Using these cards

These cards each explain how to get a basic project, or component, setup with an Arduino.

You should start with the **Arduino - Intro** document, to get setup instructions.

Then read the **Arduino - Basics** document to understand the basics of Arduino setup and coding.

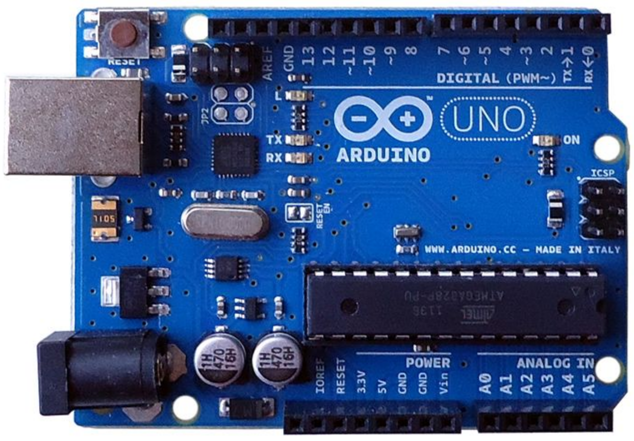
After that, pick any project that interests you, or component you have, and get hacking!

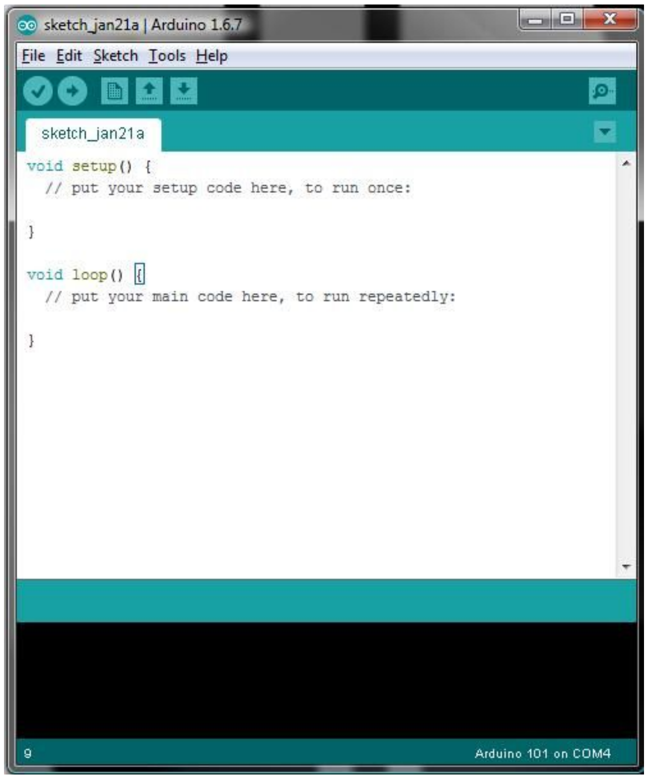
Arduino

The Arduino is a microcontroller which is used to control Hardware, It repeats the same code over and over forever unless it is turned off or reset.

The Arduino is expandable with many Shields which increase Functionality and multiple variants which offer more features.

The Arduino Uno, the most common variant has 13 Digital Pins which can be set to On or Off and 6 Analog Pins which can be set from 0 Volts up to 5 Volts.
There are 3 Ground Pins and one 5 Volt and one 3.3 Volt pin.





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

}

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

}
```

To Program with the Arduino Uno we use a tool called Arduino IDE(Integrated Development Environment) which is Open Source and compatible with most Desktop Operating Systems.
You can download the IDE here.
https://www.arduino.cc/en/Main/Software

Arduino - Basics

There are a few basics that should be kept in mind while programming with the Arduino, such as the Basic Structure of Arduino Code…

```
/* Each Arduino sketch must contain the
following two functions. */
void setup(){
/* this code runs once at the beginning of
the code execution. */
}

void loop(){
/* this code runs repeatedly over and over
as long as the board is powered. */
 }
```

Setting up pins...

```
pinMode(pin, [INPUT \ OUTPUT \ INPUT_PULL-UP]);
/* There are three options for setting up Pins, as an Input, an output or an
input with an internal Pull Up resistor. */
```

If, Else, and If Else Statements…

```
if(Condition is True){
Run this code;
} else if(Last condition wasn’t true but this one is) {
Run this code;
} else {
Run this code instead;
}

/* These type of arguments are very important in coding, they tell the
Arduino what to do if something happens such as if you press a button it will
run the code you told it to run if it detects the button was pressed. */
```

For Loops…

```
for(x <= 100){
digitalWrite(13, HIGH);
delay 100;
digitalWrite(13, LOW);
delay 100;
x++;
}

/* This code tells the Arduino that if the variable x is less than or equal
to 100 that it should turn on the LED on pin 13 for 100 Milliseconds, then
turn it off and then add 1 to the variable x.
This will make the LED blink 100 times. This is useful for repeating
something a lot of times without having to Write a lot of lines of code. */
```

Adding Libraries…

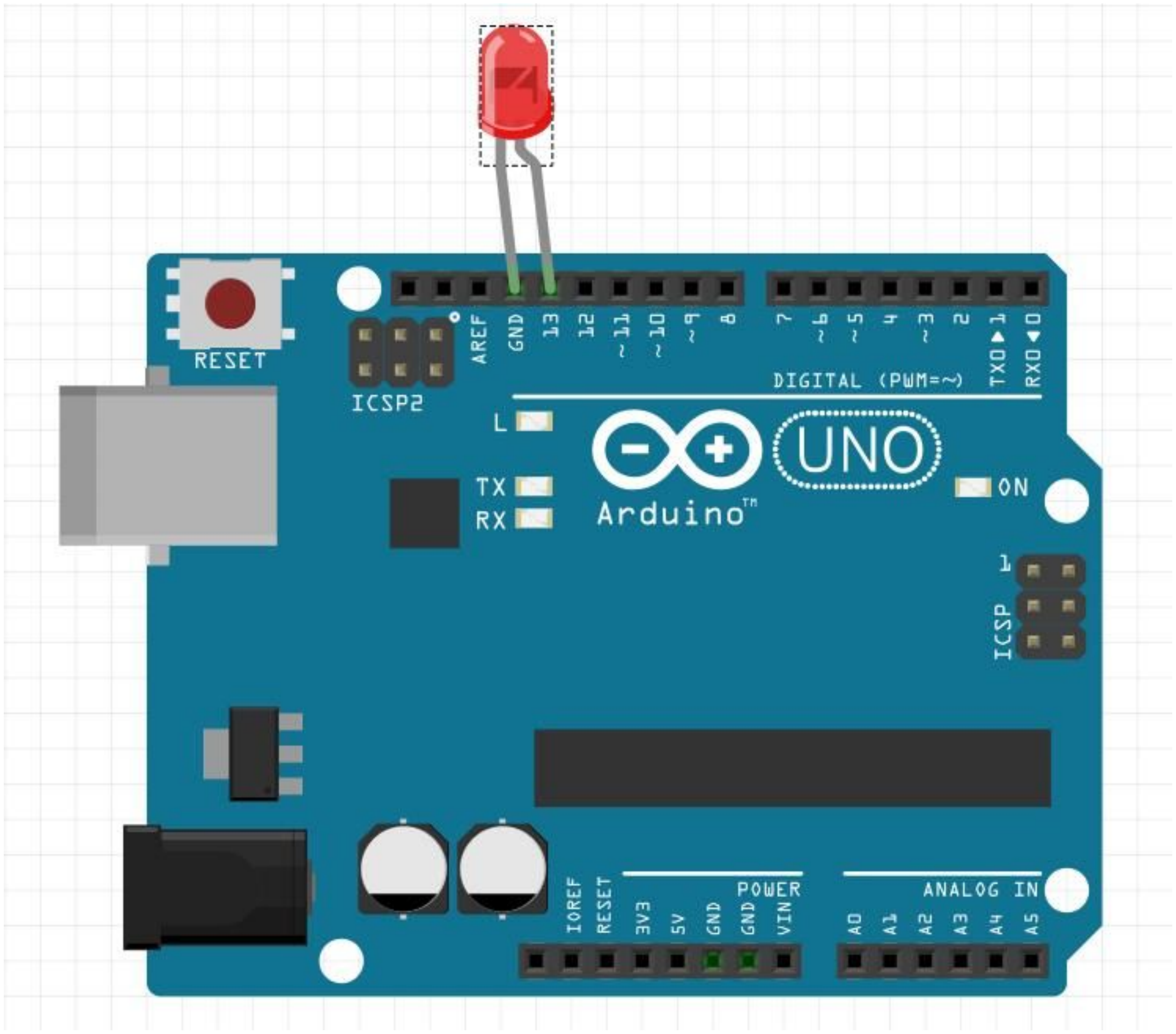
```
#include <nameOfLibrary.h>

/* Libraries are very useful bits of pre written code that expand the
Arduino’s functionality. */
```

Arduino - Blink

For this Tutorial you will need….

- An Arduino or Arduino Compatible Board
- An LED (Optional)

The Circuit….



- Connect the Positive Leg of the LED to Pin 13.
- Connect the Negative Leg of the LED to GND.

The Code….

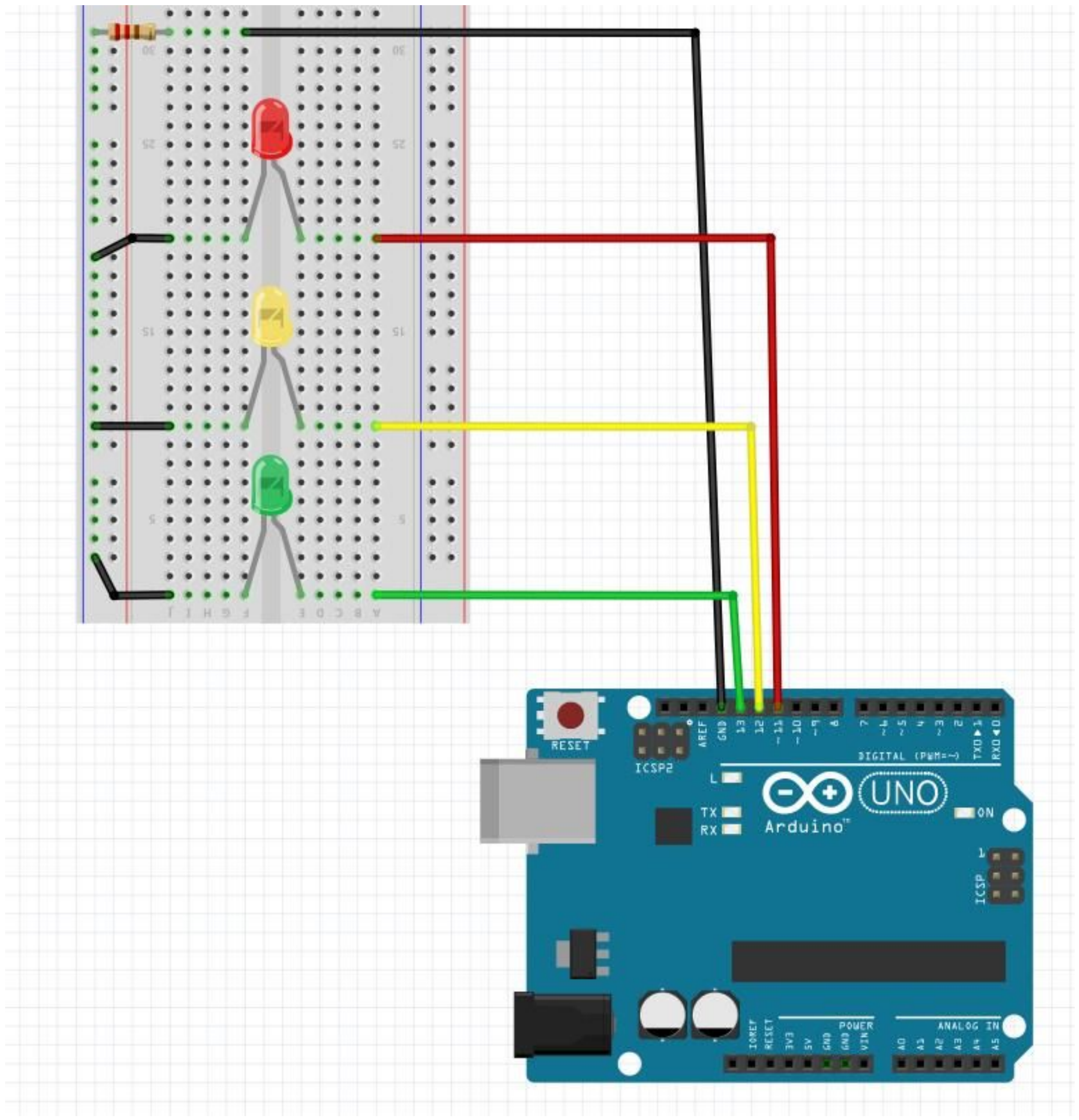
```
 // the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

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

Arduino-Traffic Lights

What you'll need....

- 3 LEDs; Red, Yellow and Green
- A Resistor; either 220 Ω or 330 Ω
- Jumper Wires
- A Breadboard

The Circuit….



- Connect the Green LED to pin 13
- Connect the Yellow LED to pin 12
- Connect the Red LED to pin 11
- Connect the Negative legs of the LEDs to the ground rail on the Breadboard
- Connect a Resistor to the Ground Rail on the Breadboard
- Connect the resistor to GND
- The Circuit is Complete!

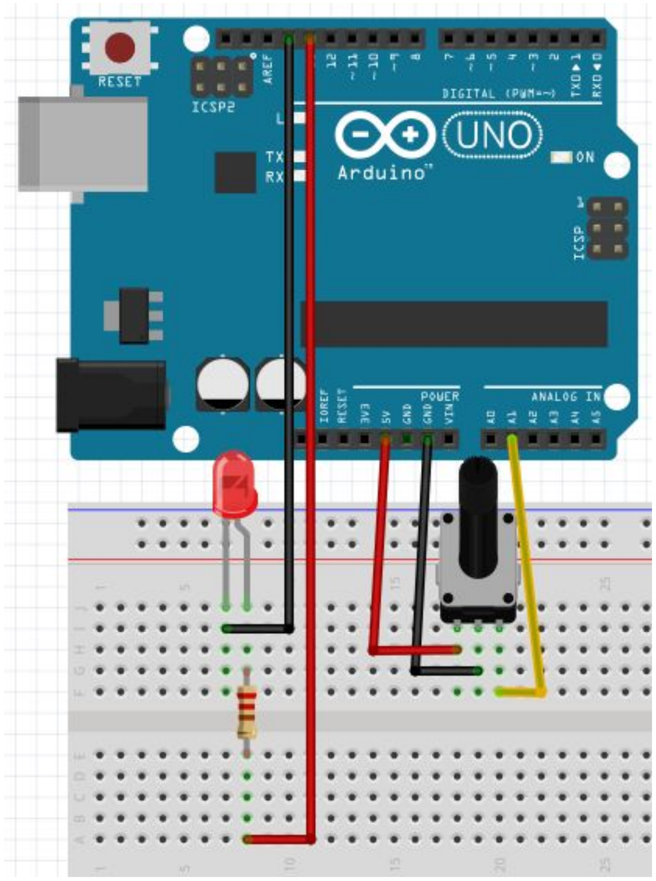
The Code….

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
  // initialize digital pin 12 as an output.
  pinMode(12, OUTPUT);
  // initialize digital pin 11 as an output.
  pinMode(11, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);      //Turn on the Green LED
  delay(1000);                //Wait 1000 milliseconds (1 Second)
  digitalWrite(13, LOW);      //Turn off the Green LED
  digitalWrite(13, HIGH);      //Turn on the Yellow LED
  delay(1000);                //Wait 1000 milliseconds (1 Second)
  digitalWrite(13, LOW);      //Turn off the Yellow LED
  digitalWrite(13, HIGH);      //Turn on the Red LED
  delay(1000);                //Wait 1000 milliseconds (1 Second)
  digitalWrite(13, LOW);      //Turn off the Red LED
}
```

Controlling an LED with a Potentiometer - Arduino

What you'll need…
An Arduino

- A Breadboard
- Jumper Wires
- A Potentiometer/Variable Resistor
- An LED
- A 220 Ohm Resistor



The Circuit…

- Connect the 5V wire to the left pin
- Connect the A1 wire to the center pin
- Connect the GND wire to the right pin
- Connect the LED to Pin 13 and ground with a resistor

The Code…

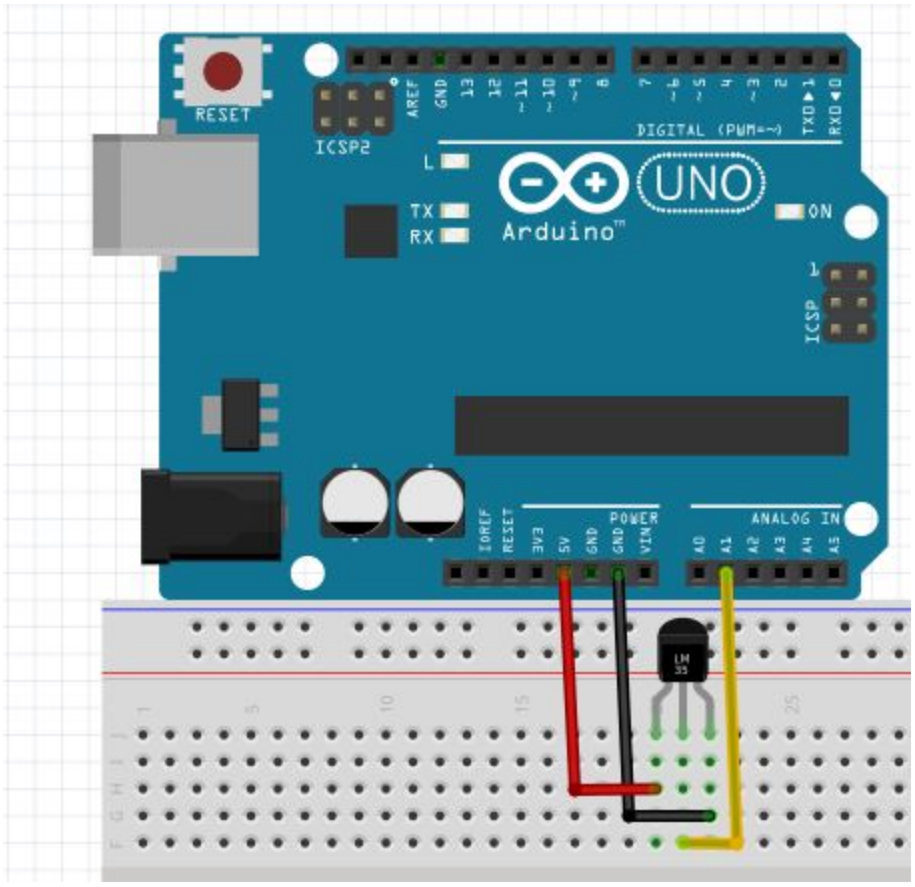
```
int potPin = 0;         // select the input pin for the potentiometer
int ledPin = 13;        // select the pin for the LED
int val = 0;            // variable to store the value coming from the sensor

void setup() {
  pinMode(ledPin, OUTPUT);  // set up the LED
}

void loop() {
  val = analogRead(potPin);         // read the value from the Potentiometer
  digitalWrite(ledPin, HIGH);       // turn the led on
  delay(val);                       // stop the program for the value of the Potentiometer
  digitalWrite(ledPin, LOW);        // turn the led off
  delay(val);                       // stop the program for the value of the Potentiometer
}
```

Arduino - Temperature Sensor

What you'll need…

- An Arduino
- A breadboard
- Jumper Wires
- An LM35 or similar temperature sensor

The Circuit…



- With the flat side of the sensor facing you connect the 5v wire to the left pin
- Then connect the A1 signal wire to the center pin
- Then connect the GND wire to the right pin

The code…

```
int val;
int tempPin = 1;       //set up the variables

void setup(){

Serial.begin(9600);  //set up a serial monitor

}

void loop(){

val = analogRead(tempPin);         //read the voltage coming to analog pin 1
float mv = ( val/1024.0)*5000;

/*divide the voltage detected by 1024 and then multiply it by 5000 */

float cel = mv/10;     //divide by 10 for the temperature in degrees celsius

Serial.print("TEMPERATURE = ");          //print temperature
Serial.print(cel);                       //then the value of the variable "cel"
Serial.print("*C");              //then put "*C" after to show that it's in degrees celsius
Serial.println();                //then go to the next line in the console
delay(1000);                     //wait 1 second before updating the temperature
}
```

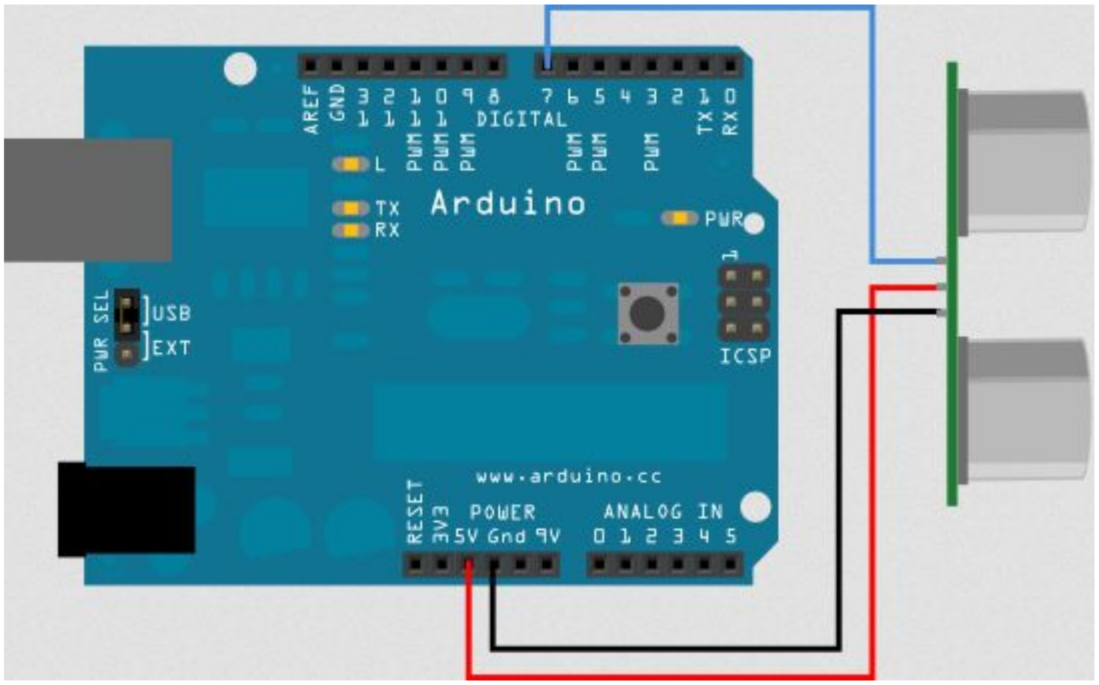
To see the temperature go to tools/Serial Monitor or press Ctrl+Shift+M

Arduino - Ultrasonic Range Sensor

What you'll need…

- An Arduino
- Jumper wires
- An Ultrasonic Range Sensor, the PING))) or Adafruit sensors will work fine

The Circuit…



- Connect the 5v pin to Vcc
- Connect the Gnd pin to Gnd
- Connect Pin 7 to Sig or Trig

The Code…

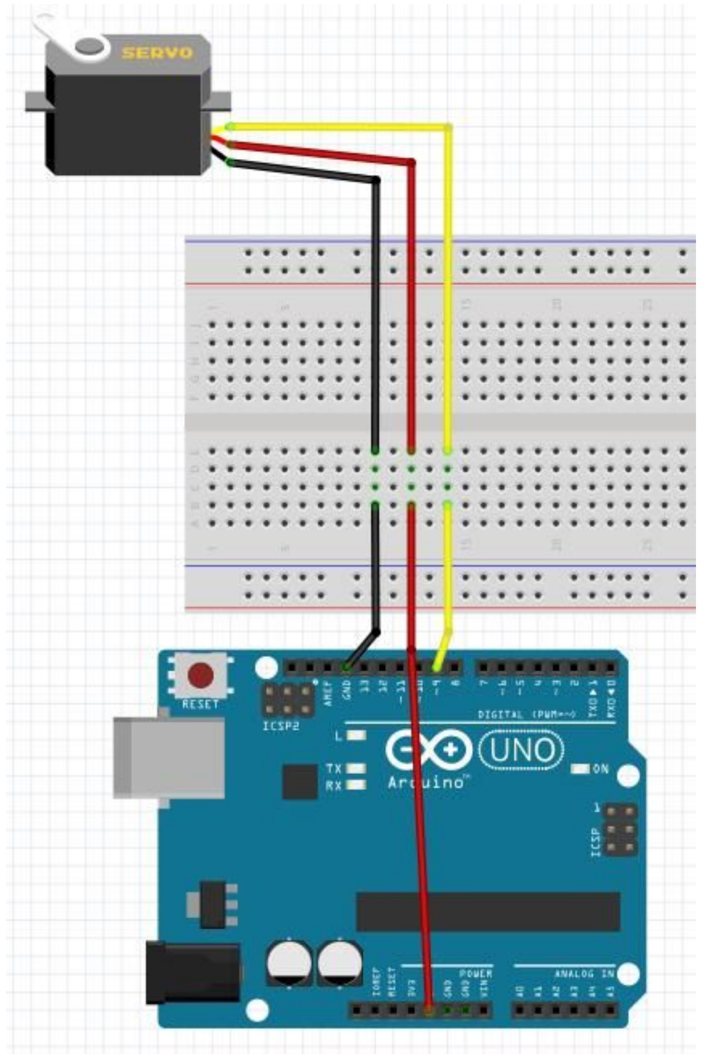
```
int pingPin = 7; // Use pin 7

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

void loop() {

  // The PING))) is triggered by a HIGH pulse of 2 or more microseconds.
  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
  pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(pingPin, LOW);
  pinMode(pingPin, INPUT);
  duration = pulseIn(pingPin, HIGH);

  cm = microsecondsToCentimeters(duration);

  Serial.print(cm);
  Serial.print("cm");  //Print the Variable “cm”
  Serial.println();
  delay(100);
}

long microsecondsToCentimeters(long microseconds) {
return microseconds / 29 / 2;
/* The speed of sound is 340 m/s or 29 microseconds per centimeter so to get the
distance we divide the time needed for the echo to reach the sensor by 2. */
}
```

Arduino - Servo

What you'll need….

- A servo motor
- A breadboard
- Jumper Wires

The Circuit….



The Code….

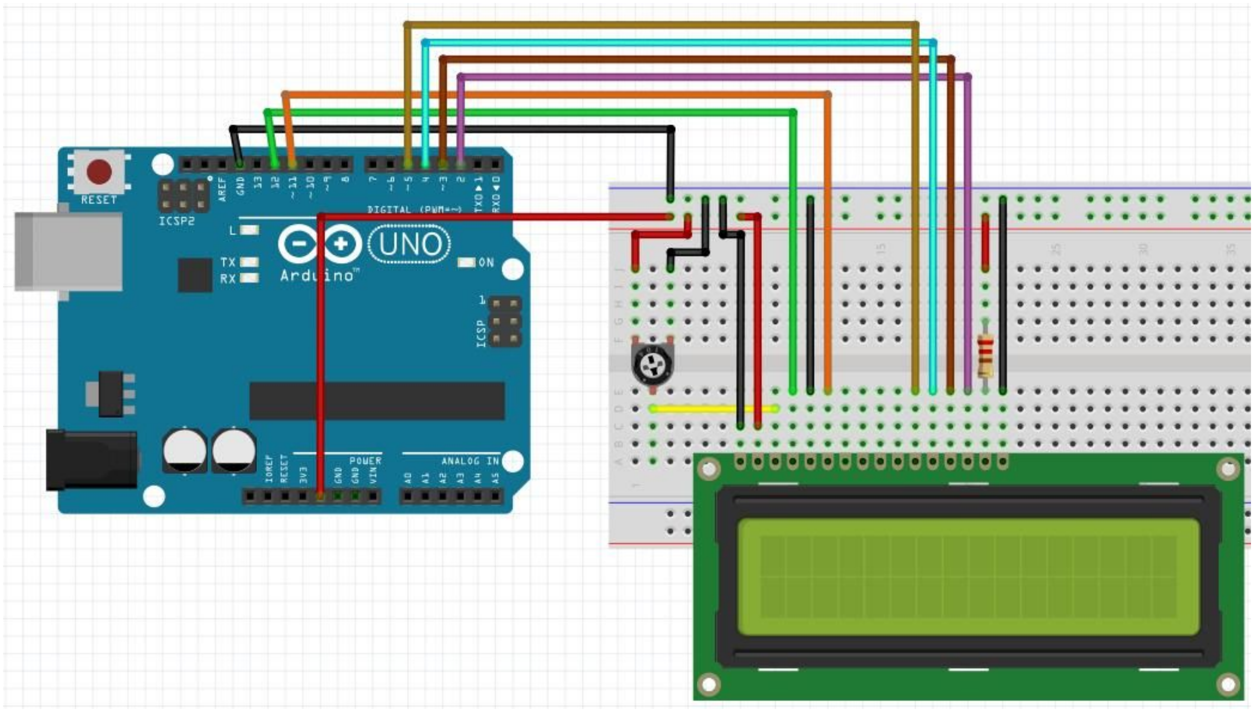
```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo
                // a maximum of eight servo objects can be created

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

void setup()
{
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}


void loop()
{
  for(pos = 0; pos < 180; pos += 1)  // goes from 0 degrees to 180 degrees
  {                                  // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for(pos = 180; pos>=1; pos-=1)     // goes from 180 degrees to 0 degrees
  {
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
}
```

Arduino - LCD

What you'll need….

- LCD display (I used an LCM1602C)
- 10k Potentiometer
- 220 Ohm Resistor
- Jumper Wires
- A breadboard

The Circuit…



The code….

```
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

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() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
}
```