Co-funded by the
Erasmus+ Programme
of the European Union

UNIT 10: Driving motors

AIMS

The aim of this unit is to Give the basic ideas and simple examples for driving DC and Stepper motors with Arduino.

THEORY SECTION

- Explain what is an electric motor
- Present what kind of motors we use in this unit
- Give the basic driving scheme of an electric motor
- Drive DC motors with a transistor and an H-Bridge
- Drive Stepper motors with two H-Bridges
- Explain The Stepper library for Arduino

PRACTICE SECTION

- EXAMPLE 10-1: DC motor drive with an NPN Transistor
- EXAMPLE 10-2: DC motor drive with the IC L293D
- EXAMPLE 10-3: A motor shield driver for Arduino board based on L298P Dual H-bridge
- EXAMPLE 10-4: Stepper motor drive with the IC L293D
- EXAMPLE 10-5: Stepper motor drive with the motor shield for Arduino

MATERIALS REQUIRED

-Lap top or desk top computer
-Arduino IDE work environment; this should include the supplementary material already installed and configured.
-Arduino UNO controller board
-Arduino UNO motor shield
-The IC L293D
- A DC (PM) motor
-A stepper motor
-A PN2222A or TIP120 Transistor
-A 0,5KΩ resistor
- A 1N4001 Diode
-A USB cable

THEORY SECTION

10.1 DRIVING MOTORS

Electric motors impact almost every aspect of modern living. Refrigerators, vacuum cleaners, air conditioners, fans, computer hard drives, automatic car windows, and multitudes of other appliances and devices all use electric motors to convert electrical energy into useful mechanical energy. Electric motors are used at some point in the manufacturing process of nearly every conceivable product that is produced in modern factories. Because of the nearly unlimited number of applications for electric motors, it is not hard to imagine that there are over 700 million motors of various sizes in operation across the world. This enormous number of motors and motor drives has a significant impact on the world because of the amount of power they consume.

The systems that controlled electric motors in the past suffered from very poor performance and were very inefficient and expensive. In recent decades, the demand for greater performance and precision in electric motors, combined with the development of better solid-state electronics and cheap microprocessors has led to the creation of modern precise and cheap control systems.

In this unit we will briefly explain and present two different kind of electric motors that are wildly spread in industrial and mechatronic applications, the permanent magnet direct current motor and the stepper motor. The basic scheme for driving and controlling such motors will be also explained while a number of basic circuits will be given as application examples.

10.2 DC PERMANENT MAGNET (PM) DIRECT CURRENT (DC) MOTOR

An electric motor is an actuator, which converts electrical energy into mechanical energy. Most electric motors operate through the interaction between an electric motor's magnetic field and winding currents, as shown in see figure 1, to generate force and eventually torque to the shaft of the motor. The simplest electric motor is the standard permanent magnet brushed motor, which is commonly used for high-speed applications, or high torque when gearing is used. Brushed direct current motors are also commonly used in gear motors and servo motors.

The Permanent Magnet Direct Current (PMDC) motor has only one motor coil with two wires for operation. This is by far the easiest type of motor to drive, control, and manipulate, also the basic idea of the motor is shown in figure 1. In order to drive them we apply a voltage to the motor terminals, the higher the voltage, the faster the spinning. We can also reverse the rotational direction of a DC motor output shaft by reversing the polarity of the voltage to the motor terminals. In other words, polarity determines which way it rotates.

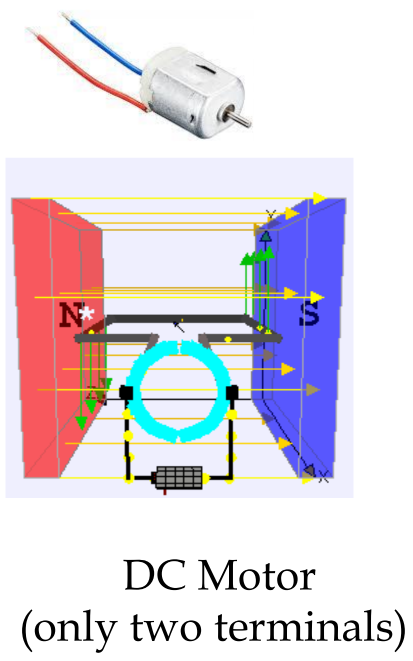


Figure 1: The basic principle of a DC Motor

<u>10.3 THE BASIC IDEA FOR DRIVING AND CONTROLLING A PMDC MOTOR</u>

The basic driving scheme for a DC motor in shown in figure 2. In fact, to control a DC or stepper motor high current signals (HCS) are required (e.g. from 500mA up to 6A or more) depending on their size and power rating. However, microcontrollers in their outputs can handle low current control signal (LCCS) that are less than 50mA. Therefore we utilize motor drivers that are able to provide the appropriate high current signals and are comprised of simple transistors or more complicated integrated circuits like the well-known H-Bridge circuit. The drivers are also able to change the direction of rotation and can be combined in open or closed loop systems. In this unit we are concerned with open loop systems that are simpler to understand, easier to implement and cheaper to buy in the market.

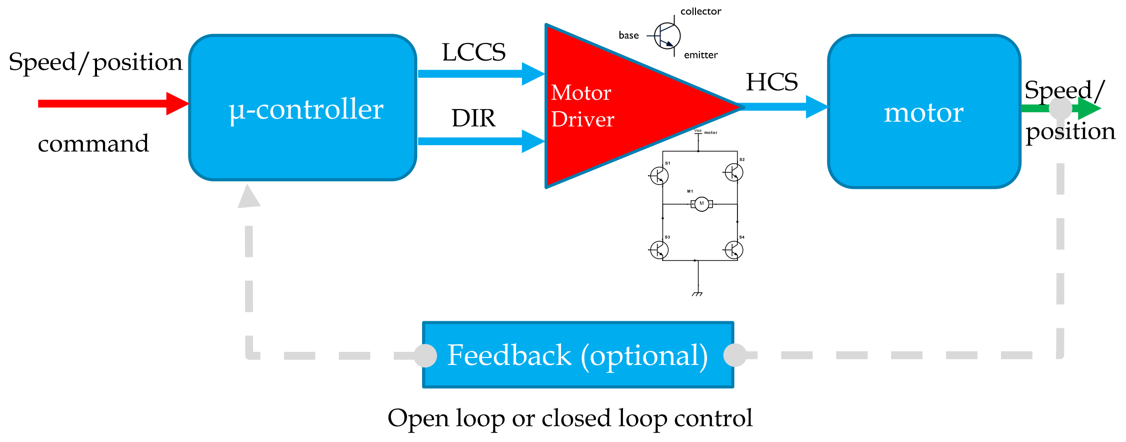


Figure 2: The basic driving scheme to control an electric motor

In order to control the speed of the PMDC motor you have to simply control the input voltage to the motor and the most common method of doing that is by using a PWM signal. Pulse width modulation is a technique which allows us to adjust the average value of the voltage that is going to the electronic device by turning on and off the power at a fast rate. The average voltage depends on the duty cycle, or the amount of time the signal is ON versus the amount of time the signal is OFF in a single period of time as shown in figure 3.

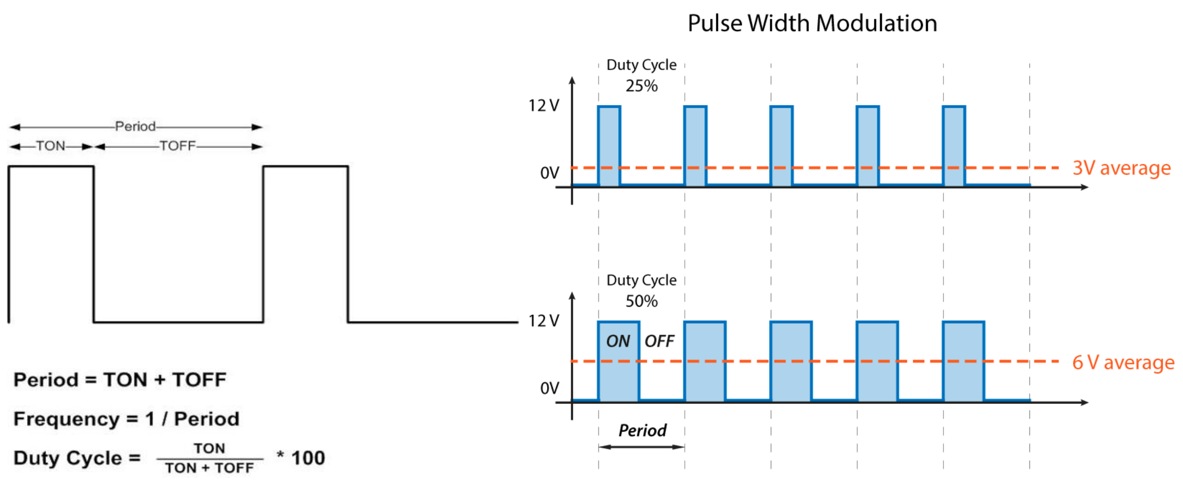


Figure 3: The basic idea of the PWM signal

So depending on the size of the motor, we can simply connect an Arduino PWM output to the base of a transistor and control the speed of the motor by controlling the PWM output. The low power Arduino PWM signal switches ON and OFF the transistor through which the high power motor is driven by a high current signal provided by the external power supply of the motor as shown in figure 4.

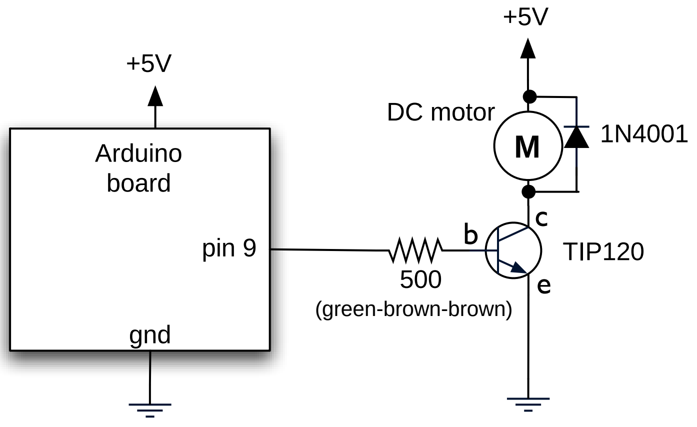


Figure 4: The basic circuit to control a PMDC motor idea to control an electric motor

On the other hand, for controlling the direction of rotation, we just need to inverse the direction of the current flow through the motor, and the most common method of doing that is by using an H-Bridge. An H-Bridge circuit contains four switching elements with the motor at the center forming an H-like configuration as shown in figure 5. By activating two particular switches at the same time we can change the direction of the current flow, thus change the rotation direction of the motor.

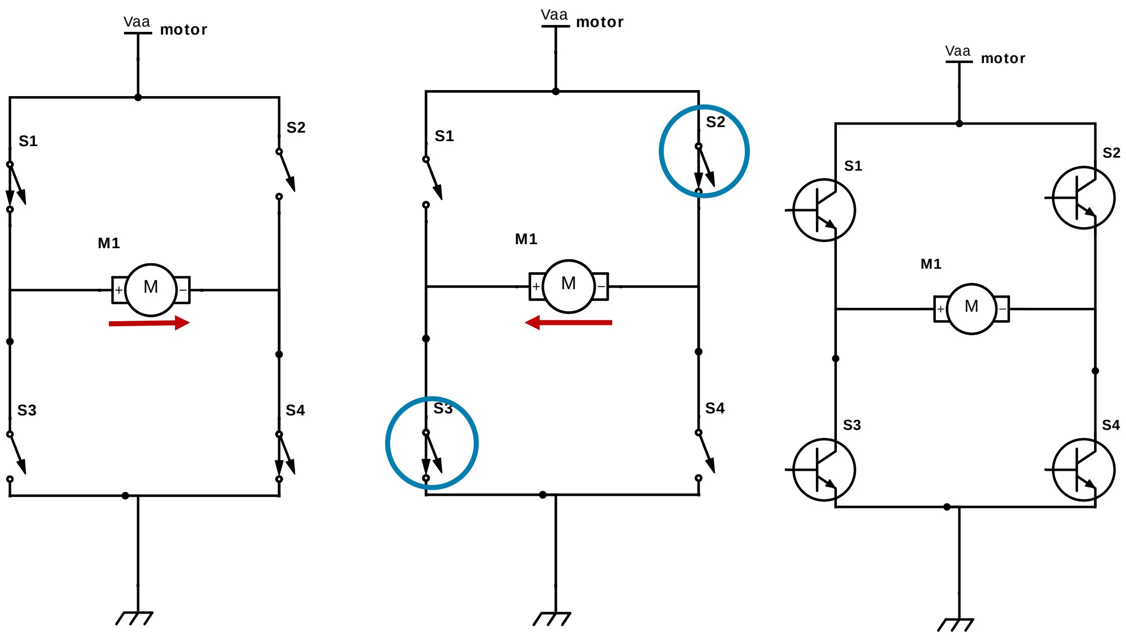


Figure 5: Controlling the direction of rotation for the PMDC motor by utilizing an H-Bridge.

For example closing switches S_1-S_4 causes current to flow through the motor in one direction, making the motor spin clockwise. In contrast closing switches S_2-S_3 causes current to flow through the motor in the opposite direction, making the motor to spin counterclockwise. In practice switches are replaced by transistors (i.e. transistors S_1,S_2,S_3,S_4), so a monolithic integrated circuit (IC) is used like L293D and L298 that are shown in figure 6. Many times ICs can be incorporated in a simple printed circuit board (PCB) or as an Arduino-based shield, which is based on the IC L298P Dual H-bridge. In general H-bridge acts as a current amplifier and also can be used to drive coils or stepper motors.

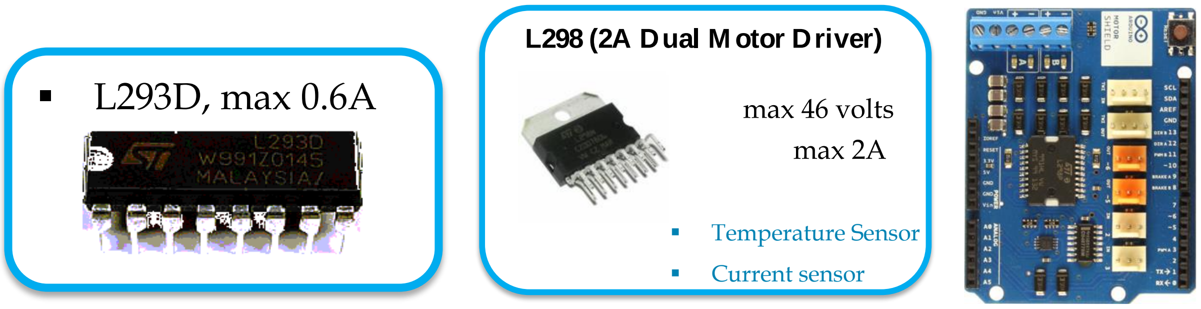
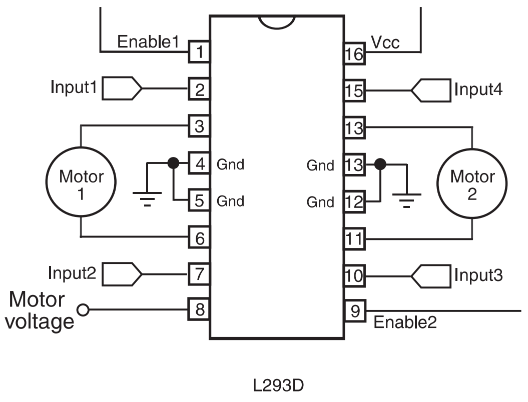


Figure 6: Commercial L293D and L298 ICs along with the Arduino shield

The schematic diagram for controlling two DC motors with the IC L293D is shown below in figure 7. In order to control the direction of motor 1 we must define the *Input1* and *Input2* along with *Enable1* according to the values of the table shown in figure 7.



Motor 1			
Enable	**Input 1**	**Input 2**	**Motor Action**
LOW	either	either	Low stop
HIGH	LOW	LOW	Brake
HIGH	LOW	HIGH	forward
HIGH	HIGH	LOW	backword
HIGH	HIGH	HIGH	Brake
PWM	LOW	LOW	Pulse Brake
PWM	LOW	HIGH	Forward-spd
PWM	HIGH	LOW	Backword-spd
PWM	HIGH	HIGH	Pulse Brake

Figure 7: Schematic diagram for the L293D IC along with its function table

In general, the *Enable1* and *Enable2* pins are used for enabling and controlling the speed of the motors 1 and 2 respectively. Next, the pair of pins *Input1* and *Input2* are used for controlling the rotation direction of the motor A, and the *inputs3* and *4* for the motor *2* respectively. Using these pins we actually control the switches of the H-Bridge inside the

L293D or L298 ICs. If *input1* is LOW and *input2* is HIGH the motor will move forward, and vice versa, if *input1* is HIGH and *input2* is LOW the motor will move backward. In case both inputs are equal, either LOW or HIGH the motor will stop. The same applies for the *inputs 3* and *4* and the motor B. Therefore, if we combine the PWM method and the H-Bridge circuit we can have a complete control over the PMDC motor.

10.4 STEPPER MOTORS

A stepper motor is a motor controlled by a series of electromagnetic coils. The center shaft has a series of magnets mounted on it, and the coils surrounding the shaft are alternately given current or not, creating magnetic fields which repulse or attract the magnets on the shaft, causing the motor to rotate (Figure 8).

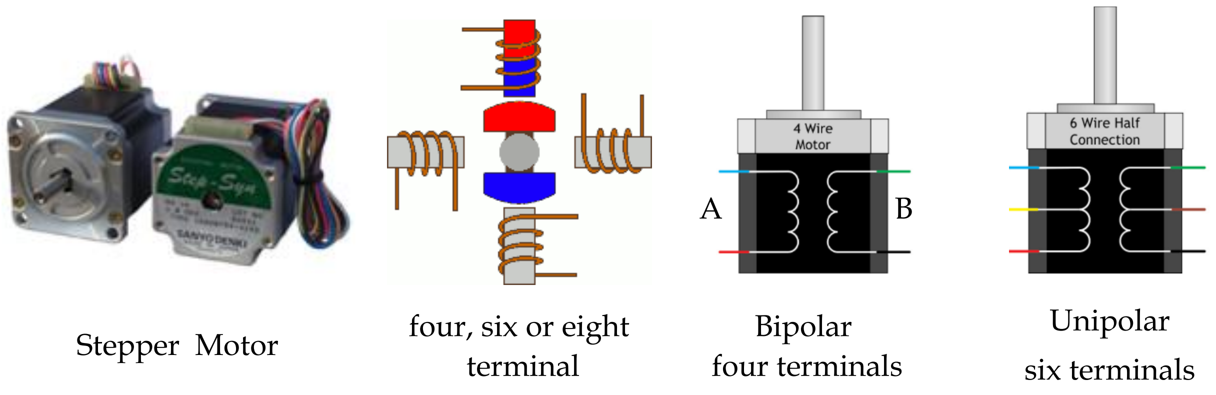


Figure 8: The basic principle of a stepper motor with 4 or six cables

Therefore, the shaft of a stepper motor rotates in discrete step increments when electrical command pulses are applied to it in the proper sequence. The proper energizing of the coils allows for very precise control of the motor that can be turned in very accurate steps of rotation increments that depends on the specifications of the stepper motor (e.g. 30°, 15°, 5°, 2.5°, 2° or 1.8°). Although the sequence of the applied pulses is related to the direction of motor shafts rotation, the speed of the motor shafts rotation is directly related to the frequency of the input pulses. Thus, a stepper motor can be a good choice whenever controlled rotational movement is required. They are used in scanners, computer printers, plotters, slot machines and more recently in 3D printers. There are two basic types of stepper motors, unipolar steppers and bipolar steppers.

The unipolar stepper motor has five or six wires and two coils divided by center connections on each coil. The center connections of the coils are tied together and used as the power connection. They are called unipolar steppers because power always comes in on this one pole (Figure 8).

The bipolar stepper motor has four wires (Figure 8) coming out of it that correspond in two independent sets of coils (A, B) and unlike unipolar steppers, bipolar steppers have no common center connection. We can distinguish them from unipolar steppers by measuring the resistance between the wires.

Stepper motors need a separate power supply provided by the manufacturer, but in many cases the rated supply voltage is not known. Thus, you can use a variable DC power supply, apply the minimum voltage (3V or so) across two wires of a coil and slowly raise the voltage until the motor is difficult to turn. Typical voltages for a stepper might be 5V, 9V, 12V, 24V.

10.5 THE BASIC IDEA TO DRIVE A STEPPER MOTOR WITH THE STEPPER LIBRARY

Stepper motors, due to their unique design, can be controlled to a high degree of accuracy without any feedback mechanisms. Two H-Bridge circuits can be combined in order to control either a unipolar or a bipolar stepper motor. Thus the ICs L293D, L298 or the motor shield for Arduino can be used. The schematic circuit for controlling the two coils A, B of a bipolar stepper motor with the L293D IC is given in the following figure 9. However, for a unipolar stepper motor the only difference is to connect the extra common terminals of the coils A' and B' (figure 9-b) with the rated supply voltage of the motor.

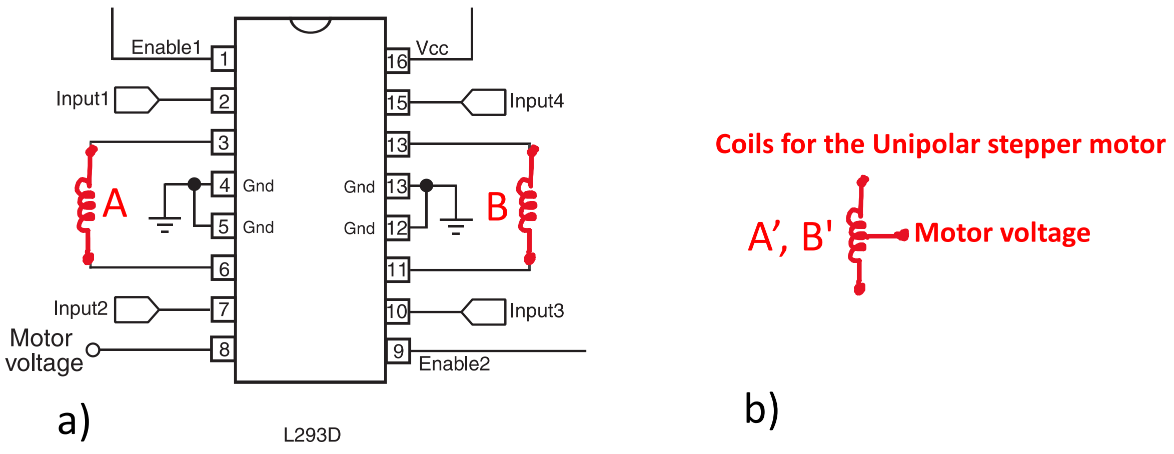


Figure 9: The basic connection of the L293D IC with the coils of a) a bipolar and b) a unipolar stepper motor

As we described in the previous section, the shaft of a stepper motor is controlled by a series of electromagnetic coils that are charged positively and negatively in a specific sequence. This can be implemented with the pins 2,7, 10 and 15 of the IC chip L293D, which is called the *four wire control of the stepper motor*. In addition, the pins 1, 9 and 16 are set to 5volts with the rated motor voltage in pin 8. Fortunately, the specific sequence can be easily implemented from functions that are utilized by the Stepper Library for Arduino (Stepper.h).

This library allows you to control unipolar or bipolar stepper motors with easy and intuitive way by using functions that are based on simple parameters like the number of wires used in the Arduino microcontroller, the number of steps per revolution, the maximum speed during revolute motion and the desired steps for the motor. These functions are given in the following figure 10 and allows you to control unipolar or bipolar stepper motors.

Functions that are utilized by the Stepper library *(stepper library for Arduino)*

```
#include <Stepper.h>
Stepper my_Stepper_1( value1_define_steps_per_revolution, pin1, pin2 )            Two wire control
        my_Stepper_1(value1_define_steps_per_revolution, pin1, pin2, pin3, pin4 )  Four wire control
my_Stepper_1.setSpeed(value2_define_max_speed_during_motion)  Set max speed during the motion of the motor
my_Stepper_1.step(value3_define_desired steps)                Move the rotor for the number of desired steps
```

Figure 10: The functions used in the stepper library for controlling a unipolar or bipolar stepper motor

For example the function **Stepper my_Stepper_1(** *value1_define_steps_per_revolution*, pin1, pin2 **)** or **Stepper my_Stepper_1(***value1_define_steps_per_revolution*, pin1, pin2, pin3, pin4 **)** creates a new instance of the **Stepper** class named ***my_Stepper1*** that represents a particular stepper motor attached to the Arduino board. The number of parameters (pin 1, pin2 or pin 1, pin2, pin3, pin4) depends on how we have wired our motor either using two or four control pins of the Arduino board. The parameter *value1_define_steps_per_revolution* corresponds to the number of steps in one revolution of the stepper motor. So, if it is known the number of degrees per step, by dividing that number into 360 we get the number of steps.

In addition, the function **my_Stepper_1**.**setSpeed**(value2_define_max_speed_during_motion) sets the motor speed in rotations per minute (RPMs). This function does not make the motor **my_Stepper_1** to turn, just sets the speed at which it will when we call the function **step()** which is explained next .

Finally, the function **my_Stepper_1.step(***value3_define_desired_steps***)** turns the motor a specific number of steps given by *value3_define_desired steps*, at a speed determined by the most recent call of **setSpeed()**. This function is blocking; that is, it will wait until the motor has finished moving to pass control to the next line in our sketch. For example, if we set the speed to 1 RPM and called step(100) on a 100-step motor, this function would take a full minute to run. For that reason it is better to keep the speed high and only go a few steps with each call of **step()**.

PRACTICE AREA: Example Programs

EXAMPLE 10-1: PMDC MOTOR DRIVE WITH THE NPN TRANSISTOR

In this example we are going to drive a PMDC motor with the use of a negative type transistor (NPN) transistor as shown in figure 11. The base of the transistor is connected through a 500 Ohm resistor to the pin 9 of Arduino microcontroller and a 1N2001 diode is connected in parallel with the PMDC motor for protecting the TIP120 transistor that is rated at 60V and 5A. In contrast, for a smaller motor a P2N2222AG transistor can be used that is rated at 40V and 200mA. The circuit is very simple and can be implemented in a breadboard while a 5 volt power supply different from that of Arduino board must be used.

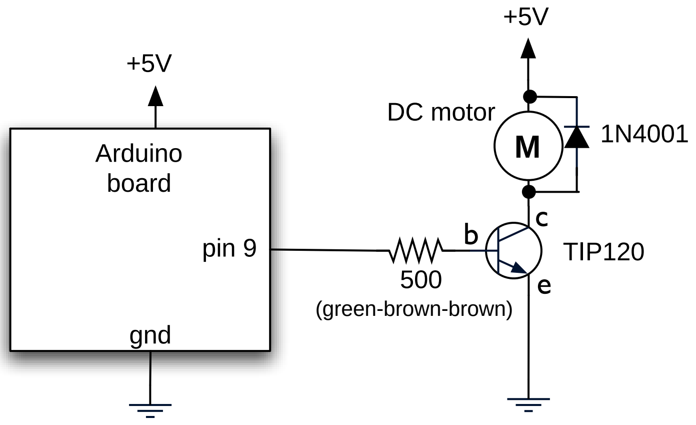


Figure 11: Drive a PMDC with a negative type transistor

The following code can be used to change the speed of the motor by choosing one of the three different functions that are the *motorOnThenoff()*, the motorOnThenoffWithSpeed() and the *motorAcceleration()*. The first one switches on then off the transistor with the use of the *delay() and digitalWrite()* functions. The second one uses the *analogwrite()* function and finally the last one uses two for loops in order to accelerate and decelerate the rotation of the motor. The code is given below:

```
int motorPin = 9;
void setup() { pinMode(motorPin, OUTPUT); }
void loop()  {
                motorOnThenOff();
              //motorOnThenOffWithSpeed();
              //motorAcceleration();
             }

void motorOnThenOff(){
                        int onTime = 2500;
                       int offTime = 1000;

                      digitalWrite(motorPin, HIGH);
                             delay(onTime);
                      digitalWrite(motorPin, LOW);
                             delay(offTime);
                    }

void motorOnThenOffWithSpeed(){
                                  int onTime = 2500;
                                 int offTime = 1000;
                                 int onSpeed = 200;
                                int offSpeed = 50;

                                analogWrite(motorPin, onSpeed);
                                      delay(onTime);
                                analogWrite(motorPin, offSpeed);
                                      delay(offTime);
                              }

void motorAcceleration(){
                          int delayTime = 50;
  //Accelerates the motor
                          for(int i = 0; i < 256; i++){
                                                         analogWrite(motorPin, i);
                                                               delay(delayTime);
                                                        }
  //Decelerates the motor
                          for(int i = 255; i >= 0; i--){
                                                          analogWrite(motorPin, i);
                                                                delay(delayTime);
                                                         }
                        }
```

EXAMPLE 10-2: DRIVE THE PMDC MOTOR WITH THE IC L293D

In the circuit of the previous example it is not possible to change the direction of rotation for the PMDC motor. To do so, we can effectively drive the motor as shown in figure 12 by utilizing one of the two H-Bridge circuits that are incorporated in the IC L293D chip.

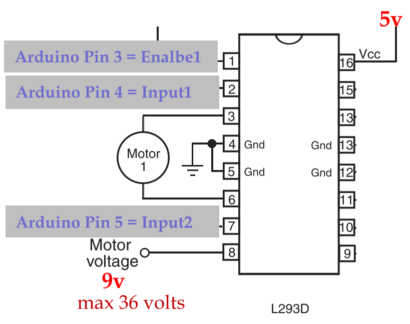


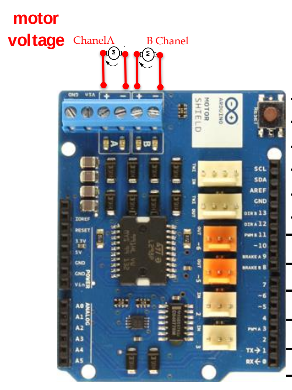
Figure 12: Driving a PMDC motor with the IC L293D chip.

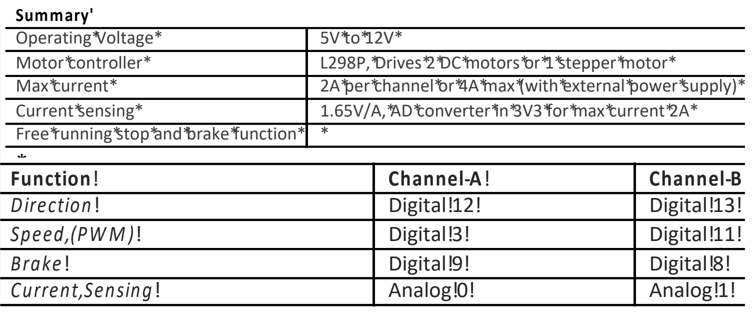
The pins 1, 2 and 7 of the IC chip L293D are connected directly to the Arduino pins 3, 4 and 5 respectively, while the motor terminals are connected to the pins 3 and 6 of the IC chip. For a PMDC motor with nominal power supply of 9V we connect pin 8 of the chip to 9V and pins 4, 5 at ground (0V). Now, it is possible to control the direction of rotation with Arduino pins 4 and 5 and the speed of the motor with Arduino pin 3. The following program can be used along with the above circuit to change the direction of rotation and accelerate or decelerate the motor's shaft.

```
int pin_Enable1 = 3;
int pin_Input1 =  4;
int pin_Input2 =  5;
int i = 0;

void setup() {
              pinMode(pin_Enable1, OUTPUT);
              pinMode(pin_Input1, OUTPUT);
              pinMode(pin_Input2, OUTPUT);
             }

void loop (){
              digitalWrite(pin_Enable1,HIGH); //Clockwise Rotation
              digitalWrite(pin_Input1,LOW);
              digitalWrite(pin_Input2,HIGH);

delay(4000);

              digitalWrite(pin_Input1, HIGH); //Counter-clockwise Rotation
              digitalWrite(pin_Input2, LOW);

delay(4000);

              digitalWrite(pin_Enable1, LOW); //Low Stop
delay(4000);

              for ( i=0; i<256; i=i+2) { analogWrite(pin_Enable1,i); delay(100);} // accelerate
              for ( i=255; i>=0; i=i-2) { analogWrite(pin_Enable1,i); delay(100);} // decelerate
 delay(4000);
              digitalWrite(pin_Enable1,HIGH); //Clockwise Rotation
              digitalWrite(pin_Input1,LOW);
              digitalWrite(pin_Input2,HIGH);

 delay(4000);
              digitalWrite(pin_Input1,LOW);     //Break
              digitalWrite(pin_Input2,LOW);
 delay(10000);


             }
```

EXAMPLE 10-3: MOTOR SHIELD DRIVER FOR ARDUINO BASED ON IC L298P (2x H-Bridge)

In order to simplify the implementation of the circuit shown in the previous example and reduce the pins used to control the PMDC motor we can use the Arduino motor shield shown in figure 13. In the same figure it is also given the specifications of the motor shield and the digital pins that can be used to control up to two motors (channel A and channel B) along with measuring their current consumption from analog inputs A0 and A1 respectively.





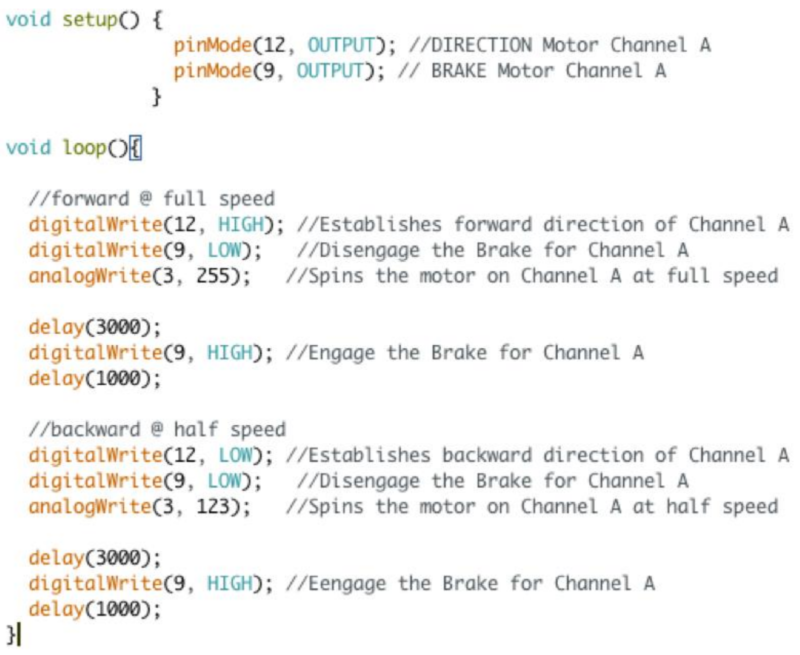
Summary'

Operating*Voltage*	5V*to*12V*
Motor*controller*	L298P,*Drives*2*DC*motors*or*1*stepper*motor*
Max*current*	2A*per*channel*or*4A*max*(with*external*power*supply)*
Current*sensing*	1.65V/A,*AD*converter*in*3V3*for*max*current*2A*
Free*running*stop*and*brake*function*	*

Function!	**Channel-A!**	**Channel-B**
Direction!	Digital!12!	Digital!13!
Speed,(PWM)!	Digital!3!	Digital!11!
Brake!	Digital!9!	Digital!8!
Current,Sensing!	Analog!0!	Analog!1!

Figure 13: The motor shield driver for Arduino based on IC L298P

The following program controls the direction and the velocity of the motor connected in channel A. The motor shield must be connected to a power supply of 9V for a corresponding PMDC motor. The direction of rotation, the speed and the brake of the motor are controlled through the Arduino pins 12, 3 and 9 respectively. Appropriate comments are used in the following code in order to be clear the main idea and the instructions of the program.



```
void setup() {
               pinMode(12, OUTPUT); //DIRECTION Motor Channel A
               pinMode(9, OUTPUT); // BRAKE Motor Channel A
             }

void loop(){

  //forward @ full speed
  digitalWrite(12, HIGH); //Establishes forward direction of Channel A
  digitalWrite(9, LOW);   //Disengage the Brake for Channel A
  analogWrite(3, 255);   //Spins the motor on Channel A at full speed

  delay(3000);
  digitalWrite(9, HIGH); //Engage the Brake for Channel A
  delay(1000);

  //backward @ half speed
  digitalWrite(12, LOW); //Establishes backward direction of Channel A
  digitalWrite(9, LOW);   //Disengage the Brake for Channel A
  analogWrite(3, 123);   //Spins the motor on Channel A at half speed

  delay(3000);
  digitalWrite(9, HIGH); //Eengage the Brake for Channel A
  delay(1000);
}
```

EXAMPLE 10-4: CONTROL THE ROTATION ANGLE OF A STEPPER MOTOR WITH A POTENTIOMETER BASED ON IC L293D (2x H-Bridge)

In this example we control the rotational angle of a stepper motor with an Arduino microcontroller and two H-Bridge circuits that are incorporated in the IC L293D chip. A potentiometer, connected in the Arduino analog input A0, is used for the setpoint of the desire angle. Thus, if you turn the potentiometer clockwise then stepper will rotate clockwise and if you turn potentiometer anticlockwise then it will rotate anticlockwise.

The schematic diagram of the circuit for connecting a bipolar stepper motor is given in figure 14. In this case, four cables are needed to control the stepper motor through Arduino pins D8, D9, D10 and D11 along with an extra power supply for the stepper motor that is connected with the pin8 of the IC L298D. The enable pins 1 and 9 along with the pin 16 of the IC L293D are connected in the Arduino 5v power supply.

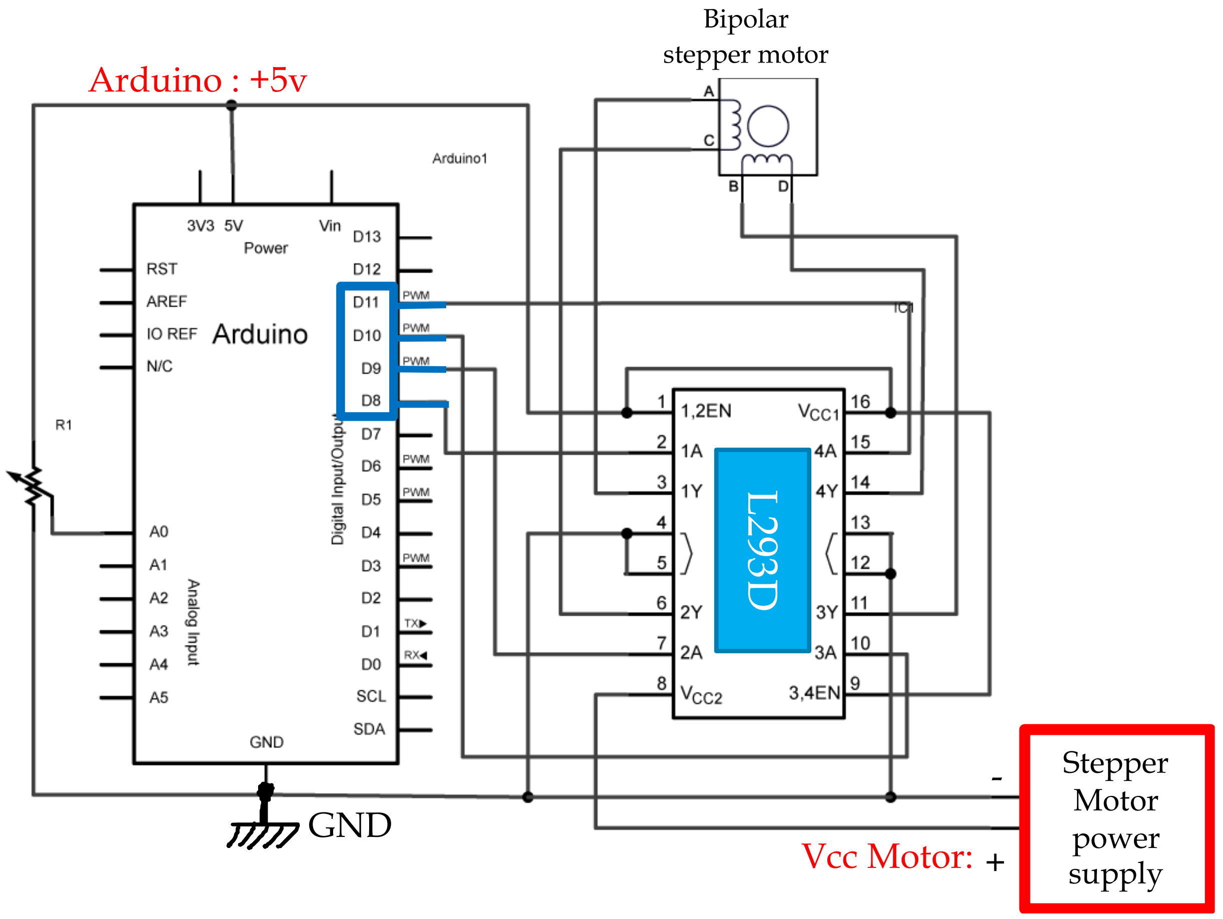
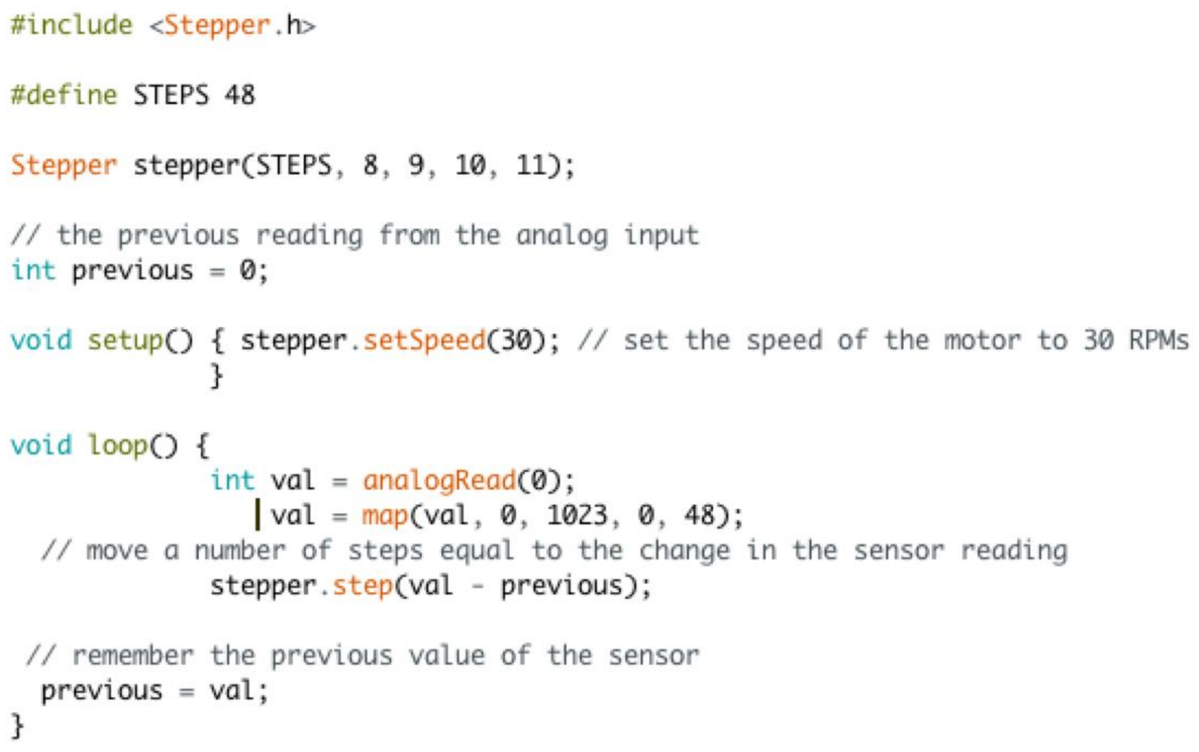


Figure 14: Controlling the rotation of a stepper motor with an Arduino microcontroller, a potentiometer and the IC L293D.

It should be noted that for a unipolar stepper motor the only difference in the connection of the above circuit is the extra common terminals of the coils connected with the rated supply voltage of the stepper motor.

Finally, the following program can be used for the Arduino microcontroller to control both a unipolar or bipolar stepper motor with the use of the stepper library.



```
#include <Stepper.h>

#define STEPS 48

Stepper stepper(STEPS, 8, 9, 10, 11);

// the previous reading from the analog input
int previous = 0;

void setup() { stepper.setSpeed(30); // set the speed of the motor to 30 RPMs
             }

void loop() {
             int val = analogRead(0);
                |val = map(val, 0, 1023, 0, 48);
  // move a number of steps equal to the change in the sensor reading
             stepper.step(val - previous);

 // remember the previous value of the sensor
  previous = val;
}
```

EXAMPLE 10-5: CONTROL THE ROTATION ANGLE OF A STEPPER MOTOR WITH A THE ARDUINO MOTOR SHIELD

In order to simplify the implementation of the circuit shown in figure 14 and reduce the pins used to control the stepper motor we utilize the Arduino motor shield as shown in figure 15 with the two coils of the motor are connected in channels A and B. On this case two wire control is used with the Arduino pins 12 and 13 while the pins 3, 11 are set in 5volts and pins 8,9 in 0v through the *digitalWrite()* function.

Note that the potentiometer is connected in the analog channel A2 since A0 is reserved for current sensing in the channel A. Finally the *stepper.step()* function is used to move the motors shaft in the desired position as shown in the following program.

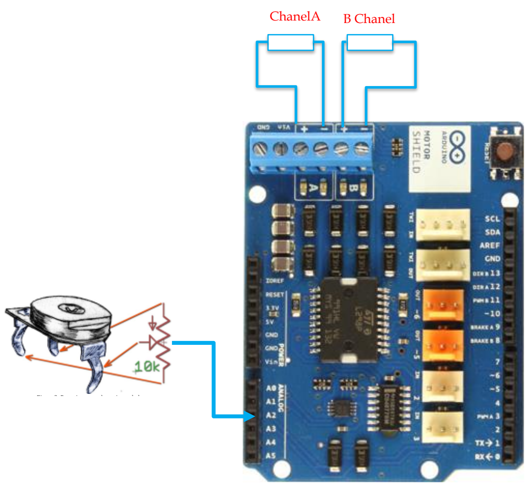


Figure 15: Controlling the rotation of a stepper motor with an Arduino microcontroller, a potentiometer and the motor shield,

```
#include<Stepper.h>
#define STEPS 48

Stepper stepper(STEPS, 12, 13); //two wire control

int previous = 0;

void setup()
{

  stepper.setSpeed(30);

  pinMode(3, OUTPUT); // Enable chanel A (pwm)
  pinMode(11, OUTPUT); //Enable chanel B (pwm)
  pinMode(9, OUTPUT);// Brake chanel A
  pinMode(8, OUTPUT);// Brake chanel B

}

void loop()
{

  int val = analogRead(A2); //A0 is reserved for current Sensing in the Chanel A
      val = map(val, 0, 1023, 0, 48); // to restrict the values of the potentiometer

  digitalWrite(3, HIGH);   // Enable chanelA or Speed
  digitalWrite(11, HIGH);  // Enable chanelB or Speed
  digitalWrite(9, LOW);    // chanelA - Brake
  digitalWrite(8, LOW);    // chanelB - Brake

  stepper.step(val - previous);
  previous = val;

}
```