Lab 3
Some challenges and extra circuits

Objective

The aim of this lab is to enable students to read the sensors/actuators datasheet and create a circuit without any help. You will find a series of challenges to be completed with the help of your instructors.

After completing your challenges, you can create additional circuits.

The components

- Arduino
- Step Motor
- Jumper wires
- Breadboard
- DC motor
- LEDs
- RGB LED
- Resistors
- KY-037 (Big Sound sensor)
- KY-033 (Tracking sensor)
- KY-032 (Infrared avoidance sensor)
- KY-036 (Human touch sensor)
- KY-028 (Digital temperature sensor)
- KY-039 (Heartbeat sensor)
- KY-002 (Vibration sensor)
- KY-040 (Rotary sensor)

- KY-017 (Tilt switch)
- KY-020 (Ball switch)
- KY-018 (Photo resistance sensor)
- KY-015 (Temperature and Humidity sensor)
- KY-005/KY-022 (IR transmetter/receiver)
- KY-031 (Knock sensor)

Your instructor will **randomly** assign you the sensors and you will choose an actuator of your choice to create your circuit.

In order to obtain the datasheet of the modules mentioned above, you will need to access to the following website: https://datasheetspdf.com/

Note

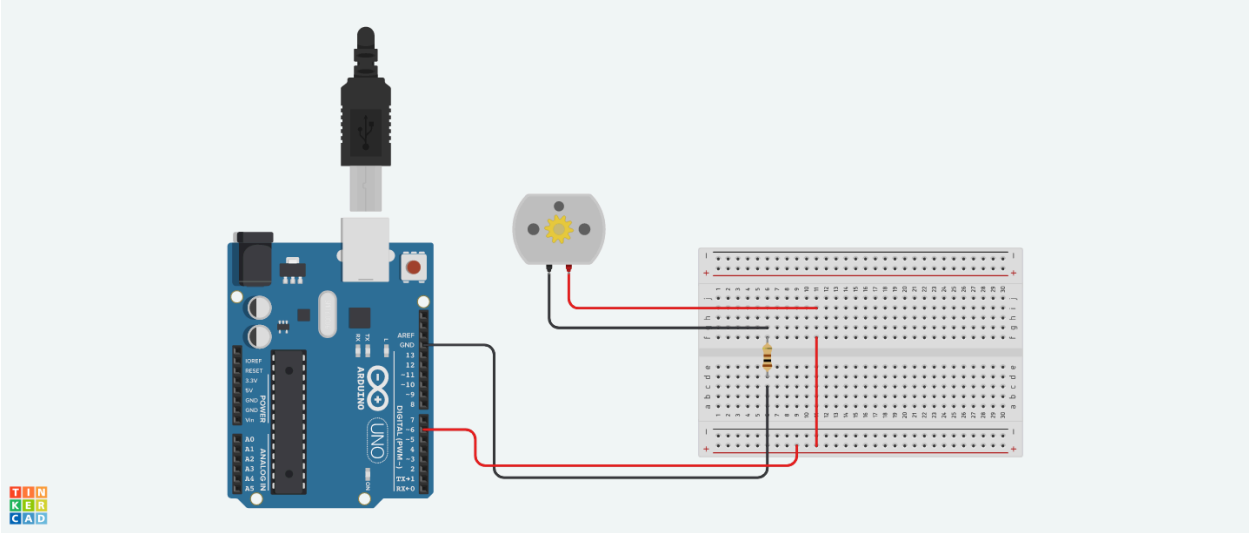
You can find on the Internet a lot of datasheet websites

Bonus implementation 1

In this implementation, you will learn how to control the speed of a DC motor using an input value given by the user using the Serial Monitor.

Create the Arduino circuit

We must create this circuit:



Then, we write the following source code:

```
1.   int basePin = 5;
2.   int speed;
3.   void setup()
4.   {
5.     pinMode(basePin, OUTPUT);
6.     Serial.begin(9600);
7.     while(!Serial);
8.     Serial.println("speed of motor");
9.   }
10.  void loop()
11.  {
12.     if(Serial.available()) //to input the data
13.      {
14.        speed = Serial.parseInt(); //to convert data to Integer
15.        if (speed >=0 && speed <=255)
16.         {
17.           analogWrite(basePin, speed);
18.         }
19.      }
20.  }
```

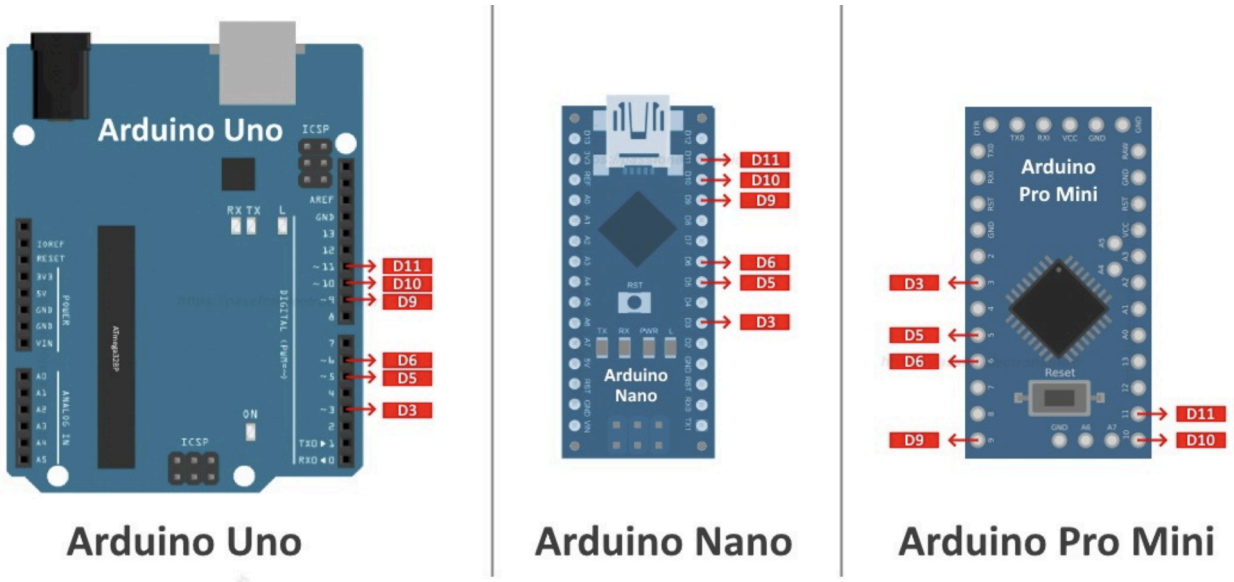
How it works

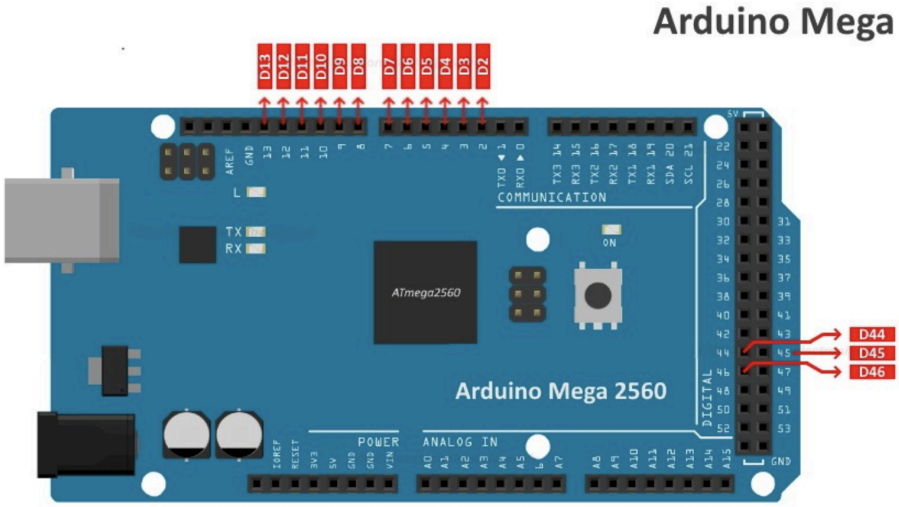
So far, we have used **PINs** with the **digitalWrite** functions, which send **LOW** or **HIGH** values.

In this implementation, we are going to use the PINs that allow us to send analog signals: these are the **PWM PINs**.

PWM - Pulse Width Modulation, defines an electrical signal with a given frequency, whose duty cycle can vary over time. The duty cycle is the ratio between the time when the signal is in the "high state" and the time of a complete period of this signal (the sum of the time spent in the "high state" + the time spent in the "low state", in fact).

In our case, we will use PIN number 5, with the ~ symbol in front (this may vary from one arduino type to another).





In order to use the **PWM PINs**, we need to declare the output Pin with the **`pinMode()`** function and use the **`analogWrite(duty cycle value)`** function, where the **`duty cycle value`**, in the arduino code, is expressed as a number between **0** and **255**. So, in theory :

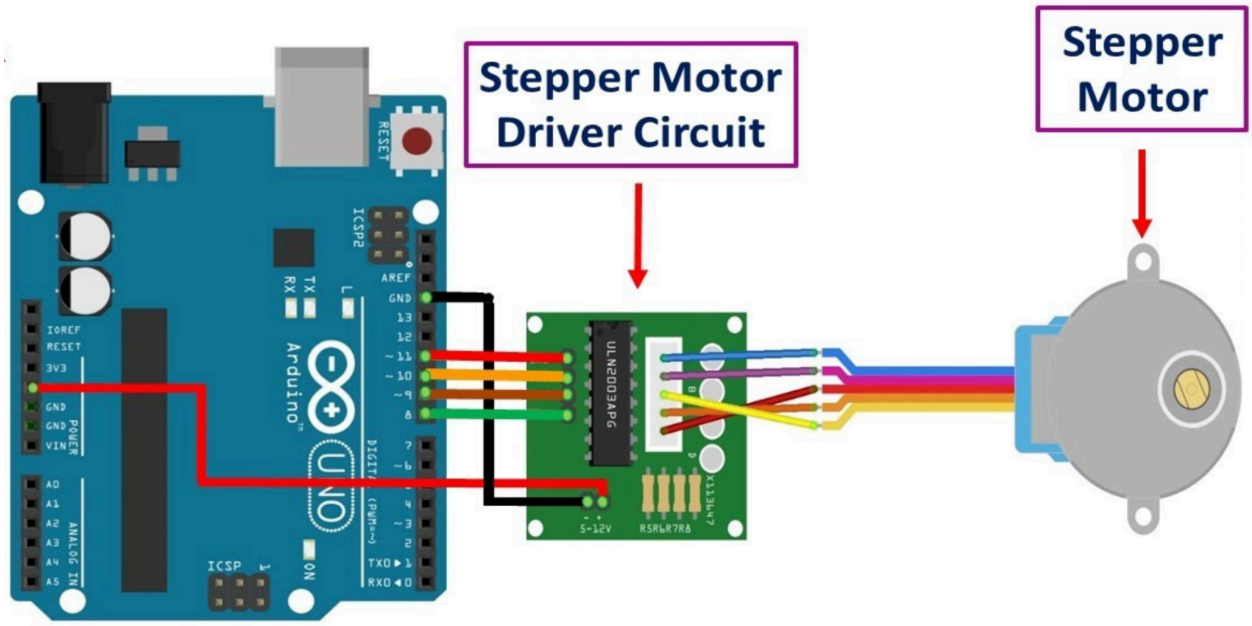
- A duty cycle of **0** means no PWM signal at the output (i.e. a permanent low state, equal to 0 volts).
- And a duty cycle of **255** equates to a 100% PWM signal (i.e. a permanent high state, e.g. equal to +5V if your microcontroller operates at 5 volts).

Bonus implementation 2

In this implementation, we will learn how to manipulate a **stepper motor**. It can be controlled to a high degree of accuracy without any feedback mechanisms. The shaft of a stepper, mounted with a series of magnets, is controlled by a series of electromagnetic coils that are charged positively and negatively in a specific sequence, precisely moving it forward or backward in small "steps".

Create the Arduino circuit

We have to create the following circuit



The source code

Then, we write the following source code. For this implementation, we will use the Arduino **Stepper Library**, which comes with the Arduino IDE.

The Arduino stepper library handles the stepping sequence and allows you to control a wide range of unipolar and bipolar stepper motors.

```
#include <Stepper.h>
// Defines the number of steps per rotation
const int stepsPerRevolution = 2038;
// Creates an instance of stepper class
```

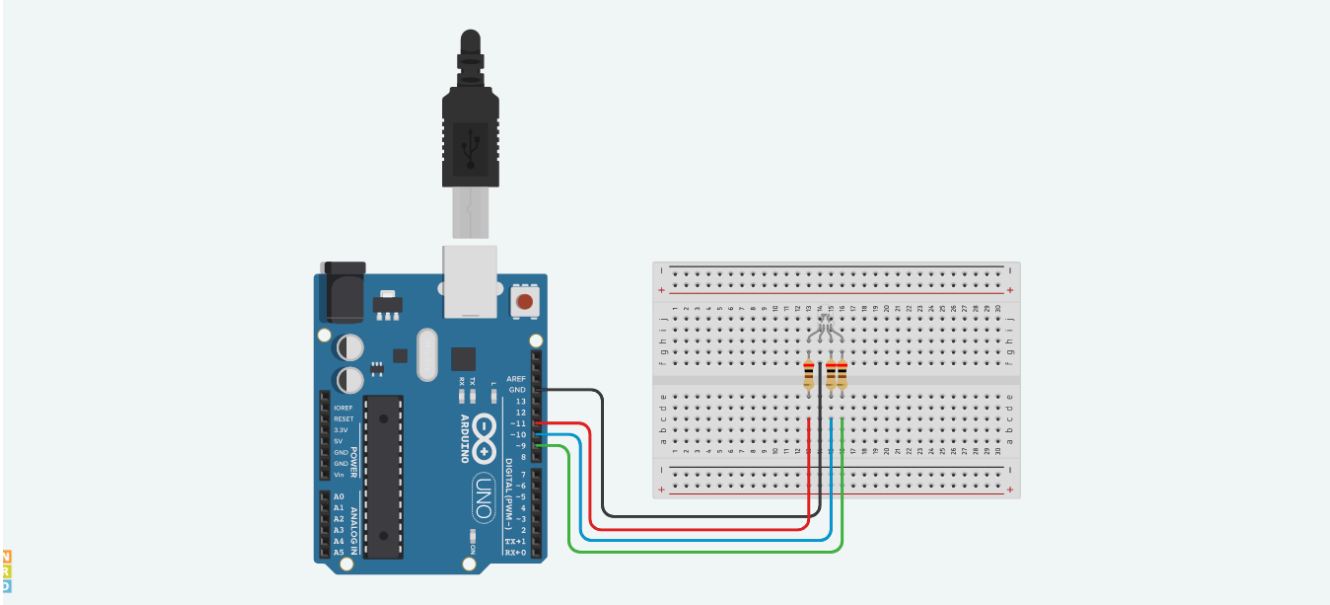
```
5.   // Pins entered in sequence IN1-IN3-IN2-IN4 for proper step
sequence
6.   Stepper myStepper = Stepper(stepsPerRevolution, 8, 10, 9, 11);
7.   void setup() {
8.   }
9.   void loop() {
10.    // Rotate CW slowly at 5 RPM
11.    myStepper.setSpeed(5);
12.    myStepper.step(stepsPerRevolution);
13.    delay(1000);
14.    // Rotate CCW quickly at 10 RPM
15.    myStepper.setSpeed(10);
16.    myStepper.step(-stepsPerRevolution);
17.    delay(1000);
18.  }
```

Bonus implementation 3

In this project, we will learn how to interface **RGB LED**. The RGB LED is controlled by **PWM signals**.

Create the Arduino circuit

We must create this circuit:



After that, use the following code:

```
1.   const int greenPin = 9;
2.   const int bluePin = 10;
3.   const int redPin = 11;
4.   void setup()
5.   {
6.     pinMode(greenPin, OUTPUT);
7.     pinMode(bluePin, OUTPUT);
8.     pinMode(redPin, OUTPUT);
9.   }
10.  void loop()
11.  {
12.    SetColor(255,0,0); //Red color
13.    delay(1000); // Wait for 1000 millisecond(s)
14.    SetColor(0,255,0);  //Green color
15.    delay(1000);
16.    SetColor(0,0,255);  //Blue color
17.    delay(1000);
18.    SetColor(255,255,255);  //White color
19.    delay(1000);
20.    SetColor(170,0,255);   //Purple color
21.    delay(1000);
22.    SetColor(50,100,90);   //turquoise color
23.    delay(1000);
24.  }
25.  void SetColor(int red, int green, int blue)
26.  {
27.    analogWrite(redPin, red);
28.    analogWrite(greenPin, green);
29.    analogWrite(bluePin, blue);
20.  }
```

Using **`analogWrite(PIN,value)`** function, we can control the intensity of the color by varying the second parameter: **`value`** (from 0 to 255).