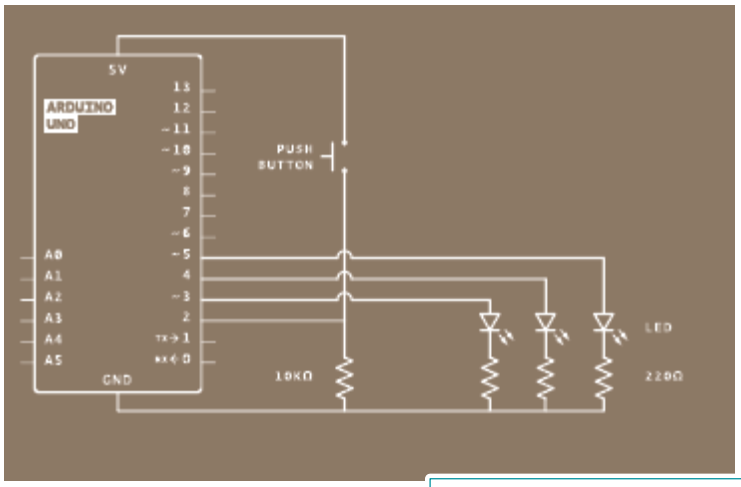
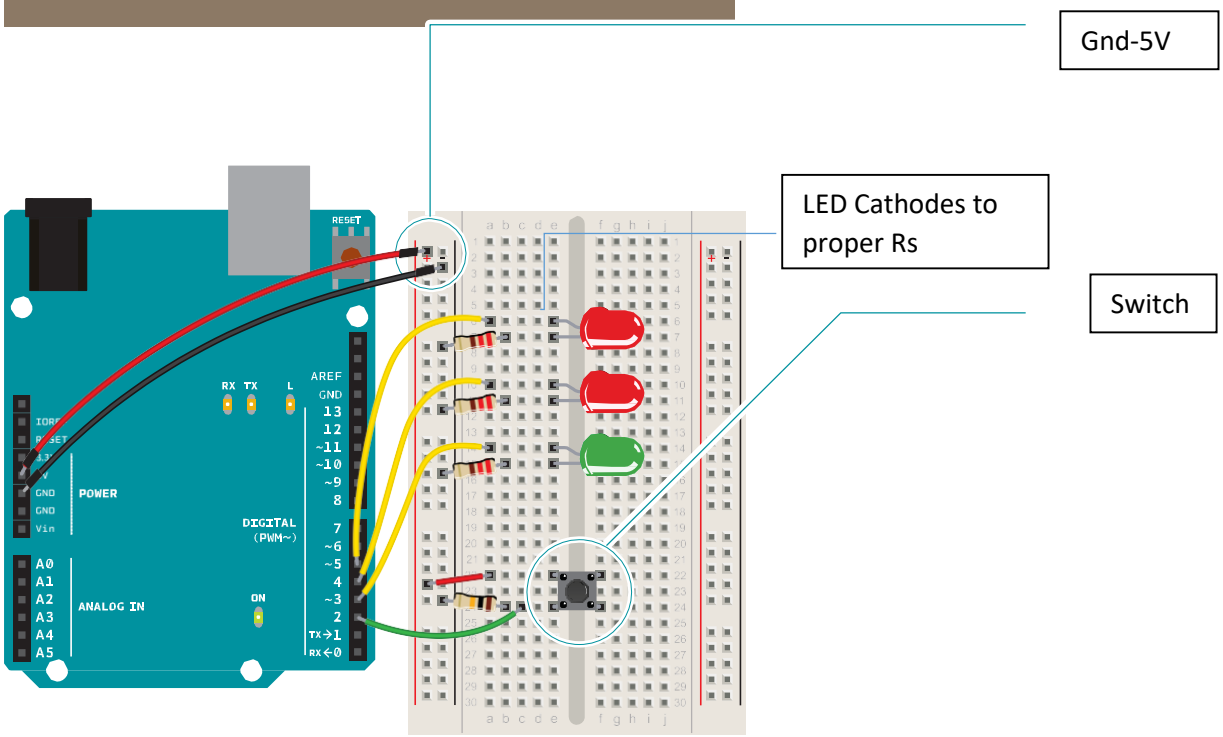
2 My first code

The Arduino's digital pins can read only two states: when there is voltage on an input pin, and when there's not. This kind of input is normally called digital (or sometimes binary, for two-states). These states are commonly referred to as **`HIGH`** and **`LOW`**.

When you turn an `OUTPUT` pin `HIGH` using a command called `digitalWrite()`, you're turning it on. Measure the voltage between the pin and ground, you'll get 5 volts. When you turn an `OUTPUT` pin `LOW`, you're turning it off.

The Arduino's digital pins can act as both inputs and outputs. In your code, you'll configure them depending on what you want their function to be. When the pins are outputs, you can turn on components like LEDs. If you configure the pins as inputs, you can check if a switch is being pressed or not. Since pins 0 and 1 are used for communicating with the computer, it's best to start with pin 2.





```
// Create a global variable to hold the state of the switch. This variable is
// persistent throughout the program. Whenever you refer to switchState, you're
// talking about the number it holds
int switchstate = 0;

void setup() {
  // declare the LED pins as outputs
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);

  // declare the switch pin as an input
  pinMode(2, INPUT);
}

void loop() {

  // read the value of the switch
  // digitalRead() checks to see if there is voltage on the pin or not
  switchstate = digitalRead(2);

  // if the button is not pressed turn on the green LED and off the red LEDs
  if (switchstate == LOW) {
    digitalWrite(3, HIGH); // turn the green LED on pin 3 on
    digitalWrite(4, LOW);  // turn the red LED on pin 4 off
    digitalWrite(5, LOW);  // turn the red LED on pin 5 off
  }
  // this else is part of the above if() statement.
  // if the switch is not LOW (the button is pressed) turn off the green LED and
  // blink alternatively the red LEDs
  else {
    digitalWrite(3, LOW);  // turn the green LED on pin 3 off
    digitalWrite(4, LOW);  // turn the red LED on pin 4 off
    digitalWrite(5, HIGH); // turn the red LED on pin 5 on
    // wait for a quarter second before changing the light
    delay(250);
    digitalWrite(4, HIGH); // turn the red LED on pin 4 on
    digitalWrite(5, LOW);  // turn the red LED on pin 5 off
    // wait for a quarter second before changing the light
    delay(250);
  }
}
```

Every Arduino program has two main functions. Functions are parts of a computer program that run specific commands. Functions have unique names, and are "called" when needed. The necessary functions in an Arduino program are called **setup()** and **loop().** These functions need to be declared, which means that you need to tell the Arduino what these functions will do.

In this program, you're going to create a variable before you get into the main part of the program. A variable named **switchState** tells you what it stores: the state of a switch. To create a variable, you need to declare what type it is. The data type **int** will hold a whole number (also called an integer). When you declare a variable, you usually give it an initial value as well. **The declaration of the variable as every statement must end with a semicolon (;).**

The **setup() runs once**, when the Arduino is first powered on. This is where you configure the digital pins to be either inputs or outputs using a function named **pinMode()**. The pins connected to LEDs will be OUTPUTs and the switch pin will be an INPUT.
The **loop() runs continuously after the setup() has completed**. The loop() is where you'll check for voltage on the inputs, and turn outputs on and off. To check the voltage level on a digital input, you use the function **digitalRead()** that checks the chosen pin for voltage. To know what pin to check, digitalRead() expects an argument. Arguments are information that you pass to functions, telling them how they should do their job. For example, digitalRead() needs one argument: what pin to check. In your program, digitalRead() is going to check the state of pin 2 and store the value in the switchState variable. If there's voltage on the pin when digitalRead() is called, the switchState variable will get the value HIGH (or 1). If there is no voltage on the pin, switchState will get the value LOW (or 0).
An if() statement in programming compares two things, and determines whether the comparison is true or false. Then it performs actions you tell it to do. When comparing two things in programming, you use two equal signs ==. If you use only one sign, you will be setting a value instead of comparing it.

digitalWrite() is the command that allows you to send 5V or 0V to an output pin. digitalWrite() takes two arguments: what pin to control, and what value to set that pin, HIGH or LOW.
Describe the operation of the code.
After setting the LEDs to a certain state, you'll want the Arduino to pause for a moment before changing them back. If you don't wait, the lights will go back and forth so fast that it will appear as if they are just a little dim, not on and off. This is because the Arduino goes through its loop() thousands of times each second, and the LED will be turned on and off quicker than we can perceive. The **delay()** function lets you stop the Arduino from executing anything for a period of time.
delay() takes an argument that determines the number of milliseconds before it executes the next set of code. delay(250) will pause for a quarter second.