recessed screw head, which you can rotate to adjust the shunt capacitance of the probe.

Figure 118: Compensating an Attenuated Probe

Try using a small screwdriver to rotate this trimmer, and look at what happens to the waveform.

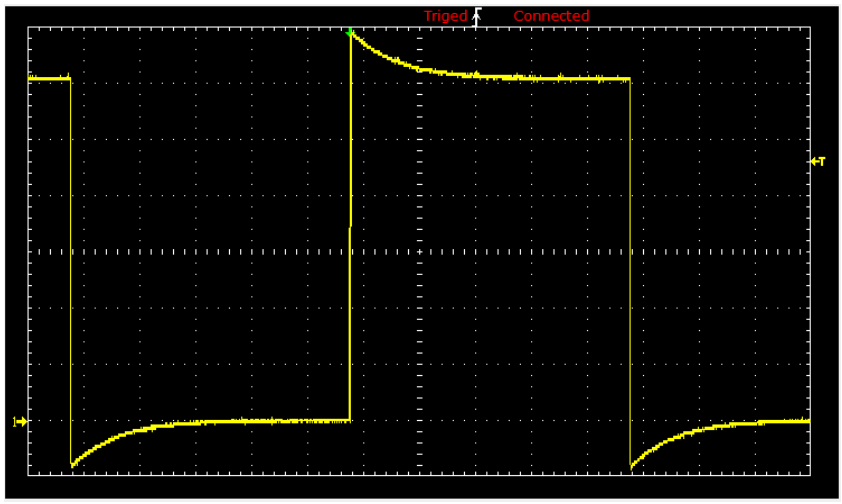


Figure 119: Output

Adjust the trimming cap on the probe handle until you have a **straight-edged** square wave. Compensation is only necessary if your probe is attenuated (e.g. 10X), in which case it's critical (especially if you don't know who used your scope last!).

Probing, Triggering, and Scaling Tips

Once you've compensated your probe, it's time to measure a real signal! Go find a signal source (frequency generator?, Terror-Min?) and come back.

The first key to probing a signal is finding a solid, reliable **grounding point**. Clasp your ground clip to a known ground, sometimes you may have to use a small wire to intermediate between the ground clip and your circuit's ground point. Then connect your probe tip to the signal under test. Probe tips exist in a variety of form factors – the spring-loaded clip, fine point, hooks, etc. – try to find one that doesn't require you to hold it in place all the time.

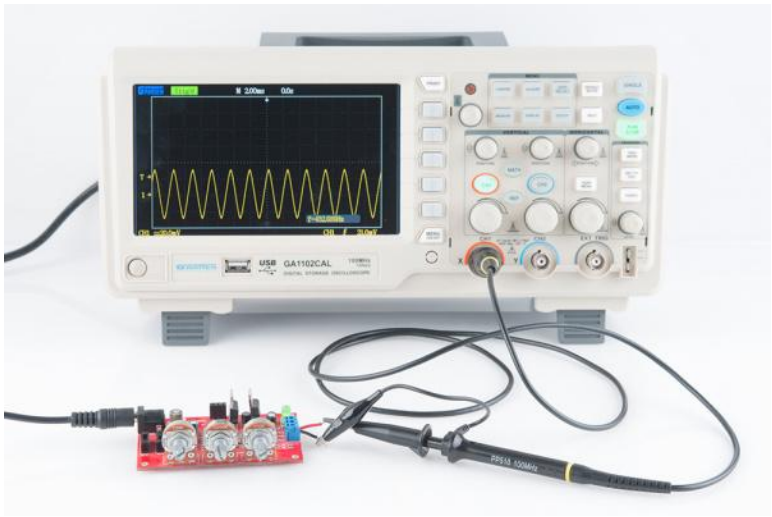


Figure 120: Connecting to a circuit

Once your signal is on the screen, you may want to begin by adjusting the horizontal and vertical scales into at least the "ballpark" of your signal. If you're probing a 5V 1kHz square wave, you'll probably want the volts/div somewhere around 0.5-1V, and set the seconds/div to around 100µs (14 divisions would show about one and a half periods).

If part of your wave is rising or falling of the screen, you can adjust the **vertical position** to move it up or down. If your signal is purely DC, you may want to adjust the 0V level near the bottom of your display.

Once you have the scales ballparked, your waveform may need some triggering. **Edge triggering** – where the scope tries to begin its scan when it sees voltage rise (or fall) past a set point – is the easiest type to use. Using an edge trigger, try to set the trigger level to a point on your waveform that only sees a **rising edge once per period**.

Now just **scale, position, trigger and repeat** until you're looking at exactly what you need.

Measure Twice, Cut Once

With a signal scoped, triggered, and scaled, it comes time to measure transients, periods, and other waveform properties. Some scopes have more measurement tools than others, but they'll all at least have divisions, from which you should be able to at least estimate the amplitude and frequency.

Many scopes support a variety of automatic measurement tools, they may even constantly display the most relevant information, like frequency. To get the most out of your scope, you'll want to explore all of the **measure functions** it supports. Most scopes will calculate frequency, amplitude, duty cycle, mean voltage, and a variety of other wave characteristics for you automatically.

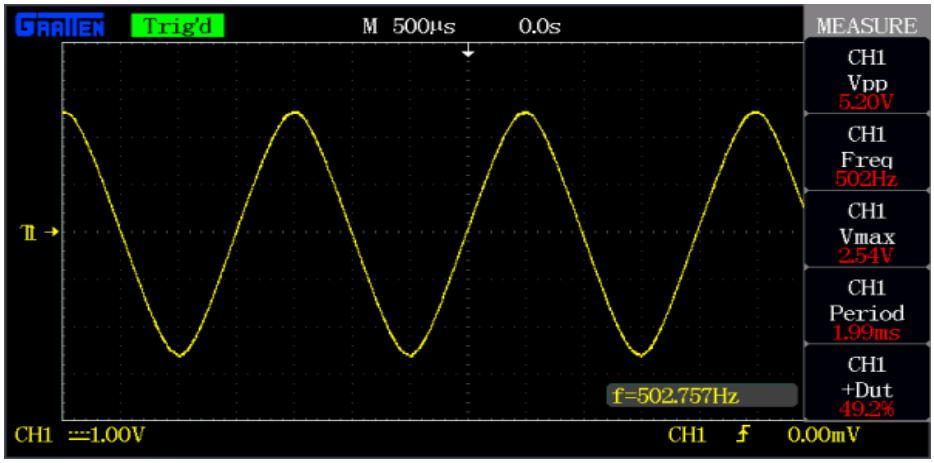


Figure 121: Using the scope's measure tools to find VPP, VMax, frequency, period, and duty cycle.

A third measuring tool many scopes provide is **cursors**. Cursors are on-screen, movable markers which can be placed on either the time or voltage axis. Cursors usually come in pairs, so you can measure the difference between one and the other.

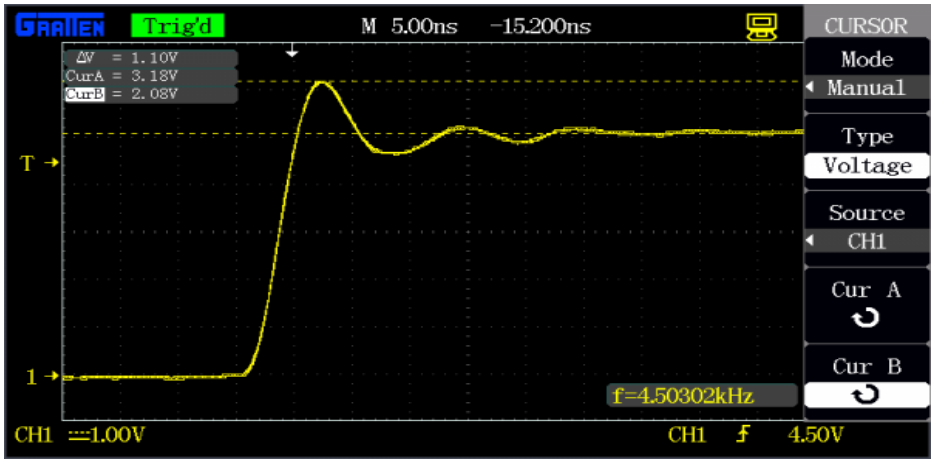


Figure 122: Measuring the ringing of a square wave with cursors.

Once you've measured the quantity you were looking for, you can begin to make adjustments to your circuit and measure some more! Some scopes also support **saving**, **printing**,

or **storing** a waveform, so you can recall it and remember those good ol' times when you scoped that signal.

To find out more about what your scope can do, consult its user's manual!

Hands on: A 555 Timer circuit will be made in astable mode and all experiments(amplitude, frequency,period,…) about an oscilloscope will done on this circuit.

- **Troubleshooting concerning opens, shorts and grounds in circuits**

Troubleshooting is a process whereby you diagnose a malfunctioning system and identify the specific defect. Once the defect is identified, a judgment must be made regarding the feasibility of repair and there are a number of considerations that have to be made. One is the time, how it will take to repair this. Sometimes there are critical systems that there is no time to repair them and simply replace the entire circuit board and just to continue to use the system. There is the time involved and also there a cost because when you go to repair a system, there is going to be the wage a person repairing it, the cost of the components to make it work, so there are considerations to be made, is it going to cost more to fix this than simply to replace it.

This chapter examines troubleshooting strategies and techniques. Troubleshooting is a lot of fun for those people that make a career, out of doing this and they can be a rewarding career. People in our electronics program in North Seattle, many students will go out into the industry they worked in, biomedical repairs, some work in telecom, some work in broadband, industrial power and for the most part, they are maintaining and repairing electronic systems that we come to depend upon in our daily lives.

Troubleshooting Series Circuits

This particular section 5.1, we're going to do it in two parts. Peer series circuits are rare and practical electronics system. You don't commonly see a disappear series circuits. However, what you will see are parts of circuits that are series within a given system but usually, an entire system isn't just a series circuit. You did not know how to troubleshoot them because you will find a series of circuits within systems.

Troubleshooting strategies applicable to series circuits play a major role in troubleshooting advance circuit configurations.

Basic Concepts

Logical troubleshooting procedures will aid in analyzing and troubleshooting all circuits, simple or complex.

Three goals of a troubleshooter should be:

1. Make a measuring only if you know what a good reading should be. The idea here is that you can make all kinds of measurements but if you don't know what the correct measurement should be, you're probably just wasting your time.
2. Make as few measurements as possible. The idea here is not to waste your time.
3. Select the best tool for the task at hand. Typically, electronic troubleshooting tools would be things like Oscilloscopes and volt readers. Depending on the task, if you're

out in the broadband industry, you'll probably want signal level meters. There is wide variety of tools out there that might be appropriate for a given task.

Know what is normal in a circuit. There are a number of sources to get this information. The first we will mention is a schematic. A schematic is just a printed readout of the circuit configuration where you could look at each component on paper and view the overall circuit. This is one of the primary troubleshooting tools many times the schematics will have voltages written in so that you'll know what you should read at specific points.

Another good way to know what is normal is to have a good known circuit for comparison. Let's say if I have a box here and we'll call this a bad one and here is the good one and let's pretend that we have a troubleshooting points or voltages that we are measuring and we suspect that something is wrong. Here, we could measure the known good box and make a determination whether there is something wrong or not. So, a known good circuit is a good way to know what is normal in a given circuit.

Finally, there is simply an intuitive understanding. Intuitive understanding usually comes with time a given technician will be able to work in a box, look at it and intuitively he'll have a sense of what is wrong oftentimes based upon his experience.

Intuitive Troubleshooting

Developing a solid intuitive understanding of a circuit behavior is important to becoming a skilled troubleshooter. There are some general rules that form the basis for all troubleshooting efforts. These are the effective power supplies, open circuits, short circuits and components which values have changed or important to understand.

Effects of an Open Circuit

First, we are going to look at the effects of an open circuit. Open circuits are characterized by having infinite resistance. In this case, we're considering resistance and open circuits will have infinite resistance. The circuit here, were looking at, have a component showing a resistor that is open. Commonly what causes this is too much current has flowed through the circuit and this particular component may have disintegrated, burned up and you will see a charred component. If you were to measure test point 4 to test point 5 with an Ohmmeter, you will commonly expect to see infinite resistance. This component seizes to exist because it's simply burned up.

Typically, you will expect to see infinite resistance but that wouldn't always be the case because sometimes when you measure the surface, you will think that you are measuring across here. What actually is happening is that the Ohmmeter doesn't know what you are just measuring across. It may measure around like this, and you may get a resistance. It may be rather measure a high one but it may not necessarily, be infinite because it may measure a different path through the circuit.

An open circuit will appear to have the applied voltage across it. The things that often surprise students is they go in … now this is what you're measuring, voltage. When they measured resistance, the circuit was turned off. When they measured voltage, we have power applied when they measure test point 4 to test point 5. Rather than seeing zero volts, we oftentimes times see, in a case like this, you might read 25 volts. People will wonder why. Consider the fact that proven the circuit is operating correctly; we have a current flowing through the circuit. We have that loop which is a requirement for all circuits. When this component opens, we no longer have that loop. So now, there is no current flowing through this component and you might notice the little zero here, on all these components indicating zero volts. Well, why is there zero volts?

Let's remember Ohms law, voltage=current x resistance. If the resistance has remained unchanged but now the current is zero, so zero times the resistance is going to equate to zero volts. In this circuit, there are no voltage drops across these components. When I insert my meter into the circuit, I will measure the supply voltage because there are no drops and any point in this circuit.

If the open circuit appears between a monitor point and ground, the meter will read a source voltage, otherwise it will be zero. When you're measuring something like this, you'll have your meter and a probe going to ground the other probe. In that situation is what we just previously described that in case, I will read the supply voltage.

There's a note down here. If the circuit is open, how much current is flowing? None. How much voltage will be dropped across each resistor? Zero. What voltage will it measure across the open? That's the power supply voltage.

Effects of Short Circuit

The previous screen you looked at the effects of an open. We're going to look at the same circuit and we're going to look at what happens if we have a short.

A short is characterized by having a zero resistance or very low resistance. Again, if the circuit is turned off and we are measuring resistance, then we measure from this point to this point, we're going to measure … this is the case of a short, either a very low resistance or zero resistance.

When a short circuit occurs within a series component; others components in the series have higher currents. In this entire circuit, we have a … maybe I'll put an R or T here, for total resistance. Remember, that current = voltage/resistance. If this component shorts, the total resistance is going to decrease and so the current in the circuit will increase. This blue arrows, here, that have a little V tied to them. This is indicating that the voltage across this voltage will suddenly be a little bit higher. The reason that the voltage is a little bit higher is because one of the resistors has been removed, shortage across, so the resistance has gone down, current has gone up and so all the voltages have increased a little bit. When a short circuit occurs within series component, other components in the series will have higher currents which we've shown. The voltage across a short circuit is zero. Now, if we measure voltage and place a volt meter in the circuit, measure it form here to here, remember it is a short we will measure zero volts.

The best tool for diagnosis is a VOM. With a VOM or volt ohmmeter, you can measure the voltage and the Ohms. The resistance across R4 is zero ohms. The voltage across R4 zero volts. The current through the short … current increases because R4 is not on the circuit. We've mentioned all of that over here.

This is the same schematic and this is review. With force shortage, there will be an increasing current. The blue arrows indicated a larger voltage across every component because of increased current. Therefore, the voltage read across every component will increase.

The voltage read across every component with respect to ground will measure closer to the applied voltage. I mention this because, in the summary statements in this chapter, they make mention of the fact that the voltage across every component will measure closer to the applied voltage. They wouldn't measure the applied voltage but since every voltage increases, they will be closer to that applied voltage.

In this section, we introduced troubleshooting, we looked at the effects of short circuit and we looked at the effects of an open circuit. We talked briefly about intuitive

troubleshooting, the goals of troubleshooting and what is normal and how to determine what is normal in a given circuit.

Hands on: Make a circuit and demonstrate the effect of open and short circuit.

Chap 4: Embedded system with atmega328/arduino

Introduction of µp and µc

Computer: A computer is a multipurpose programmable machine that reads binary instructions from its memory , accepts binary data as input ,processes the data according to those instructions and provides results as output. It is a programmable device made up of both *hardware* and *software*. The various components of the computer are called *hardware*. A set of instructions written for the computer to solve a specific task is called program and collection of programs is called *software* .
The computer hardware consists of four main components. The central processing unit which acts as computer's brain. Input unit through which program and data can be entered to computer, output unit on which the results of the computations can be displayed. Memory in which data and program are stored.

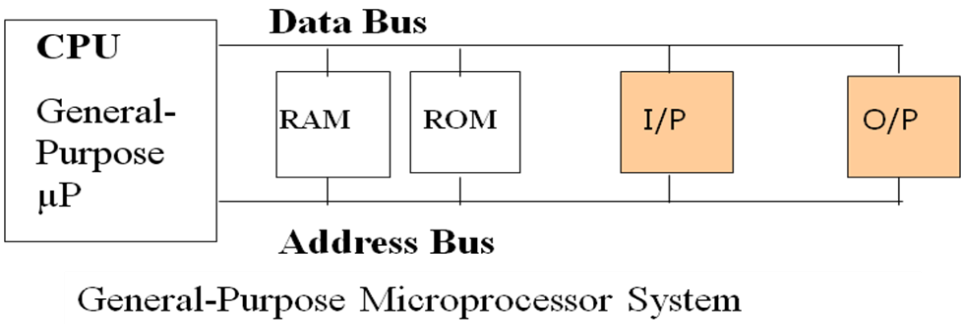


Figure 123: Microprocessor System

A computer that is designed using a microprocessor as its CPU , is known as a microcomputer.
Microprocessor or 'Computer on Chip' first became a commercial reality in 1971 with the introduction of the 4 bit 4004 by Intel. A byproduct of Microprocessor development was Microcontroller. The same fabrication technology and programming concept that make the general purpose microprocessor also yielded the Microcontroller.

Microprocessors

A microprocessor is a general purpose digital computer central processing unit (CPU). Although known as a 'Computer on Chip' the Microprocessor in no sense a complete digital computer. Block diagram of a Microprocessor CPU which contains ALU; Program counter (PC), a stack pointer (SP) ,some working registers , a clock timing circuit and interrupt circuit s is shown in the following figure

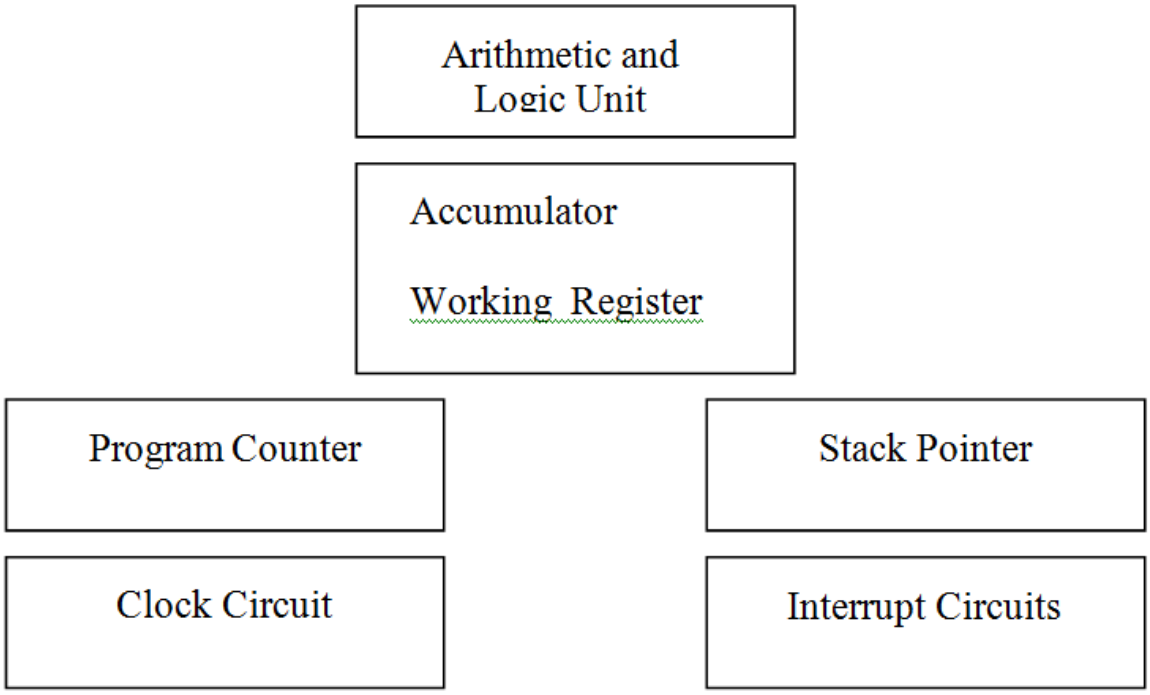


Figure 124: Microprocessor Structure

To make a computer microcomputer one must add memory usually RAM and ROM, memory decoders , an oscillator and a number of Input ,Output devices such as serial and parallel ports. In addition special purpose devices such as interrupt handler and counters may be added to relieve the CPU from time consuming counting or timing cores. When the Microcomputer is equipped with mass storage devices , I/O peripherals such as a key board and a display CRT it yields a small computer that can be applied to a range of general purpose applications.

The hardware design of a microprocessor is arranged such that a very small or very large system can be configured around the CPU as the application demands as shown in Fig1. The prime use of the Microprocessor is to read data , perform extensive calculations on that data, and store those calculations in a mass storage device or display the results for human use. The programs used by microprocessor are stored in the mass storage device and loaded into RAM as user directs. A few microprocessor program are stored in ROM . The ROM based programs are primarily small fixed programs that operate peripherals and other fixed devices that are connected to the system.

Microcontroller: A Microcontroller is a programmable digital processor with necessary peripherals. Both microcontrollers and microprocessors are complex sequential digital circuits meant to carry out job according to the program / instructions. Sometimes analog input/output interface makes a part of microcontroller circuit as mixed mode(both analog and digital) in nature.

A microcontroller can be compared to a Swiss knife with multiple functions incorporated in the same Integrated Circuits. Block diagram of a typical Microcontroller which is a true computer on a chip is shown below. The design incorporates all the features found in microprocessor CPU : ALU,PC, SP and registers. It also has other features needed to make a complete computer: ROM, RAM, Parallel I/O, serial I/O, Counters and clock circuits. Like the microprocessor , a microcontroller is a general purpose device, but one that is meant to read data, perform limited calculations on that data and control its environment based on those calculations. The prime use of microcontroller is to control the operation of a machine

using a fixed program that is stored in ROM and that does not change over the lifetime of the system.

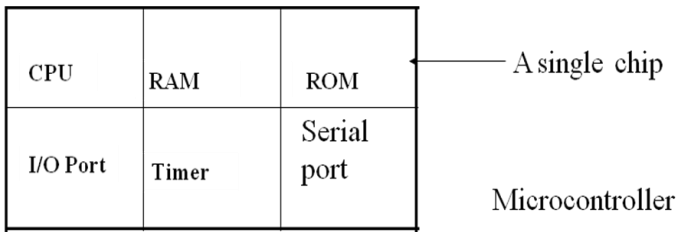


Figure 125: Block diagram of a single chip computer

Complex Instruction Set Computer (CISC):

Memory in those days was expensive. Bigger programs required more storage which included more money . There was a need to reduce the number of instructions per program . This was achieved by having multiple operations within single instruction. Multiple operations lead to many different kinds of instructions .Access to memory in turn makes the instruction length variable and fetch-decode execute time unpredictable – making it more complex. Thus hardware was made to understand the complexity of instruction set. The computer having such instruction set was named as Complex Instruction Set Computer (CISC). Intel 8051 is an example for CISC architecture.

Reduced Instruction Set Computer (RISC):

In applications which require more of input , output related operations having few simple instructions that are of the same length allows memory access only with explicit load and store instructions. Hence each instruction performs less work but instruction execution time among different instructions is consistent. This would lead to instruction execution by hardware including multiple number of registers inside CPU. The computer using such instructions is called Reduced Instruction Set Computer (RISC). PIC microcontroller manufactured by Microchip Company is an example for RISC architecture.

Vonneumann (Princeton) and Harvard Architecture :

Intel's 8051 employs Harvard architecture. A microcontroller has some embedded peripherals and Input/Output (I/O) devices. The data transfer to these devices takes place through I/O registers.

In a microprocessor, input /output (I/O) devices are externally interfaced and are mapped either to memory address (memory mapped I/O) or a separate I/O address space (I/O mapped I/O). There are two possible architectures one is Princeton (Von Neumann) and another is Harvard .I/O Registers space in Princeton architecture have only one memory interface for program memory (ROM) and data memory (RAM). One option is to map the I/O Register as a part of data memory or variable RAM area (memory mapped I/O). Alternatively a separate I/O register space can be assigned (I/O Mapped I/O) . Both the arrangements are shown in Fig.4.

Memory Mapped I/O I/O Mapped I/O

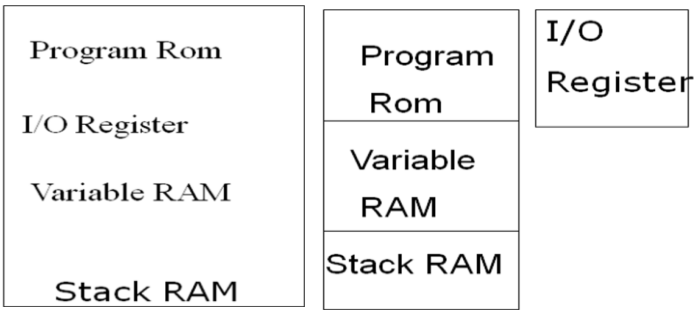


Figure 126: Input/Output Registers in Princeton Architecture

As shown in Fig 126. Program memory and Data memory are together in both the arrangements. The Princeton or Vonneumann architecture one bus is used to carry the address and data with an appropriate multiplexing technique ,which in turn reduces the cost. But Harvard architecture which 8051 employs has separate Data memory and separate Code or Program memory . The Fig. 127 and Fig126 show the need for separate address and data bus for each Program and Data memory in Harvard architecture. Since there are separate bus for access the operation of fetching the code and data can happen simultaneously which increases the speed of operation of execution inside CPU.

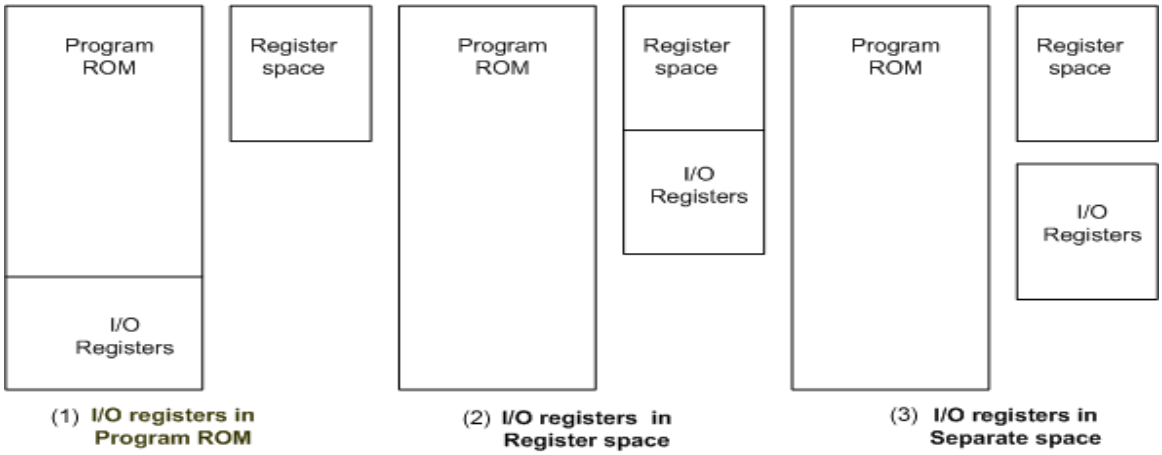
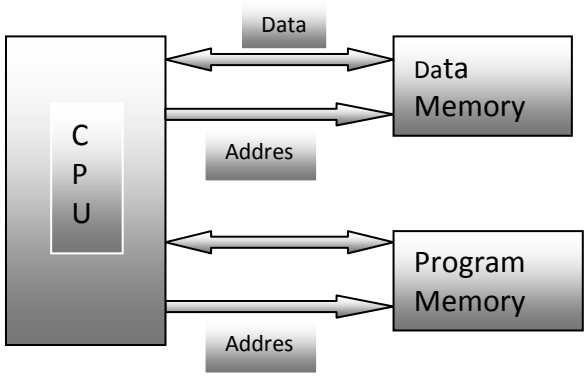


Figure 127: Organization of I/O registers in Harvard Architecture

In Fig. 127 , the first option is difficult to implement as there is no means to write to program ROM area. It is also complicated to have a separate I/O space as shown in (3). Hence the second option where I/O registers are placed in the register space is widely used in Harvard architecture.



Computer Software: A set of instructions written in a specific sequence for computer to solve a specific task is called a program, and software is collection of programs. The program stored in the computer memory in the form of 0s and 1sand it is called as machine level instructions. Since it would be difficult to remember machine codes in the form of binary numbers an intermediate level of language for programming, between higher and machine level was developed and is known as assembly level language . Assembly language programs are written using assembly instructions known as mnemonics.

For example in CLR A, instruction CLR means clear and A means accumulator. The program mnemonics are converted to machine codes in the form of binary by a software called *Assembler*.

The Assembly language programming requires a detailed knowledge of the architecture with which the program is executed. In order to overcome the drawback of assembly language programming Higher level language like C,C++ are introduced where an interpreter or a compiler takes care of translating a higher level source code into machine codes.

Development/Classification of microcontrollers : Microcontrollers have gone through a silent evolution (invisible). The evolution can be rightly termed as silent as the impact or application of a microcontroller is not well known to a common user, although microcontroller technology has undergone significant change since early 1970's. Development of some popular microcontrollers is given as follows.

Table 5: Classification of Microcontrollers

Intel 4004	4 bit (2300 PMOS trans, 108 kHz)	1971
Intel 8048	8 bit	1976
Intel 8031	8 bit (ROM-less)	.
Intel 8051	8 bit (Mask ROM)	1980
Microchip PIC16C64	8 bit	1985
Motorola 68HC11	8 bit (on chip ADC)	.
Intel 80C196	16 bit	1982
Atmel AT89C51	8 bit (Flash memory)	.
Microchip PIC 16F877	8 bit (Flash memory + ADC)	.

We use more number of microcontrollers compared to microprocessors. Microprocessors are primarily used for computational purpose, whereas microcontrollers find wide application in devices needing real time processing and control. Application of microcontrollers are numerous. Starting from domestic applications such as in washing machines, TVs, air conditioners, microcontrollers are used in automobiles, process control industries , cell phones, electrical drives, robotics and in space applications.

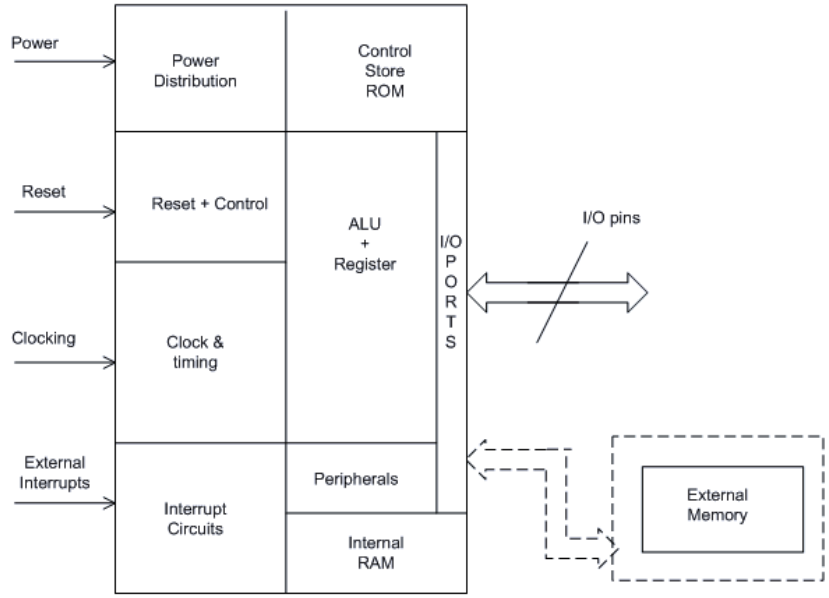


Figure 128: Internal Structure of a typical Microcontroller

The one we are studying is a 8 bit Embedded Microcontroller introduced by Intel, 8051.

Introduction of Arduino

What is Arduino?

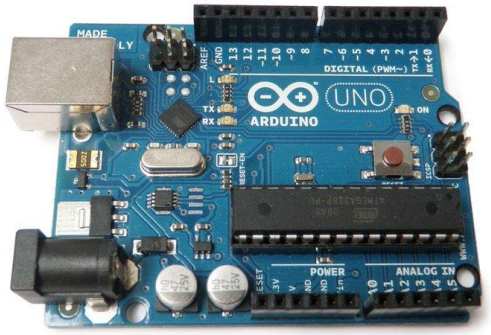


Figure 129: arduino Uno

Where to locally buy this?: http://naradaelectronics.rw/

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of

instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

Why Arduino?

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step by step instructions of a kit, or sharing ideas online with other members of the Arduino community.

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- **Inexpensive** - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50
- **Cross-platform** - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
- **Simple, clear programming environment** - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
- **Open source and extensible software** - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can

make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.

- **Open source and extensible hardware** - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

Different Types of Arduino

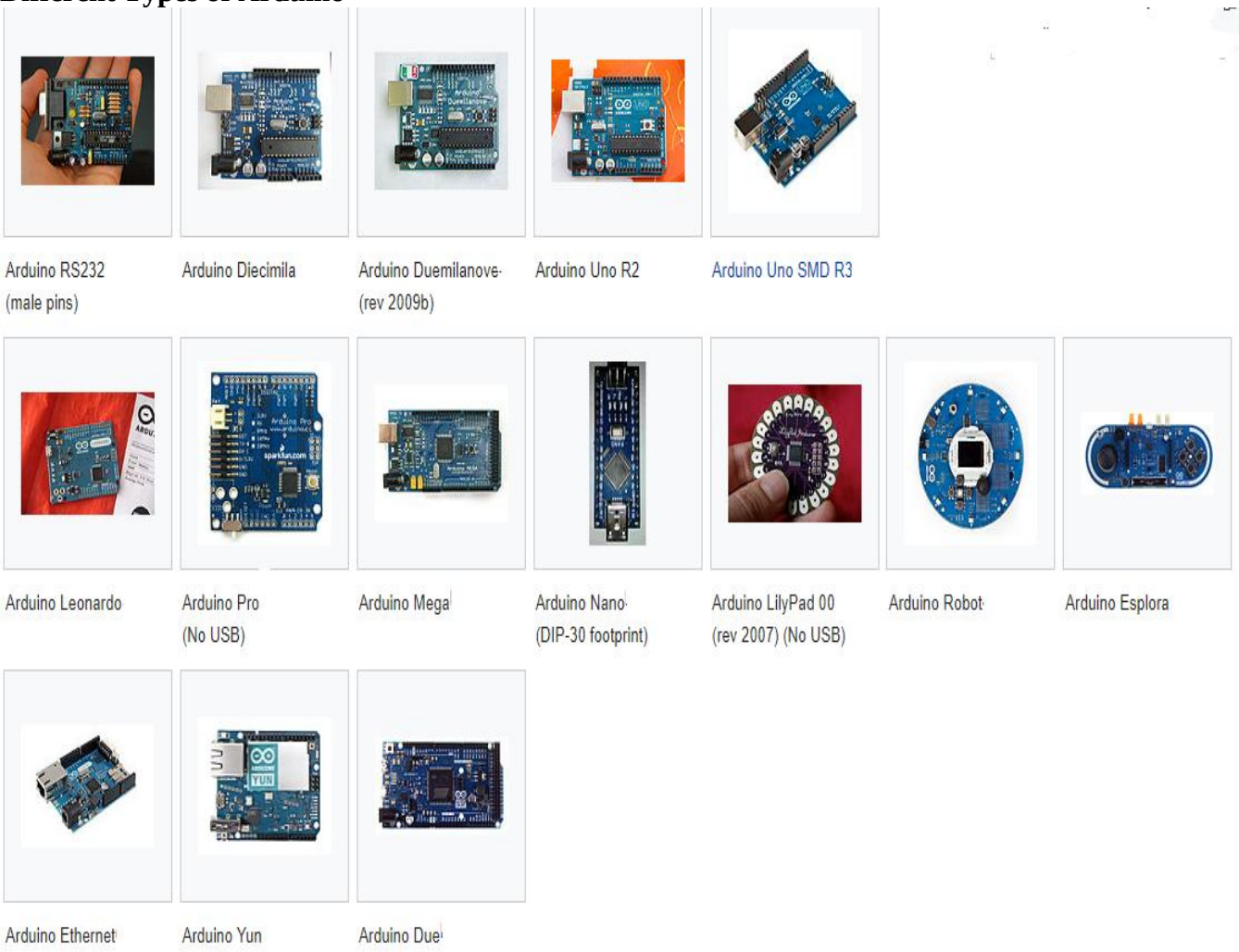


Figure 130: Different Arduino Types

There are a number of different types of Arduinos to choose from. This is a brief overview of some of the more common types of Arduino boards you may encounter.

Arduino Uno

The most common version of Arduino is the Arduino Uno. This board is what most people are talking about when they refer to an Arduino. In the next step, there is a more complete rundown of its features.

Arduino NG, Diecimila, and the Duemilanove (Legacy Versions)

Legacy versions of the Arduino Uno product line consist of the NG, Diecimila, and the Duemilanove. The important thing to note about legacy boards is that they lack particular feature of the Arduino Uno. Some key differences:

- The Diecimila and NG use an ATMEGA168 chips (as opposed to the more powerful ATMEGA328),
- Both the Diecimila and NG have a jumper next to the USB port and require manual selection of either USB or battery power.
- The Arduino NG requires that you hold the rest button on the board for a few seconds prior to uploading a program.

Arduino Mega 2560

The Arduino Mega 2560 is the second most commonly encountered version of the Arduino family. The Arduino Mega is like the Arduino Uno's beefier older brother. It boasts 256 KB of memory (8 times more than the Uno). It also had 54 input and output pins, 16 of which are analog pins, and 14 of which can do PWM. However, all of the added functionality comes at the cost of a slightly larger circuit board. It may make your project more powerful, but it will also make your project larger.

Arduino Mega ADK

This specialized version of the Arduino is basically an Arduino Mega that has been specifically designed for interfacing with Android smartphones. This too is now a legacy version.

Arduino Yun

The Arduino Yun uses a ATMega32U4 chip instead of the ATmega328. However, what really sets it apart is the addition of the Atheros AR9331 microprocessor. This extra chip allows this board to run Linux in addition to the normal Arduino operating system. If all of that were not enough, it also has onboard wifi capability. In other words, you can program the board to do stuff like you would with any other Arduino, but you can also access the Linux side of the board to connect to the internet via wifi. The Arduino-side and Linux-side can then easily communicate back and forth with each other. This makes this board extremely powerful and versatile.

Arduino Nano

If you want to go smaller than the standard Arduino board, the Arduino Nano is for you! Based on a surface mount ATmega328 chip, this version of the Arduino has been shrunk down to a small footprint capable of fitting into tight spaces. It can also be inserted directly into a breadboard, making it easy to prototype with.

Arduino LilyPad

The LilyPad was designed for wearable and e-textile applications. It is intended to be sewn to fabric and connected to other sewable components using conductive thread. This board requires the use of a special FTDI-USB TTL serial programming cable. For more information, the Arduino LilyPad page is a decent starting point.

Hands on: Different types of arduino boards will shown.

Definition and types , advantages ,

Atmega328

The **ATmega328** is a single-chip microcontroller created by Atmel in the megaAVR family (later Microchip Technology acquired Atmel in 2016). It has a modified Harvard architecture 8-bit RISC processor core.

Specifications

The Atmel 8-bit AVR RISC-based microcontroller combines 32 kB ISP flash memory with read-while-write capabilities, 1 kB EEPROM, 2 kB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART, a byte-oriented 2-wire serial interface, SPI serial port, 6-channel 10-bit A/D converter (8-channels in TQFP and QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes. The device operates between 1.8-5.5 volts. The device achieves throughput approaching 1 MIPS per MHz.

Key parameters

Table 6: key parameter

Parameter	Value
CPU type	8-bit AVR
Performance	20 MIPS at 20 MHz[2]
Flash memory	32 kB
SRAM	2 kB
EEPROM	1 kB
Pin count	28 or 32 pin: PDIP-28, MLF-28, TQFP-32, MLF-32[2]
Maximum operating frequency	20 MHz
Number of touch channels	16
Hardware QTouch Acquisition	No
Maximum I/O pins	23
External interrupts	2
USB Interface	No
USB Speed	–

Pinout configuration

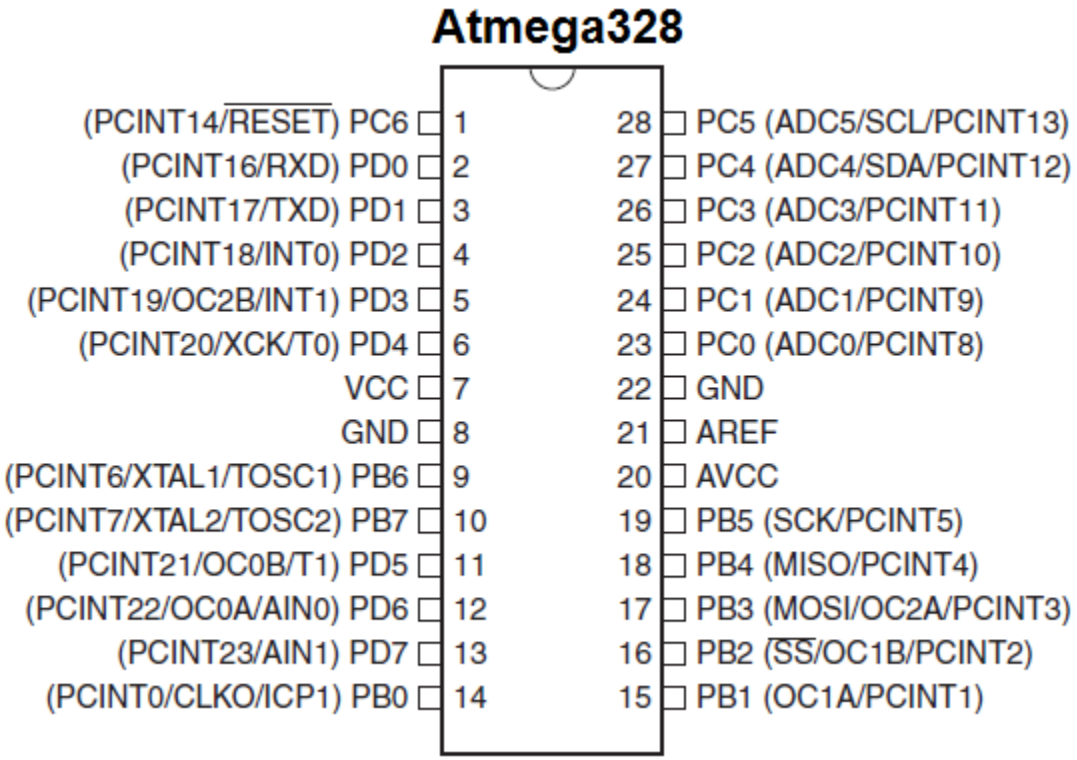


Figure 131: Pinout configuration

Internal Structure

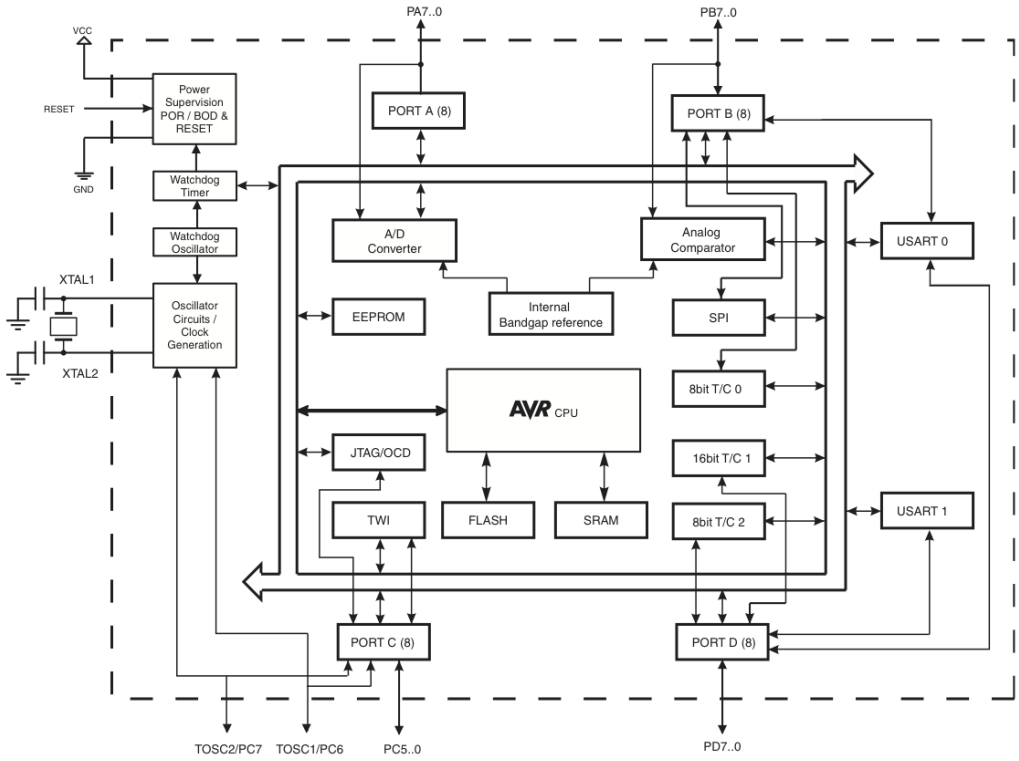


Figure 132: Internal Structure

Arduino IDE

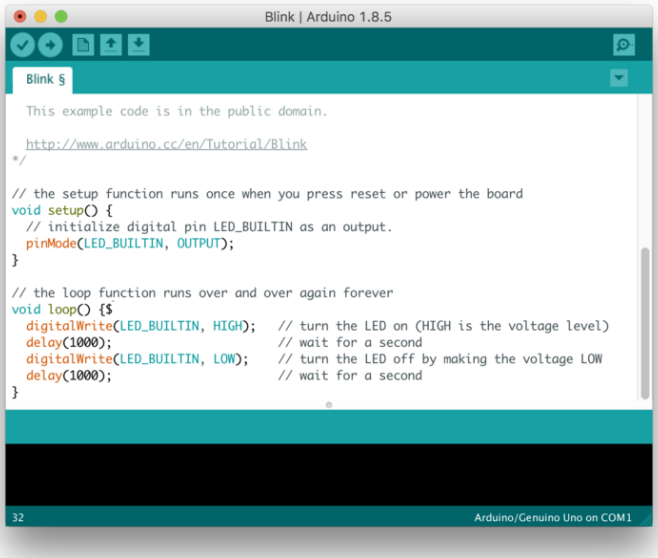


Figure 133: Arduino Ide

Hands on: Different Types of IDE Menu will be shown and Explained .

The **Arduino integrated development environment (IDE)** is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It is used to write and upload programs to Arduino board.

The source code for the IDE is released under the GNU General Public License, version 2.The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiringproject, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub *main()* into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution.The Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

Basic programming

Blinking LEDs

This example shows the simplest thing you can do with an Arduino or Genuino to see physical output: it blinks the on-board LED.

Hardware Required

- Arduino or Genuino Board

Optional

- LED
- 220 ohm resistor

Circuit

This example uses the built-in LED that most Arduino and Genuino boards have. This LED is connected to a digital pin and its number may vary from board type to board type. To make your life easier, we have a constant that is specified in every board descriptor file. This constant is *LED_BUILTIN* and allows you to control the built-in LED easily. Here is the correspondence between the constant and the digital pin.

- D13 - 101
- D13 - Due
- D1 - Gemma
- D13 - Intel Edison
- D13 - Intel Galileo Gen2
- D13 - Leonardo and Micro
- D13 - LilyPad
- D13 - LilyPad USB
- D13 - MEGA2560
- D13 - Mini
- D6 - MKR1000

- D13 - Nano
- D13 - Pro
- D13 - Pro Mini
- D13 - UNO
- D13 - Yún
- D13 - Zero

If you want to lit an external LED with this sketch, you need to build this circuit, where you connect one end of the resistor to the digital pin correspondent to the *LED_BUILTIN* constant. Connect the long leg of the LED (the positive leg, called the anode) to the other end of the resistor. Connect the short leg of the LED (the negative leg, called the cathode) to the GND. In the diagram below we show an UNO board that has D13 as the LED_BUILTIN value.

The value of the resistor in series with the LED may be of a different value than 220 ohm; the LED will lit up also with values up to 1K ohm.

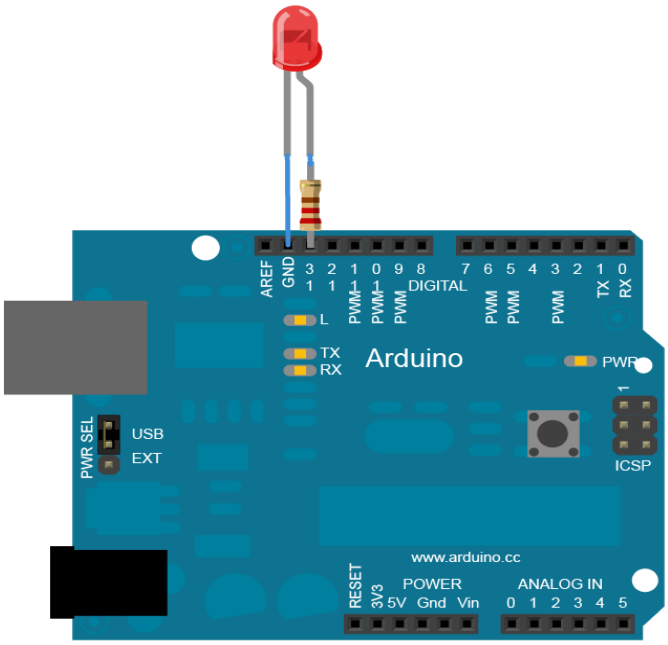


Figure 134: LED Interfacing

Schematic

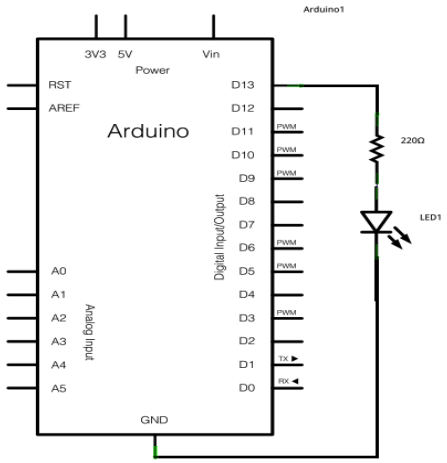


Figure 135: LED Circuit Diagram

Code

After you build the circuit plug your Arduino or Genuino board into your computer, start the Arduino Software (IDE) and enter the code below. You may also load it from the menu File/Examples/01.Basics/Blink . The first thing you do is to initialize LED_BUILTIN pin as an output pin with the line

```
pinMode(LED_BUILTIN, OUTPUT);
```

In the main loop, you turn the LED on with the line:

```
digitalWrite(LED_BUILTIN, HIGH);
```

This supplies 5 volts to the LED anode. That creates a voltage difference across the pins of the LED, and lights it up. Then you turn it off with the line:

```
digitalWrite(LED_BUILTIN, LOW);
```

That takes the LED_BUILTIN pin back to 0 volts, and turns the LED off. In between the on and the off, you want enough time for a person to see the change, so the delay() commands tell the board to do nothing for 1000 milliseconds, or one second. When you use the delay() command, nothing else happens for that amount of time. Once you've understood the basic examples, check out the BlinkWithoutDelay example to learn how to create a delay while doing other things.

Code:

```
void setup()

{
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

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

Hands on: Each and every trainee will be able to write a blinking LED program .

Analog read Serial

This example shows you how to read analog input from the physical world using a potentiometer. A potentiometer is a simple mechanical device that provides a varying amount of resistance when its shaft is turned. By passing voltage through a potentiometer and into an analog input on your board, it is possible to measure the amount of resistance produced by a potentiometer (or *pot* for short) as an analog value. In this example you will monitor the state of your potentiometer after establishing serial communication between your Arduino or Genuino and your computer running the Arduino Software (IDE).

Hardware Required

- Arduino or Genuino Board
- 10k ohm Potentiometer

Circuit

Connect the three wires from the potentiometer to your board. The first goes from one of the outer pins of the potentiometerto ground . The second goes from the other outer pin of the potentiometer to 5 volts. The third goes from the middle pin of the potentiometer to the analog pin A0.

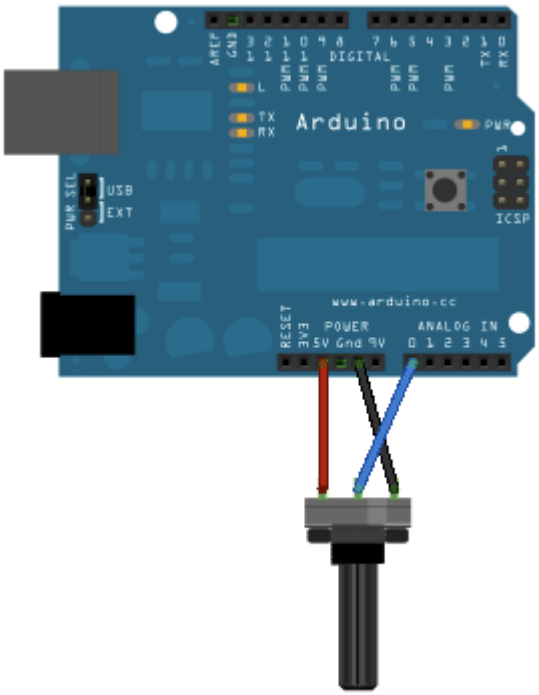
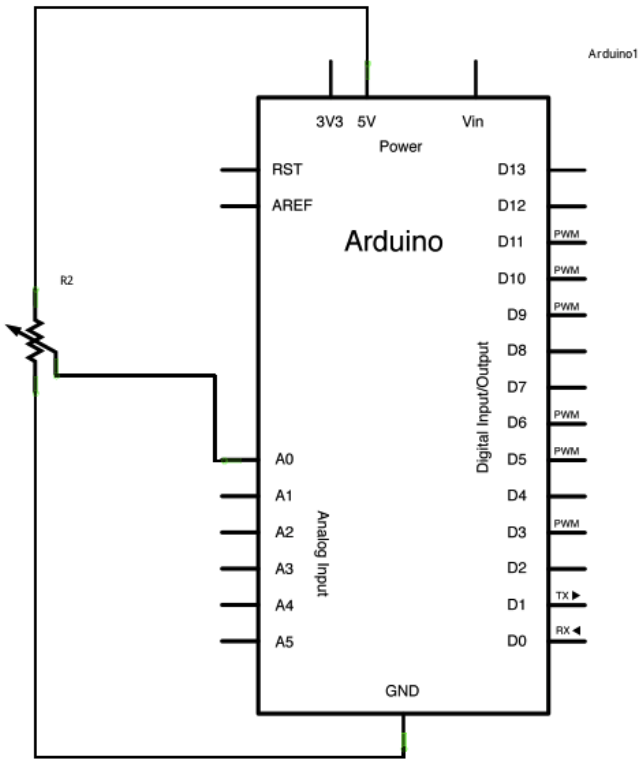


Figure 136:image developed using Fritzing

By turning the shaft of the potentiometer, you change the amount of resistance on either side of the wiper, which is connected to the center pin of the potentiometer. This changes the voltage at the center pin. When the resistance between the center and the side connected to 5 volts is close to zero (and the resistance on the other side is close to 10k ohm), the voltage at the center pin nears 5 volts. When the resistances are reversed, the voltage at the center pin nears 0 volts, or ground. This voltage is the *analog voltage* that you're reading as an input.

The Arduino and Genuino boards have a circuit inside called an *analog-to-digital converter or ADC* that reads this changing voltage and converts it to a number between 0 and 1023. When the shaft is turned all the way in one direction, there are 0 volts going to the pin, and the input value is 0. When the shaft is turned all the way in the opposite direction, there are 5 volts going to the pin and the input value is 1023. In between, analogRead() returns a number between 0 and 1023 that is proportional to the amount of voltage being applied to the pin.

Schematic



Code

In the sketch below, the only thing that you do in the setup function is to begin serial communications, at 9600 bits of data per second, between your board and your computer with the command:

```
Serial.begin(9600);
```

Next, in the main loop of your code, you need to establish a variable to store the resistance value (which will be between 0 and 1023, perfect for an int datatype) coming in from your potentiometer:

```
int sensorValue = analogRead(A0);
```

Finally, you need to print this information to your serial monitor window. You can do this with the command Serial.println() in your last line of code:

```
Serial.println(sensorValue)
```

Now, when you open your Serial Monitor in the Arduino Software (IDE) (by clicking the icon that looks like a lens, on the right, in the green top bar or using the keyboard shortcut Ctrl+Shift+M), you should see a steady stream of numbers ranging from 0-1023, correlating to the position of the pot. As you turn your potentiometer, these numbers will respond almost instantly.

```
void setup()
{
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);        // delay in between reads for stability
}
```

Hands on: Each and Every trainee will be able to know to connect a sensor a microcontroller and finally convert a physical quantity to its corresponding electrical signal .

Digital programming

- Push button

This example shows you how to monitor the state of a switch by establishing serial communication between your Arduino or Genuino and your computer over USB.

Hardware Required

- Arduino or Genuino Board
- A momentary switch, button, or toggle switch
- 10k ohm resistor
- hook-up wires
- breadboard

Circuit

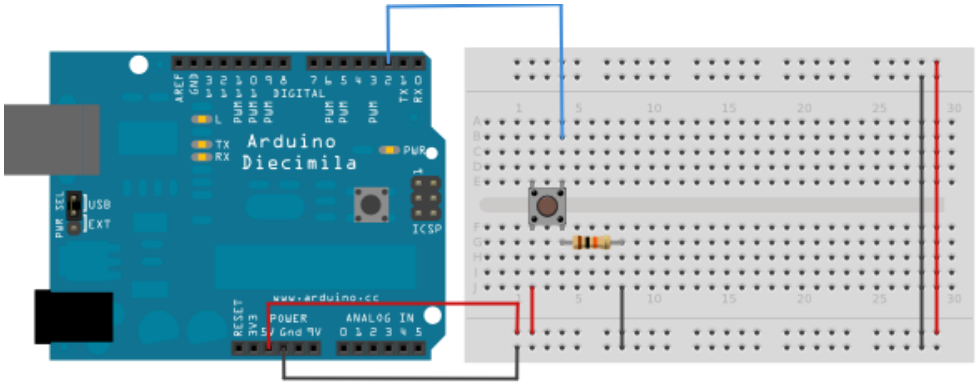


Figure 137: Pushbutton interfacing

Connect three wires to the board. The first two, red and black, connect to the two long vertical rows on the side of the breadboard to provide access to the 5 volt supply and ground. The third wire goes from digital pin 2 to one leg of the pushbutton. That same leg of the button connects through a pull-down resistor (here 10k ohm) to ground. The other leg of the button connects to the 5 volt supply.

Pushbuttons or switches connect two points in a circuit when you press them. When the pushbutton is open (unpressed) there is no connection between the two legs of the pushbutton, so the pin is connected to ground (through the pull-down resistor) and reads as LOW, or 0. When the button is closed (pressed), it makes a connection between its two legs, connecting the pin to 5 volts, so that the pin reads as HIGH, or 1.

If you disconnect the digital i/o pin from everything, the LED may blink erratically. This is because the input is "floating" - that is, it doesn't have a solid connection to voltage or ground, and it will randomly return either HIGH or LOW. That's why you need a pull-down resistor in the circuit.

Schematic

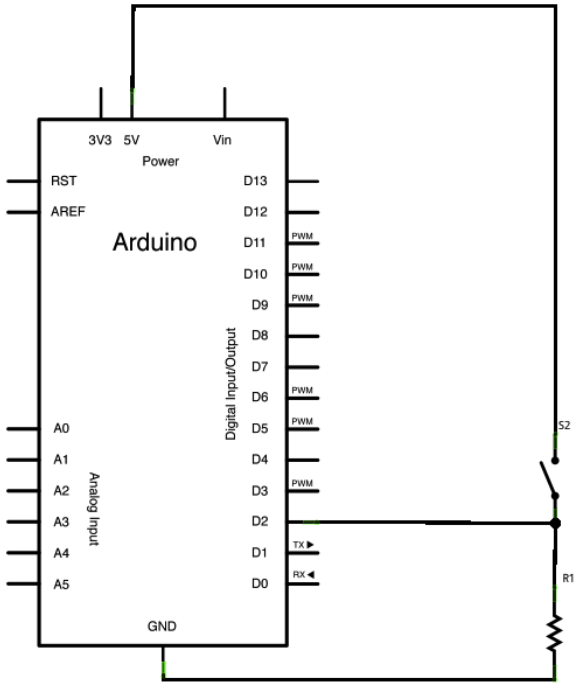


Figure 138: Arduino Input

Code

In the program below, the very first thing that you do will in the setup function is to begin serial communications, at 9600 bits of data per second, between your board and your computer with the line:

```
Serial.begin(9600);
```

Next, initialize digital pin 2, the pin that will read the output from your button, as an input:

```
pinMode(2,INPUT);
```

Now that your setup has been completed, move into the main loop of your code. When your button is pressed, 5 volts will freely flow through your circuit, and when it is not pressed, the input pin will be connected to ground through the 10k ohm resistor. This is a digital input, meaning that the switch can only be in either an on state (seen by your Arduino as a "1", or HIGH) or an off state (seen by your Arduino as a "0", or LOW), with nothing in between.

The first thing you need to do in the main loop of your program is to establish a variable to hold the information coming in from your switch. Since the information coming in from the switch will be either a "1" or a "0", you can use an `int` datatype. Call this variable `sensorValue`, and set it to equal whatever is being read on digital pin 2. You can accomplish all this with just one line of code:

```
int sensorValue = digitalRead(2);
```

Once the bord has read the input, make it print this information back to the computer as a decimal value. You can do this with the command Serial.println() in our last line of code:

```
Serial.println(sensorValue);
```

Now, when you open your Serial Monitor in the Arduino Software (IDE), you will see a stream of "0"s if your switch is open, or "1"s if your switch is closed.

```
/*
                                                    DigitalReadSerial

  Reads a digital input on pin 2, prints the result to the Serial Monitor

       This example code is in the public domain.

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

// digital pin 2 has a pushbutton attached to it. Give it a name:
int pushButton = 2;

// the setup routine runs once when you press reset:
void setup() {
 // initialize serial communication at 9600 bits per second:
 Serial.begin(9600);
 // make the pushbutton's pin an input:
 pinMode(pushButton, INPUT);
}

// the loop routine runs over and over again forever:
void loop() {
 // read the input pin:
 int buttonState = digitalRead(pushButton);
```

```
  // print out the state of the button:
  Serial.println(buttonState);
  delay(1);        // delay in between reads for stability
}
```

Digital write

Write a HIGH or a LOW value to a digital pin.

If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.

If the pin is configured as an INPUT, digitalWrite() will enable (HIGH) or disable (LOW) the internal pullup on the input pin. It is recommended to set the pinMode() to INPUT_PULLUP to enable the internal pull-up resistor. See the digital pins tutorial for more information.

If you do not set the pinMode() to OUTPUT, and connect an LED to a pin, when calling digitalWrite(HIGH), the LED may appear dim. Without explicitly setting pinMode(), digitalWrite() will have enabled the internal pull-up resistor, which acts like a large current-limiting resistor.

Syntax

digitalWrite(pin, value)

Parameters

pin: the pin number

value: HIGH or LOW

Returns

Nothing

Example Code

The code makes the digital pin 13 an OUTPUT and toggles it by alternating between HIGH and LOW at one second pace.

```
void setup()
{
  pinMode(13, OUTPUT);          // sets the digital pin 13 as output
}

void loop()
{
```

```
  digitalWrite(13, HIGH);       // sets the digital pin 13 on
  delay(1000);                  // waits for a second
  digitalWrite(13, LOW);        // sets the digital pin 13 off
  delay(1000);                  // waits for a second
}
```

Analog programming

Analog read

Description

Reads the value from the specified analog pin. Arduino boards contain a multichannel, 10-bit analog to digital converter. This means that it will map input voltages between 0 and the operating voltage(5V or 3.3V) into integer values between 0 and 1023. On an Arduino UNO, for example, this yields a resolution between readings of: 5 volts / 1024 units or, 0.0049 volts (4.9 mV) per unit. See the table below for the usable pins, operating voltage and maximum resolution for some Arduino boards.

The input range can be changed using analogReference(), while the resolution can be changed (only for Zero, Due and MKR boards) using analogReadResolution().

On ATmega based boards (UNO, Nano, Mini, Mega), it takes about 100 microseconds (0.0001 s) to read an analog input, so the maximum reading rate is about 10,000 times a second.

BOARD	OPERATING VOLTAGE	USABLE PINS	MAX RESOLUTION
Uno	5 Volts	A0 to A5	10 bits
Mini, Nano	5 Volts	A0 to A7	10 bits
Mega, Mega2560, MegaADK	5 Volts	A0 to A14	10 bits
Micro	5 Volts	A0 to A11*	10 bits
Leonardo	5 Volts	A0 to A11*	10 bits
Zero	3.3 Volts	A0 to A5	12 bits**
Due	3.3 Volts	A0 to A11	12 bits**
MKR Family boards	3.3 Volts	A0 to A6	12 bits**

*A0 through A5 are labelled on the board, A6 through A11 are respectively available on pins 4, 6, 8, 9, 10, and 12

**The default analogRead() resolution for these boards is 10 bits, for compatibility. You need to use analogReadResolution() to change it to 12 bits.

Syntax

analogRead(pin)

Parameters

pin: the name of the analog input pin to read from (A0 to A5 on most boards, A0 to A6 on MKR boards, A0 to A7 on the Mini and Nano, A0 to A15 on the Mega).

Returns

The analog reading on the pin (int). Although it is limited to the resolution of the analog to digital converter (0-1023 for 10 bits or 0-4095 for 12 bits).

Example Code

The code reads the voltage on analogPin and displays it.

```
int analogPin = A3;     // potentiometer wiper (middle terminal) connected to analog pin 3
                        // outside leads to ground and +5V
int val = 0;            // variable to store the value read

void setup()
{
  Serial.begin(9600);              //  setup serial
}

void loop()
{
  val = analogRead(analogPin);     // read the input pin
  Serial.println(val);             // debug value
}
```

Analog write

Description

Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds. After a call to analogWrite(), the pin will generate a steady square wave of the specified duty cycle until the next call to analogWrite() (or a call to digitalRead() or digitalWrite()) on the same pin. The frequency of the PWM signal on most pins is approximately 490 Hz. On the Uno and similar boards, pins 5 and 6 have a frequency of approximately 980 Hz.

On most Arduino boards (those with the ATmega168 or ATmega328P), this function works on pins 3, 5, 6, 9, 10, and 11. On the Arduino Mega, it works on pins 2 - 13 and 44 - 46. Older Arduino boards with an ATmega8 only support analogWrite() on pins 9, 10, and 11.
The Arduino DUE supports analogWrite() on pins 2 through 13, plus pins DAC0 and DAC1. Unlike the PWM pins, DAC0 and DAC1 are Digital to Analog converters, and act as true analog outputs.
You do not need to call pinMode() to set the pin as an output before calling analogWrite().
The analogWrite function has nothing to do with the analog pins or the analogRead function.

Syntax

analogWrite(pin, value)

Parameters

pin: the pin to write to. Allowed data types: int.
value: the duty cycle: between 0 (always off) and 255 (always on). Allowed data types: int

Returns

Nothing

Example Code

Sets the output to the LED proportional to the value read from the potentiometer.

```
int ledPin = 9;      // LED connected to digital pin 9
int analogPin = 3;   // potentiometer connected to analog pin 3
int val = 0;         // variable to store the read value

void setup()
{
  pinMode(ledPin, OUTPUT);   // sets the pin as output
}

void loop()
{
  val = analogRead(analogPin);   // read the input pin
  analogWrite(ledPin, val / 4);  // analogRead values go from 0 to 1023, analogWrite values from 0 to 255
}
```

Communication

Serial communication

Description

Used for communication between the Arduino board and a computer or other devices. All Arduino boards have at least one serial port (also known as a UART or USART): Serial. It communicates on digital pins 0 (RX) and 1 (TX) as well as with the computer via USB. Thus, if you use these functions, you cannot also use pins 0 and 1 for digital input or output.
You can use the Arduino environment's built-in serial monitor to communicate with an Arduino board. Click the serial monitor button in the toolbar and select the same baud rate used in the call to begin().

Serial communication on pins TX/RX uses TTL logic levels (5V or 3.3V depending on the board). Don't connect these pins directly to an RS232 serial port; they operate at +/- 12V and can damage your Arduino board.

The **Arduino Mega** has three additional serial ports: Serial1 on pins 19 (RX) and 18 (TX), Serial2 on pins 17 (RX) and 16 (TX), Serial3 on pins 15 (RX) and 14 (TX). To use these pins to communicate with your personal computer, you will need an additional USB-to-serial adaptor, as they are not connected to the Mega's USB-to-serial adaptor. To use them to communicate with an external TTL serial device, connect the TX pin to your device's RX pin, the RX to your device's TX pin, and the ground of your Mega to your device's ground.

The **Arduino DUE** has three additional 3.3V TTL serial ports: Serial1 on pins 19 (RX) and 18 (TX); Serial2 on pins 17 (RX) and 16 (TX), Serial3 on pins 15 (RX) and 14 (TX). Pins 0 and 1 are also connected to the corresponding pins of the ATmega16U2 USB-to-TTL Serial chip, which is connected to the USB debug port. Additionally, there is a native USB-serial port on the SAM3X chip, SerialUSB'.

The **Arduino Leonardo** board uses Serial1 to communicate via TTL (5V) serial on pins 0 (RX) and 1 (TX). Serial is reserved for USB CDC communication. For more information, refer to the Leonardo getting started page and hardware page.

Functions

If (Serial)
available()
availableForWrite()
begin()
end()
find()
findUntil()
flush()
parseFloat()
parseInt()
peek()
print()

println()
read()
readBytes()
readBytesUntil()
readString()
readStringUntil()
setTimeout()
write()
serialEvent()

Control

For loop

Description

The for statement is used to repeat a block of statements enclosed in curly braces. An increment counter is usually used to increment and terminate the loop. The for statement is useful for any repetitive operation, and is often used in combination with arrays to operate on collections of data/pins.

Syntax

```
for (initialization; condition; increment) {
        //statement(s);
}
```

The **initialization** happens first and exactly once. Each time through the loop, the **condition** is tested; if it's true, the statement block, and the **increment** is executed, then the **condition** is tested again. When the **condition** becomes false, the loop ends.

Example Code

```
// Dim an LED using a PWM pin
int PWMpin = 10; // LED in series with 470 ohm resistor on pin 10

void setup()
{
  // no setup needed
}

void loop()
{
   for (int i=0; i <= 255; i++){
      analogWrite(PWMpin, i);
```

```
    delay(10);
  }
}
```

Notes and Warnings

The C for loop is much more flexible than for loops found in some other computer languages, including BASIC. Any or all of the three header elements may be omitted, although the semicolons are required. Also the statements for initialization, condition, and increment can be any valid C statements with unrelated variables, and use any C datatypes including floats. These types of unusual for statements may provide solutions to some rare programming problems.

For example, using a multiplication in the increment line will generate a logarithmic progression:

```
for(int x = 2; x < 100; x = x * 1.5){
println(x);
}
```

Generates: 2,3,4,6,9,13,19,28,42,63,94

Another example, fade an LED up and down with one for loop:

```
void loop()
{
  int x = 1;
  for (int i = 0; i > -1; i = i + x){
    analogWrite(PWMpin, i);
    if (i == 255) x = -1;            // switch direction at peak
    delay(10);
  }
}
```

If statement

The if() statement is the most basic of all programming control structures. It allows you to make something happen or not, depending on whether a given condition is true or not. It looks like this:

```
if (someCondition) {

  // do stuff if the condition is true

}
```

There is a common variation called if-else that looks like this:

```
if (someCondition) {

  // do stuff if the condition is true

} else {

  // do stuff if the condition is false

}
```

There's also the else-if, where you can check a second condition if the first is false:

```
if (someCondition) {

  // do stuff if the condition is true

} else if (anotherCondition) {

  // do stuff only if the first condition is false

  // and the second condition is true

}
```

You'll use if statements all the time. The example below turns on an LED on pin 13 (the built-in LED on many Arduino boards) if the value read on an analog input goes above a certain threshold.

Hardware Required

Arduino or Genuino Board
Potentiometer or variable resistor

Circuit

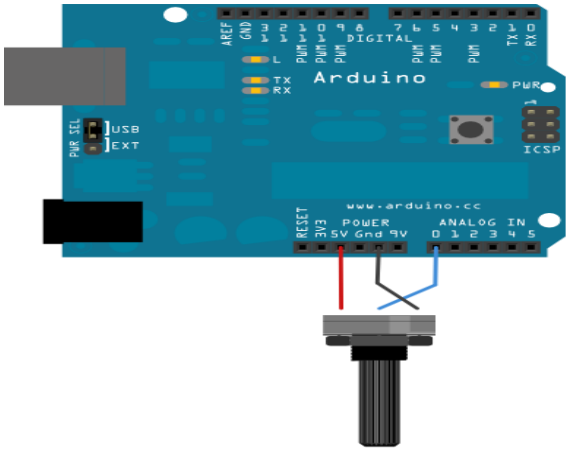


Figure 139: PoT Interfaing
image developed using Fritzing. For more circuit examples, see the Fritzing project page

Schematic
click the image to enlarge

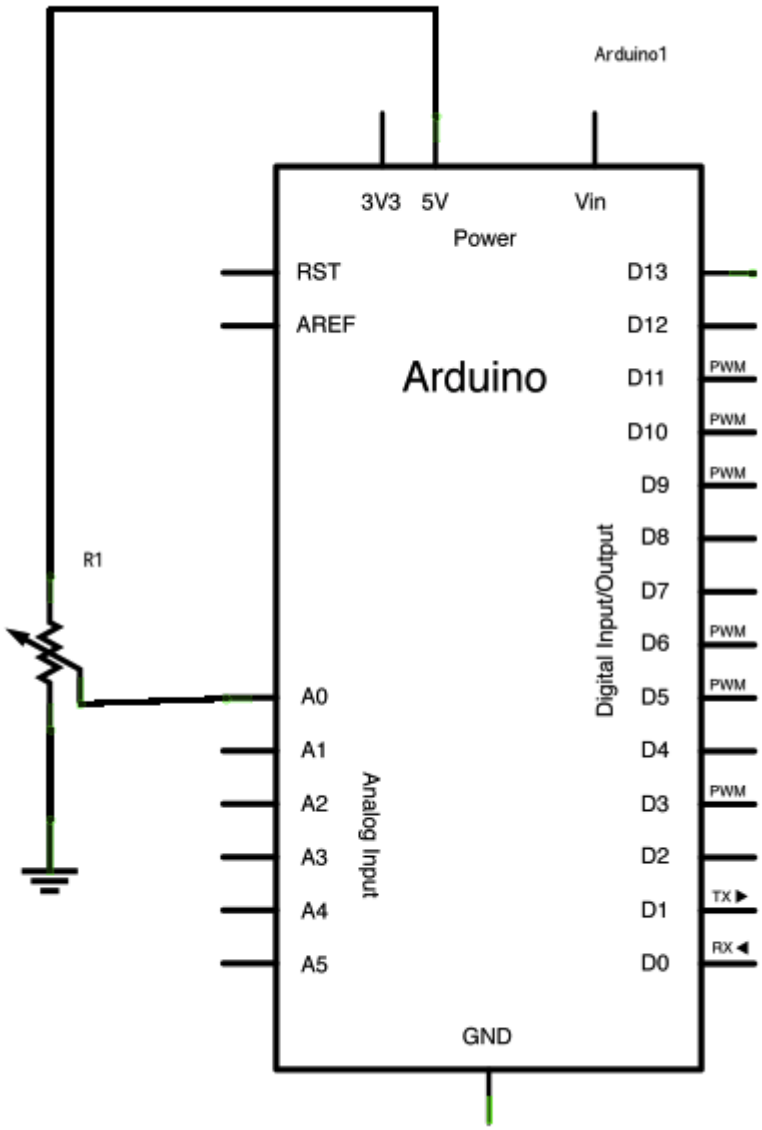


Figure 140: POT Circuit Diagram

Code

In the code below, a variable called analogValue is used to store the data collected from a potentiometer connected to the board on analogPin 0. This data is then compared to a threshold value. If the analog value is found to be above the set threshold the built-in LED connected to digital pin 13 is turned on. If analogValue is found to be < (less than) threshold, the LED remains off.

```
// These constants won't change:
const int analogPin = A0;   // pin that the sensor is attached to
const int ledPin = 13;      // pin that the LED is attached to
const int threshold = 400;  // an arbitrary threshold level that's in the range of the analog input
```

```
void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize serial communications:
  Serial.begin(9600);
}

void loop() {
  // read the value of the potentiometer:
  int analogValue = analogRead(analogPin);

  // if the analog value is high enough, turn on the LED:
  if (analogValue > threshold) {
                                                    digitalWrite(ledPin, HIGH);
  } else {
                                                    digitalWrite(ledPin, LOW);
  }

  // print the analog value:
  Serial.println(analogValue);
  delay(1);        // delay in between reads for stability
}
```

Array

This variation on the For Loop Iteration example shows how to use an array. An array is a variable with multiple parts. If you think of a variable as a cup that holds values, you might think of an array as an ice cube tray. It's like a series of linked cups, all of which can hold the same maximum value.

The For Loop Iteration example shows you how to light up a series of LEDs attached to pins 2 through 7 of the Arduino or Genuino board, with certain limitations (the pins have to be numbered contiguously, and the LEDs have to be turned on in sequence).

This example shows you how you can turn on a sequence of pins whose numbers are neither contiguous nor necessarily sequential. To do this is, you can put the pin numbers in an array and then use for loops to iterate over the array.

This example makes use of 6 LEDs connected to the pins 2 - 7 on the board using 220 ohm resistors, just like in the For Loop. However, here the order of the LEDs is determined by their order in the array, not by their physical order.

This technique of putting the pins in an array is very handy. You don't have to have the pins sequential to one another, or even in the same order. You can rearrange them in any order you want.

Hardware Required

- Arduino or Genuino Board
- 6 LEDs
- 6 220 ohm resistors
- hook-up wires
- breadboard

Circuit

Connect six LEDs, with 220 ohm resistors in series, to digital pins 2-7 on your board.

click the image to enlarge

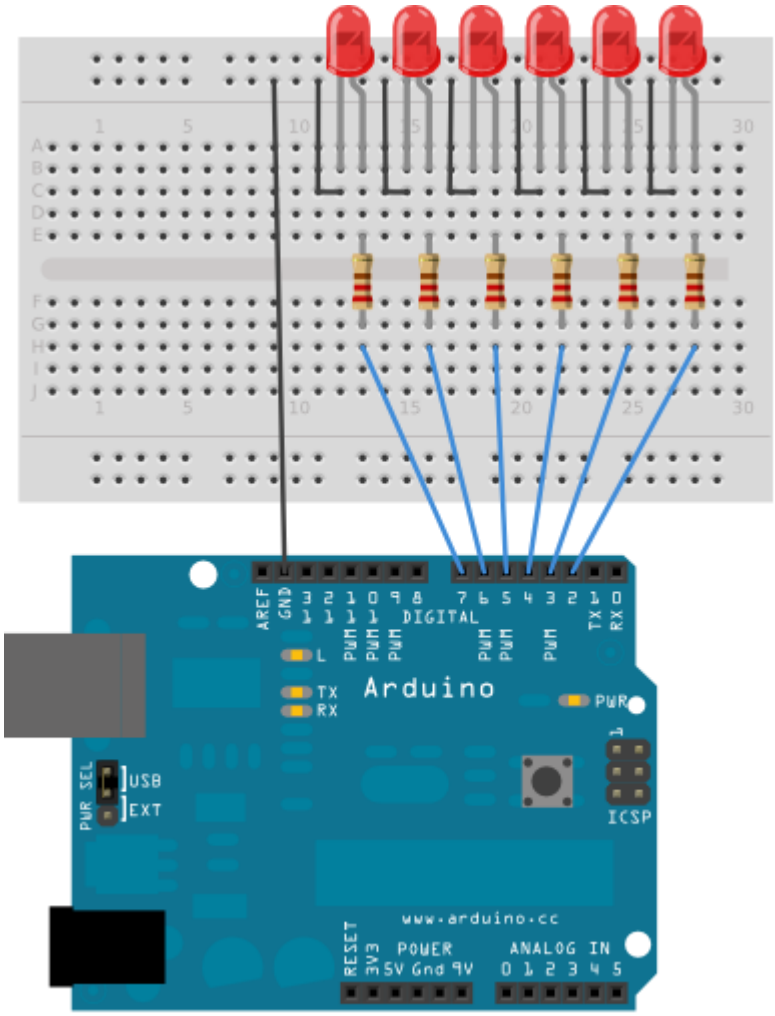


Figure 141: LED Interfacing

image developed using Fritzing. For more circuit examples, see the Fritzing project page

Schematic:

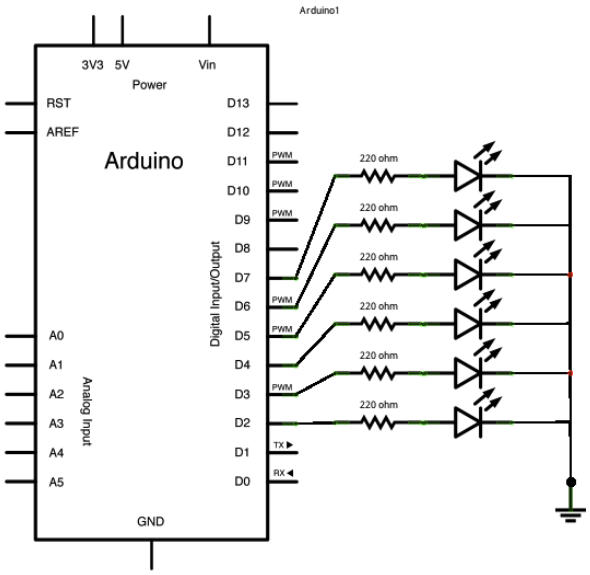


Figure 142: Circuit Diagram

Code

```
int timer = 100;           // The higher the number, the slower the timing.
int ledPins[] = {
  2, 7, 4, 6, 5, 3
};          // an array of pin numbers to which LEDs are attached
int pinCount = 6;           // the number of pins (i.e. the length of the array)

void setup() {
  // the array elements are numbered from 0 to (pinCount - 1).
  // use a for loop to initialize each pin as an output:
  for (int thisPin = 0; thisPin < pinCount; thisPin++) {
                                                    pinMode(ledPins[thisPin], OUTPUT);
  }
}

void loop() {
  // loop from the lowest pin to the highest:
  for (int thisPin = 0; thisPin < pinCount; thisPin++) {
                    // turn the pin on:
                                                    digitalWrite(ledPins[thisPin], HIGH);
                                                                                delay(timer);
                    // turn the pin off:
                                                    digitalWrite(ledPins[thisPin], LOW);


  }

  // loop from the highest pin to the lowest:
  for (int thisPin = pinCount - 1; thisPin >= 0; thisPin--) {
```

```
                // turn the pin on:
                    digitalWrite(ledPins[thisPin], HIGH);
                                    delay(timer);
                // turn the pin off:
                    digitalWrite(ledPins[thisPin], LOW);
  }
}
```

Sensors

Temperature

Connecting to a Temperature Sensor
These sensors have little chips in them and while they're not that delicate, they do need to be handled properly. Be careful of static electricity when handling them and make sure the power supply is connected up correctly and is between 2.7 and 5.5V DC - so don't try to use a 9V battery!

They come in a "TO-92" package which means the chip is housed in a plastic hemi-cylinder with three legs. The legs can be bent easily to allow the sensor to be plugged into a breadboard. You can also solder to the pins to connect long wires. If you need to waterproof the sensor, you can see below for an Instructable for how to make an excellent case.
Reading the Analog Temperature Data
Unlike the FSR or photocell sensors we have looked at, the TMP36 and friends doesn't act like a resistor. Because of that, there is really only one way to read the temperature value from the sensor, and that is plugging the output pin directly into an Analog (ADC) input.

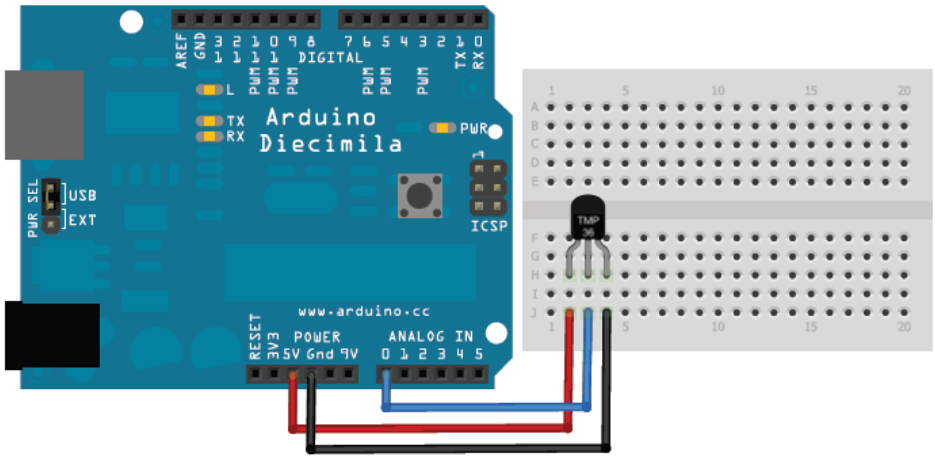


Figure 143: LM35 Interfacing

Remember that you can use anywhere between 2.7V and 5.5V as the power supply. For this example I'm showing it with a 5V supply but note that you can use this with a 3.3v supply just as easily. No matter what supply you use, the analog voltage reading will range from about 0V (ground) to about 1.75V.
If you're using a 5V Arduino, and connecting the sensor directly into an Analog pin, you can use these formulas to turn the 10-bit analog reading into a temperature:
Voltage at pin in milliVolts = (*reading from ADC*) * (5000/1024)
This formula converts the number 0-1023 from the ADC into 0-5000mV (= 5V)

If you're using a 3.3V Arduino, you'll want to use this:
Voltage at pin in milliVolts = (*reading from ADC*) * (3300/1024)
This formula converts the number 0-1023 from the ADC into 0-3300mV (= 3.3V)
Then, to convert millivolts into temperature, use this formula:
Centigrade temperature = [(analog voltage in mV) - 500] / 10
Arduino Sketch - Simple Thermometer

This example code for Arduino shows a quick way to create a temperature sensor, it simply prints to the serial port what the current temperature is in both Celsius and Fahrenheit.

```
1.  //TMP36 Pin Variables
2.  int sensorPin = 0; //the analog pin the TMP36's Vout (sense) pin is connected to
3.                     //the resolution is 10 mV / degree centigrade with a
4.                     //500 mV offset to allow for negative temperatures
5.
6.  /*
7.   * setup() - this function runs once when you turn your Arduino on
8.   * We initialize the serial connection with the computer
9.   */
10. void setup()
11. {
12.   Serial.begin(9600);  //Start the serial connection with the computer
13.                        //to view the result open the serial monitor
14. }
15.
16. void loop()                 // run over and over again
17. {
18.  //getting the voltage reading from the temperature sensor
19.  int reading = analogRead(sensorPin);
20.
21.  // converting that reading to voltage, for 3.3v arduino use 3.3
22.  float voltage = reading * 5.0;
23.  voltage /= 1024.0;
24.
25.  // print out the voltage
26.  Serial.print(voltage); Serial.println(" volts");
27.
28.  // now print out the temperature
29.  float temperatureC = (voltage - 0.5) * 100 ;  //converting from 10 mv per
     degree wit 500 mV offset
30.                                  //to degrees ((voltage - 500mV) times 100)
31.  Serial.print(temperatureC); Serial.println(" degrees C");
32.
33.  // now convert to Fahrenheit
34.  float temperatureF = (temperatureC * 9.0 / 5.0) + 32.0;
35.  Serial.print(temperatureF); Serial.println(" degrees F");
36.
37.  delay(1000);                          //waiting a second
38. }
```

Distance

The HC-SR04 ultrasonic sensor uses SONAR to determine the distance of an object just like the bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package from 2 cm to 400 cm or 1" to 13 feet.

The operation is not affected by sunlight or black material, although acoustically, soft materials like cloth can be difficult to detect. It comes complete with ultrasonic transmitter and receiver module.

Figure 144: Ultrasonic Sensor

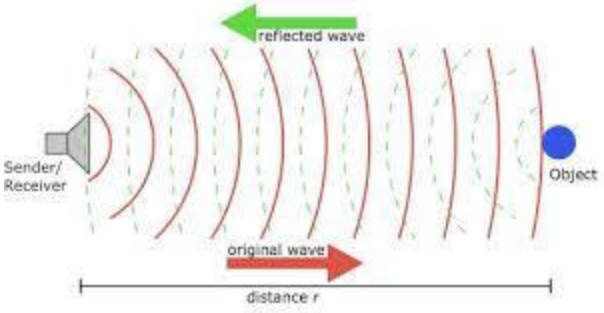


Figure 145: working Principle

Technical Specifications

- Power Supply − +5V DC
- Quiescent Current − <2mA
- Working Current − 15mA
- Effectual Angle − <15°
- Ranging Distance − 2cm – 400 cm/1″ – 13ft
- Resolution − 0.3 cm
- Measuring Angle − 30 degree

Components Required

You will need the following components −

- 1 × Breadboard
- 1 × Arduino Uno R3
- 1 × ULTRASONIC Sensor (HC-SR04)

Procedure

Follow the circuit diagram and make the connections as shown in the image given below.

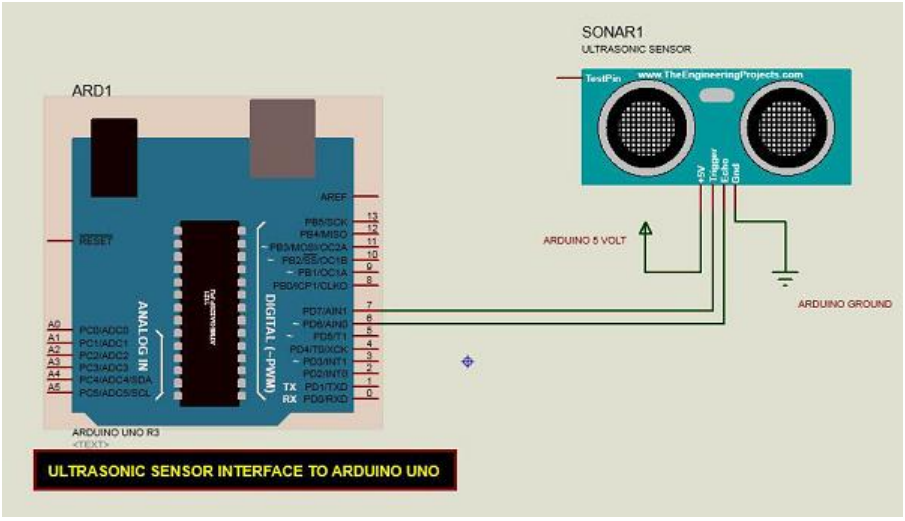


Figure 146: ULtrasonic Sensor Interfacing

Sketch

Open the Arduino IDE software on your computer. Coding in the Arduino language will control your circuit. Open a new sketch File by clicking New.

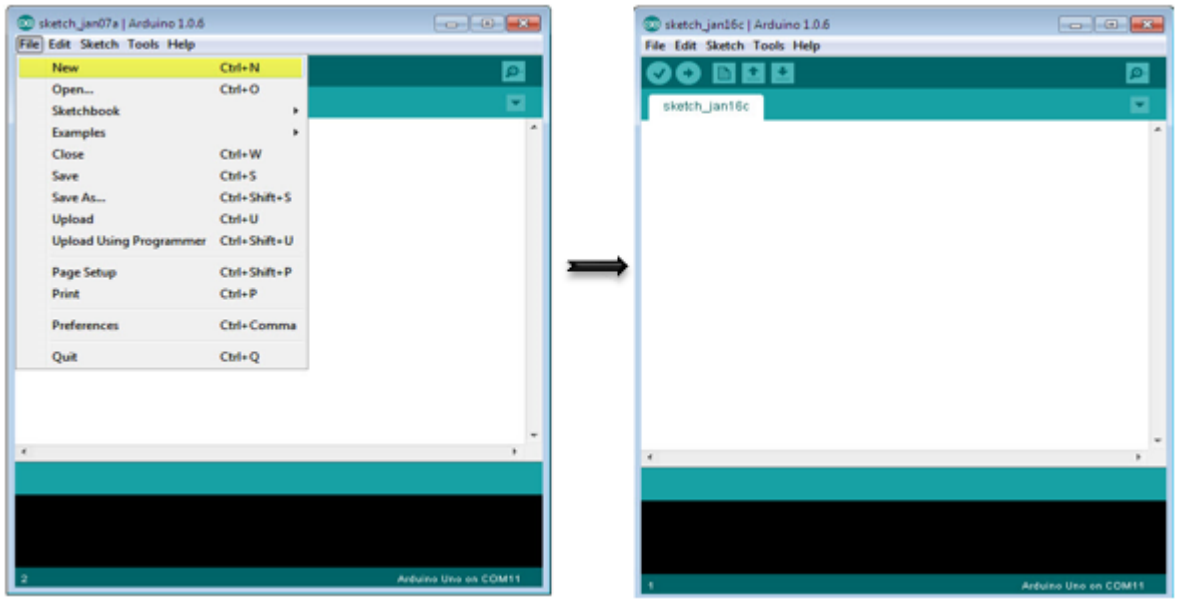


Figure 147: Arduino IDE

Arduino Code

```
const int pingPin = 7; // Trigger Pin of Ultrasonic Sensor
const int echoPin = 6; // Echo Pin of Ultrasonic Sensor

void setup() {
  Serial.begin(9600); // Starting Serial Terminal
}

void loop() {
  long duration, inches, cm;
  pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(pingPin, LOW);
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);
  inches = microsecondsToInches(duration);
  cm = microsecondsToCentimeters(duration);
  Serial.print(inches);
  Serial.print("in, ");
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();
  delay(100);
}

long microsecondsToInches(long microseconds) {
  return microseconds / 74 / 2;
}
```

```
long microsecondsToCentimeters(long microseconds) {
  return microseconds / 29 / 2;
}
```

Code to Note

The Ultrasonic sensor has four terminals - +5V, Trigger, Echo, and GND connected as follows −

- Connect the +5V pin to +5v on your Arduino board.
- Connect Trigger to digital pin 7 on your Arduino board.
- Connect Echo to digital pin 6 on your Arduino board.
- Connect GND with GND on Arduino.

In our program, we have displayed the distance measured by the sensor in inches and cm via the serial port.

Result

You will see the distance measured by sensor in inches and cm on Arduino serial monitor.

Light

Analog Voltage Reading Method

The easiest way to measure a resistive sensor is to connect one end to Power and the other to a **pull-down** resistor to ground. Then the point between the fixed pulldown resistor and the variable photocell resistor is connected to the analog input of a microcontroller such as an Arduino (shown)

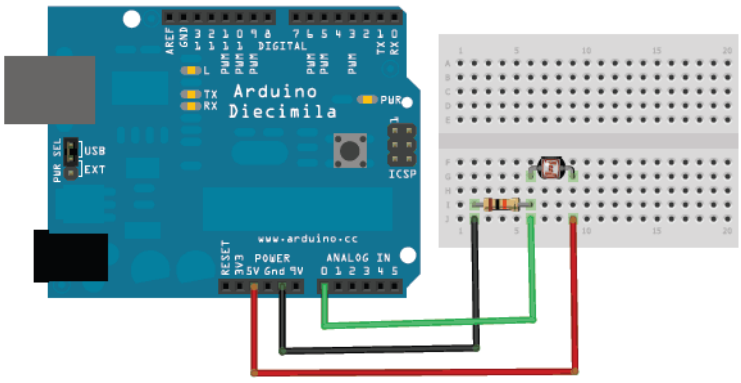


Figure 148: LDR Interfacing

For this example I'm showing it with a 5V supply but note that you can use this with a 3.3v supply just as easily. In this configuration the analog voltage reading ranges from 0V (ground) to about 5V (or about the same as the power supply voltage).

The way this works is that as the resistance of the photocell decreases, the total resistance of the photocell and the pulldown resistor decreases from over 600KΩ to 10KΩ. That means that the current flowing through both resistors *increases* which in turn causes the voltage across the fixed 10KΩ resistor to increase. It's quite a trick!

Ambient light like…	Ambient light (lux)	Photocell resistance (Ω)	LDR + R (Ω)	Current thru LDR +R	Voltage across R
Dim hallway	**0.1 lux**	600KΩ	610 KΩ	0.008 mA	0.1 V
Moonlit night	**1 lux**	70 KΩ	80 KΩ	0.07 mA	0.6 V
Dark room	**10 lux**	10 KΩ	20 KΩ	0.25 mA	2.5 V
Dark overcast day / Bright room	**100 lux**	1.5 KΩ	11.5 KΩ	0.43 mA	4.3 V
Overcast day	**1000 lux**	300 Ω	10.03 KΩ	0.5 mA	5V

- *This table indicates the approximate analog voltage based on the sensor light/resistance w/a 5V supply and 10KΩ pulldown resistor.*
- If you're planning to have the sensor in a bright area and use a 10KΩ pulldown, it will quickly *saturate*. That means that it will hit the 'ceiling' of 5V and not be able to differentiate between kinda bright and really bright. In that case, you should replace the 10KΩ pulldown with a 1KΩ pulldown. In that case, it will not be able to detect dark level differences as well but it will be able to detect bright light differences better. This is a tradeoff that you will have to decide upon!
- You can also use the "Axel Benz" formula by first measuring the minimum and maximum resistance value with the multimeter and then finding the resistor value with: Pull-Down-Resistor = squareroot(Rmin * Rmax), this will give you slightly better range calculations

Ambient light like…	Ambient light (lux)	Photocell resistance (?)	LDR + R (?)	Current thru LDR+R	Voltage across R
Moonlit night	**1 lux**	70 KΩ	71 KΩ	0.07 mA	0.1 V
Dark room	**10 lux**	10 KΩ	11 KΩ	0.45 mA	0.5 V
Dark overcast day / Bright room	**100 lux**	1.5 KΩ	2.5 KΩ	2 mA	2.0 V
Overcast day	**1000 lux**	300 Ω	1.3 KΩ	3.8 mA	3.8 V
Full daylight	**10,000 lux**	100 Ω	1.1 KΩ	4.5 mA	4.5 V

- *This table indicates the approximate analog voltage based on the sensor light/resistance w/a 5V supply and 1K pulldown resistor.*
- Note that our method does not provide linear voltage with respect to brightness! Also, each sensor will be different. As the light level increases, the analog voltage goes up even though the resistance goes down:
- **Vo = Vcc (R / (R + Photocell))**
- That is, the voltage is proportional to the **inverse** of the photocell resistance which is, in turn, inversely proportional to light levels.

Simple Demonstration of Use

This sketch will take the analog voltage reading and use that to determine how bright the red LED is. The darker it is, the brighter the LED will be! Remember that the LED has to be connected to a PWM pin for this to work, I use pin 11 in this example.

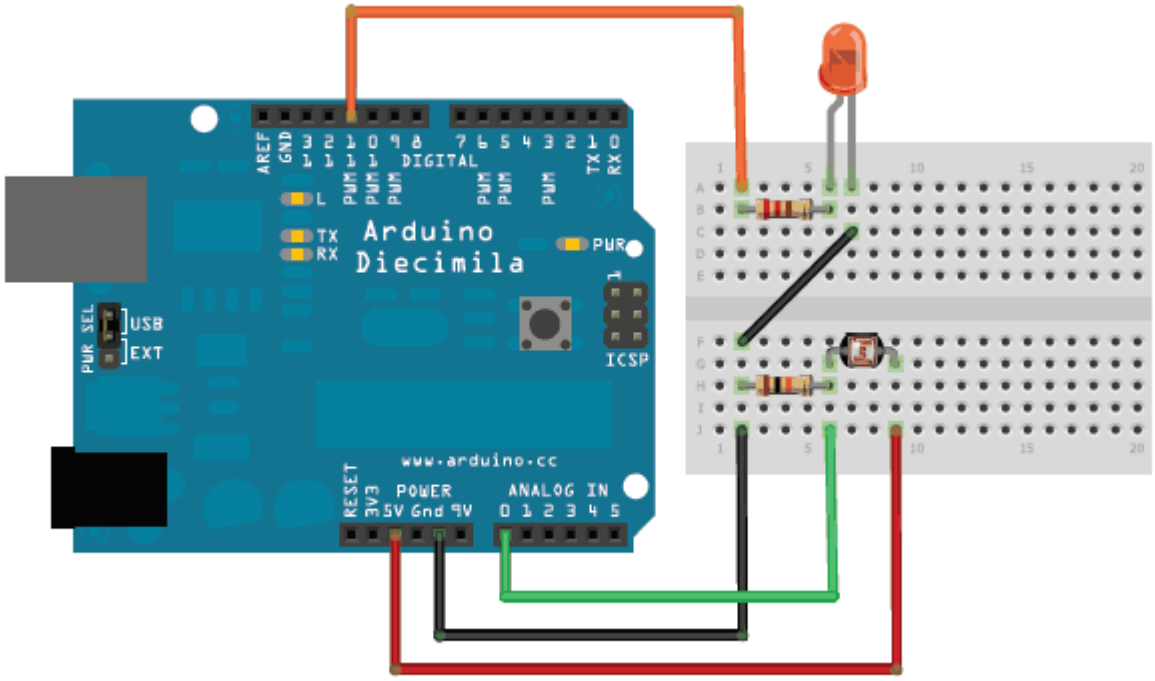


Figure 149: LDR and LED Interfacing

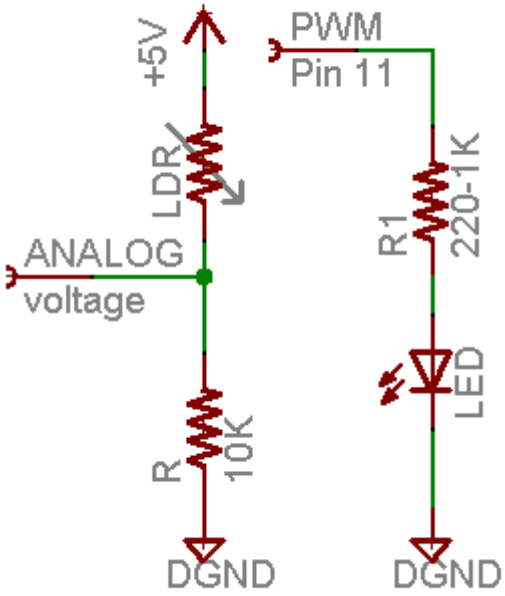


Figure 150: LED and LDR Circuit

These examples assume you know some basic Arduino programming. If you don't, maybe spend some time reviewing the basics at the Arduino tutorial?

Code

```
/* Photocell simple testing sketch.

Connect one end of the photocell to 5V, the other end to Analog 0.
Then connect one end of a 10K resistor from Analog 0 to ground
Connect LED from pin 11 through a resistor to ground
For more information see http://learn.adafruit.com/photocells */

int photocellPin = 0;     // the cell and 10K pulldown are connected to a0
int photocellReading;     // the analog reading from the sensor divider
int LEDpin = 11;          // connect Red LED to pin 11 (PWM pin)
int LEDbrightness;        //
void setup(void) {
  // We'll send debugging information via the Serial monitor
  Serial.begin(9600);
}

void loop(void) {
  photocellReading = analogRead(photocellPin);

  Serial.print("Analog reading = ");
  Serial.println(photocellReading);     // the raw analog reading

  // LED gets brighter the darker it is at the sensor
  // that means we have to -invert- the reading from 0-1023 back to 1023-0
  photocellReading = 1023 - photocellReading;
  //now we have to map 0-1023 to 0-255 since thats the range analogWrite uses
  LEDbrightness = map(photocellReading, 0, 1023, 0, 255);
  analogWrite(LEDpin, LEDbrightness);

  delay(100);
}
```

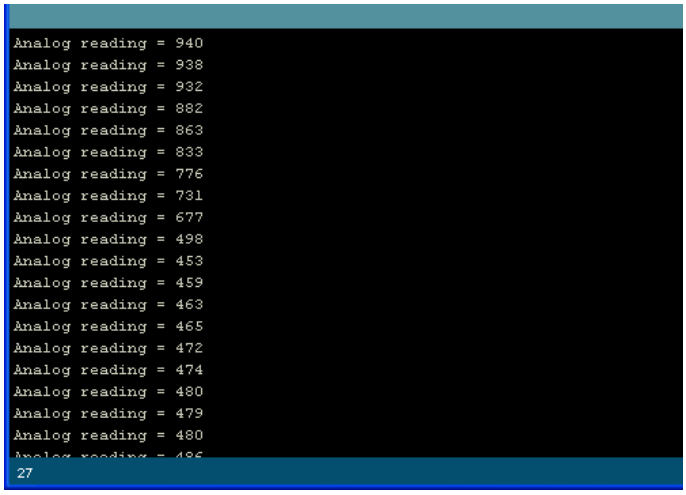


Figure 151: Serial output

You may want to try different pulldown resistors depending on the light level range you want to detect!

Simple Code for Analog Light Measurements
This code doesn't do any calculations, it just prints out what it interprets as the amount of light in a qualitative manner. For most projects, this is pretty much all thats needed!

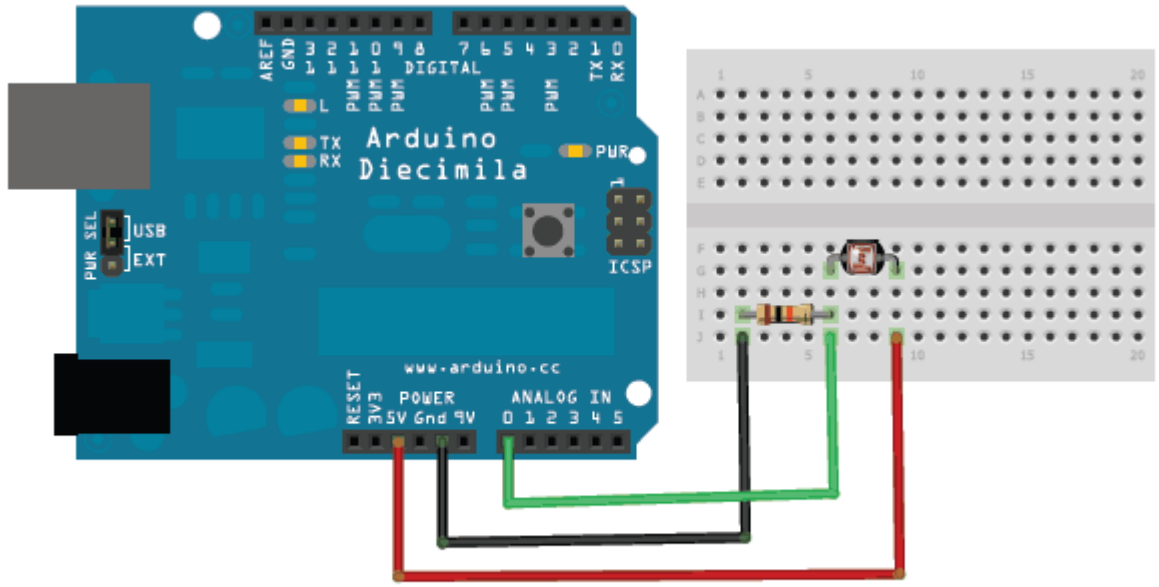


Figure 152: Simple LDR Circuit

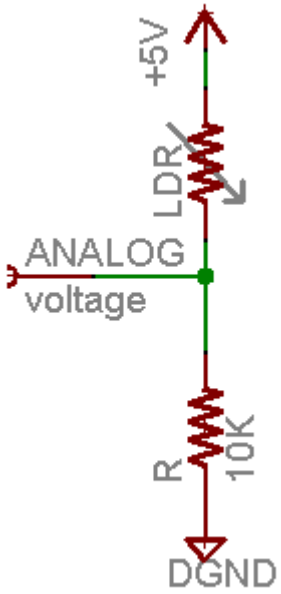


Figure 153: LDR Circuit

Code

```
1.  /* Photocell simple testing sketch.
2.
3.  Connect one end of the photocell to 5V, the other end to Analog 0.
4.  Then connect one end of a 10K resistor from Analog 0 to ground
```

```
5.
6.    For more information see http://learn.adafruit.com/photocells */
7.
8.    int photocellPin = 0;     // the cell and 10K pulldown are connected to a0
9.    int photocellReading;     // the analog reading from the analog resistor divider
10.
11.   void setup(void) {
12.     // We'll send debugging information via the Serial monitor
13.     Serial.begin(9600);
14.   }
15.
16.   void loop(void) {
17.     photocellReading = analogRead(photocellPin);
18.
19.     Serial.print("Analog reading = ");
20.     Serial.print(photocellReading);     // the raw analog reading
21.
22.     // We'll have a few threshholds, qualitatively determined
23.     if (photocellReading < 10) {
24.       Serial.println(" - Dark");
25.     } else if (photocellReading < 200) {
26.       Serial.println(" - Dim");
27.     } else if (photocellReading < 500) {
28.       Serial.println(" - Light");
29.     } else if (photocellReading < 800) {
30.       Serial.println(" - Bright");
31.     } else {
32.       Serial.println(" - Very bright");
33.     }
34.     delay(1000);
35.   }
```

To test it, I started in a sunlit (but shaded) room and covered the sensor with my hand, then covered it with a piece of blackout fabric.

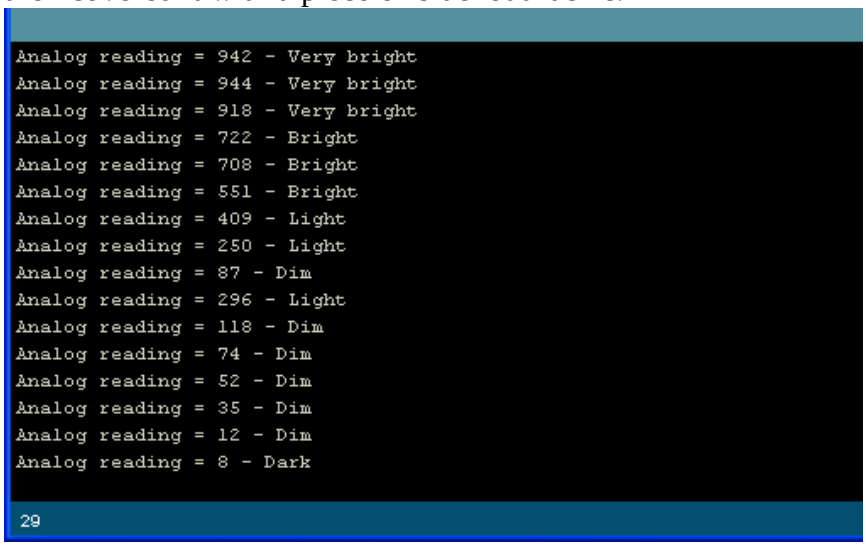


Figure 154: Output

BONUS! Reading Photocells Without Analog Pins

Because photocells are basically resistors, its possible to use them even if you don't have any analog pins on your microcontroller (or if say you want to connect more than you have analog input pins). The way we do this is by taking advantage of a basic electronic property of resistors and capacitors. It turns out that if you take a capacitor that is initially storing no voltage, and then connect it to power (like 5V) through a resistor, it will charge up to the power voltage slowly. The bigger the resistor, the slower it is.

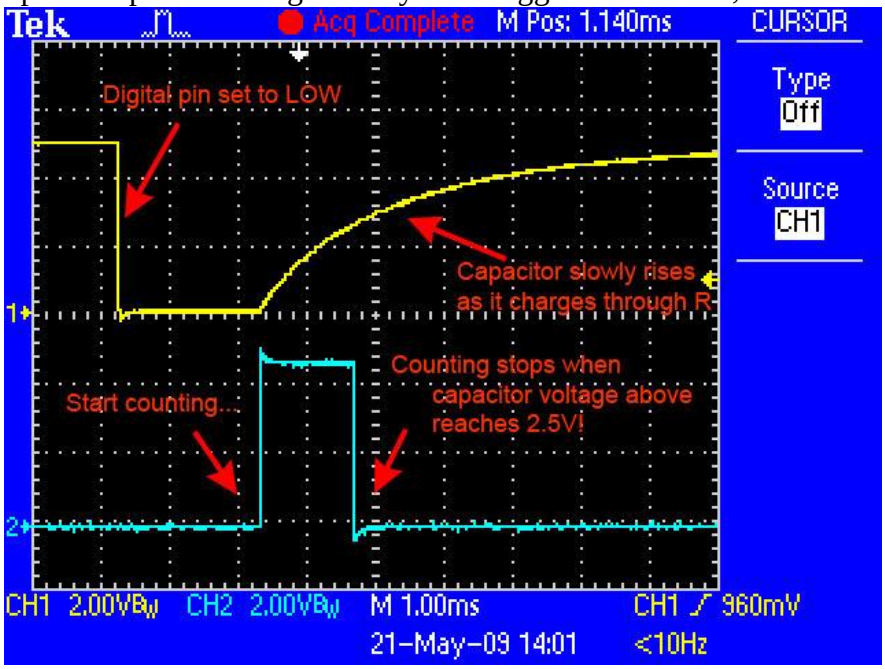


Figure 155: Scope Output

This capture from an oscilloscope shows whats happening on the digital pin (yellow). The blue line indicates when the sketch starts counting and when the couting is complete, about 1.2ms later.

This is because the capacitor acts like a bucket and the resistor is like a thin pipe. To fill a bucket up with a very thin pipe takes enough time that you can figure out how wide the pipe is by timing how long it takes to fill the bucket up halfway.

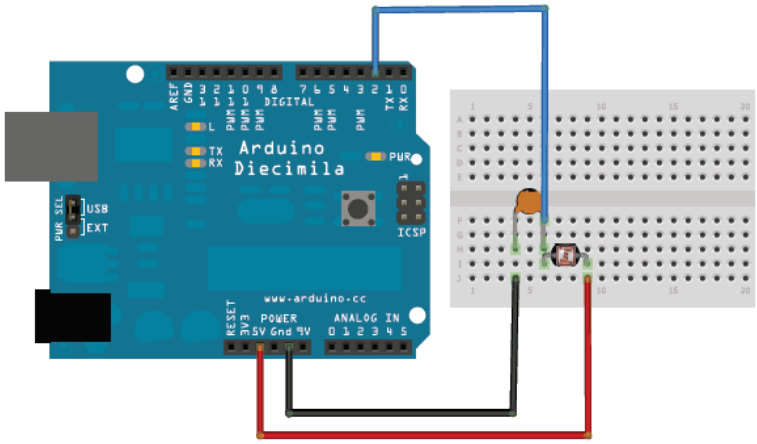


Figure 156: LDR Intefacing

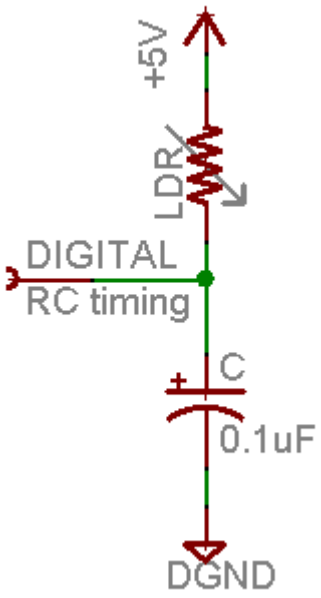


Figure 157: LRD Circuit

In this case, our 'bucket' is a 0.1uF ceramic capacitor. You can change the capacitor nearly any way you want but the timing values will also change. 0.1uF seems to be an OK place to start for these photocells. If you want to measure brighter ranges, use a 1uF capacitor. If you want to measure darker ranges, go down to 0.01uF.

Code

```
/* Photocell simple testing sketch.
Connect one end of photocell to power, the other end to pin 2.
Then connect one end of a 0.1uF capacitor from pin 2 to ground
For more information see http://learn.adafruit.com/photocells */

int photocellPin = 2;     // the LDR and cap are connected to pin2
int photocellReading;     // the digital reading
int ledPin = 13;    // you can just use the 'built in' LED

void setup(void) {
  // We'll send debugging information via the Serial monitor
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);  // have an LED for output
}

void loop(void) {
  // read the resistor using the RCtime technique
  photocellReading = RCtime(photocellPin);

  if (photocellReading == 30000) {
    // if we got 30000 that means we 'timed out'
    Serial.println("Nothing connected!");
  } else {
    Serial.print("RCtime reading = ");
```

```
25.       Serial.println(photocellReading);     // the raw analog reading
26.
27.       // The brighter it is, the faster it blinks!
28.       digitalWrite(ledPin, HIGH);
29.       delay(photocellReading);
30.       digitalWrite(ledPin, LOW);
31.       delay(photocellReading);
32.     }
33.     delay(100);
34.   }
35.
36.   // Uses a digital pin to measure a resistor (like an FSR or photocell!)
37.   // We do this by having the resistor feed current into a capacitor and
38.   // counting how long it takes to get to Vcc/2 (for most arduinos, thats 2.5V)
39.   int RCtime(int RCpin) {
40.     int reading = 0;  // start with 0
41.
42.     // set the pin to an output and pull to LOW (ground)
43.     pinMode(RCpin, OUTPUT);
44.     digitalWrite(RCpin, LOW);
45.
46.     // Now set the pin to an input and...
47.     pinMode(RCpin, INPUT);
48.     while (digitalRead(RCpin) == LOW) { // count how long it takes to rise up to
    HIGH
49.       reading++;      // increment to keep track of time
50.
51.       if (reading == 30000) {
52.         // if we got this far, the resistance is so high
53.         // its likely that nothing is connected!
54.         break;          // leave the loop
55.       }
56.     }
57.     // OK either we maxed out at 30000 or hopefully got a reading, return the
    count
58.
59.     return reading;
60.   }
```

```
RCtime reading = 25
RCtime reading = 26
RCtime reading = 27
RCtime reading = 28
RCtime reading = 31
RCtime reading = 33
RCtime reading = 38
RCtime reading = 148
RCtime reading = 188
RCtime reading = 240
RCtime reading = 270
RCtime reading = 281
RCtime reading = 284
RCtime reading = 282
RCtime reading = 18
RCtime reading = 14
27
```

Figure 158: Output

Actuators

- DC motor

In this lesson, you will learn how to control a small DC motor using an Arduino and a transistor.

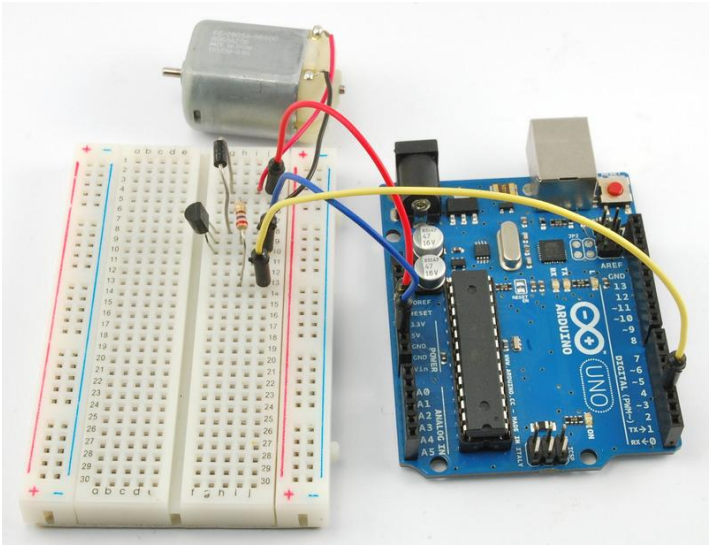


Figure 159: Dc Motor Interfacing

You will use an Arduino analog output (PWM) to control the speed of the motor by sending a number between 0 and 255 from the Serial Monitor.

Hands on: In this section , we will know how to connect different type of motors on a arduino board , program them targeting a given application .

Parts

To build the project described in this lesson, you will need the following parts.

Part

Qty

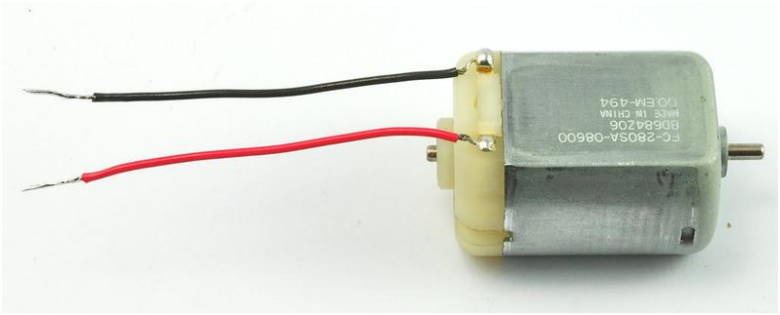


Figure 160:Small 6V DC Motor

1

Figure 161:PN2222 Transistor

1

Figure 162:1N4001 diode

1

Figure 163: 270 Ω Resistor (red, purple, brown stripes)

1

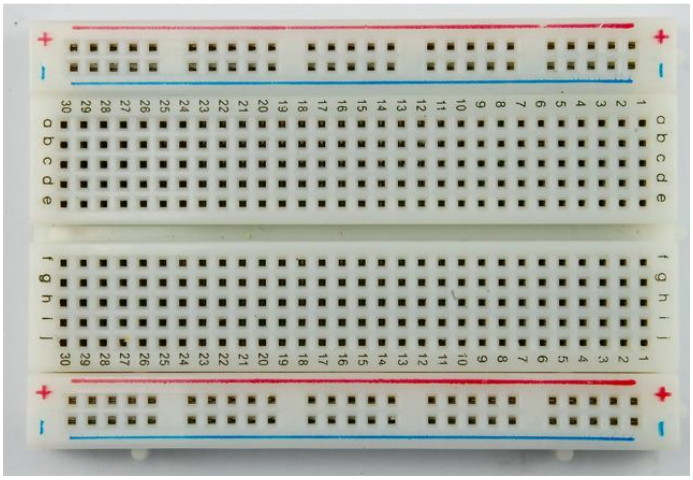


Figure 164: Half-size Breadboard

1

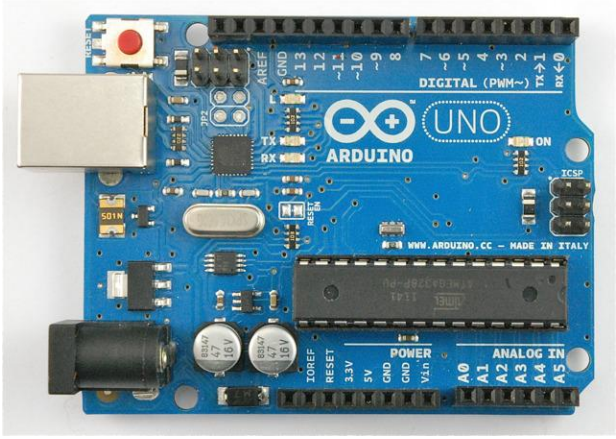


Figure 165:Arduino Uno R3

1

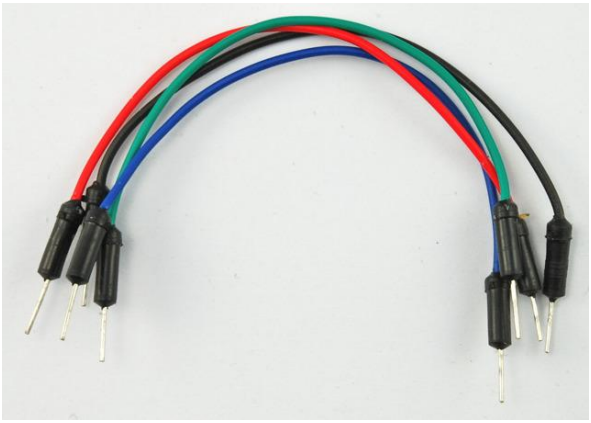


Figure 166:Jumper wire pack

Breadboard Layout

by Simon Monk

When you put together the breadboard, there are two things to look out for.

Firstly, make sure that the transistor is the right way around. The flat side of the transistor should be on the right-hand side of the breadboard.

Secondly the striped end of the diode should be towards the +5V power line - see the image below!

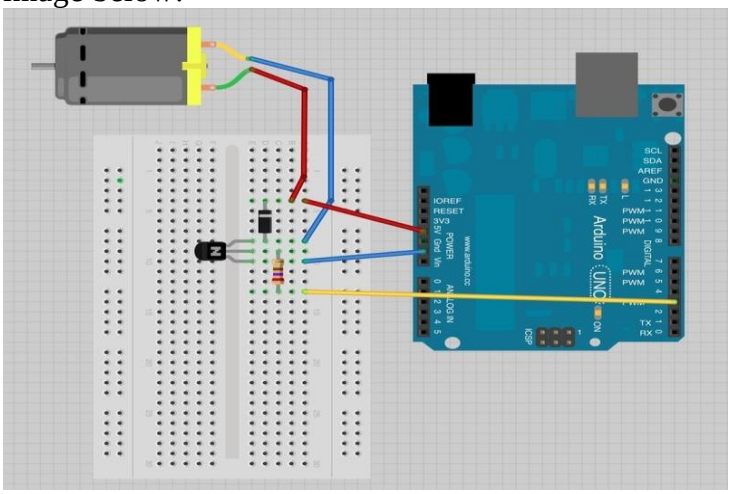


Figure 167: Dc Motor Interfacing

Arduino Code

Load up the following sketch onto your Arduino.

```
1.  /*
2.  Adafruit Arduino - Lesson 13. DC Motor
3.  */
4.
5.
6.  int motorPin = 3;
7.
8.  void setup()
9.  {
10.   pinMode(motorPin, OUTPUT);
11.   Serial.begin(9600);
12.   while (! Serial);
13.   Serial.println("Speed 0 to 255");
14. }
15.
16.
17. void loop()
18. {
19.   if (Serial.available())
20.   {
21.     int speed = Serial.parseInt();
22.     if (speed >= 0 && speed <= 255)
23.     {
24.       analogWrite(motorPin, speed);
25.     }
26.   }
27. }
```

The transistor acts like a switch, controlling the power to the motor, Arduino pin 3 is used to turn the transistor on and off and is given the name 'motorPin' in the sketch.
When the sketch starts, it prompts you, to remind you that to control the speed of the motor you need to enter a value between 0 and 255 in the Serial Monitor.

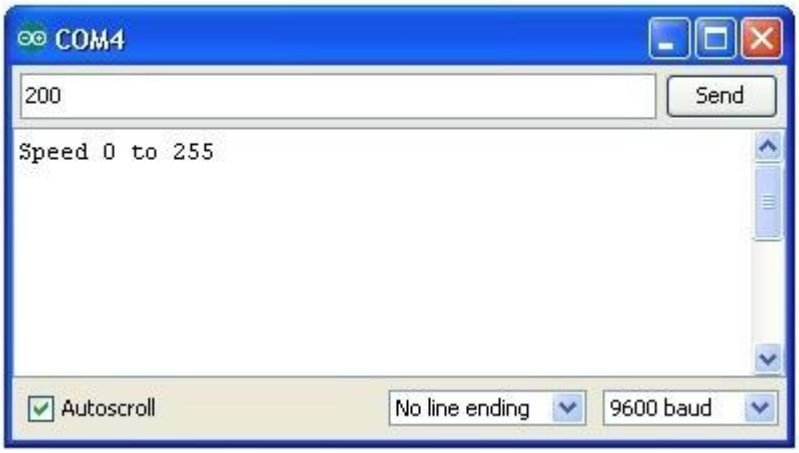


Figure 168:Output

In the 'loop' function, the command 'Serial.parseInt' is used to read the number entered as text in the Serial Monitor and convert it into an 'int'.
You could type any number here, so the 'if' statement on the next line only does an analog write with this number if the number is between 0 and 255.

Transistors

The small DC motor, is likely to use more power than an Arduino digital output can handle directly. If we tried to connect the motor straight to an Arduino pin, there is a good chance that it could damage the Arduino.
A small transistor like the PN2222 can be used as a switch that uses just a little current from the Arduino digital output to control the much bigger current of the motor.

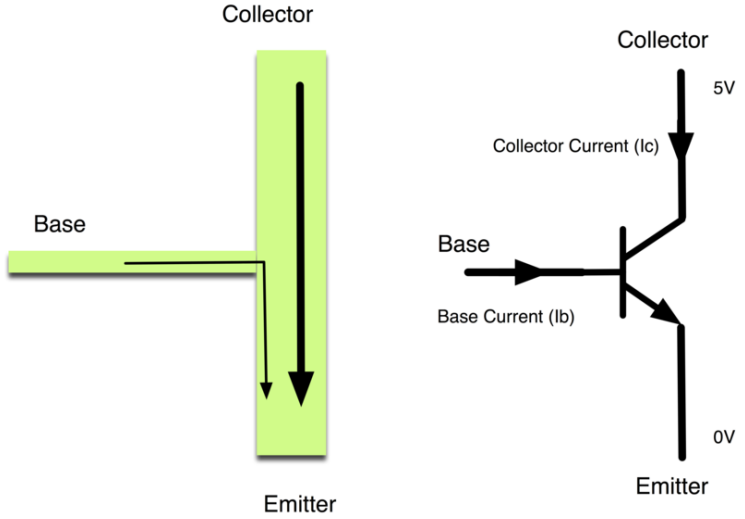


Figure 169: Current Flow Diagram

The transistor has three leads. Most of the electricity flows from the Collector to the Emitter, but this will only happen if a small amount is flowing into the Base connection. This small current is supplied by the Arduino digital output.
The diagram below is called a schematic diagram. Like a breadboard layout, it is a way of showing how the parts of an electronic project are connected together.

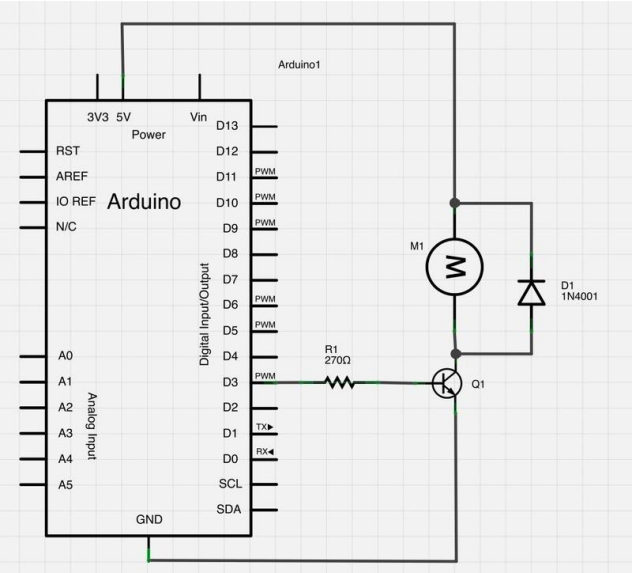


Figure 170: DC Motor Controlling Circuit

The pin D3 of the Arduino is connected to the resistor. Just like when using an LED, this limits the current flowing into the transistor through the base.
There is a diode connected across the connections of the motor. Diodes only allow electricity to flow in one direction (the direction of their arrow).

When you turn the power off to a motor, you get a negative spike of voltage, that can damage your Arduino or the transistor. The diode protects against this, by shorting out any such reverse current from the motor.

Servo motor

Servo Motor Control with an Arduino

You can connect small servo motors directly to an Arduino to control the shaft position very precisely.

Because servo motors use feedback to determine the position of the shaft, you can control that position very precisely. As a result, servo motors are used to control the position of objects, rotate objects, move legs, arms or hands of robots, move sensors etc. with high precision. Servo motors are small in size, and because they have built-in circuitry to control their movement, they can be connected directly to an Arduino.

Most servo motors have the following three connections:

- Black/Brown ground wire.
- Red power wire (around 5V).
- Yellow or White PWM wire.

In this experiment, we will connect the power and ground pins directly to the Arduino 5V and GND pins. The PWM input will be connected to one of the Arduino's digital output pins.

Experiment 1

Hardware Required

- 1 x TowerPro SG90 servo motor
- 1 x Arduino Mega2560
- 3 x jumper wires

Wiring Diagram

The best thing about a servo motor is that it can be connected directly to an Arduino. Connect to the motor to the Arduino as shown in the table below:

- Servo red wire – 5V pin Arduino
- Servo brown wire – Ground pin Arduino
- Servo yellow wire – PWM(9) pin Arduino

Caution: Do not try to rotate the servo motor by hand, as you may damage the motor.

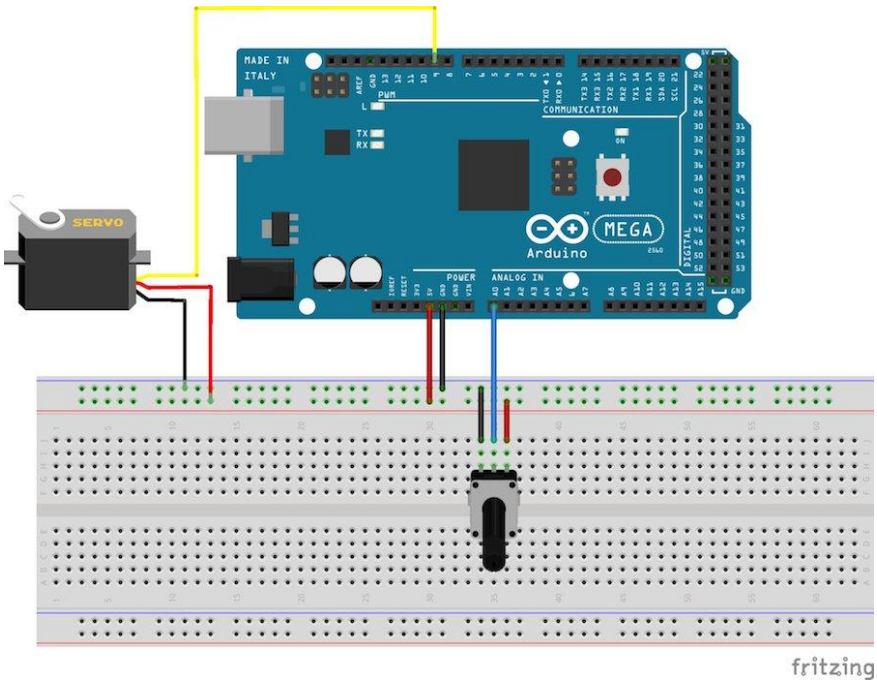


Figure 171: Pot Controlling a Servo Motor

Code

When the program starts running, the servo motor will rotate slowly from 0 degrees to 180 degrees, one degree at a time. When the motor has rotated 180 degrees, it will begin to rotate in the other direction until it returns to the home position.

```
#include          //Servo library

Servo servo_test;                    //initialize a servo object for the connected servo

int angle = 0;

void setup()
{
  servo_test.attach(9);              // attach the signal pin of servo to pin9 of arduino
}

void loop()
{
  for(angle = 0; angle < 180; angle += 1)          // command to move from 0 degrees to 180 degrees
  {
    servo_test.write(angle);         //command to rotate the servo to the specified angle
```

```
  delay(15);
 }
```

```
 delay(1000);

 for(angle = 180; angle>=1; angle-=5)     // command to move from 180 degrees to 0 degrees
 {
  servo_test.write(angle);              //command to rotate the servo to the specified angle
  delay(5);
 }

  delay(1000);
}
```

Experiment 2

This experiment is essentially the same as Experiment 1, except that we have added a potentiometer for position control. The Arduino will read the voltage on the middle pin of the potentiometer and adjust the position of the servo motor shaft.

Hardware Required

- 1 x TowerPro SG90 servo motor
- 1 x Arduino Mega2560
- 1 x 20kΩ potentiometer
- 1 x breadboard
- 6 x jumper wires

Wiring Diagram

Connect the circuit as show in the figure below:

- Servo red wire – 5V pin Arduino
- Servo brown wire – Ground pin Arduino
- Servo yellow wire – PWM(9) pin Arduino
- Potentiometer pin 1 - 5V pin Arduino
- Potentiometer pin 3 - Ground pin Arduino
- Potentiometer pin 2 – Analog In (A0) pin Arduino

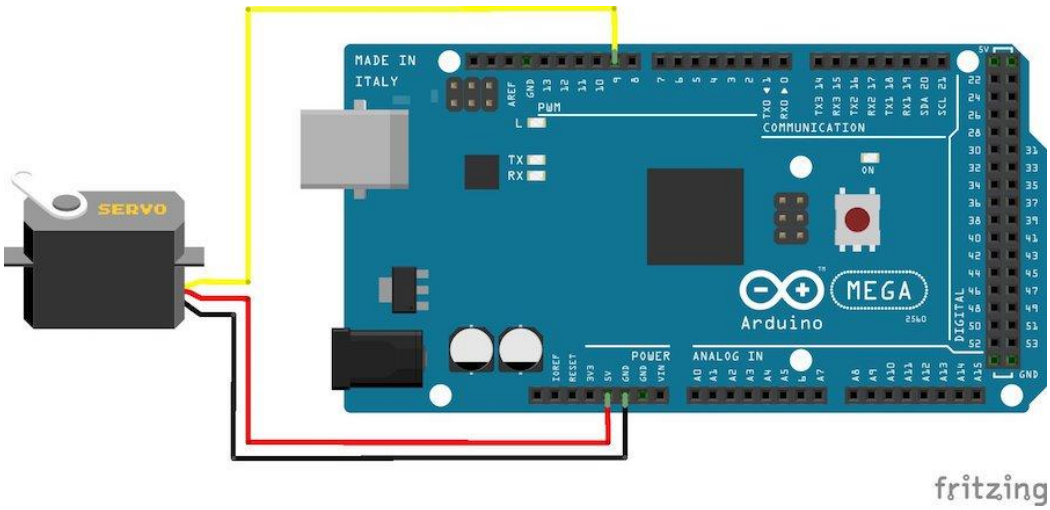


Figure 172: Servo motor interfacing

Code

Once the program is started, rotating the potentiometer should cause the shaft of the servo motor to rotate.

```
#include        //Servo library

Servo servo_test;               //initialize a servo object for the connected servo

int angle = 0;
int potentio = A0;      // initialize the A0analog pin for potentiometer



void setup()
{
 servo_test.attach(9);          // attach the signal pin of servo to pin9 of arduino
}

void loop()
{
 angle = analogRead(potentio);          // reading the potentiometer value between 0 and 1023
 angle = map(angle, 0, 1023, 0, 179);     // scaling the potentiometer value to angle value for servo between 0 and 180)
 servo_test.write(angle);               //command to rotate the servo to the specified angle
 delay(5);
}
```

Display

- LCD

The Liquid Crystal Library allows you to control LCD displays that are compatible with the Hitachi HD44780 driver. There are many of them out there, and you can usually tell them by the 16-pin interface.

This example sketch shows how to use the display() and noDisplay() methods to turn on and off the display. The text to be displayed will still be preserved when you use noDisplay() so it's a quick way to blank the display without losing everything on it.

Hardware Required

- Arduino or Genuino Board
- LCD Screen (compatible with Hitachi HD44780 driver)
- pin headers to solder to the LCD display pins
- 10k ohm potentiometer
- 220 ohm resistor
- hook-up wires
- breadboard

Circuit

Before wiring the LCD screen to your Arduino or Genuino board we suggest to solder a pin header strip to the 14 (or 16) pin count connector of the LCD screen, as you can see in the image above.
To wire your LCD screen to your board, connect the following pins:

- LCD RS pin to digital pin 12
- LCD Enable pin to digital pin 11
- LCD D4 pin to digital pin 5
- LCD D5 pin to digital pin 4
- LCD D6 pin to digital pin 3
- LCD D7 pin to digital pin 2

Additionally, wire a 10k pot to +5V and GND, with it's wiper (output) to LCD screens VO pin (pin3). A 220 ohm resistor is used to power the backlight of the display, usually on pin 15 and 16 of the LCD connector

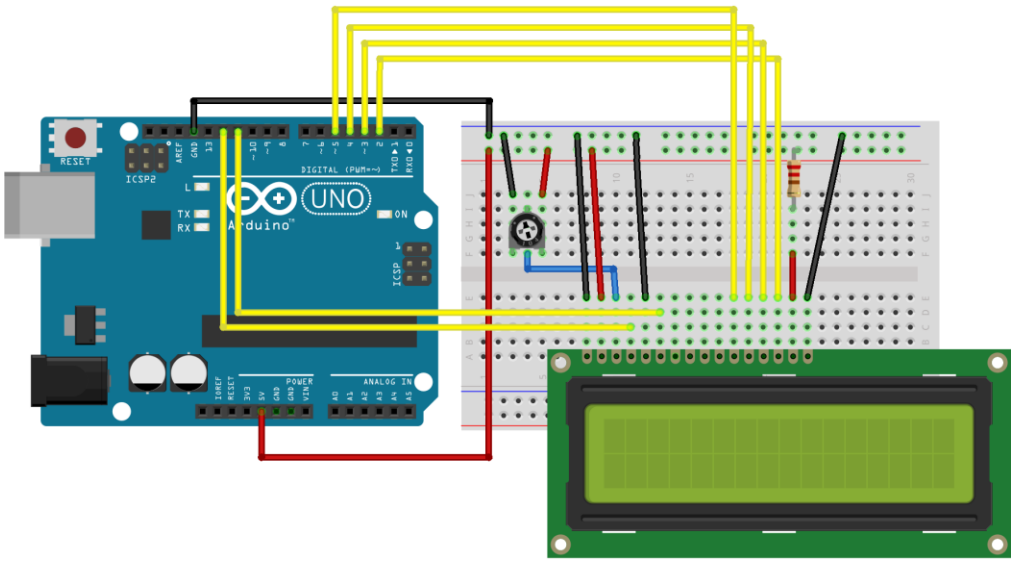


Figure 173: LCD Interfacing

Schematic

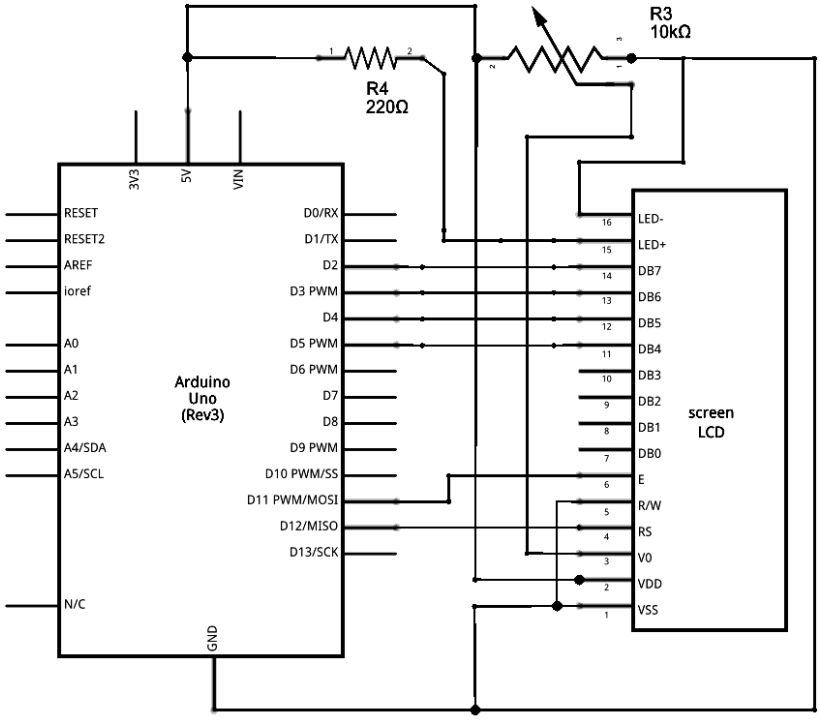


Figure 174: Arduino and LCD Circuit Diagram

Code

```
/*
  LiquidCrystal Library - display() and noDisplay()


// include the library code:
#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
```

```
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
}

void loop() {
  // Turn off the display:
  lcd.noDisplay();
  delay(500);
  // Turn on the display:
  lcd.display();
  delay(500);
}
```

Wireless communication

- GPS

  This section shows how to use the NEO-6M GPS module with the Arduino to get GPS data. GPS stands for *Global Positioning System* and can be used to determine position, time, and speed if you're travelling.

Figure 175: arduino and GPS Module

You'll learn how to:

- Wire the NEO-6M GPS module to the Arduino UNO
- Get raw GPS data
- Parse raw data to obtain selected and readable GPS information

- Get location

Introducing the NEO-6M GPS Module

The NEO-6M GPS module is shown in the figure below. It comes with an external antenna, and does't come with header pins. So, you'll need to get and solder some.

Figure 176: GPS Module

- This module has an external antenna and built-in EEPROM.
- Interface: RS232 TTL
- Power supply: 3V to 5V
- Default baudrate: 9600 bps
- Works with standard NMEA sentences

The NEO-6M GPS module is also compatible with other microcontroller boards. To learn how to use the NEO-6M GPS module with the Raspberry Pi, you can read: Email Alert System on Location Change with Raspberry Pi and GPS Module.

Pin Wiring

The NEO-6M GPS module has four pins: VCC, RX, TX, and GND. The module communicates with the Arduino via serial communication using the TX and RX pins, so the wiring couldn't be simpler:

NEO-6M GPS Module	**Wiring to Arduino UNO**

VCC	5V
RX	TX pin defined in the software serial
TX	RX pin defined in the software serial
GND	GND

Getting GPS Raw Data

To get raw GPS data you just need to start a serial communication with the GPS module using Software Serial. Continue reading to see how to do that.

Parts Required

For testing this example you'll need the following parts:

- Arduino –
- NEO-6M GPS module
- Jumper wires

Schematics

Wire the NEO-6M GPS module to your Arduino by following the schematic below.

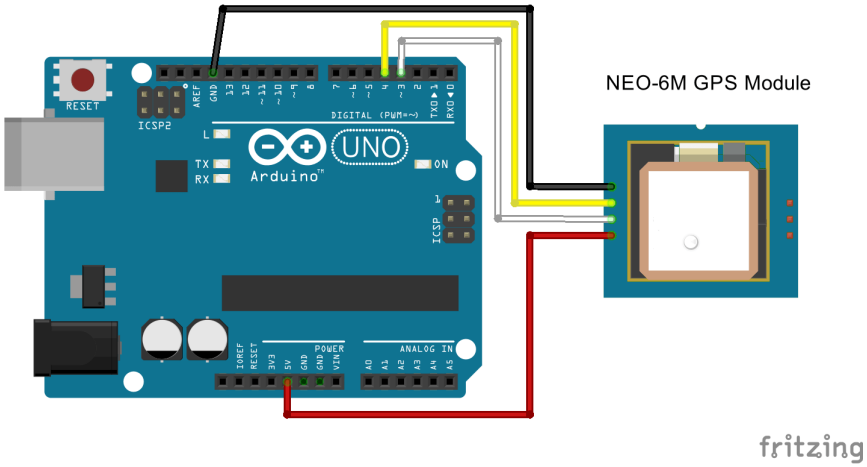


Figure 177: Arduino and GPS connection

- The module GND pin is connected to Arduino GND pin
- The module RX pin is connected to Arduino pin 3
- The module TX pin is connected to Arduino pin 4
- The module VCC pin is connected to Arduino 5V pin

Code

Copy the following code to your Arduino IDE and upload it to your Arduino board.

```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
 */

#include <SoftwareSerial.h>

// The serial connection to the GPS module
SoftwareSerial ss(4, 3);

void setup(){
  Serial.begin(9600);
  ss.begin(9600);
}

void loop(){
  while (ss.available() > 0){
    // get the byte data from the GPS
    byte gpsData = ss.read();
    Serial.write(gpsData);
  }
}
```

This sketch assumes you are using pins 4 and 3 as RX and TX serial pins to establish serial communication with the GPS module. If you're using other pins you should edit that on the following line:

SoftwareSerial ss(**4**, **3**);

Also, if your module uses a different default baud rate than 9600 bps, you should modify the code on the following line:

ss.begin(**9600**);

This sketch listen to the GPS serial port, and when data is received from the module, it is sent to the serial monitor.

```
while (ss.available() > 0){
  // get the byte data from the GPS
  byte gpsData = ss.read();
  Serial.write(gpsData);
}
```

Open the Serial Monitor at a baud rate of 9600.

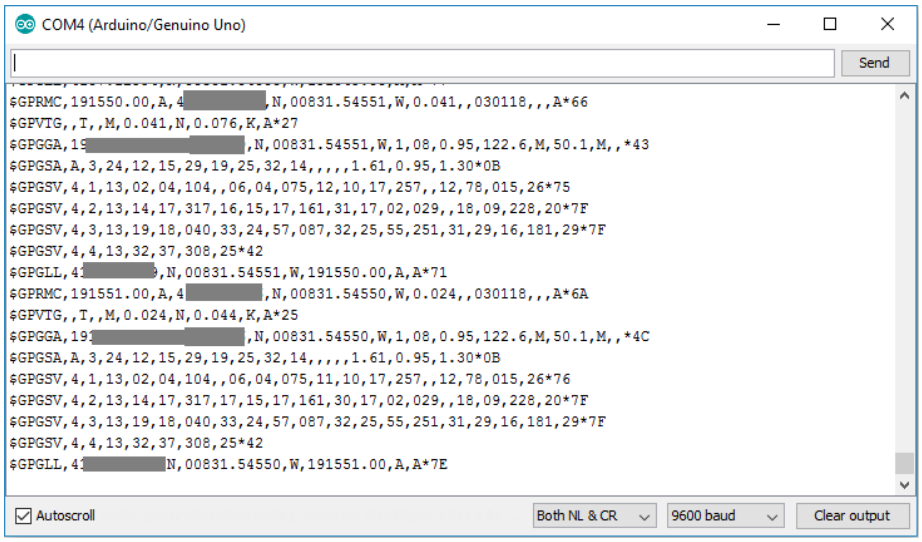


Figure 178: OutPut On Serial Monitor

You should get a bunch of information in the GPS standard language, NMEA. Each line you get int the serial monitor is an NMEA sentence.

NMEA stands for National Marine Electronics Association, and in the world of GPS, it is a standard data format supported by GPS manufacturers.

Understanding NMEA Sentences

NMEA sentences start with the $ character, and each data field is separated by a comma.

$GPGGA,110617.00,41XX.XXXXX,N,00831.54761,W,1,05,2.68,129.0,M,50.1,M,,*42
$GPGSA,A,3,06,09,30,07,23,,,,,,,,4.43,2.68,3.53*02
$GPGSV,3,1,11,02,48,298,24,03,05,101,24,05,17,292,20,06,71,227,30*7C
$GPGSV,3,2,11,07,47,138,33,09,64,044,28,17,01,199,,19,13,214,,*7C
$GPGSV,3,3,11,23,29,054,29,29,01,335,,30,29,167,33*4E
$GPGLL,41XX.XXXXX,N,00831.54761,W,110617.00,A,A*70
$GPRMC,110618.00,A,41XX.XXXXX,N,00831.54753,W,0.078,,030118,,,A*6A
$GPVTG,,T,,M,0.043,N,0.080,K,A*2C

There are different types of NMEA sentences. The type of message is indicated by the characters before the first comma.

The GP after the $ indicates it is a GPS position. The $GPGGA is the basic GPS NMEA message, that provides 3D location and accuracy data. In the following sentence:

$GPGGA,110617.00,41XX.XXXXX,N,00831.54761,W,1,05,2.68,129.0,M,50.1,M,,*42

- **110617** – represents the time at which the fix location was taken, 11:06:17 UTC
- **41XX.XXXXX,N** – latitude 41 deg XX.XXXXX' N
- **00831.54761,W** – Longitude 008 deg 31.54761′ W
- **1** – fix quality (0 = invalid; 1= GPS fix; 2 = DGPS fix; 3 = PPS fix; 4 = Real Time Kinematic; 5 = Float RTK; 6 = estimated (dead reckoning); 7 = Manual input mode; 8 = Simulation mode)
- **05** – number of satellites being tracked
- **2.68** – Horizontal dilution of position
- **129.0, M** – Altitude, in meters above the sea level
- **50.1, M** – Height of geoid (mean sea level) above WGS84 ellipsoid
- empty field – time in seconds since last DGPS update
- empty field – DGPS station ID number
- ***42** – the checksum data, always begins with *

The other NMEA sentences provide additional information:

- **$GPGSA** – GPS DOP and active satellites
- **$GPGSV** – Detailed GPS satellite information
- **$GPGLL** – Geographic Latitude and Longitude
- **$GPRMC** – Essential GPS pvt (position, velocity, time) data
- **$GPVTG** – Velocity made good

Parsing NMEA Sentences with TinyGPS++ Library

You can work with the raw data from the GPS, or you can convert those NMEA messages into a readable and useful format, by saving the characters sequences into variables. To do that, we're going to use the TinyGPS++ library.

Installing the TinyGPS++ Library

Follow the next steps to install the TinyGPS++ library in your Arduino IDE:

1. Download the TinyGPSPlus library. You should have a .zip folder in your Downloads folder
2. Unzip the .zip folder and you should get TinyGPSPlus-master folder
3. Rename your folder from ~~TinyGPSPlus-master~~ to TinyGPSPlus
4. Move the TinyGPSPlus folder to your Arduino IDE installation libraries folder
5. Finally, re-open your Arduino IDE

The library provides several examples on how to use it. In your Arduino IDE, you just need to go to **File** > **Examples** > **TinyGPS++**, and choose from the examples provided.
Note: the examples provided in the library assume a baud rate of 4800 for the GPS module. You need to change that to 9600 if you're using the NEO-6M GPS module.

Getting Location Using the NEO-6M GPS Module and the TinyGPS++ Library

You can get the location in a format that is convenient and useful by using the TinyGPS++ library. Below, we provide a code to get the location from the GPS. This is a simplified version of one of the library examples.

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>

static const int RXPin = 4, TXPin = 3;
static const uint32_t GPSBaud = 9600;

// The TinyGPS++ object
TinyGPSPlus gps;

// The serial connection to the GPS device
SoftwareSerial ss(RXPin, TXPin);

void setup(){
  Serial.begin(9600);
  ss.begin(GPSBaud);
}

void loop(){
  // This sketch displays information every time a new sentence is correctly encoded.
  while (ss.available() > 0){
    gps.encode(ss.read());
    if (gps.location.isUpdated()){
      Serial.print("Latitude= ");
      Serial.print(gps.location.lat(), 6);
      Serial.print(" Longitude= ");
      Serial.println(gps.location.lng(), 6);
```

```
                                                                                    }
                                                                                    }
}
```

You start by importing the needed libraries: TinyGPSPlus and SoftwareSerial:

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
```

Then, you define the software serial RX and TX pins, as well as the GPS baud rate. If you are using other pins for software serial you need to change that here. Also, if your GPS module uses a different default baud rate, you should also modify that.

```
static const int RXPin = 4, TXPin = 3;
static const uint32_t GPSBaud = 9600;
```

Then, you create a TinyGPS++ object:

```
TinyGPSPlus gps;
```

And start a serial connection on the pins you've defined earlier

```
SoftwareSerial ss(RXPin, TXPin);
```

In the setup(), you initialize serial communication, both to see the readings on the serial monitor and to communicate with the GPS module.

```
void setup() {
  Serial.begin(9600);
  ss.begin(GPSBaud);
}
```

In the loop is where you request the information. To get TinyGPS++ to work, you have to repeatedly funnel the characters to it from the GPS module using the encode() method.

```
while (ss.available() > 0){
 gps.encode(ss.read());
```

Then, you can query the gps object to see if any data fields have been updated:

```
 if (gps.location.isUpdated()){
  Serial.print("Latitude="); Serial.print(gps.location.lat(), 6);
  Serial.print("Longitude="); Serial.println(gps.location.lng(), 6);
}
```

Getting the latitude and longitude is has simple has using **gps.location.lat()**, and **gps.location.lng()**, respectively.

Upload the code to your Arduino, and you should see the location displayed on the serial monitor. After uploading the code, wait a few minutes while the module adjusts the position to get a more accurate data.

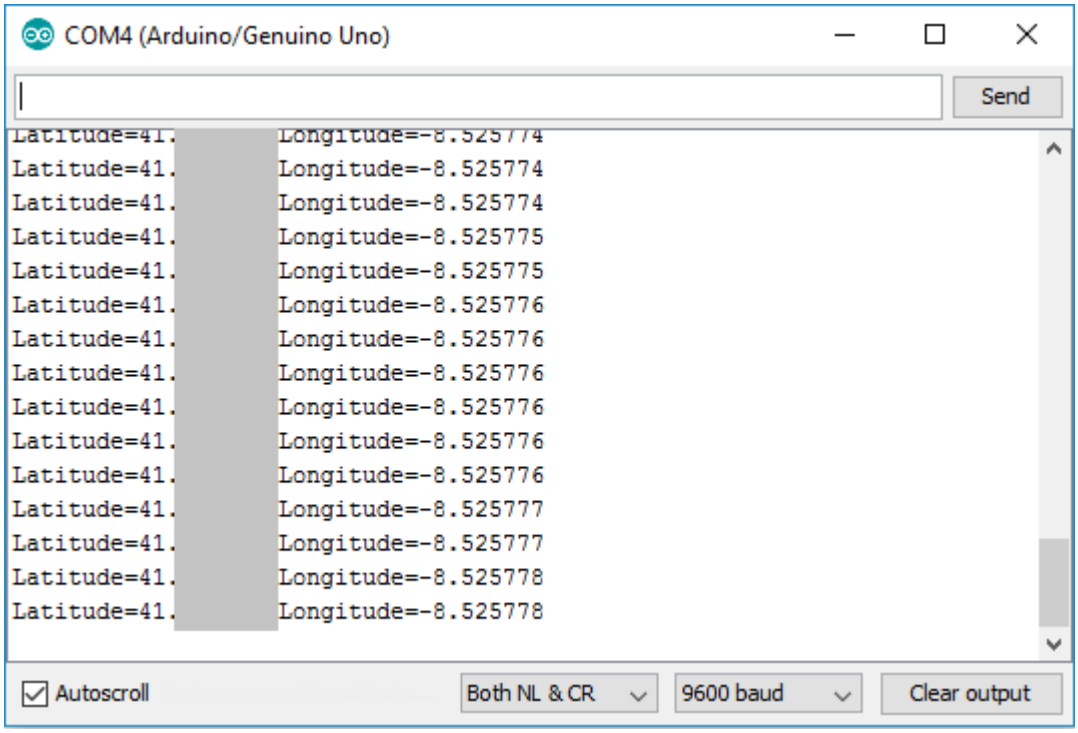


Figure 179:Output on Serial Monotor

Getting More GPS Information Using the TinyGPS++ Library

The TinyGPS++ library allows you to get way more information than just the location, and in a simple way. Besides the location, you can get:

- date
- time
- speed
- course
- altitude
- satellites
- hdop

The code below exemplifies how you can get all that information in a simple way.

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>

static const int RXPin = 4, TXPin = 3;
static const uint32_t GPSBaud = 9600;

// The TinyGPS++ object
TinyGPSPlus gps;

// The serial connection to the GPS device
SoftwareSerial ss(RXPin, TXPin);

void setup(){
  Serial.begin(9600);
  ss.begin(GPSBaud);
}

void loop(){
  // This sketch displays information every time a new sentence is correctly encoded.
  while (ss.available() > 0){
    gps.encode(ss.read());
    if (gps.location.isUpdated()){
      // Latitude in degrees (double)
      Serial.print("Latitude= ");
      Serial.print(gps.location.lat(), 6);
      // Longitude in degrees (double)
      Serial.print(" Longitude= ");
      Serial.println(gps.location.lng(), 6);

      // Raw latitude in whole degrees
      Serial.print("Raw latitude = ");
      Serial.print(gps.location.rawLat().negative ? "-" : "+");
      Serial.println(gps.location.rawLat().deg);
      // ... and billionths (u16/u32)
      Serial.println(gps.location.rawLat().billionths);

      // Raw longitude in whole degrees
```

```
            Serial.print("Raw longitude = ");
Serial.print(gps.location.rawLng().negative ? "-" : "+");
                Serial.println(gps.location.rawLng().deg);
        // ... and billionths (u16/u32)
            Serial.println(gps.location.rawLng().billionths);

    // Raw date in DDMMYY format (u32)
        Serial.print("Raw date DDMMYY = ");
                Serial.println(gps.date.value());

            // Year (2000+) (u16)
            Serial.print("Year = ");
                Serial.println(gps.date.year());
            // Month (1-12) (u8)
            Serial.print("Month = ");
                Serial.println(gps.date.month());
            // Day (1-31) (u8)
            Serial.print("Day = ");
                Serial.println(gps.date.day());

    // Raw time in HHMMSSCC format (u32)
    Serial.print("Raw time in HHMMSSCC = ");
                Serial.println(gps.time.value());

            // Hour (0-23) (u8)
            Serial.print("Hour = ");
                Serial.println(gps.time.hour());
            // Minute (0-59) (u8)
            Serial.print("Minute = ");
                Serial.println(gps.time.minute());
            // Second (0-59) (u8)
            Serial.print("Second = ");
                Serial.println(gps.time.second());
    // 100ths of a second (0-99) (u8)
            Serial.print("Centisecond = ");
                Serial.println(gps.time.centisecond());

// Raw speed in 100ths of a knot (i32)
    Serial.print("Raw speed in 100ths/knot = ");
```

```
                Serial.println(gps.speed.value());
        // Speed in knots (double)
        Serial.print("Speed in knots/h = ");
                Serial.println(gps.speed.knots());
    // Speed in miles per hour (double)
        Serial.print("Speed in miles/h = ");
                Serial.println(gps.speed.mph());
    // Speed in meters per second (double)
        Serial.print("Speed in m/s = ");
                Serial.println(gps.speed.mps());
    // Speed in kilometers per hour (double)
        Serial.print("Speed in km/h = ");
                Serial.println(gps.speed.kmph());

// Raw course in 100ths of a degree (i32)
    Serial.print("Raw course in degrees = ");
                Serial.println(gps.course.value());
        // Course in degrees (double)
        Serial.print("Course in degrees = ");
                Serial.println(gps.course.deg());

    // Raw altitude in centimeters (i32)
    Serial.print("Raw altitude in centimeters = ");
                Serial.println(gps.altitude.value());
        // Altitude in meters (double)
        Serial.print("Altitude in meters = ");
                Serial.println(gps.altitude.meters());
        // Altitude in miles (double)
        Serial.print("Altitude in miles = ");
                Serial.println(gps.altitude.miles());
        // Altitude in kilometers (double)
        Serial.print("Altitude in kilometers = ");
                Serial.println(gps.altitude.kilometers());
        // Altitude in feet (double)
        Serial.print("Altitude in feet = ");
                Serial.println(gps.altitude.feet());

    // Number of satellites in use (u32)
    Serial.print("Number os satellites in use = ");
```

```
                                        Serial.println(gps.satellites.value());

                    // Horizontal Dim. of Precision (100ths-i32)
                                    Serial.print("HDOP = ");
                                        Serial.println(gps.hdop.value());
                                                }
                                                }
}
```

- GSM

The Arduino GSM shield allows an Arduino board to connect to the internet, send and receive SMS, and make voice calls using the GSM library.

The shield will work with the Arduino Uno out of the box. The shield will work with the Mega, Mega ADK, Yun, and Leonardo boards with a minor modification. The Due is not supported at this time.

The GSM library is included with Arduino IDE 1.0.4 and later.

What is GSM

GSM is an international standard for mobile telephones. It is an acronym that stands for Global System for Mobile Communications. It is also sometimes referred to as 2G, as it is a second-generation cellular network.

To use GPRS for internet access, and for the Arduino to request or serve webpages, you need to obtain the Access Point Name (APN) and a username/password from the network operator. See the information in Connecting to the Internet for more information about using the data capabilities of the shield.

Among other things, GSM supports outgoing and incoming voice calls, Simple Message System (SMS or text messaging), and data communication (via GPRS).

The Arduino GSM shield is a a GSM modem. From the mobile operator perspective, the Arduino GSM shield looks just like a mobile phone. From the Arduino perspective, the Arduino GSM shield looks just like a modem.

What is GPRS

GPRS is a packet switching technology that stands for General Packet Radio Service. It can provide idealized data rates between 56-114 kbit per second.

A number of technologies such as SMS rely on GPRS to function. With the GSM shield, it is also possible to leverage the data communication to access the internet. Similar to the Ethernet and WiFi libraries, the GSM library allows the Arduino to act as a client or server, using http calls to send and receive web pages.

Network operator requirements

To access a network, you must have a subscription with a mobile phone operator (either prepaid or contract), a GSM compliant device like the GSM shield or mobile phone, and a Subscriber Identity Module (SIM) card. The network operator provides the SIM card, which contains information like the mobile number, and can store limited amounts of contacts and SMS messages.

To use GPRS for internet access, and for the Arduino to request or serve webpages, you need to obtain the Access Point Name (APN) and a username/password from the network operator. See the information in Connecting to the Internetfor more information about using the data capabilities of the shield.

SIM cards

In addition to the GSM shield and an Arduino, you need a SIM card. The SIM represents a contract with a communications provider. The communications provider selling you the SIM has to either provide GSM coverage where you are, or have a roaming agreement with a company providing GSM coverage in your location.

It's common for SIM cards to have a four-digit PIN number associated with them for security purposes. Keep note of this number, as it's necessary for connecting to a network. If you lose the PIN associated with your SIM card, you may need to contact your network operator to retrieve it. *Some SIM cards become locked if an incorrect PIN is entered too many times. If you're unsure of what the PIN is, look at the documentation that came with your SIM.*

Using a PUK (PIN Unlock Code), it is possible to reset a lost PIN with the GSM shield and an Arduino. The PUK number will come with your SIM card documentation.

Look at the PIN Management example in the "tools" folder, bundled with the GSM library for an example of how to manage your PIN number with the PUK.

There are a few different sizes of SIM cards; the GSM shield accepts cards in the mini-SIM format (25mm long and 15mm wide).

Notes on the Telefonica/Movilforum SIM included with the shield

The GSM shield comes bundled with a SIM from Telefonica/Movilforum that will work well for developing machine to machine (M2M) applications. It is not necessary to use this specific card with the shield. You may use any SIM that works on a network in your area.

The Movilforum SIM card includes a roaming plan. It can be used on any supported GSM network. There is coverage throughout the Americas and Europe for this SIM, check the Movilforum service availability page for specific countries that have supported networks.

Activation of the SIM is handled by Movilforum. Detailed instructions on how to register and activate your SIM online and add credit are included on a small pamphlet that comes with your shield. The SIM must be inserted into a powered GSM shield that is mounted on an Arduino for activation.

These SIM card come without a PIN, but it is possible to set one using the GSM library's GSMPIN class.

You cannot use the included SIM to place or receive voice calls.

You can only place and receive SMS with other SIMs on the Movilforum network.

It's not possible to create a server that accepts incoming requests from the public internet. However, the Movilforum SIM will accept incoming requests from other SIM cards on the Movilforum network.

For using the voice, and other functions of the shield, you'll need to find a different network provider and SIM. Operators will have different policies for their SIM cards, check with them directly to determine what types of connections are supported.

Connecting the Shield

If you are using an Arduino Uno, follow the instructions below. If you are using an Arduino Mega, Mega ADK, Yun, or Leonardo, you must follow these instructions. The GSM shield is not currently supported on the Due.

To use the shield, you'll need to insert a SIM card into the holder. Slide the metal bracket away from the edge of the shield and lift the cradle up.

Figure 180: SIMCard Slot

Insert the SIM in the plastic holder so the metal contacts are facing the shield, with the notch of the card at the top of the bracket.

Figure 181: GSM Shield onto arduino Uno

To upload sketches to the board, connect it to your computer with a USB cable and upload your sketch with the Arduino IDE. Once the sketch has been uploaded, you can disconnect the board from your computer and power it with an external power supply.

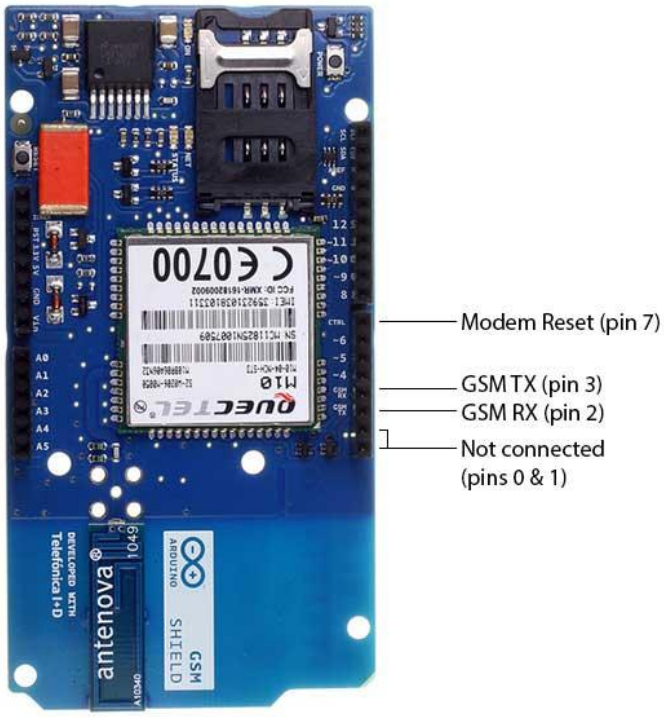


Figure 182: GSM Shield

Digital pins 2, 3 and 7 are reserved for communication between the Arduino and modem and cannot be used by your sketches. Communication between the moden and Arduino is handled by the Software Serial library on pins 2 and 3. Pin 7 is used for the modem reset.

When the yellow *status* LED turns on, it means the modem is powered, and you can try connecting to the network.

Developer versions of the GSM shield required you to press press the *Power* button on the shield for a few moments to turn the modem on. If you have an early version of the shield,

and it does not turn on automatically, you can solder a jumper to the CTRL/D7 pad on the reverse side of the board, and it will turn on when an attached Arduino receives power.

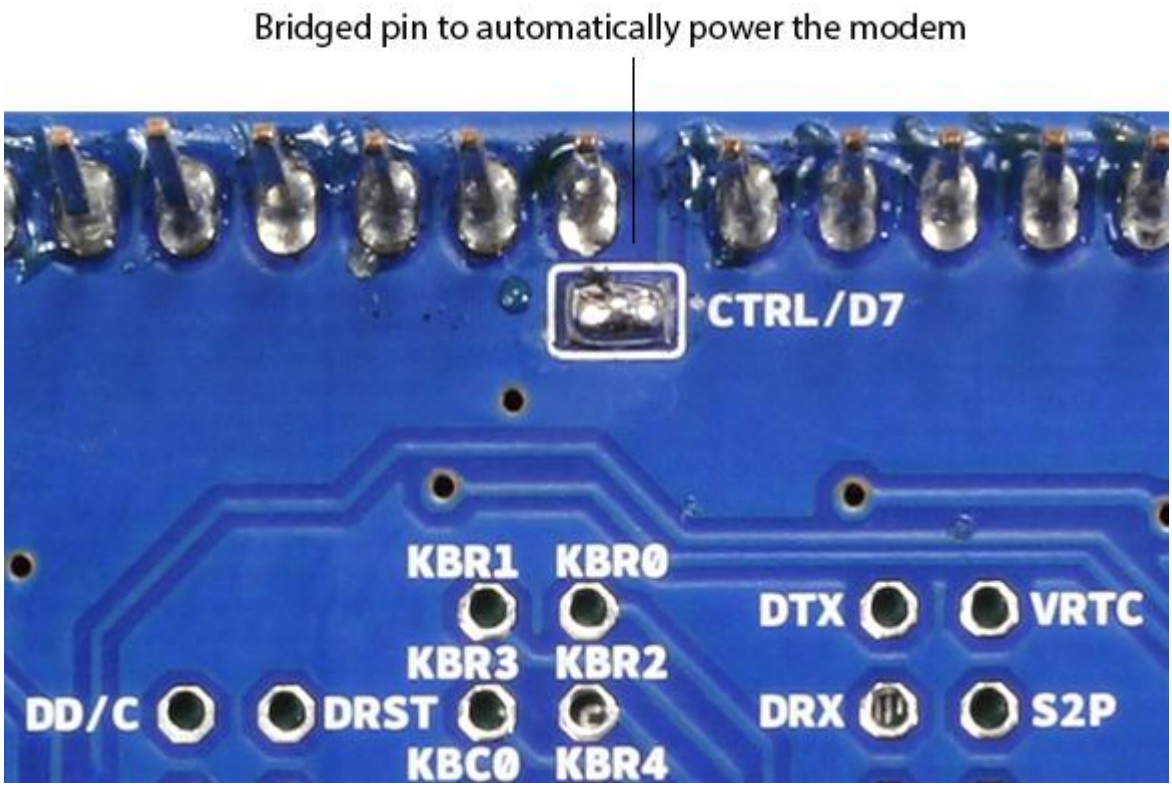


Figure 183:GSM

The shield should work in any area with GSM coverage. Before buying the shield please verify that there is this kind of coverage where you plan to use it.

GSM Library

The GSM library handles communication between Arduino and the GSM shield. The majority of functions are for managing data, voice, and SMS communication. There are also a number of utilities for managing information about the modem and the SIM card's PIN. See the library reference pages for more information and a complete set of examples.

Testing the modem and network connection

This sketch will check the modem's IMEI number. This number is unique to each modem, and is used to identify valid devices that can connect to a GSM network. Once the number has been read from the modem, the Arduino will print out the network carrier it is connected to, and the signal strength of the network over the serial port.

```
// import the GSM library
#include <GSM.h>

// PIN Number
#define PINNUMBER ""

// initialize the library instance
GSM gsmAccess(true); // include a 'true' parameter for debug enabled
```

```
GSMScanner scannerNetworks;
GSMModem modemTest;

// Save data variables
String IMEI = "";

// serial monitor result messages
String errortext = "ERROR";

void setup()
{
 // initialize serial communications
 Serial.begin(9600);
 Serial.println("GSM networks scanner");
 scannerNetworks.begin();

 // connection state
 boolean notConnected = true;

 // Start GSM shield
 // If your SIM has PIN, pass it as a parameter of begin() in quotes
 while(notConnected)
 {
 if(gsmAccess.begin(PINNUMBER)==GSM_READY)
 notConnected = false;
 else
 {
 Serial.println("Not connected");
 delay(1000);
 }
 }

 // get modem parameters
 // IMEI, modem unique identifier
 Serial.print("Modem IMEI: ");
 IMEI = modemTest.getIMEI();
 IMEI.replace("\n","");
 if(IMEI != NULL)
 Serial.println(IMEI);

 // currently connected carrier
 Serial.print("Current carrier: ");
 Serial.println(scannerNetworks.getCurrentCarrier());

 // returns strength and ber
 // signal strength in 0-31 scale. 31 means power > 51dBm
 // BER is the Bit Error Rate. 0-7 scale. 99=not detectable
 Serial.print("Signal Strength: ");
 Serial.print(scannerNetworks.getSignalStrength());
 Serial.println(" [0-31]");
}
```

```
void loop()
{
 // scan for existing networks, displays a list of networks
 Serial.println("Scanning available networks. May take some seconds.");

 Serial.println(scannerNetworks.readNetworks());

 // currently connected carrier
 Serial.print("Current carrier: ");
 Serial.println(scannerNetworks.getCurrentCarrier());

 // returns strength and ber
 // signal strength in 0-31 scale. 31 means power > 51dBm
 // BER is the Bit Error Rate. 0-7 scale. 99=not detectable
 Serial.print("Signal Strength: ");
 Serial.print(scannerNetworks.getSignalStrength());
 Serial.println(" [0-31]");

}
```

Sending a SMS message

Once you have connected to your network with the sketch above, you can test some of the other functionality of the board. This sketch will connect to a GSM network and send a SMS message to a phone number of your choosing.

```
#include <GSM.h>

#define PINNUMBER ""

// initialize the library instance
GSM gsmAccess; // include a 'true' parameter for debug enabled
GSM_SMS sms;

// char array of the telephone number to send SMS
// change the number 1-212-555-1212 to a number
// you have access to
char remoteNumber[20]= "12125551212";

// char array of the message
char txtMsg[200]="Test";

void setup()
{
 // initialize serial communications
 Serial.begin(9600);

 Serial.println("SMS Messages Sender");

 // connection state
```

```
 boolean notConnected = true;

 // Start GSM shield
 // If your SIM has PIN, pass it as a parameter of begin() in quotes
 while(notConnected)
 {
                                    if(gsmAccess.begin(PINNUMBER)==GSM_READY)
                                                        notConnected = false;
                                                                    else
                                                                        {
                                        Serial.println("Not connected");
                                                                    delay(1000);
                                                                        }
 }
 Serial.println("GSM initialized");
                                                                     sendSMS();
}

void loop()
{
// nothing to see here
}

void sendSMS(){

 Serial.print("Message to mobile number: ");
 Serial.println(remoteNumber);

 // sms text
 Serial.println("SENDING");
 Serial.println();
 Serial.println("Message:");
 Serial.println(txtMsg);

 // send the message
                                        sms.beginSMS(remoteNumber);
                                                        sms.print(txtMsg);
                                                          sms.endSMS();
 Serial.println("\nCOMPLETE!\n");
}
```

Connecting to the internet

In addition to the SIM card and a data plan, you will need some additional information from your cellular provider to connect to the internet. Every cellular provider has an Access Point Name (APN) that serves as a bridge between the cellular network and the internet. Sometimes, there is a username and password associated with the connection point. For example, the Movilforum APN is sm2ms.movilforum.es, but it has no password or login name.

The sketch below will connect to arduino.cc/latest.txt and print out its contents.

NB: Some network operators block incoming IP traffic. You should be able to run client functions, such as the sketch below, with no issues.

```
// include the GSM library
#include <GSM.h>

// PIN number if necessary
#define PINNUMBER ""

// APN information obrained from your network provider
#define GPRS_APN "GPRS_APN" // replace with your GPRS APN
#define GPRS_LOGIN "login" // replace with your GPRS login
#define GPRS_PASSWORD "password" // replace with your GPRS password

// initialize the library instances
GSMClient client;
GPRS gprs;
GSM gsmAccess;

// This example downloads the URL "http://arduino.cc/latest.txt"

char server[] = "arduino.cc"; // the base URL
char path[] = "/latest.txt"; // the path
int port = 80; // the port, 80 for HTTP

void setup()
{
 // initialize serial communications
 Serial.begin(9600);
 Serial.println("Starting Arduino web client.");
 // connection state
 boolean notConnected = true;

 // Start GSM shield
 // pass the PIN of your SIM as a parameter of gsmAccess.begin()
 while(notConnected)
 {
 if((gsmAccess.begin(PINNUMBER)==GSM_READY) &

 (gprs.attachGPRS(GPRS_APN, GPRS_LOGIN, GPRS_PASSWORD)==GPRS_READY))
 notConnected = false;
 else
 {
 Serial.println("Not connected");
 delay(1000);
 }
 }

 Serial.println("connecting...");

 // if you get a connection, report back via serial:
```

```
 if (client.connect(server, port))
 {
                                                    Serial.println("connected");
            // Make a HTTP request:
                                                client.print("GET ");
                                                client.print(path);
                                            client.println(" HTTP/1.0");
                                                client.println();
 }
 else
 {
      // if you didn't get a connection to the server:
                                        Serial.println("connection failed");
 }
}

void loop()
{
 // if there are incoming bytes available
 // from the server, read them and print them:
 if (client.available())
 {
                                                char c = client.read();
                                                Serial.print(c);
 }

 // if the server's disconnected, stop the client:
 if (!client.available() && !client.connected())
 {
                                                Serial.println();
                                        Serial.println("disconnecting.");
                                                client.stop();

            // do nothing forevermore:
                                                for(;;)
                                                ;
 }
}
```

Making voice calls

Through the modem, it is possible to make voice calls. In order to speak to and hear the other party, you will need to add a speaker and microphone.

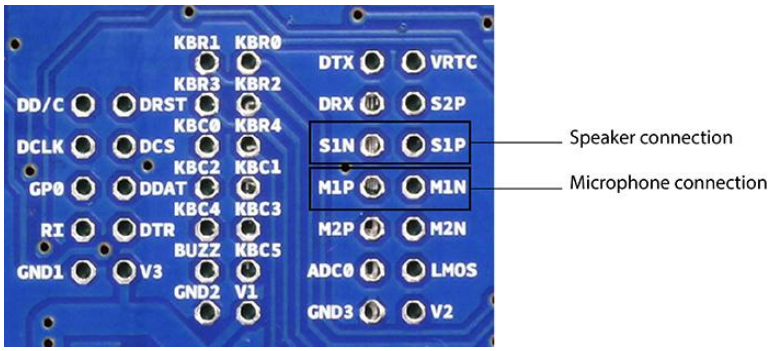
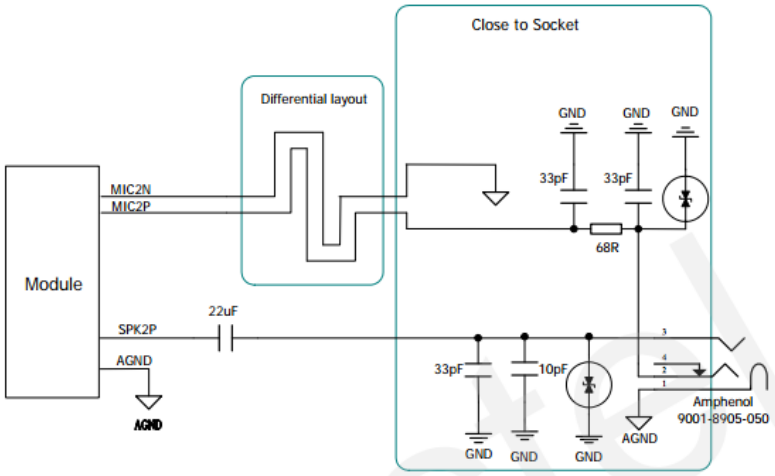


Figure 184: GSM shield

On the underside of the shield, there are through-holes labeled M1P and M1N. These are the positive and negative voice input pins for a microphone. The through-holes labeled S1P and S1N are the positive and negative voice output pins, to which you need to connect a speaker.

3.9.3 Earphone interface configuration



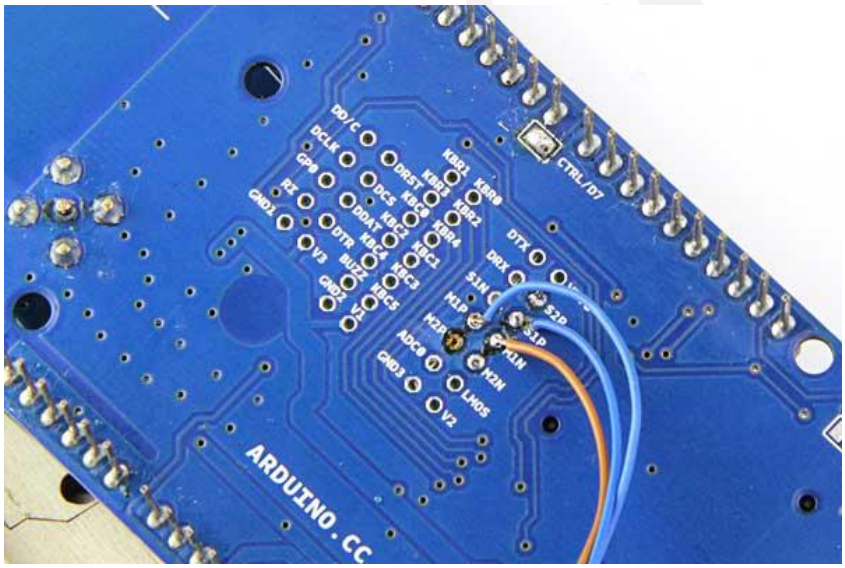


Figure 185:GSM

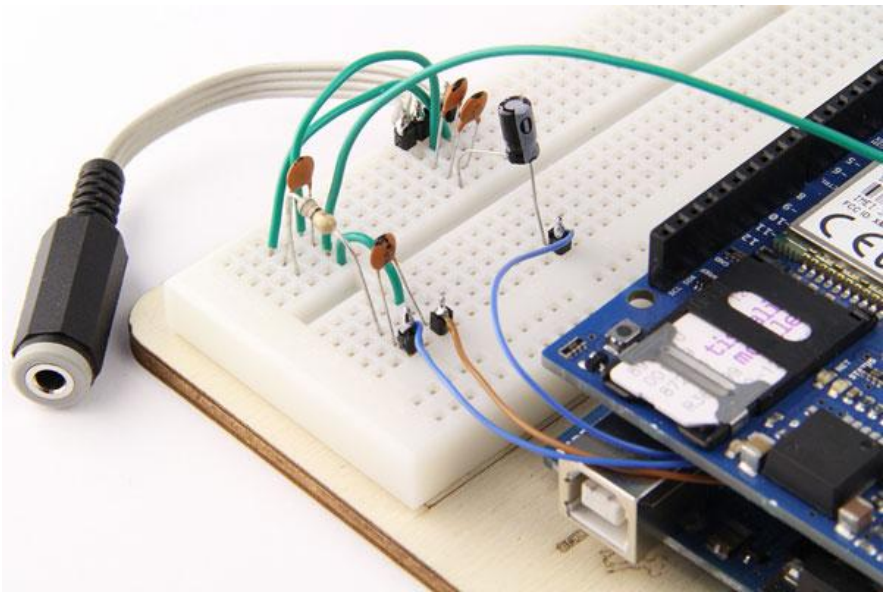


Figure 186: Arduino and GSM Shield

The following sketch allows you to place a voice call. Using the serial monitor, you can enter the remote phone number and terminate the call. When you see the READY message, type a phone number. Make sure the serial monitor is set to send a just newline when you press return.

```
#include <GSM.h>

// PIN Number
#define PINNUMBER ""

// initialize the library instance
GSM gsmAccess; // include a 'true' parameter for debug enabled
GSMVoiceCall vcs;

String remoteNumber = ""; // the number you will call
char charbuffer[20];

void setup()
{

 // initialize serial communications
 Serial.begin(9600);

 Serial.println("Make Voice Call");

 // connection state
 boolean notConnected = true;

 // Start GSM shield
 // If your SIM has PIN, pass it as a parameter of begin() in quotes
 while(notConnected)
 {
                    if(gsmAccess.begin(PINNUMBER)==GSM_READY)
                                notConnected = false;
```

```
                                                                    else
                                                                    {
                                                Serial.println("Not                 connected");
                                                                    delay(1000);
                                                                    }
 }

 Serial.println("GSM                                                initialized.");
 Serial.println("Enter          phone          number          to          call.");

}

void loop()
{

 // add any incoming characters to the String:
 while (Serial.available() > 0)
 {
                                                                    char inChar = Serial.read();
     // if it's a newline, that means you should make the call:
                                                                    if (inChar == '\n')
                                                                    {
          // make sure the phone number is not too long:
                                                                    if (remoteNumber.length() < 20)
                                                                    {
                              // show the number you're calling:
                                                Serial.print("Calling          to          :          ");
                                                                    Serial.println(remoteNumber);
                                                                    Serial.println();

                                  // Call the remote number
                                                                    remoteNumber.toCharArray(charbuffer, 20);


            // Check if the receiving end has picked up the call
                                                                    if(vcs.voiceCall(charbuffer))
                                                                    {
                                    Serial.println("Call Established. Enter line to end");
                                  // Wait for some input from the line
          while(Serial.read()!='\n' && (vcs.getvoiceCallStatus()==TALKING));
                                                  // And hang up
                                                                    vcs.hangCall();
                                                                    }
                                                Serial.println("Call                Finished");
                                                                    remoteNumber="";
                                    Serial.println("Enter phone number to call.");
                                                                    }
                                                                    else
                                                                    {
        Serial.println("That's too long for a phone number. I'm forgetting it");
                                                                    remoteNumber = "";
```

```
                                                                    }
                                                                  }
                                                                else
                                                                  {
            // add the latest character to the message to send:
                                                                if(inChar!='\r')
                                                              remoteNumber += inChar;
                                                                  }
  }
}
```

- Bluetooth
- Infrared

Mini projects

- Home automation
- Car speed sensor
- Traffic light control

Hands on: We will learn how to use different wireless communication module , we will do different mini projects like Home automation, car speed sensing, traffic sign control, especially those which are direcltly applied to automobile,.

Chap5: Electronic engine control

Electronic Control Module (ECM)

In automotive electronics, an electronic control module (ECM), also called a control unit or control module is an embedded system that controls one or more of the electrical subsystems in a vehicle. Some of which are as mentioned.

 Engine Control Unit, now even referred to as a Powertrain Control Module (PCM). This is explained in detail below.

 Transmission Control Unit (TCU): This is common in automatic transmission which facilitates gear changes and various gear ratios in automatic/semi-automatic transmission system.

 Telephone Control Unit (TCU): deals with telematics and telemetry (see chapter 12).

 Man Machine Interface (MMI): Deals with the ergonomical aspects of the vehicle.

 Door Control unit: deals with aspects such as door locking, child safe door locking etc.

 Seat Control Unit: deals with automatic seat positioning as per driver needs in case of multiple driver driven vehicle (one car may be driven by more than one people; car remembers this and adjusts driver seat accordingly). It also in-cludes seat belt warning system etc.

 Climate Control Unit: adjusts the climate inside the car in context to temperature and humidity. Controls operation of air conditioner and car heater in conjunction to the set value.

Of these, some are of more importance than the others. In the following articles, and even following chapters, these shall be dealt with in detail

Engine control Unit (ECU)

An engine control unit (ECU) is an electronic control unit which controls various aspects of an internal combustion engines operation. The simplest ECUs control only the quantity of fuel injected into each cylinder each engine cycle. More advanced ECUs found on most modern cars also control the ignition timing, variable valve timing (VVT), the level of boost maintained by the turbocharger (in turbocharged cars), and control other peripherals.

ECUs determine the quantity of fuel, ignition timing and other parameters by monitoring the engine through sen-sors. These can include, MAP sensor, throttle position sensor, air temperature sensor, oxygen sensor and many others. Often this is done using a control loop.

> Before ECUs most engine parameters were fixed. The quantity of fuel per cylinder per engine cycle was deter-mined by a carburetor or injector pump.

Operating modes

There are 2 main types of operating modes of the ECM. One is open loop and other is closed loop. As the name suggests, closed loop systems are more adaptive than open loop system because of the simple fact that there is a dy-namic behavior in the operation which makes it all the more reliable. They are explained below with suitable examples

Open loop

In this case, there is no feedback loop back to the input of the electronic control unit. An example of this system is **cruise control**. In this case, the car can cruise at a set speed without the driver having to press the accelerator pedal. The ECM shall monitor the car speed with the set speed, and in turn make changes in the engine intake parameters such as air mass, air pressure, fuel mass, fuel pressure etc and vary these parameters to maintain the 2 values of rec-orded speed and set speed in close allowable tolerances.

Closed loop

The closed loop function incorporates a feedback loop which monitors and evaluates performance of the system not only with the standard set values but also with the dynamic values that are recorded in conjunction with the out-put. An example of this kind of a system is **Adaptable cruise control.** Not only does the ECM measure the speed of the car and compare with the set speed. It also senses the distance of the car ahead of you and even slows the car down to maintain a safe distance.

Terminologies associated with ECM

In the ECM, there are various terminologies associated. We shall discuss the terms used in conjunction to the ECU as it is the most important aspect of the ECM.

ECU Algorithm

An ECU algorithm is a complex set of instructions based on which the ECU works. It is a set of instructions which gives the exact detail of the way the ECU should act in all possible engine situations. A complex ECU shall have an algo-rithm that if printed can take thousands of pages! Its complexity is magnified by the fact that there are more inputs and more outputs to the ECU day by day as is seen in newer high end cars. An ECU algorithm takes care of each and every input to the ECU, and how is the ECU supposed to act, i.e. which outputs to give to the actuators is stored in the algo-rithms. These are locked and in most cases are a deal of high amount of confidentiality of companies.

Engine maps

Engine maps are a set of graphs on which an engine works. The task of the ECU algorithm is to make sure that the operating point of the engine always coincides with these graphs. For example, there will be an ideal value for every engine speed at every engine load. This means that if engine speed is plotted on X axis and engine load on Y axis. We shall get a graph of engine speed vs engine load. The ECU algorithm shall read that get inputs such as engine speed and engine load. If the 2 match as per the graph then it is fine, however if say they do not match, the ECU algorithm shall send signals to actuators to make it match. A modern day ECM can contain thousands of such maps. They are also 3D maps and surface maps.

Lookup tables

A lookup table is a way in which a dependable variable is ‘written’ in conjunction to an independent variable in the engine management. These variables are obtained from sensors. The values thus obtained are stored in an array which is termed as lookup tables. The ECU algorithm also uses these values in engine operation

Inputs and Outputs to the ECM

In an ECM, the inputs are from sensors and outputs are to actuators. The sensor *'senses'* the measured parameter and in context with it, sends a suitable signal to the ECM. The ECM reads the signal, and runs it in the ECU algorithm ac-cording to which corrective action to be taken is determined. These corrective actions are then taken by sending corre-sponding signals to the actuators which actually *'take action.'*

The signals that are exchanged are either of two types,

 Analogue: these signals are based on voltage pulses and controlled by Pulse width modulation (PWM). Although useful in itself, it is slowly being overrun by digital signals. The reason is that PWM usually deals with either true (indicated by 1) or false (indicated by 0). Thus it gives an idea of the value of the parameter in question but does not give an extent of the value.

 Digital: these signals are based on data of bits each bit with a specific function. This makes it more useful in monitoring and controlling a system. A very simple example of this is the CAN system used

Electronic spark timing

Newer engines typically use electronic ignition systems (ignition controlled by a computer). The computer has a timing map which is a table with engine speed on one axis and engine load on another axis. Timing advance values are inserted in this table. The computer will send a signal to the ignition coil at the indicated time in the timing map in order to spark the spark plug. Most computers from original equipment manufacturers (OEM) are not able to be modified so changing the timing advance curve is not possible. Overall timing changes are still possible, depending on the engine design. Aftermarket engine control units allow the tuner to make changes to the timing map. This allows the timing to be advanced or retarded based on various engine applications.

Electronic spark control

In its simplest form, electronic ignition retained the conventional distributor with its mechanical spark advance, merely replacing the points with a non-wearing electronic means of sensing crankshaft position and firing the spark. Later analogue spark control computers had more sensors to gather information on throttle position, throttle opening speed, engine speed, manifold vacuum and coolant temperature. The spark advance was set by the computer rather than by a centrifugal unit in the distributor. Carburetors could be jetted for leaner fuel-air mixtures which burned more completely with less polluting residue. Today, at the end of the twentieth century, quite ordinary family cars have elec-tronic engine management systems which have done away with the familiar distributor and carburetor entirely and al-low previously unheard of performance and fuel economy.

Idle speed control system

Idle speed control is used by some engine manufacturers to prevent engine stall using engine management. The goal is to allow engine at as low rpm as possible, yet keep the engine from running rough and stalling when power con-suming accessories such as air conditioners are turned on.

The control mode selection logic switches to idle speed control when the throttle angle reaches zero.(completely closed position) position and engine rpm falls below a certain value, and when the vehicle is stationary. Idle speed is controlled by an electronically controlled throttle bypass valve that allows air to flow around the throttle plate and pro-duce the same effect as a slightly open throttle would create.

There are various schemes of operating a valve to introduce bypass air for idle control. One relatively common method is to use a stepper motor to actuate the valve. A stepper motor is a motor which can rotate in either directions. It rotates in small increments in angular position when pulses are given to it. This makes the stepper motor ideal for the operation of idle control valve

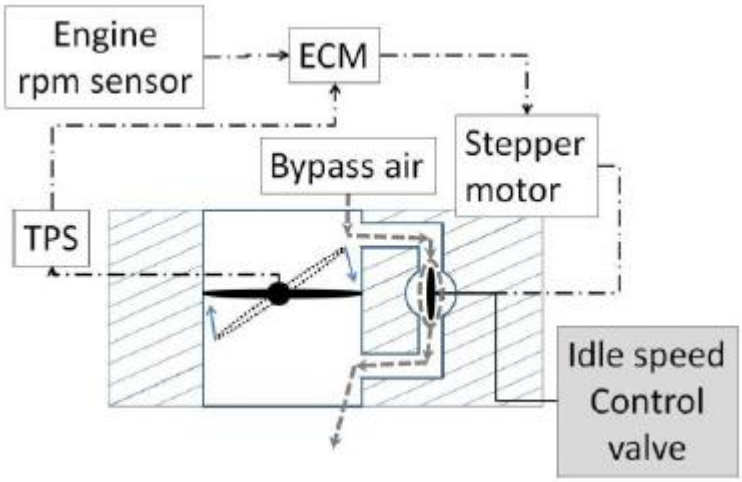


Figure 187: Idle speed control

Hands on: Try to perform some ECU tasks with arduino Microcontroller Board. Different task can even be proposed trainees themself .

Chap 6: Sensor and actuator

SENSORS PRINCIPLES

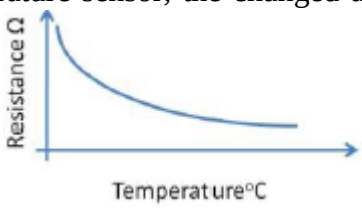
Sensors are those systems/components that give an electrical signal that is relative to the parameter they are measuring. These are extensions of transducers. All sensor use transducer technology, in conjunction with other sys-tems/technology to measure the given parameter and give a suitable output.

Thermisters

A thermistor is a type of resistor with resistance varying according to its temperature. The word is a combination of thermal and resistor. Thermisters are of 2 types, they are given as follows

NTC Type

To measure the temperature of air and fluid in vehicle sensor materials which change their electrical resistance with effect of heat is used. Most of these sensors are NTC Resistors, for example the coolant temperature sensor, the changed air temperature sensor. Consider the

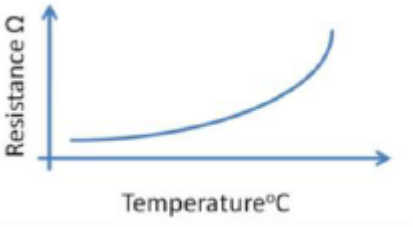


coolant temperature sensor.

The sensor consists of a semi conductor material which has the negative temperature co-efficient. This means the resistance of sensor drops as temperature in-creases. NTC Resistors are therefore also known as negative temperature co-efficient thermisters. The drop is resistors as temperature increases is caused by the increased reaching of electrons from the atomic bombs. The higher the tem-perature, the more free electrons there are available for electrical flow. This increases the conductivity of the material. The drop in resistor is registered by the control module and converted into corresponding temperature value.

PTC Type

NTC Resistors cannot be used for vary high temperature because the semi conductors will be destroy in the pro-cess. Instead PTC resistors, the positive temperature co-efficient are used. As in the case of measuring emission tem-perature PTC resistors consists of special metal alloy whose resistance increases with temperature. They therefore are also known as positive



temperature co-efficient thermistors.

Figure 188:PTC Type sensor

The increase in resistance is caused by the increase in thermal oscillation of the atoms as temperature rises. These thermal oscillation impede the flow of electrons. In other words the

resistance increases. The increase in resistance is registered by the control volume and converted into corresponding temperature value.

Inductive sensors

Inductive sensors are based on the principle of electromagnetic induction. In this usually there is a moving magnet and a stationary coil. The change in the flux cutting the coil generates an emf in the coil which is then measured. It gives a relative position and velocity of the magnet. Inductive sensors such as wheel speed sensors and crank shaft position sensors are extensively used in a vehicle.

The Crank shaft position sensor, for example consists of a permanent magnet, a soft iron core and a stationery coil. The sensor is located on the engine housing and is separated from the fly wheel by an air gap. The fly wheel may have teeth or grooves. As the flywheel rotates, it hinders the flux which is interacting with the coil. This generates and emf which is sensed by the ECM.

Resistive sensors

These are based on resistance change of a conductor. It is called a potentiometer. Its principle of operation is that the resistance of any conductor is directly proportional to its length and inversely proportional to the cross sectional ar-ea. A sliding jockey is connected to a coiled resistor, one end of the coil is connected to a voltage supply and the other end is connected to the jockey. As V=IR, Changing resistance causes a linear change in voltage which is measured.

Optical sensors

Optical sensors are based on photodiodes. A photodiode is a type of photodetector capable of converting light into either current or voltage, depending upon the mode of operation.

Photodiodes are similar to regular semiconductor diodes except that they may be either exposed (to detect vacu-um UV or X-rays) or packaged with a window or optical fiber connection to allow light to reach the sensitive part of the device. In this sense, there is a toothed member, which alternatively exposes and opposes the light beam from hitting the photo diode. This causes a pulse to be generated, the frequency of which is the same as the frequency of obstruc-tion

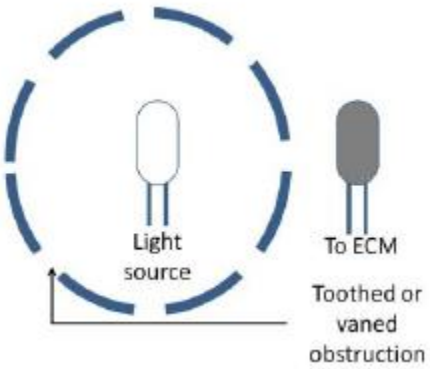


Figure 189:Principle of Optical sensor

Piezoelectric transducer

Piezoelectricity is the ability of some materials (notably crystals and certain ceramics) to generate an electric po-tential in response to applied mechanical stress. This means that when force is applied on these materials, it induces a voltage that can be measured

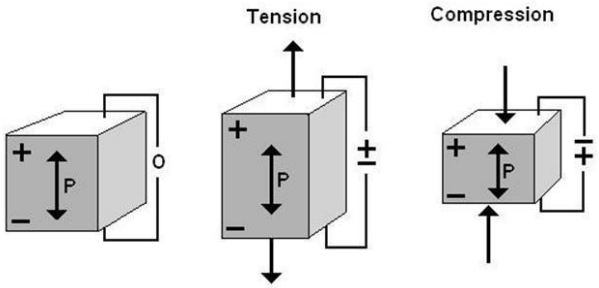


Figure 190: Piezoelectric Phenomen

Actuators

An actuator is a mechanical device for moving or controlling a mechanism or system. In automobiles, actuators are used for many purposes. The actuators in modern automobiles work when a signal is received from the ECM. This signal is interpreted and the actuator takes the necessary action. Actuators are also based on hydraulic principles. It is not necessary that all actuators respond to electric signal only. However, as the ECM is the device which controls almost everything in a vehicle, it is the electric signal which becomes inevitable. Further on from here, we shall see the basic principles of some of the actuators used in vehicles.

Solenoid Actuators

A typical electric solenoid actuator is shown.

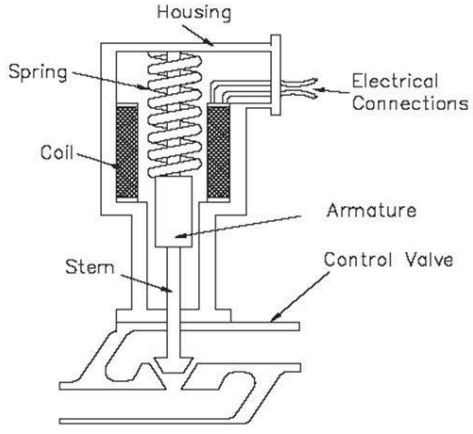


Figure 191:Solenoid actuator

It consists of a coil, armature, spring, and stem.

The coil is connected to an external current supply. The spring rests on the armature to force it downward. The armature moves vertically inside the coil and transmits its motion through the stem to the valve. When current flows through the coil, a magnetic field forms around the coil. The magnetic field attracts the armature toward the center of the coil. As the armature

moves upward, the spring collapses and the valve opens. When the circuit is opened and current stops flowing to the coil, the magnetic field collapses. This allows the spring to expand and shut the valve.

A major advantage of solenoid actuators is their quick operation. Also, they are much easier to install than pneu-matic or hydraulic actuators. However, solenoid actuators have two disadvantages. First, they have only two positions: fully open and fully closed. Second, they don't produce much force, so they usually only operate relatively small valves.

Electrohydraulic actuators

Electrohydraulic valve actuators and hydraulic valve actuators convert fluid pressure into motion in response to a signal. They use an outside power source and receive signals that are measured in amperes, volts, or pressure. Some electrohydraulic valve actuators and hydraulic valve actuators move rotary motion valves such as ball, plug, and butter-fly valves through a quarter-turn or more from open to close. Other valve actuators move linear valves such as gate, globe, diaphragm, and pinch valves by sliding a stem that controls the closure element. Throttling valves can be moved to any position, including fully open or fully closed, within the stroke of the valve. Typically, valve actuators are added to throttling valves as part of a control loop that includes a sensing device and circuitry.

Electrohydraulic valve actuators and hydraulic valve actuators use several different types of actuators. Diaphragm actuators are used mainly with linear motion valves, but are suitable for rotary motion valves with a linear-to-rotary motion linkage. Rack-and-pinion actuators transfer the linear motion of a piston cylinder actuator to rotary motion. They are ideal for automating manually-operated valves. Scotch yoke actuators also transfer linear motion to rotary motion. With lever and link actuators, a splined or slotted lever attaches to the valve shaft in order to transfer the linear motion of a diaphragm or piston cylinder to rotary motion. Vane actuators are used only with rotary motion valves.

Relay

A relay is an electrical switch that opens and closes under the control of another electrical circuit. It works on the principle of electromagnetic induction. In the original form, the switch is operated by an electromagnet to open or close one or many sets of contacts.

When a current flows through the coil, the resulting magnetic field attracts an armature that is mechanically linked to a moving contact. The movement either makes or breaks a connection with a fixed contact. When the current to the coil is switched off, the armature is returned by a force approximately half as strong as the magnetic force to its relaxed position. Usually this is a spring, but gravity is also used commonly in industrial motor starters. Most relays are manu-factured to operate quickly. In a low voltage application, this is to reduce noise. In a high voltage or high current appli-cation, this is to reduce arcing.

If the coil is energized with DC, a diode is frequently installed across the coil, to dissipate the energy from the col-lapsing magnetic field at deactivation, which would otherwise generate a spike of voltage and might cause damage to circuit components. Some automotive relays already include that diode inside the relay case.

Electro mechanic actuators

All actuators discussed above can be considered as electromechanical actuators. These are devices that convert electric signal into mechanical motion. All the actuators discussed above are of such type.

Thermal Actuators

Thermal microactuators are commonly either of the "bimetallic" type, or rely on the expansion of a liquid or gas.

Two metals of different coefficient of thermal expansion are used for the purpose. The 2 are fused to gather such that there can be no relative motion between the fusing surfaces. When the element is heated, the metal with higher coefficient of linear expansion expands more than the other. This causes the strip to bend. The strip, as it is made of metals, can be used to conduct electricity. A screw can be adjusted to adjust the position of the contact w.r.t. to the bimetallic strip.

Whilst thermally actuated devices can develop relatively large forces, the heating elements consume quite large amounts of power.

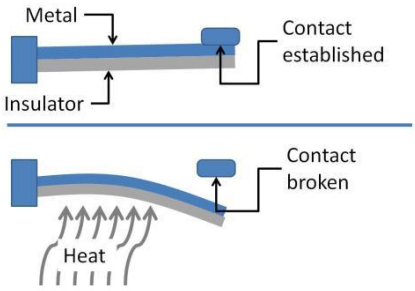


Figure 192: Bimetallic strip

Also, the heated material has to cool down to return the actuator to its original position; so the heat has to be dissipated into the surrounding structure. This will take a finite amount of time, and may affect the speed at which such actuators can be operated

Motorized actuators

These are linear travel actuators which control a linear parameter. This is usually based on a simple DC motor, which has a stationary magnet and a moving coil. These actuators can be made to be precise and accurate with a lease count in micrometers. As mentioned above, most of them are based on electric DC motor. However, high precision ones (which are generally not used in cars) use piezoelectric effect to have a travel of a few microns on application of a high voltage.

The actuator is based on a screw of very fine pitch, which is splined internally and connected to a splined armature shaft. Rotation of armature causes screw action which actuates the control arm

Some Automotive Actuators

- Throttle control valve
- EGR valve
- Air pump in air management system
- Automatic door locks

- Heated seat element cutoff actuator
- Remote trunk opener
- Motors used in opening rooftop of convertibles
- TCS/ABS hydraulic braking/pressure release devices

Hands on: Try to interface different types of sensors and actuators reffering to how they are been used in a vehicle.

Chap 7: Lighting

Types of lamps

There are various lamps used in cars. They are as follows
Forward Illumination

Head lamps:

dipped beam: Dipped-beam (also called low, passing, or meeting beam) headlamps provide a light distribution to give adequate forward and lateral illumination without blinding other road users with excessive glare
Main beam: (also called high, driving, or full beam) headlamps provide an intense, centre-weighted distribution of light with no particular control of glare**.**

Auxiliary lamps:

Driving lamps "Driving lamp" is a term deriving from the early days of nighttime driving, when it was relatively rare to encounter an opposing vehicle. Only on those occasions when opposing drivers passed each other would the dipped or "passing" beam be used. The full beam was therefore known as the driving beam, and this terminology is still found in international ECE Regulations, which do not distinguish between a vehicle's primary (mandatory) and auxiliary (op-tional) upper/driving beam lamps**.**
Fog lamps: Front fog lamps provide a wide, bar-shaped beam of light with a sharp cutoff at the top, and are gener-ally aimed and mounted low They may be either white or selective yellow. They are intended for use at low speed to in-crease the illumination directed towards the road surface and verges in conditions of poor visibility due to rain, fog, dust or snow.
Cornering lamps: On some models in North America and Japan, white cornering lamps provide extra lateral illumi-nation in the direction of an intended turn or lane change. These are actuated in conjunction with the turn signals, though they burn steadily, and they may also be wired to illuminate when the vehicle is shifted into reverse gear
Spot lights: Police cars, emergency vehicles, and those competing in road rallyes are sometimes equipped with an auxiliary lamp in a swivel-mounted housing attached to one or both a-pillars, directable by a handle protruding through the pillar into the vehicle.

Conspicuity devices

Retro-reflectors: The most basic vehicle conspicuity devices are retroreflectors (also reflex reflectors or, archaically, cat's eyes - not to be confused with the reflective road markings), which despite emitting no light on their own, are reg-ulated as automotive lighting devices. These devices reflect light from other vehicles' headlamps back towards the light source, that is, other vehicles' drivers. Thus, vehicles are conspicuous even when their electrically-powered lighting sys-tem is deactivated or disabled

Front position lamps (parking lamps): Nighttime standing-vehicle conspicuity to the front is provided by front position lamps, known as parking lamps or parking lights in North America, sidelights in UK English, and in other regions as posi-tion lamps, standing lamps, or city lights.

 Rear position lamps-tail lamps (power requirement 5W): Night time vehicle conspicuity to the rear is provided by rear position lamps (North American terms: taillamp, taillight, tail lamp, tail light; UK term rear light). These are required to produce only red light, and to be wired such that they are lit whenever the front position lamps are illuminated—including when the headlamps are on. Rear position lamps may be combined with the vehicles brake lamps, or separate
from them. In combined-function installations, the lamps produce brighter red light for the brake lamp function, and dimmer red light for the rear position lamp function.

 Rear registration plate lamp: The rear registration plate must be illuminated by a white lamp whenever the position lamps are active. The light may however not be directed to the rear.

 Daytime running lamps: Some countries permit or require vehicles to be equipped with daytime running lamps (DRL). These may be functionally-dedicated lamps, or the function may be provided by e.g. the low beam or high beam head-lamps, the front turn signals, or the front fog lamps, depending on local regulations

 Rear fog lamps (power requirement 21W): In Europe and other countries adhering to ECE Regulation 48, vehicles must be equipped with one or two bright red "rear fog lamps" (or "fog tail lamps"), which are switched on manually by the driver in conditions of poor visibility to enhance vehicle conspicuity from the rear

 Emergency vehicle lights: Emergency vehicles such as fire engines, ambulances, police cars, snow-removal vehicles and tow trucks are usually equipped with intense warning lights of particular colours. These may be motorised rotating bea-cons, xenon strobes, or arrays of LEDs.

 Taxi displays: Taxicabs are distinguished by special lights according to local regulations. They may have an illuminated "Taxi" sign, a light to signal that they are ready to take passengers, and an emergency panic light the driver can activate in the event of a robbery to alert passersby to call the police.

Signaling Devices

 Turn signals (power requirement 7W): Turn signals (properly directional indicators or directional signals, also "indica-tors," "directionals," "blinkers," or "flashers") are signal lights mounted near the left and right front and rear corners, and sometimes on the sides of vehicles, used to indicate to other drivers that the operator intends a lateral change of position (turn or lanechange).

 Hazard lights: International regulations have since the 1960s required vehicles to be equipped with a control which, when activated, flashes the left and right directional signals, front and rear, all at the same time and in phase. This func-tion is meant to be used to indicate a hazard such as a vehicle stopped in or alongside moving traffic, a disabled vehicle, an exceptionally slow-moving vehicle, or a vehicle participating in a funeral procession.

 Brake lights (power requirement 15W to 36W): Red steady-burning rear lights, brighter than the taillamps, are activat-ed when the driver applies the vehicle's brakes. These are called brake lights or stop lamps. They are required to be fit-ted in multiples of two, symmetrically at the left and right edges of the rear of every vehicle.

 Reversing light(power requirement 24W): To provide illumination to the rear when backing up, and to warn adjacent vehicle operators and pedestrians of a vehicle's rearward motion, each vehicle must be equipped with at least one rear-mounted, rear-facing reversing lamp (or "backup light")

Head lamps

Halogen bulb

A halogen lamp is an incandescent lamp where a tungsten filament is sealed into a compact transparent envelope filled with an inert gas, plus a small amount of halogen such as iodine or bromine. The halogen cycle prevents darkening of the bulb. The halogen lamp can operate its filament at a higher temperature than in a standard gas filled lamp of sim-ilar wattage without loss of operating life. This gives it a higher efficacy (10-30%). It also gives light of a higher color temperature compared to a non-halogen incandescent lamp. Alternatively, it may be designed to have perhaps twice the life with the same or slightly higher efficacy.

The reason that these bulbs don't blacken is that in older gas bulbs, over a period of time, about 10% of filament metal evaporates and gets deposited on the glass wall. The halogen in the halogen bulb prevents that. When tungsten filament metal evaporates, it forms tungsten halide; this is not deposited on the wall due to temperature. The convec-tion current causes this tungsten halide to move back to the filament at some point or the other and tungsten is again deposited on the filament.

Headlight reflectors

The light produced by the bulb is insufficient as it is directed toward a particular direction only. To give the light emitted from the bulb, a definite pattern to cover considerate amount of longitudinal and lateral visibility. Headlights are used which reflect the light from the bulb into a particular pattern on the road.

The object of the headlight reflector is to direct the random rays of light produced by the source (i.e. the bulb) into a concentrated beam of light by applying laws of reflection. Bulb position relative to the reflector is important. It de-termines the exact pattern of the light beam. Slight deviations in bulb position w.r.t. reflector can cause magnified dis-tortions in headlight pattern. A reflector is basically a layer of silver, chrome or aluminum deposited on a smooth polished surface such as brass or glass. As shown in the figure above, the reflectors used in automobile headlamps are all concave reflectors

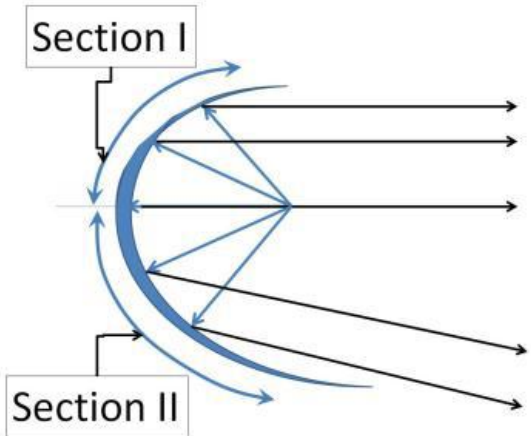
PARABOLIC REFLECTOR

A parabolic reflector is one in which if the source of light is kept at the focal point, all the light rays will end up par-allel to the principle axis. The light intensity is maximum at the centre, except from the light cut off by the bulb itself. The intensity diminishes as one moves away from the centre. The parabolic reflector is as shown in fig

BIFOCAL REFLECTOR

The bifocal reflector, as the name suggests has 2 different curved surfaces with 2 focal points. This helps to take the advantage of light striking the lower reflector area. The parabolic section of the down section is designed to reflect the light further down to improve near the car visibility. This system is not suitable with twin filaments

bulbs and is only used for vehicles with 4 head lamps (Mercedes E class, Jaguar S type).



HOMIFOCAL REFLECTOR

A homifocal reflector is made up of a number of sections, each with a common focal point, this allows shorter focal length and hence overall depth of the light unit decreases. The effective luminous flux is also increased. It can be used with a twin filament bulb to give a dip and main beam. The light from the main reflector gives normal long range light-ing, and the auxiliary reflectors improve near field and lateral visibility.

POLYELIPSOIDAL HEADLIGHT SYSTEM

It was introduced by Bosch in 1983. It allows the light produced to be as good, or in some cases even better than conventional lights, but with a light opening area of less than 30cm2. This is achieved by using a CAD designed elliptical reflector. A shield Is used to ensure particular pattern. This can be a clearly defined cutoff lines or even intentional blur-riness in the image. These can be only used with single filament bulbs and are found in four headlamp vehicles.

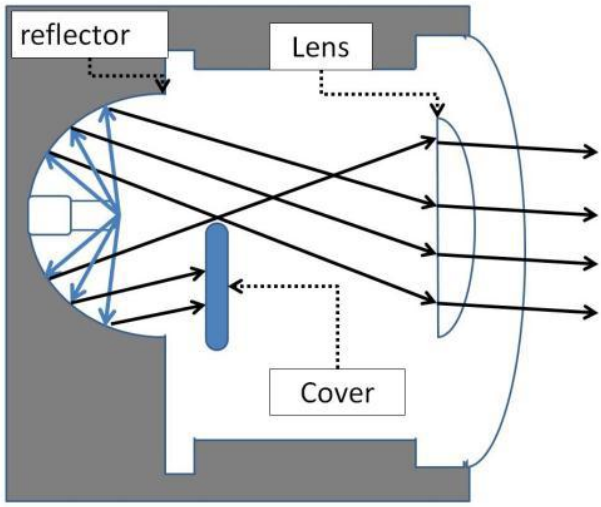


Figure 193:Poly Ellipsoidal headlight system

Headlight lenses

A good headlight should have a powerful and far reaching central beam, around which the light is distributed both horizontally and vertically in order to illuminate as much an area of

the road surface as possible. The beam formation can be considerably improved by passing the reflected light through a transparent block of lenses. It is the function of the lenses to partially redistribute the reflected light beam and any stray rays so that overall road illumination is im-proved with minimum glare.
Lenses work on the principle of refraction. The headlamp front glass is made up of a large number of small rectan-gular zones, each zone being formed optically in the shape of a concave flute or a combination of flutes and prisms. The shape of these sections is such that when a roughly parallel beam of light passes through the glass, each individual ele-ment is redirected to obtain better beam design.
The flutes control the horizontal distribution of light. At the same time, they sharply bend the rays downward, to give diffused local lighting just near the vehicle. Many headlights are now made with clear lenses. This means that all the reflection is being done by the reflector only (e.g. BMW series 5, series 7).

Electronic flasher circuit

Direction indicators have a number of requirements which are governed by legislature. They are as follows.

1. Light must be amber in colour.
2. Flashing must be in phase.
3. Flashing rate must be between 1-2 per second.
4. On a fault in the circuit, there should be an indication in the instrument panel.
5. If one bulb fails, others need to continue flashing.
6. An audible noise of ‘tick-tock’ nature is necessary to indicate the flashing of the lights.

Flasher Unit

A schematic circuit diagram of an electronic flasher unit is as shown in Fig 10.3.2.1. The operation of this unit is based around an integrated circuit (IC). The type shown can operate at least four 21W bulbs (front and rear) and two 5W bulbs (sides) for several hours if required in hazard mode. Flasher units are rated by the number of bulbs they are capable of operating. When towing a trailer or caravan, it must be able to operate the bulbs on the trailer/caravan also. Most units use a relay for the actual switching as this is not susceptible to voltage spikes and also provides the audible signal.
The electronic circuit is constructed together with the relay, on a printed circuit board. Very few components are used as the IC is specially designed for the purpose. The IC itself has 3 main sections, the relay driver, an oscillator and a bulb failure circuit. A zener diode is built in the oscillator section to ensure constant voltage such that the frequency of operation will remain constant in the range of 10-15V. The timer for the oscillator is controlled by an R1 and C. The values are often set to give an ON/OFF ratio of 1:1 and a frequency of 1.5Hz. the ON-OFF signal produced by the oscillator is passed to the driver circuit, which is a Darling-ton pair with a diode connected to prevent damage due to generation of back emf, when relay turns ON and OFF.
Bulb failure is recognized when the volt drop across the low value resistor R2 falls. The bulb failure circuit causes the oscillator to double the speed of operation. Extra capacitors can be used to protect the circuit against transient voltages and for interference problems. Fig –

10.3.2.2 shows the actual ‘packaging’ of the flasher unit.

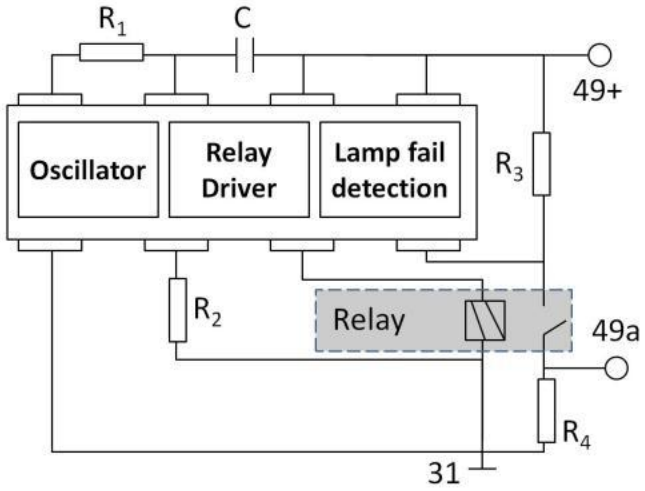


Figure 194: Electronics Flasher Circuit

Hands on: By using arduino, perform some projects which are linked to vehicle lighting .

Chap 8: Accessories

Visual displays

Although the analogue system has almost become obsolete in other applications, we find that in vehicles, even digital displays are represented in an analogue manner. This is because, analogue displays reduce driver processing time, leaving more time to interpret the actual driving conditions.
For example, an analogue engine temperature gauge, with its needle in the middle (not on H or C, but in the mid-dle) is easy to read and interpret. The driver can easily see that the value is within limits. The same display, however if showed 70oC, it would be difficult to interpret.

Over the years, there has been considerate amount of advancement in display technologies of automobiles. There are those that are still used such as analogue ones, and the digital ones have also been evolving. Some of the most commonly used display techniques is given below

Light emitting diode displays

A light-emitting diode, usually called an LED is a semiconductor diode that emits incoherent narrow-spectrum light when electrically biased in the forward direction of the p-n junction, as in the common LED circuit. This effect is a form of electroluminescence.

In automobile instrumentations, it is widely used for showing speed, to odometer, to fuel level and other things. The number is represented by the '8' Fashion, and other things are shown by bar graphs 

Figure 195: LED based Instrumentation basic shape

Liquid crystal displays

LCDs use liquid crystals that do not melt directly, but get transformed to a paracrystalline form in which the mole-cules are partially ordered. In this stage, the material is a cloudy, translucent fluid, still having optical properties of sol-ids. There three types of these crystals

1. Smectic: parallel rod shaped molecules, arranged in layers but with no pattern in each layer

2. Nemetic: parallel rod shaped molecules, not arranged in layers

3. Cholestric (twisted nematic): parallel rod shaped molecules, arranged in layers, with each layer having spiral or helical orientation

Mechanical stress, electric and magnetic fields, pressure and temperature can alter molecular structure of liquid crystals. A liquid crystal also scatters light that shines on it. Because of these properties, liquid crystals are used to dis-play letters, and numbers in calculators and such devices. Advancement in technology has caused to arrange these in an array to create a pixel matrix. These can be used to give colour display by the use of three colours on each pixel, viz RGB.

One type of display incorporating choleristic type is explained here. This display is achieved by only allowing polar-ized light to enter the crystal. As the light passes

through the crystal, it rotates by 90o. The light then passes to a second polarizer which is at 90o to the first one. Thus reflection takes place and all the light is reflected. However, when a volt-age of 10V at 50Hz is applied, the crystal orientation changes and it no longer turns the light by 90o. This causes a dark spot to appear on the screen. These areas are strategically placed on the display panel in the same manner as LEDs are arranged, to get meaningful designs.

Fuel level Gauge

The fuel level gauge is usually of resistive type. Other types using electromagnetic induction, variable capacitance are also in use, however, the variable resistive gauge is by far the most used in today's automobiles.

the float is attached to a guide which travels on a rod in the fuel tank. The float is directly attached to the jockey. According to the fuel level, the float rises/falls and changes the resistance. This change in resistance of the circuit causes a change in the voltage drop across the circuit, which is measured and fuel level calculated.

Electric horn

Automobile horns are usually electric klaxons, driven by a flat circular steel diaphragm that has an electromagnet acting upon it and is attached to a contactor that repeatedly interrupts the current to the electromagnet. This arrange-ment works like a buzzer or electric bell. There is usually a screw to adjust the distance/tension of the electrical con-tacts for best operation.

Refer fig 11.5.1, when the horn switch is closed, the relay coil is energized which closes relay switch and current flows through the horn winding. This magnetizes the core and attracts the tone disc through the connection and dia-phragm. As soon as that happens, the contact breaks, which de-energizes the winding, magnetic flux falls, and tone disc is sent back. Again contact is made and again the cycle continues. This happens several times in a second, and causes the tone disc to vibrate against the diaphragm to produce sound.

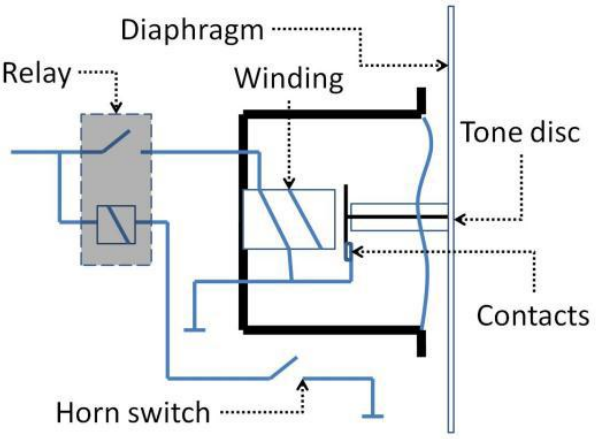


Figure 196: Electric horn

Wipers

A windscreen wiper (windshield wiper in North America) is a device used to wipe rain and dirt from a windscreen. Almost all automobiles are equipped with windscreen wipers, often by legal requirement.
Wipers can also be fitted to other vehicles, such as buses, trams, locomotives, aircraft and ships.
A wiper generally consists of an arm, pivoting at one end and with a long rubber blade attached to the other. The blade is swung back and forth over the glass, pushing water from its surface. The speed is normally adjustable, with several continuous speeds and often one or more "intermittent" settings. Most automobiles use two synchronized radi-al type arms, while many commercial vehicles use one or more pantograph arms. Mercedes-Benz pioneered a system called the Monoblade in which a single wiper extends outward to get closer to the top corners, and pulls in at the ends and middle of the stroke, sweeping out a somewhat 'W'-shaped path.
Wipers may be powered by a variety of means, although most in existence today are powered by an electric motor through a series of mechanical components, typically two 4-bar linkages in series or parallel. Vehicles with air operated brakes sometimes use air operated wipers, run by bleeding a small amount of air pressure from the brake system to a small air operated motor mounted just above the windscreen. These wipers are activated by opening a valve which al-lows pressurized air to enter the motor.

Most windscreen wipers operate together with a windscreen washer; a pump that supplies water and detergent (usually a blend called windscreen wiper fluid) from a tank to the windscreen through small nozzles, mounted on the hood or on the wipers, known as a 'wet-arm' system.

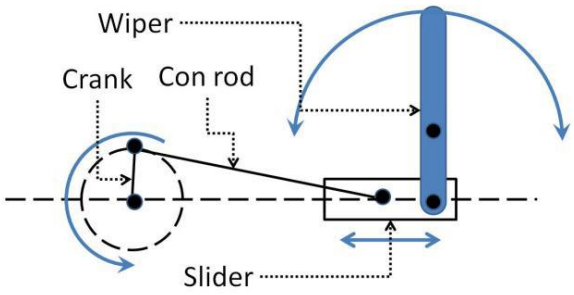


Figure 197:Wiper mechanism run by motor using single slider crank mechanism

Fuel pump

Nowadays, the fuel pump is located inside of the fuel tank and is usually electric. The pump creates positive pres-sure in the fuel lines, pushing the gasoline to the engine. The higher gasoline pressure raises the boiling point. Placing the pump in the tank puts the component least likely to handle gasoline vapor well (the pump itself) farthest from the engine, submersed in cool liquid. Another benefit to placing the pump inside the tank is that it is less likely to start a fire. Though electrical components (such as a fuel pump) can spark and ignite fuel vapors, liquid fuel will not explode due to absence of air, and therefore submerging the pump in the tank is one of the safest places to put it.
The ignition switch does not carry the power to the fuel pump, instead it activates a relay which will handle the higher current load.

Modern engines utilize solid-state control which allows the fuel pressure to be controlled via pulse-width modula-tion of the pump voltage.[1] This increases the life of the pump, allows a smaller and lighter device to be used, and re-duces electrical load and thereby fuel consumption.

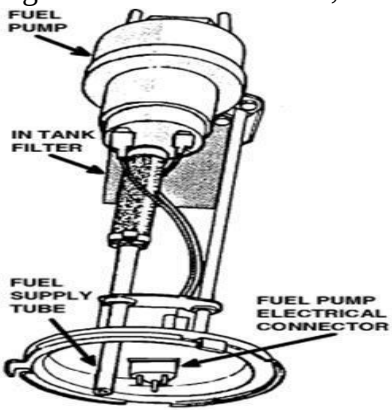


Figure 198: Electric Fuel Pump

i. Power operated windows

THE LIFTING MECHANISM

The window lift on most cars uses a mechanical linkage to lift the window glass while keeping it level. A small elec-tric motor is attached to a worm gear and several other spur gears to create a large gear reduction, giving it enough torque to lift the window.

An important feature of power windows is that they cannot be forced open -- the worm gear in the drive mecha-nism takes care of this. Many worm gears have a self-locking feature because of the angle of contact between the worm and the gear. The worm can spin the gear, but the gear cannot spin the worm -- friction between the teeth causes the gears to bind.

The linkage has a long arm, which attaches to a bar that holds the bottom of the window. The end of the arm can slide in a groove in the bar as the window rises. On the other end of the bar is a large plate that has gear teeth cut into it, and the motor turns a gear that engages these teeth. The same linkage is often used on cars with manual windows, but instead of a motor turning the gear, the crank handle turns it. In the next section we'll learn about some of the neat features some power windows have, including the child lockout and automatic-up.

ELICTRICAL AND WIRING SYSTEM

On this system, the power is fed to the driver's door through a 20-amp circuit breaker. The power comes into the window-switch control panel on the door and is distributed to a contact in the center of each of the four window switches. Two contacts, one on either side of the power contact, are connected to the vehicle ground and to the motor. The power also runs through the lockout switch to a similar window switch on each of the other doors.

When the driver presses one of the switches, one of the two side contacts is disconnected from the ground and connected to the center power contact, while the other one remains grounded. This provides power to the window mo-tor. If the switch

is pressed the other way, then power runs through the motor in the opposite direction.

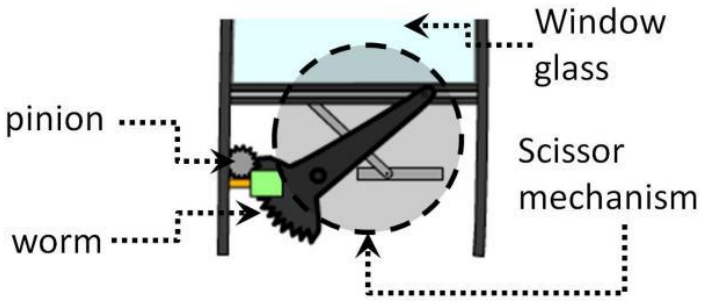


Figure 199: Lifting Mechanism of a window

Hands on: Each By using arduino , perform some projects related to the above vehicle accessories .

Chap 9: Telematics

The term telematics is used in a number of ways:
 It can be defined as the integrated use of telecommunications and informatics, also known as ICT (Information and Communications Technology). More specifically it is the science of sending, receiving and storing information via tele-communication devices.

 More commonly, telematics have been applied specifically to the use of Global Positioning System technology integrat-ed with computers and mobile communications technology in automotive navigation systems.

 Most narrowly, the term has evolved to refer to the use of such systems within road vehicles, in which case the term vehicle telematics may be used

Vehicle telematics systems may be used for a number of purposes, including collecting road tolls, managing road usage (intelligent transportation systems), pricing auto insurance, tracking fleet vehicle locations (fleet telematics), cold store logistics, recovering stolen vehicles, providing automatic collision notification, location-driven driver information services — and more particularly, dedicated short range communications DSRC in-vehicle early warning (car accident prevention) notification alerts.
Vehicle telematics systems are also increasingly being used to provide remote diagnostics; a vehicle's built-in sys-tem will identify a mechanical or electronic problem, and the telematics package can automatically make this infor-mation known to the vehicle manufacturer service organization. The telematics monitored system is also capable of no-tifying any problems to the owner of the vehicle via e-mail. Other forthcoming applications include on-demand naviga-tion, audio and audio-visual entertainment content.
While there are many potential applications for vehicle telematics, the main advantage for transportation safety advocates is that it will help reduce and ideally eliminate road injuries and road traffic related deaths worldwide.

Telematics architecture

The architecture of telematics is a very complex data transfer process, much of the same way that internet works, or mobile phones works. In simple forms, it can be explained by Fig

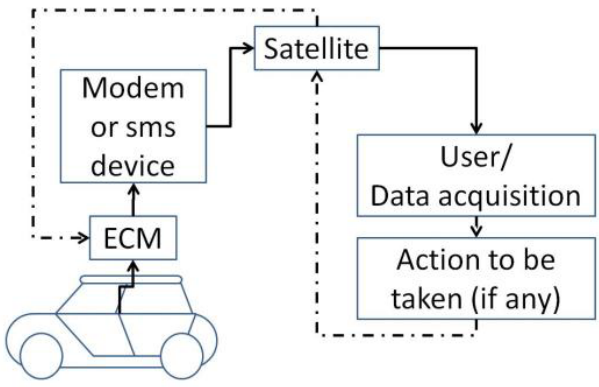


Figure 200:Telematics architecture

What data is to be collected is pre-fed in the ECM algorithm. This data is collected and sent to the modem or sms device, it must be noted that these 2 devices are integral

with the car, but for simplicity sake, these are shown as sepa-rate blocks outside the vehicle. The data is then sent to a satellite, which reads redirects the data back to the data ac-quisition system or the user. If there is any action to be taken, it is done and the data is sent back to the car via the same route. The action taken may not be directed to the same car. For example, in case the car meets an accident, the 'action' taken is to alert the authorities such as police and the hospital in that area. The data is sent is in coded form, the code includes destination code, the data itself and many other such parame-ters required for such a system.

i. Vehicle tracking

Vehicle tracking is a way of monitoring the location, movements, status and behavior of a vehicle or fleet of vehi-cles. This is achieved through a combination of a GPS receiver and an electronic device (usually comprising a GSM GPRS modem or SMS sender) installed in each vehicle, communicating with the user (dispatching, emergency or co-coordinating unit) and PC- or web-based software.

ii. Trailer tracking

Trailer tracking is the technology of tracking the movements and position of an articulated vehicle's trailer unit, through the use of a location unit fitted to the trailer and a method of returning the position data via mobile communi-cation network or geostationary satellite communications, for use through either PC- or web-based software.

iii. Cold store freight logistics

Cold store freight trailers that are used to deliver fresh or frozen foods are increasingly incorporating telematics to gather time-series data on the temperature inside the cargo container, both to trigger alarms and record an audit trail for business purposes.

iv. Fleet management

Fleet management is the management of a company's vehicle fleet. Fleet management includes the management of ships and or motor vehicles such as cars, vans and trucks. Fleet (vehicle) Management can include a range of Fleet Management functions, such as vehicle financing, vehicle maintenance, vehicle telematics (tracking and diagnostics), driver management, fuel management and health & safety management.

v. Satellite navigation

Satellite navigation in the context of vehicle telematics is the technology of using a GPS and electronic mapping tool to enable the driver of a vehicle to locate a position, then route plan and navigate a journey.

vi. Mobile data and mobile television

Mobile data is use of wireless data communications using radio waves to send and receive real time computer data to, from and between devices used by field based personnel. These devices can be fitted solely for use while in the ve-hicle (Fixed Data Terminal) or for use in and out of the vehicle (Mobile Data Terminal). See mobile Internet.

Mobile data can be used to receive TV channels and programs, in a similar way to mobile phones, but using LCD TV devices.

vii. Wireless vehicle safety communications

It is an electronic sub-system in a car or other vehicle for the purpose of exchanging safety information, about such things as road hazards and the locations and speeds of vehicles, over short range radio links. This may involve tempo-rary ad hoc wireless local area networks.

Wireless units will be installed in vehicles and probably also in fixed locations such as near traffic signals and emer-gency call boxes along the road. Sensors in the cars and at the fixed locations, as well as possible connections to wider networks, will provide the information, which will be displayed to the drivers in some way.

Intelligent vehicle technologies

Telematics comprise electronic, electromechanical, and electromagnetic devices — usually silicon micro machined components operating in conjunction with computer controlled devices and radio transceivers to provide precision re-peatability functions (such as in robotics artificial intelligence systems) emergency warning validation performance re-construction.

Intelligent vehicle technologies commonly apply to car safety systems and self-contained autonomous electrome-chanical sensors generating warnings that can be transmitted within a specified targeted area of interest, say within 100 meters of the emergency warning system for vehicles transceiver. In ground applications, intelligent vehicle tech-nologies are utilized for safety and commercial communications between vehicles or between a vehicle and a sensor along the road.

ABS

The reason for the development of anti-lock brakes (ABS) is very simple. Under braking conditions if one or more of the vehicle wheels locks (begins to skid) then this has a number of consequences:

- Braking distance increases;
- Steering control is lost;
- Tyre wear is abnormal.

The obvious consequence is that an accident is far more likely to occur. The maximum deceleration of a vehicle is achieved when maximum energy conversion is taking place in the brake system. This is the conversion of kinetic energy to heat energy at the discs and brake drums.

General system descriptions

As with other systems ABS can be considered as a central control unit with a series of inputs and outputs. An ABS system is represented by the closed loop system block diagram shown in Figure 13.1.2.1. The most important of the in-puts are the wheel speed sensors and the main output is some form of brake system pressure control. The task of the control unit is to compare signals from each wheel sensor to measure the acceleration or deceleration of an individual wheel. From this data and pre-programmed look up tables, brake pressure to one or more of the wheels can be regu-lated. Brake pressure can be reduced, held constant or

allowed to increase. The maximum pressure is determined by the driver's pressure on the brake pedal.

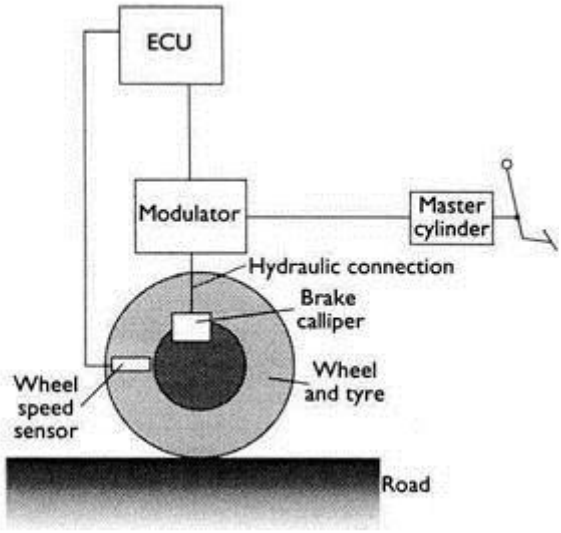


Figure 201: AVBS System

ABS components

Wheel speed sensors

Most of these devices are simple inductance sensors and work in conjunction with a toothed wheel. They consist of a permanent magnet and a soft iron rod around which is wound a coil of wire. As the toothed wheel rotates the changes in inductance of the magnetic circuit generates a signal; the frequency and voltage of which are proportional to wheel speed. The frequency is the signal used by the ECU. The coil resistance is in the order of 800 to 1000 Ω. Coaxial cable is used to prevent interference affecting the signal. Some systems now use 'Hall effect' sensors.

ECU

The function of the ECU is to take in information from the wheel sensors and calculate the best course of action for the hydraulic modulator. The heart of a modern ECU consists of two microprocessors such as the Motorola 68HC11, which run the same programme independently of each other. This ensures greater security against any fault which could adversely affect braking performance, because the operation of each processor should be identical. If a fault is detected, the ABS disconnects itself and operates a warning light. Both processors have nonvolatile memory into which fault codes can be written for later service and diagnostic access. The ECU also has suitable input signal processing stag-es and output or driver stages for actuator control. The ECU performs a self-test after the ignition is switched on. A fail-ure will result in disconnection of the system. The following list forms the self-test procedure:

- Current supply;
- Exterior and interior interfaces;
- Transmission of data;
- Communication between the two microprocessors;
- Operation of valves and relays;
- Operation of fault memory control;

 Reading and writing functions of the internal memory.

All this takes about 300 mS!

Hydraulic modulator

The hydraulic modulator as shown in Figure 9 has three operating positions:
 pressure buildup brake line open to the pump;
 pressure holding brake line closed;
 pressure release brake line open to the reservoir.

> The valves are controlled by electrical solenoids, which have a low inductance so they react very quickly. The mo-tor only runs when ABS is activated.

Electric power steering

Electric power steering (EPS or EPAS) is designed to use an electric motor to reduce effort by providing assist to the driver of a vehicle. Most EPS systems have variable assist, which allows for more assistance as the speed of a vehicle decreases and less assistance from the system during high-speed situations. This functionality requires a delicate bal-ance of power and control that has only been available to manufacturers in recent years. The EPS system has replaced the hydraulic steering system (HPS or HPAS) in many passenger cars recently. Although EPS is so far limited to passen-ger cars, as a higher voltage electrical system is necessary to operate EPS in larger vehicles.
Unlike HPS systems, EPS systems do not require a hydraulic pump, which is belted into the engine. Rather the EPS system's electric motor is powered by the vehicle's alternator which is belted into the engine. The efficiency advantage of an EPS system is derived from the fact that it is activated only when needed. Thus, a vehicle equipped with EPS may achieve an estimated improvement in fuel economy of 3% compared to the same vehicle with conventional HPS. How-ever, any fuel economy benefit of EPS over HPS can be negated in situations where a vehicle is not driven on straight-aways very often, or where a vehicle's wheels are out of alignment.

Global Positioning System

The Global Positioning System (GPS) is the only fully functional Global Navigation Satellite System (GNSS). Utilizing a constellation of at least 24 Medium Earth Orbit satellites that transmit precise microwave signals, the system enables a GPS receiver to determine its location, speed, direction, and time.
A typical GPS receiver calculates its position using the signals from four or more GPS satellites. Four satellites are needed since the process needs a very accurate local time, more accurate than any normal clock can provide, so the re-ceiver internally solves for time as well as position. In other words, the receiver uses four measurements to solve for four variables: x, y, z, and t. These values are then turned into more user-friendly forms, such as latitude/longitude or location on a map, then displayed to the user.

User segment

The user's GPS receiver is the user segment (US) of the GPS. In general, GPS receivers are composed of an antenna, tuned to the frequencies transmitted by the satellites, receiver-processors, and a highly-stable clock (often a crystal os-cillator). They may also include a

display for providing location and speed information to the user. A receiver is often described by its number of channels: this signifies how many satellites it can monitor simultaneously. Originally limited to four or five, this has progressively increased over the years so that, as of 2007, receivers typically have between 12 and 20 channels.

GPS and cars

GPS has found wide commercial applications in automobile industries. The maps of all major countries/ cities are fed in the system. This enables the system to accurately identify any address, anywhere! This helps in navigation. To reach new destinations, where the way is not known, GPS can be used with utter simplicity. The entire of Europe and America are covered by GPS and road maps. Moreover, the accuracy of these systems is such that they tell you your lo-cation with less than 10cms of error. This makes it extremely useful in automotive navigation use.

Adaptive Cruise Control

Adaptive cruise control (ACC) is a cruise control system in some modern vehicles. The system also goes under the names of active cruise control (ACC) or intelligent cruise control (ICC). These systems use either a radar or laser setup to allow the vehicle to slow when approaching another vehicle and accelerate again to the preset speed when traffic al-lows. ACC technology is widely regarded as a key component of any future generations of smart cars, as a form of artifi-cial intelligence that may usefully be employed as a driving aid.

Types

Laser-based systems are significantly lower in cost than radar-based systems; however, laser-based ACC systems do not detect and track vehicles well in adverse weather conditions nor do they track extremely dirty (non-reflective) vehicles very well. Laser-based sensors must be exposed, the sensor (a fairly-large black box) is typically found in the lower grill offset to one side of the vehicle.

Some systems also feature forward collision warning or Collision Mitigation Avoidance System, which warns the driver and/or provides brake support if there is a high risk of a rear-end collision.

Radar-based systems are available on many luxury cars as an option for approx. 1000-3000 USD/euro. Laser-based systems are available on some near luxury and luxury cars as an option for approx. 400-600 USD/euro. Radar-based sensors can be hidden behind plastic fascias; however, the fascias typically looks different from a vehicle without the feature. For example, Mercedes packages the radar behind the upper grill in the center; however, the Mercedes grill on such applications contains a solid plastic panel in front of the radar with painted slats to simulate the slats on the rest of the grill.

Drive by wire

Drive-by-wire, DbW, by-wire, or x-by-wire technology in the automotive industry replaces the traditional mechani-cal and hydraulic control systems with electronic control systems using electromechanical actuators and human-machine interfaces such as pedal and steering feel emulators. Hence, the traditional components such as the steering column, intermediate shafts, pumps, hoses, fluids, belts, coolers and brake boosters and master cylinders are eliminat-ed from the vehicle.

Examples include electronic throttle control and brake-by-wire.

Advantages

Safety can be improved by providing computer controlled intervention of vehicle controls with systems such as Electronic Stability Control (ESC), adaptive cruise control and Honda's Lane Keeping Assist System (LKAS).
Ergonomics can be improved by the amount of force and range of movement required by the driver and by greater flexibility in the location of controls. This flexibility also significantly expands the number of options for the vehicle's de-sign.
Parking can be made easier with reduced lock-to-lock steering wheel travel as with BMW's Active Steering System, or automatic parallel parking which is available in some Toyota Prius models and newer European Volkswagen models. Although neither of these are strictly Steer-by-Wire (SbW) because they retain mechanical linkages, they show the ca-pabilities that are possible.

Disadvantages

The cost of DbW systems is often greater than conventional systems. The extra costs stem from greater complexi-ty, development costs and the redundant elements needed to make the system safe. Failures in the control systems can result in an unstoppable runaway vehicle - if the throttle, ignition and transmission are all beyond the direct control of the driver there is no effective way to stop the vehicle in such an event.

Steer by Wire

This is currently used in electric forklifts and stockpickers and some tractors. Its implementation in road vehicles is limited by concerns over reliability although it has been demonstrated in several concept vehicles such as ThyssenKrupp Presta Steering's Mercedes-Benz Unimog, General Motors' Hy-wire and Sequel and the Mazda Ryuga. A rear wheel SbW system by Delphi called Quadrasteer is used on some pickup trucks but has had limited commercial success.
Competitors in the DARPA Grand Challenge, an automated driving competition, relied on 100% DbW systems, in some cases including a SbW system provided by the manufacturer.
In some concept vehicles, steer by wire completely eliminates the steering wheel, and the vehicle is steered by a joystick.

Hands on: By using some IoT skills , we will try to remotely control or monitor some parameters in a fixed or moving vehicle .

References

1. http://www.electronicsandyou.com/blog/electronic-components-parts-and-their-function.html
2. https://quasarelectronics.co.uk/component-identification-help
3. https://wiki.restarters.net/Basic_electronic_components
4. https://learn.Electronicians.com/sections/series-and-parallel-circuits/all
5. https://www.elprocus.com/types-of-electronic-testing-equipments/
6. https://learn.Electronicians.com/sections/how-to-use-a-multimeter/all
7. https://www.autodesk.com/products/eagle/blog/kirchhoffs-law-for-complex-circuits/
8. https://learn.sparkfun.com/Sections/how-to-use-an-oscilloscope/all
9. https://www.allaboutcircuits.com/video-lectures/troubleshooting-series-circuits/
10. https://www.arduino.cc/en/guide/introduction
11. https://www.instructables.com/id/Intro-to-Arduino/
12. https://en.wikipedia.org/wiki/ATmega328
13. https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/
14. https://learn.adafruit.com/photocells/arduino-code
15. https://www.arduino.cc/en/Tutorial/LiquidCrystalDisplay
16. https://randomnerdtutorials.com/guide-to-neo-6m-gps-module-with-arduino/
17. http://www.ibrahimshaikh.com/