## Arduino: Electronics Basics

*It seems that the first and most challenging part will be electronic engineering, which I am not an expert in. Arduino may offer a viable solution for wiring and connection. If the device's electronics would work, the device may be tested as a proof of concept. Until it is done, this may be the riskiest part of the whole feasibility of the project.*

Tutorial: https://www.youtube.com/watch?v=zJ-LqeX\_fLU&t=379s

**Voltage: Water Analogy**

The pressure at the end of the hose can represent the voltage, and the water in the tank is the charge or battery. So the more water in the tank, the higher the charge and the pressure at the end of the hose.

Diagram, schematic

Description automatically generated

Figure 1 | Voltage

**Current**

The rate of flow of charge is known as current. When electrons move from a battery's negative terminal, they give rise to current. The unit of current is Amperes. The direction in which the current flows is opposite to the directions of the electrons. Therefore current is the amount of electric charge that flows when one Coulomb of charge moves past somewhere in one second. Coulomb is the measuring unit of charge. We measure current with a device called an ammeter. We can also understand the flow of current by the water analogy. The diameter of the hose acts as a resistance. The wider hose offers less resistance; thus, more current can flow through it.

Diagram

Description automatically generated

Figure 2 | Current

**AC / DC**

There are two types of current, alternative and direct. In DC, the flow of electrons stays the same direction with constant voltage. In AC, the flow of electrons changes and does not stay steady with time. The current supplied in our home socket is AC, while the batteries are DC.

Chart

Description automatically generated

Figure 3 | AC / DC

**Resistance**

The electronic resistance of a component is the measure of the difficulty of passing an electric current through a substance. With more resistance to a circuit, less electricity will flow through. And based on this terminology, there is an electrical component called a resistor. The resistor is a device that limits the flow of a current in a circuit. For instance, we may require to use resistors in a current when using LED to prevent it from burning off.

Diagram

Description automatically generated

Figure 4 | Resistance

With the resistance with the water analogy, we can visualise how different sizes of pipes would deliver a certain amount of water flow. The wider the hosepipe is, the more water can flow through it. And the current always takes the least resistive path. Additionally, the value of a resistor is measured in Ohms. Therefore, the more the value of the resistor in Ohms, the more resistance it will offer to the circuit. Resistors are colour coded with five stripes that measure their resistance and tolerance.

Diagram

Description automatically generated

Figure 5 | Resistor Colour Chart

**Ohm's Law**

German physicist George Simon Ohm stated that the amount of electric current flowing through a metallic wire is directly proportional to the difference across it, provided the temperature remains the same. Mathematically, the voltage (V) is the product of the resistance (R) and current (I). If any of them is unknown, we can find them by the following formula.

Shape

Description automatically generated

Figure 6 | Ohm's Law Triangle

**Ohm's Law: Circuit Example**

An elementary circuit that has:

* A 600 Ohm resistor,
* A 12 V battery,
* Figure out the current.

A graph with numbers and letters

Description automatically generated with low confidence

Figure 7 | Calculate Current Example

**Series and Parallel Resistance**

In a series of resistors, the current of the first resistor's output flows into the second resistor and the third. Therefore, by each resistor, the voltage drop will be different. The sum of each resistor's voltage will equal the battery's voltage.

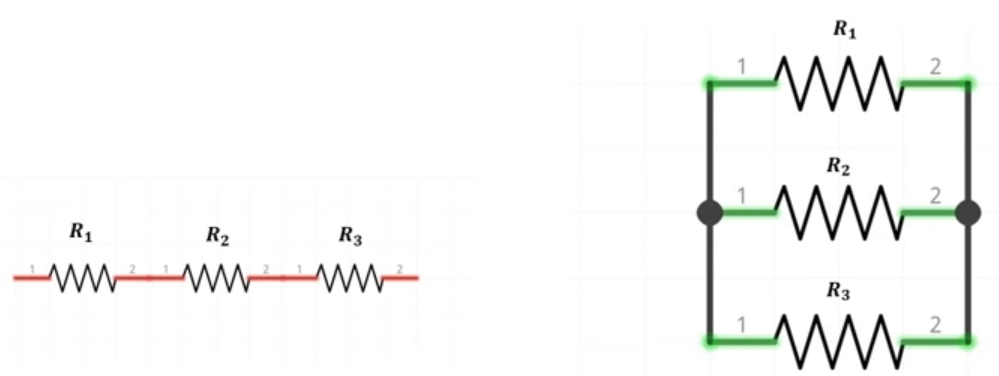


Figure 8 | Series and Parallel Resistors

**Series Connection Example**

Diagram, schematic

Description automatically generated

Figure 9 | Calculate Series of Resistors

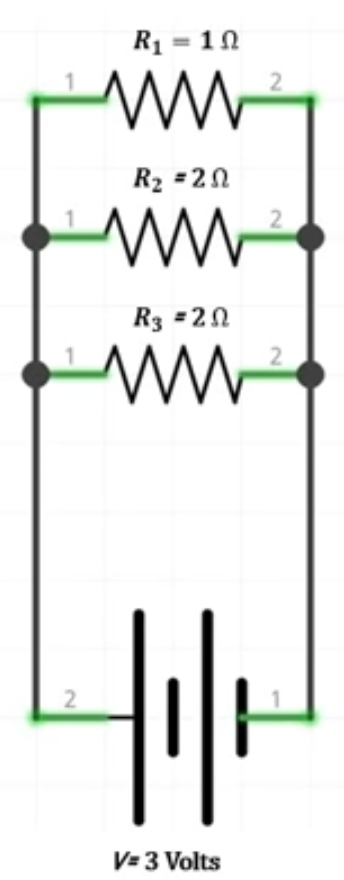
In a parallel circuit, the resistors are parallel when a continuous wire of negligible resistance connects all the resistors. Similarly, the ends are also connected. Since they are connected parallel, we can say that the voltages across them are equal. But unlike series resistors, the current has multiple paths to flow.

Icon

Description automatically generated with medium confidence

Figure 10 | Parallel Resistors

**Parallel Resistors Example**



**Combinations of Series and Parallel Resistors**

We must divide the circuit into sections when resistors are mixed (parallel and series).

**Diagram

Description automatically generated**

Figure 11 | Mixed Resistors

**Microprocessors and Microcontrollers**

Microprocessors are known as the brain of any computer. They consist of a CPU responsible for all the computations and calculations we need. In addition, we need additional peripherals, such as RAM and ROM, to make the computer function properly.

Microcontrollers are integrated chips, basically tiny computers that are powerful enough to do many tasks, like controlling motors, pumps, etc. Or send some sensor data to the internet cloud, control primary displays, or even act as brains of some intermediate-level robots. The power requirement for such chips is very low; they can even run on batteries. Moreover, they have all the peripherals like memory and flash storage on the same chip. Microcontrollers are far more economical solutions than microprocessors.

**Arduino**

Arduino is a circuit that has a microcontroller on it, acting as the heart of the board. The Arduino board given in a kit (Arduino Uno) uses a microcontroller ATMEGA328. There are other Arduino boards available that may be inferior or superior to Uno. Arduino's hardware and software are open-source and can be accessed and modified freely.

**The Arduino Uno Circuit Board**

**Graphical user interface

Description automatically generated**

Figure 12 | Arduino Uno Circuit Board Parts

1. **Digital Pins**: Digital pins are numbered from 0 to 13, and some pins have a special symbol (~), which denotes PWM capability. Other than that, these pins take digital sensors and give digital output.
2. **Built-in LED**: The LED is connected to pin number 13 by default, so whenever PIN-13 goes high, the LED will turn on.
3. **Power LED**: If this LED is turned on, the Arduino is plugged in.
4. **ATMEGA 328 Chip**: This component controls all the main functionality of the board.
5. **Analog Pins**: This board has six analogue pins, numbered A-0 to A-5. We can connect any analogue sensor to these pins.
6. **Power Pins**: 5V and GND.
7. **Power DC Jack**: This is how you power Andruino when it is not connected to a USB port for power from 7V up to 12V.
8. **TX and RX LED**: These LEDs indicate communication between the Arduino board and the computer. RX for receive, TX for Send or Transfer.
9. **USB Jack**: Used for powering the Arduino, uploading sketches, or communicating with the computer.
10. **Reset Button**: Reloads the entire code (but does not erase any already present code).

*However, it is suggested that the Arduino Uno is not ideal for my case, and I should use Arduino Leonardo instead when dealing with multiple button presses. However, for the duration of this tutorial, I will follow through with Uno's steps to get familiar with the basic controls.*

https://www.instructables.com/Arduino-Programmable-Button-Panel-As-Keyboard/

**Setup**

Download Arduino

Set communication port:

Tools >> Port (COM 3 - default) >> COM 3

Test Arduino using a built-in sketch:

File >> Examples >> 01 Basics >> Blink

Tools >> Board: Arduino Uno >> Arduino Uno

Port >> COM 3

Upload (+) Button

Arduino Emulator: TinkerCad

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure 13 | Test Arduino: Blink

**Electronics Basics Continued**

**Breadboard**

Calendar

Description automatically generated

Figure 14 | Breadboard

Breadboards are meant to make quick non-permanent connections between electronic components, and they are covered with tiny socket holes which are connected in rows. The breadboard itself is broken into four sections: two inner sections full of short horizontal rows and two outer sections long with long vertical rows. We will see a series of metal connections if we remove the adhesive at the breadboard's backside.

**Basic Push Button LED Circuit**

By convention, the battery's red wire represents the positive terminal and the black wire negative.

A picture containing text, shelf, different

Description automatically generated

Figure 15 | Push Button LED Circuit

**Arduino Safety Instructions**

* Never connect more than 5V into any I/O Pin. Any voltage more than 5V may be connected only to power the Arduino.
* Never connect any voltage above 5V on the GND Pin, as it will shortcut and turn off Arduino.
* Take care of polarity. Do not connect a positive pin to a negative. There is no polarity management on the device, and it will cause permanent damage.
* Be careful with external power. Most of the time, the USB power will be enough. Any other external powers may be used through DC jack or VN pins. In this case, we need to make sure not to connect any voltage over 12V. Arduino can handle up to 20V officially; however, it is not recommended to go beyond 12V.

**Arduino Input / Output**

**Input Devices:**

* **Rotatory Potentiometer**: This device would change the resistance it offers according to the knob rotation.
* **Momentary Push Button Switch**: When we press the switch, it will close the circuit, and when we release the switch, it will open it.
* **Force Sensing Resistor**: This resistor changes its resistance based on the force applied on it.
* **Light Dependent Resistor**: This resistor changes its resistance based on the light falling on it.
* **Tripple Axis Accelometer**: Usually, we see them in mobile phones and gaming devices, and the primary purpose of it is to get the orientation of the object on which it is placed.
* **Joystick**: a fundamental gaming input controller.
* **Microphone / Sound Sensor**: It captures environmental sound.
* **Touch Sensor:** See if someone is touching it or not.
* **Rotary Encoder:** This sensor senses the rotation. For instance, see the cycle of a motor.
* **Ultrasonic Distance Sensor:** Tells the difference between the sensor and the object.
* **Temperature Sensor**
* **Temperature and Humidity Sensor**
* **Infrared Sensor:** It is basically an obstacle detection sensor.

**Output Devices:**

* **LED:** Light Emitting Diode.
* **RGB LED:**  A particular LED that can form many colours given the amount of red, green, and blue.
* **Buzzer:** It will make some sound if a high voltage is given.
* **Loudspeaker**
* **OLED Display:** A full-fledged display that can display graphics.
* **LCD Display:** A liquid crystal display capable of displaying characters on it.
* **Seven Segment Display:** Consists of seven LEDs arranged in a rectangular fashion used to display numbers or letters.
* **Dot Matrix Display:** An array of multiple LEDs similar to some shops display the shops' names.
* **DC Motor:** It moves clockwise and anticlockwise if given some voltage across it.
* **Servo Motor:** We can stop it at any angle.#
* **Stepper Motor:** The rotation is divided into equal steps.

**Analog and Digital**

Our world is full of analogue information. Colours, fragrances, flavours, sounds, or temperature. These have an endless number of values. Likewise, in terms of electronics, something that is analogue can have an infinite number of values. So, for instance, our wall clock can show an endless number of time intervals – though they are not marked – it holds the capability to display an even smaller fraction of intervals. On the other hand, digital is something with a discrete or limited number of values, like a digital clock.

**Signals**

Signals are used to convey some information to someone. For instance, when we speak, we generate a signal containing the data we want to pass on to the listener. This signal is analogue in nature because there are infinite values which are changing. In electronics, signals are some time-varying quantities which convey some sort of information. Usually, this time-varying quantity is voltage. This means, with time, voltage changes. Those time-varying signals are passed between the devices to send and receive information. Typically an analogue signal would look like this:

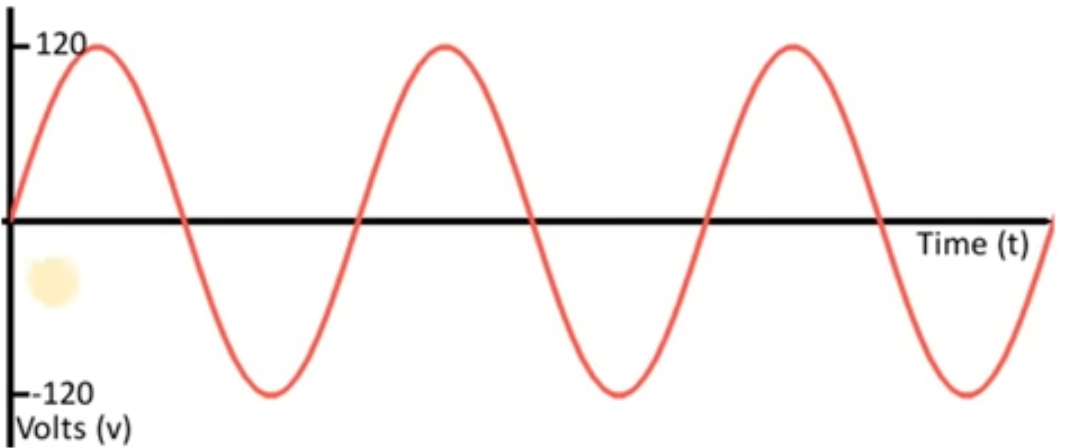


Figure 16 | Analogue Signal

In digital signals, there are only two discrete values; for instance, 5V represents on and 0V off.

Icon

Description automatically generated

Figure 17 | Digital Signal

The sensors we use can give analogue or digital signals, and we have to use Arduino to read those signals. So, for instance, let's see an Arcade gaming console. So whenever we use an analogue sensor or component, we will use Arduino's analogue pins, and respectively, we use digital pins for digital devices.

**Arduino Programming**

The language we use to program Arduino is a mixture of C and C++, which Arduino makers call Wiring. Wiring is uniform across all platforms.

Text

Description automatically generated

Figure 18 | Boilerplate

The setup function contains all the lines of code that we want Arduino to run once, when it is turned on or when we press the reset button. The loop function contains the code we wish Arduino to run repeatedly until the device is switched off. The loop runs after the setup function, and both functions are mandatory to declare.

Text

Description automatically generated

Figure 19 | Error Message

**Break Down Blink Function**

Let's break down the simple Blink function we created in the previous chapter!

Text

Description automatically generated

Figure 20 | Blink Function

**PinMode** is a built-in function configuring the specified pin as either input or output.

pinMode(pin, mode)

pin: the Arduino pin number to set the mode,

mode: INPUT, OUTPUT, or INPUT\_PULLUP.

Additionally, Arduino has a LED\_BUILTIN constant, which specifies PIN 13. More about constants:

https://www.arduino.cc/reference/en/language/variables/constants/constants/

**DigitalWrite** writes a HIGH or a LOW value to a digital pin. If the pin has been configured as OUTPUT, its voltage will be set to the corresponding value.

* 5V HIGH,
* 3.3V HIGH (3.3V boards),
* 0V (ground) LOW.

If the pin is configured as an INPUT, digitalWrite() will enable (HIGH) or disable (LOW) the internal pull-up on the input pin. Therefore, setting the pinMode() to INPUT\_PULLUP is recommended to enable the internal pull-up resistor.

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

**Serial.begin(9600).** This function enables serial communication between the Arduino and the computer.

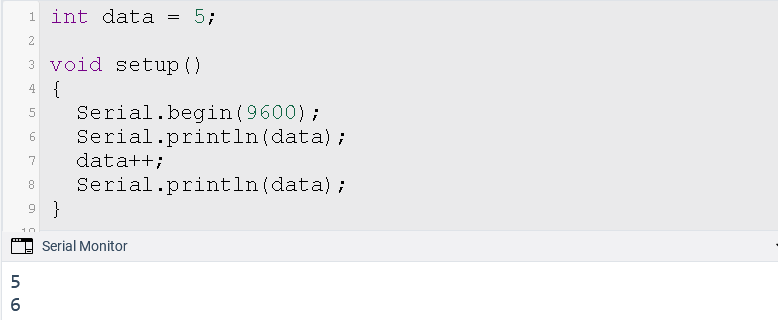
****

Figure 21 | Serial Communication

**RGB LED Setup and Function**

A simplistic solution for led lights blinking in a sequence.

**Graphical user interface, diagram

Description automatically generated**

Figure 22 | Blinking RGB LED Series

**LED Series Using Arrays and Functions**

This solution is similar to the previous one; however, it does not use a breadboard, LED pins are declared in an array, and it has a separate function for setting the LEDs to the low position.

Graphical user interface, diagram

Description automatically generated

Figure 23 | Blinking LED Solution

**Calculating LED Protection Resistance**

The typical maximum current the LED can work with is roughly 25mA, which means the total resistance of the resistor we need to add, and the LED should be R = 5V/25mA which is 200 Ohm. The resistance of the LED would be minimal in this direction, as we mentioned in the previous step. Thus, the resistance of the resistor would be greater than 200 Ohm. For safety reasons, we can use 220 Ohm, which will still work, as shown in the figure. 220 Ohm is a "standard" value.

That 40 mA max is to be avoided. The 20mA LED thing is kind of bogus. Most LEDs are bright at 10mA and not that much brighter at 20 mA.

The closest "standard" value is 330 3.2V / 330? = 9.7 mA.

**Different Ways to Declare a String in C++**

Graphical user interface, application, table, Excel

Description automatically generated

Figure 24 | String Declaration

**Push Button Switch**

This switch has four terminals, and they are internally connected in pairs.

Diagram

Description automatically generated**Diagram

Description automatically generated**

Figure 25 | Push Button Switch

**Digital Input and Output**

DidgitalWrite function has been discussed before. DigitalRead function reads the state of a pin. For instance, turn on an off LED if a switch is pressed.

A picture containing text, electronics, screenshot

Description automatically generated

Figure 26 | Switch LED State

**Set LED State with Potentiometer**

The pot meter's terminal one is connected to 5V, terminal 2 to the ground and the wiper is set on analogue pin A0. The three LEDs are on if the pot meter is in a specific position. For instance, the first third will light up red, the second yellow and the third be green. SwitchToLED function deselect any pin that is not its argument and sets the rest of the digital pins to LOW.

Graphical user interface, diagram

Description automatically generated

Figure 27 | LED Potmeter Circuit

**Simple Buzzer**

The speaker has to be connected to a digital pin and resistor. The tone function accepts an output pin, the frequency and an optional duration.

Graphical user interface, diagram

Description automatically generated

Figure 28 | Buzzer

**Speaker with Potmeter Setting Frequency (C4 – C5 Notes)**

We used a 10kOhm potentiometer where the values read on the analogue pin are between 0 and 1023. The frequency is directly proportional to the value of A0.

Graphical user interface, diagram

Description automatically generated

Figure 29 | Speaker with Potmeter

**Analogue to Digital Converter (ADC)**

The general idea is that the sensor reads a physical parameter (temperature, distance, humidity, etc.) and is converted it into an analogue signal. Then, an analogue-to-digital converter turns that analogue signal into a digital signal. Arduino has a 10 bits built-in ADC. Therefore, Arduino is capable of measuring voltages between 0V to 5V. Since the resolution of the Arduino board is 10 bits, it can hold up to 1023 values.

Chart, bar chart, waterfall chart

Description automatically generated Diagram

Description automatically generated with medium confidence

Figure 30 | ADC Conversion

**AnalogWrite Function**

A square wave has two distinct values LOW and HIGH.

Icon

Description automatically generated

Figure 31 | Square Wave

The amount of time the signal is on is known as pulse width. A duty cycle is a fraction of the period the signal is on.

Chart, waterfall chart

Description automatically generated

Figure 32 | Pulse Width & Duty Cycle

Diagram

Description automatically generated

Figure 33 | AnalogWrite vs Duty Cycle

**Piano with 8 Keys**

The eight momentary push buttons are expressed as a byte for better memory efficiency. However, this version must encode and decode the keyboard state in the main loop.

Diagram

Description automatically generated

Figure | Piano with Eight Keys