Laboratory focused on motors

ROBa - Laboratory number 3

https://github.com/Adam-Fabo/ROB-laboratories

Brno university of technology
Faculty of informatics
Adam Fabo
6.2.2023

Motors Laboratory

Welcome to the laboratory dedicated to motors. The goal of this laboratory is to introduce you to different kinds of motors and ways of controlling them.

Prerequisites

It is expected that you have knowledge from previous laboratories (prototyping electrical circuits using breadboard, creating and uploading code).

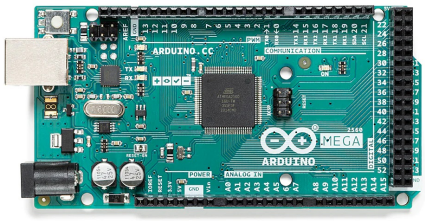
Goals of the laboratory

The goals of this laboratory are set to introduce you to 3 types of motors and ways of controlling them. You will learn:

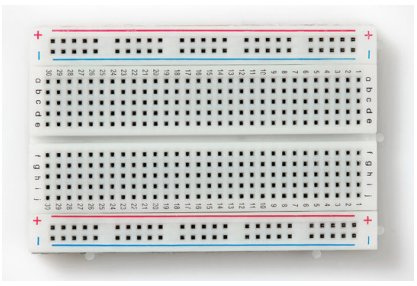
- How servo motor, stepper motor and DC motor works
- Types of control signals
- How stepper motor driver works
- Principles of controlling DC motor - transistor, H-Bridge and more advanced controllers
- Reading feedback from encoders

Hardware used in this laboratory

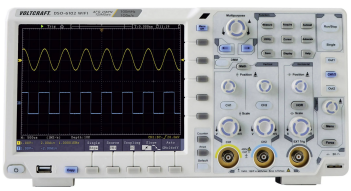
For this laboratory, you will need the following:



(a) Arduino Mega



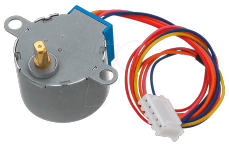
(b) Breadboard



(c) Oscilloscope

Figure 1: Core of the laboratory


(a) Servo motor


(b) Stepper motor


(c) ULN 2003


(d) DC motor


(e) DC motor drivers

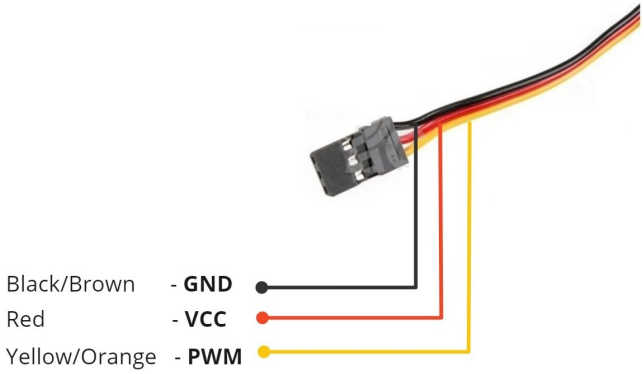
Arduino Mega	-	Contains *ATmega2560* processor. Your code will run on it
Breadboard	-	Used for creating electronics circuits
Oscilloscope	-	Used for displaying electronic signals
Servo motor	-	*FS5103B* Servo motor - Used for precise angular positioning
Stepper motor	-	*28BYJ-48* Stepper motor - Used for precise angular positioning
ULN 2003	-	Stepper motor driver - Contains Darlington transistors
DC motor	-	*Faulhaber 12V* DC Coreless Motor
DC motor drivers	-	Custom made PCB with 3 ways of controlling DC motor

Theory

Servo motor

A servo motor is a type of actuator that allows for **precise control of angular position**. Servo motors (shortened just servo) can be used in wide rage applications, for example in hobby RC cars to steer wheels or in industrial companies to control angle of electro-hydraulic valves.

(a) Servo motor



(b) Servo - connector pinout

The servo used in this laboratory is relatively small FS5103B. Operating voltage is **5 Volts** and **maximum operating current is 150 mA**. That is why there is need for **external power supply** to power this servo. Angle of rotation is **180 degrees**, but other types of servos can have wider range or be multi-turn servos. Servo is controlled by **PWM** with **frequency of 50Hz** and **duty is between 1ms and 2ms**. Angle of a servo is proportional to the duty length (bigger duty -> bigger angle)

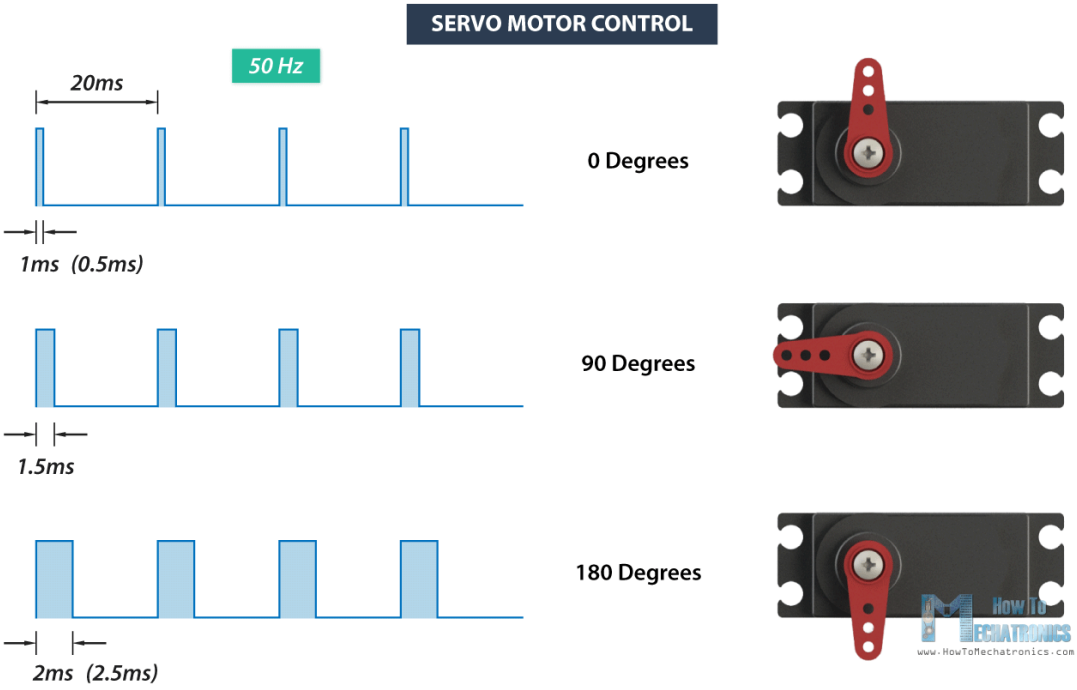


Figure 4: Servo PWM diagram

Servo's working principle is based on **closed feedback loop with controlling circuit**. Motor in a servo is small **DC motor** with high RPM and small torque. That is why servo has **reduction gears** to convert an output of this motor to low RPM and high torque. At the output of the gearbox, there is a **potentiometer** (or other element that detects rotation) and its output is fed to the controlling circuit. This controlling circuit takes in **PWM signal** and potentiometer's output and based on that rotates the motor.

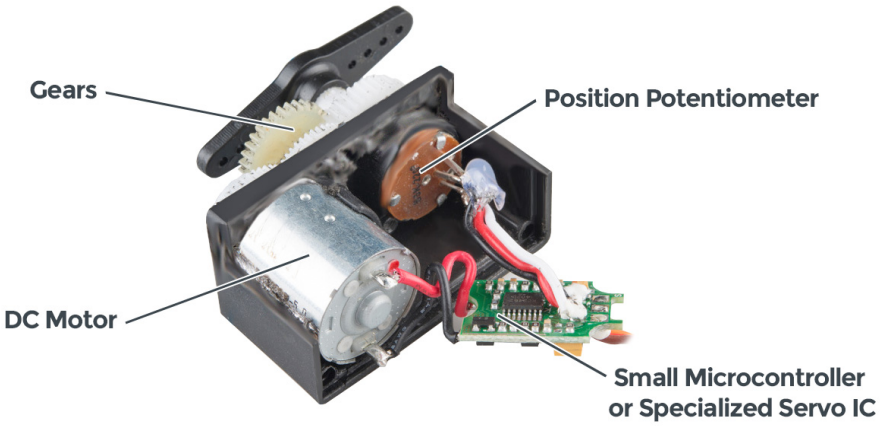


Figure 5: Servo disassembled

There are **2 ways of controlling servo with Arduino**: **Writing custom PWM** implementation or **using library <*Servo.h*>**. Writing custom implementation, great for learning the servo control but not so great for repeated use. The advantage of the built-in servo library is that it is optimized for each Arduino board and works on higher level of abstraction (you, as a user,are shielded from PWM implementation). Figure 6 shows code sample needed in order to move with servo.

Functions for custom PWM:

```
digitalWrite(pin, HIGH/LOW);   // Set pin to HIGH or LOW level
delayMicroseconds(x);          // Wait for x Microseconds
```

Methods from Servo.h library:

```
#include <Servo.h>       // Import built-in library

Servo myservo;           // Create Servo object to control a servo
myservo.attach(pin);     // Attaches the servo on given pin to the servo object

myservo.write(angle);    // Set angle to servo
```

Figure 6: Servo sample code

Stepper motor

Stepper motor, same as servo motor, is a type of actuator that allows for **precise control of angular position**. However, the working principle is different from servo's. Stepper motors can be used in 3D printers or industrial robotic arms.

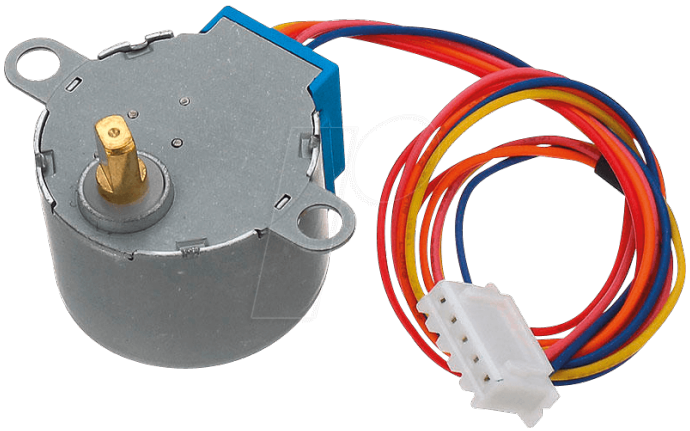


Figure 7: Stepper motor 28BYJ-48

A stepper motor is a brushless DC motor that **divides a full rotation into a number of equal steps**. It has no controlling circuit, meaning that it is up to the user to implement a feedback loop. It's rotor consist of permanent magnet and stator consists of 4 coils that are connected to the input of the motor. Main idea behind a stepper motor is that as you energize the coils one by one, the **rotor rotates in steps** - that's why it is called stepper motor.

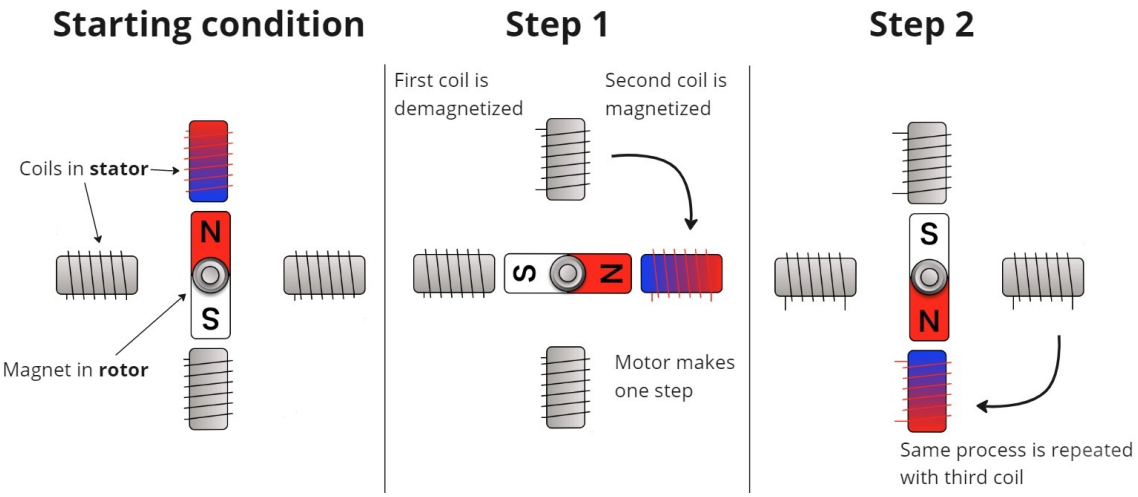


Figure 8: Simple stepper motor

In reality, **stepper motors are more complicated** than this simple example. This simple motor would produce only 4 steps, with one step advancing 90 degrees. There is need for modern stepper motors to have more steps per rotation. This can be achieved by **"toothed" electromagnets**. Toothed electromagnets offer higher precision because when coil is magnetized electromagnets are aligned **tooth to tooth** - which produces small step. Servo in this laboratory has 32 of those basic steps.

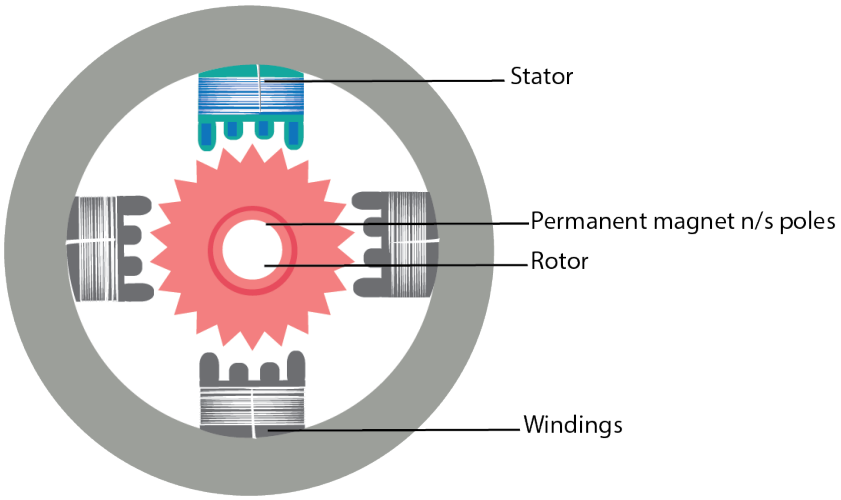
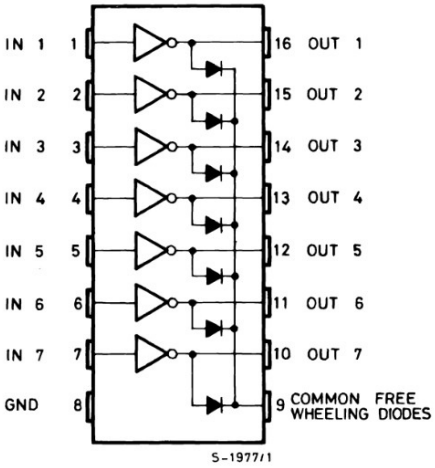


Figure 9: Diagram of a realistic stepper motor

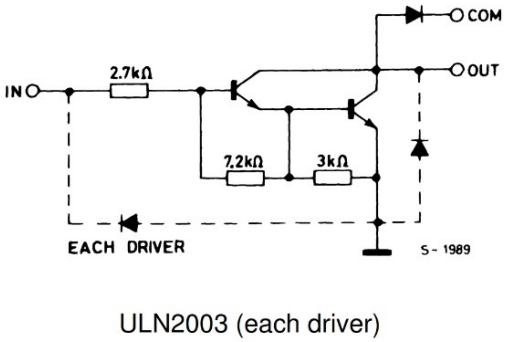
Motor used in this laboratory is **28BYJ-48**. It has **4 stator coils** and built in **gear reduction 1:64**, which means that it has in total **2048 full steps per one revolution** (360 degrees). Since stepper motors can take bigger amount of current than microcontroller can supply from its GPIO pins, it has to have a driver that amplifies it's controlling signal. Driver used in this laboratory is **ULN 2003**. It consists of simple set of **darlington transistors** that **amplifies the input signals current**. Driver and stepper's motor are rated for 5–12 Volts.

PCB with driver

Pinout



Schematic diagram



ULN2003 (each driver)

Figure 10: Stepper motor driver

In order to move properly stepper motor, coils need to be magnetized in correct order. There are 2 types of sequences: **Full step** and **half step**.

Full step method is a simpler one. **Each pulse** of this drive mode causes the motor to move a **basic step angle**... There are **2 variants** of this method - **low and high torque**. In **low torque** variant only **one coil is magnetized** in time, while in high torque method **two coils are magnetized** at once

Half step method allows for **doubling the number of steps**. It combines both low and high torque mode from full step method. Extra steps are possible by energizing two coils at once which produces half step.

Full step sequences

Lower torque method

Step Number	Coil 1	Coil 2	Coil 3	Coil 4
1	HIGH	LOW	LOW	LOW
2	LOW	HIGH	LOW	LOW
3	LOW	LOW	HIGH	LOW
4	LOW	LOW	LOW	HIGH

Higher torque method

Step Number	Coil 1	Coil 2	Coil 3	Coil 4
1	HIGH	LOW	LOW	HIGH
2	HIGH	HIGH	LOW	LOW
3	LOW	HIGH	HIGH	LOW
4	LOW	LOW	HIGH	HIGH

Figure 11: Full step sequences

Half step sequence

Step Number	Coil 1	Coil 2	Coil 3	Coil 4
1	HIGH	LOW	LOW	LOW
2	HIGH	HIGH	LOW	LOW
3	LOW	HIGH	LOW	LOW
4	LOW	HIGH	HIGH	LOW
5	LOW	LOW	HIGH	LOW
6	LOW	LOW	HIGH	HIGH
7	LOW	LOW	LOW	HIGH
8	HIGH	LOW	LOW	HIGH

Figure 12: Half step sequence

There are **2 ways of controlling stepper motor with Arduino**: **Writing custom implementation** or **using library <*Stepper.h*>**. Writing custom implementation is great for learning and allows you to choose which mode method to use - Full/Half step. Disadvantage is that it takes time to write and might use active waiting. Library <*Stepper.h*> is great that is allows for higher level of abstraction but does not support different modes - drives servo only in Full step - high torque mode.

Function for custom stepper control:

```
digitalWrite(pin, HIGH/LOW);  // Set pin to HIGH or LOW level
```

Methods from Stepper.h library:

```
#include <Stepper.h>    // Import built-in library

// Create Stepper object with given pins
// Need to know steps per revolution
Stepper myStepper = Stepper(stepsPerRevolution, p1,p2,p3,p4);

myStepper.setSpeed(x);  // Set speed for x RPM
myStepper.step(y);      // Make y steps
```

Figure 13: Stepper motor sample code

DC motor

A DC motor converts Direct Current to rotational mechanical motion. DC motors are used in fans, RC cars, electric toothbrushes, hair dryers, pumps ...

Motor consists of 2 main parts - **Stator** and **Rotor**. Stator consists of permanent magnet and rotor consists of coils with or without ferromagnetic core. The motor used in this lab is coreless. This is advantageous in some situations - for example with no extra mass motor has lower inertia and can be stopped and accelerated faster.

Motor used in this laboratory is **Faulhaber 16002**. It is rated for 12 Volts DC and has built-in encoder for feedback.

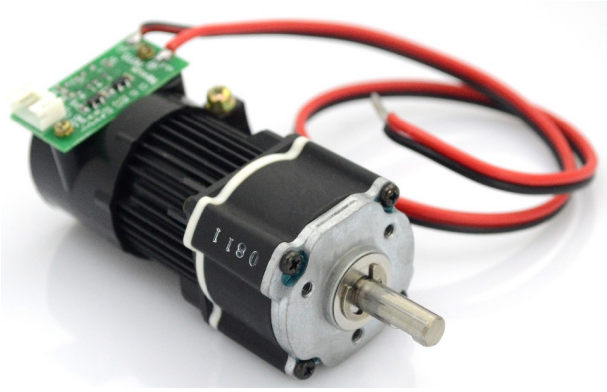


Figure 14: Faulhaber DC motor

Ways of controlling DC motor

Since bare DC motor **does not have any controlling circuits**, it is not possible to control its direction/speed without a controlling circuit. The only way how to run motor without controlling circuit is to connect it directly to the power supply. This way, it is possible to turn on the motor. Rotational speed can be regulated by changing the supply voltage. Spinning direction can be changed by reversing the polarity on the motor pins. However, **this way is unsuitable for robotics** where supply voltage is constant and motor pins are soldered to the PCB - meaning that polarity of the motor cannot be changed by manually reversing the polarity.

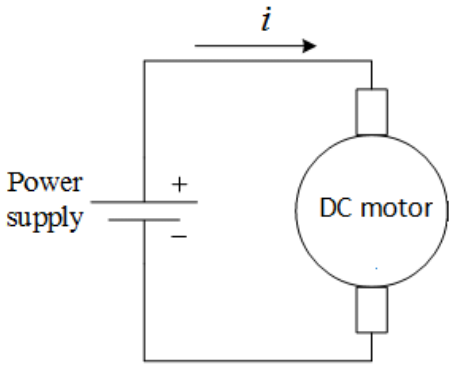


Figure 15: Simplest motor scheme

The simplest way of controlling a DC motor is to use a transistor. This way motor rotational speed can be controlled, but rotational direction can not be controlled. Since motors can demand current that exceeds microcontroller's maximum output current (∼20mA), there is need for a transistor - electrical **semiconductor part that amplifies current**. For controlling rotational speed of the motor, **PWM can be used** for average power delivery. Even if the PWM signal is not powerful enough, it can be amplified using the transistor - therefore, it can be used for controlling the motor.

A good analogue is that **a transistor acts like a switch** that can be controlled by applying voltage to one of it's 3 pins - Base. The other 2 pins are emitter and collector. By "switching" on/off this switch, it is possible to control the motor rotational speed.

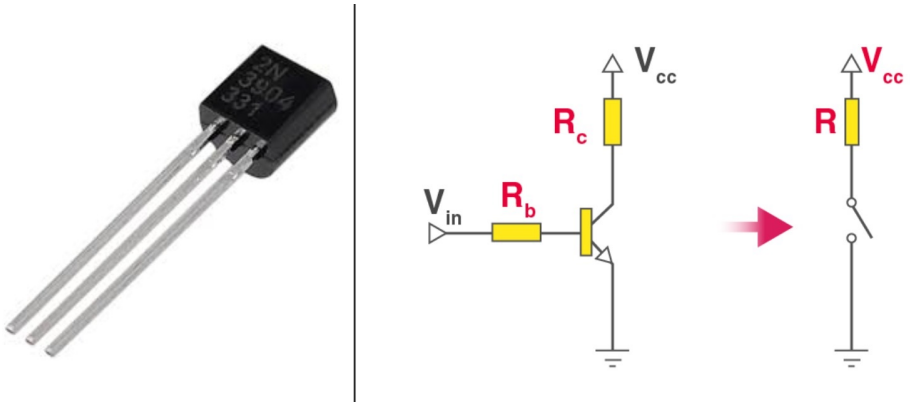


Figure 16: Transistor that acts like a switch

A more sophisticated way of controlling a DC motor is to use H-Bridge. This electrical circuits allows controlling the rotational speed as well as the rotational direction. Its disadvantage is that it is more complicated circuit and there is need for user to know what he is doing, since when mistake occurs and wrong transistors are opened, power supply might be shorted which could lead to destruction of H-bridge.

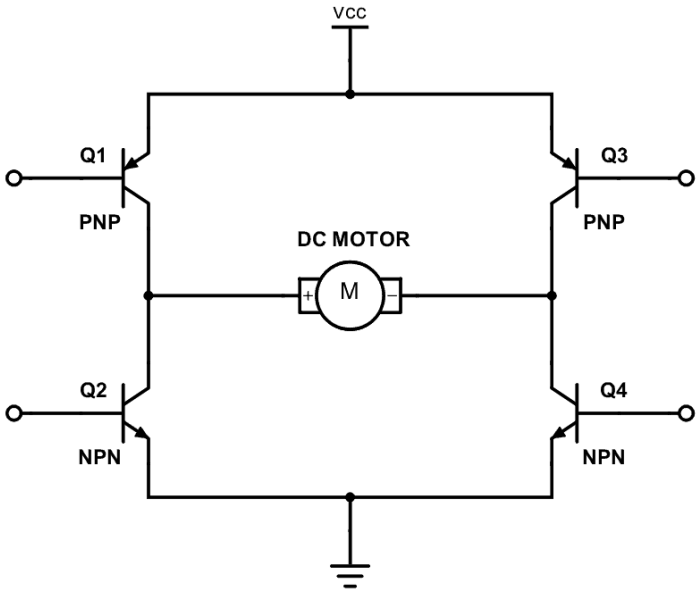


Figure 17: H-Bridge

The H-Bridge **consists of 4 transistors** that control flow of a current. In order to spin the motor **2 diagonally opposite transistors** have to be opened (no more, no less). Speed of rotation is controlled same way as with single transistor - using PWM. Direction of rotation is controlled by opening a different pair of diagonally opposite transistors.

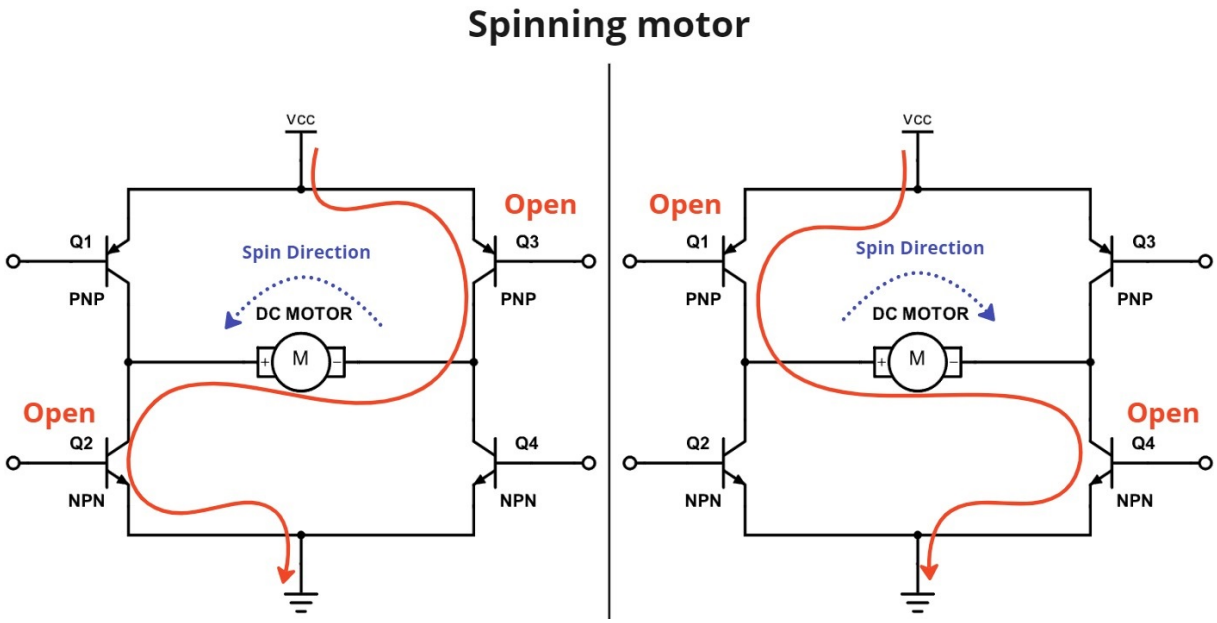


Figure 18: H-Bridge spinning motor

The H-Brigde also has an option to stop the motor by opening a pair of transistors opposite of each other.

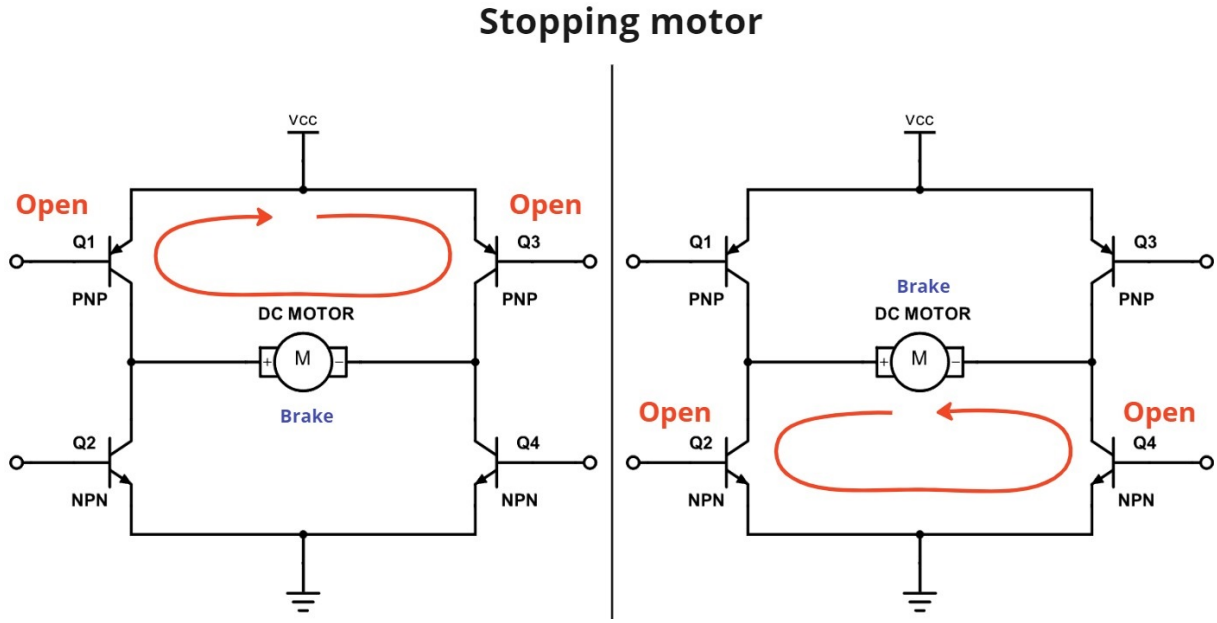


Figure 19: H-Bridge stopping motor

If two transistors that form a straight line from power supply to ground are connected, power supply will be shorted - which can lead to destruction of some components.

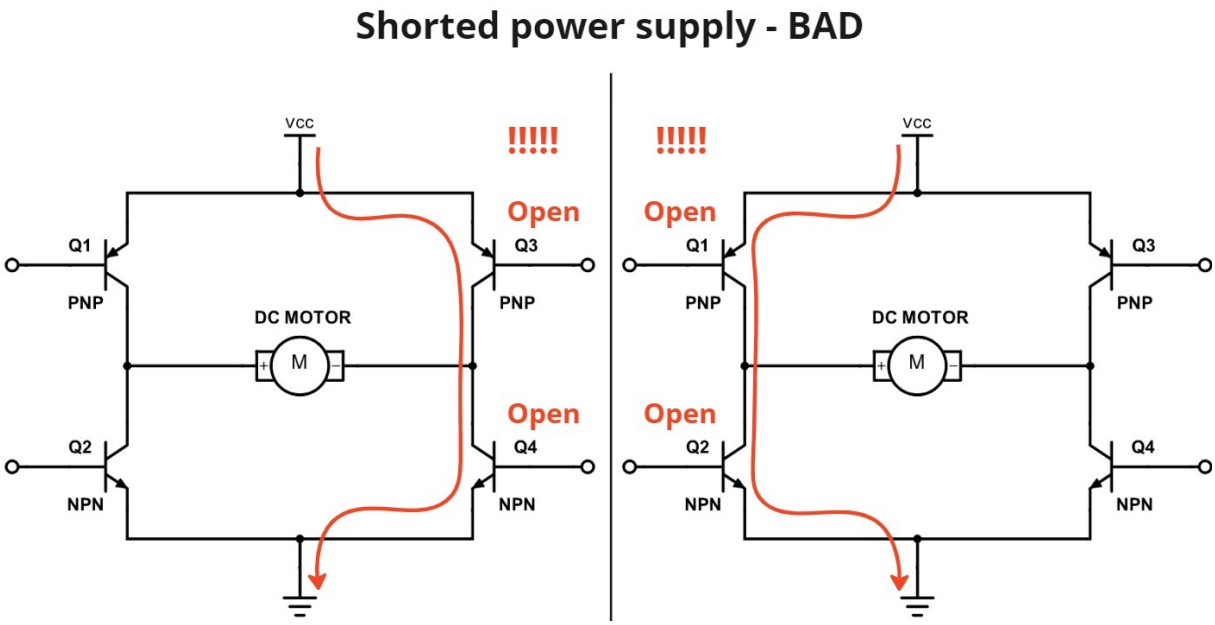


Figure 20: H-Bridge forbidden combinations

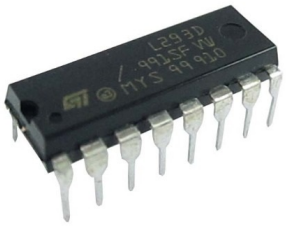
Figure 21 shows combination of all possible states of H-Bridge

Sw1	Sw2	Sw3	Sw4	Operation
1	0	0	1	Moves Right Side
0	1	1	0	Moves Left Side
1	0	1	0	Motor Brakes
0	1	0	1	Motor Brakes
1	1	0	0	Short Circuit
0	0	1	1	Short Circuit
1	1	1	1	Short Circuit

Figure 21: H-Bridge table

Another, and safest way, is to use **a dedicated motor driver**. Advantage of it is that you as a user don't have to make your own circuit while it offers same functionality as a H-Bridge - ability to control direction and speed of rotation. The disadvantage is that you have to study the datasheet in order to know how to work with specific driver. There are many drivers at the market with many functionalities. Driver used in this laboratory is **L293D**.

L293D



L293D - diagram

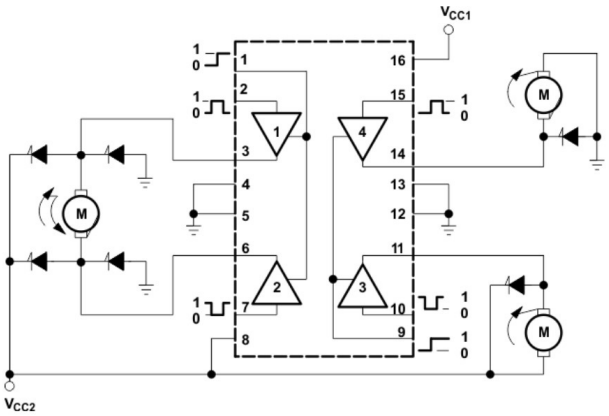


Figure 22: L293D driver with its internal diagram

In its essence, it is 2 H-Bridges with extra protection against shorting the power supply. Driver has **2 Input pins and one Enable pin**. Direction of rotation of the motor is controlled by writing different combination to the input pins, and speed of rotation can be controlled by applying PWM to the enable pin.

Enable	Input 1	Input2	Output
HIGH	HIGH	LOW	Turn right
HIGH	LOW	HIGH	Turn left
HIGH	HIGH	HIGH	Fast motor stop
HIGH	LOW	LOW	Fast motor stop
LOW	X	X	Free-running motor stop

Figure 23: Possible inputs to L293D driver

Ways of controlling DC motor - Custom PCB

Since making the circuits like H-Bridge on breadboard would be unnecessarily complicated and time consuming, custom circuit board was designed. Circuit board also contains transistor and L293D driver. Each part acts separately, but there is one common power supply and ground. This circuit is rated for 5 Volts (due to L293D driver). If you want to power the motor with higher voltage, disconnect the jumper pointed by separate power supply arrow.

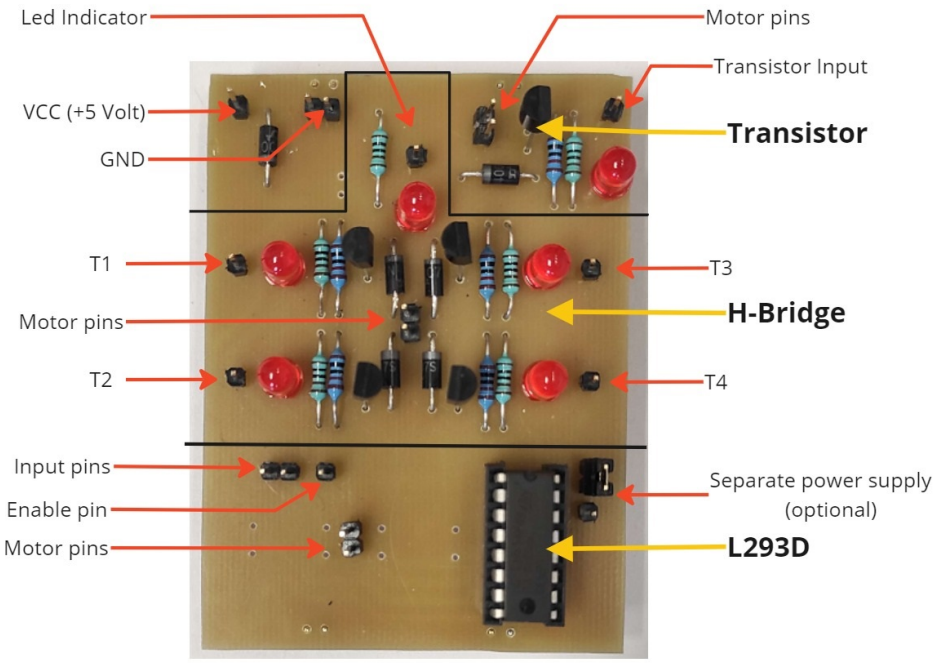


Figure 24: Explained Custom PCB

Each transistor also contains a LED that indicates if the transistor is opened or closed (if transistor is opened, LED is ON). H-Bridge also contains one extra LED that indicates the wrong combination of opened transistors. This LED is controlled by microcontroller.

Feedback - encoders

An encoder is an electrical device that is used to measure the position or speed of a rotating shaft. DC motors used in this laboratory have 2-channel incremental encoders built-in. This type of encoder has 2 outputs called A and B. These outputs have binary value - either 1 or 0. As the shaft is rotated, output signal of both channels changes in time. A diagram of this change can be seen at Figure 25.

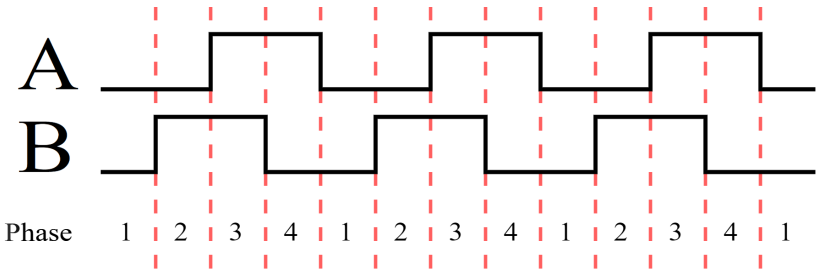


Figure 25: Quadrature output of an encoder

Since there are 2 output channels and signals are binary, it gives 4 possible combinations of output. From this output can be determined speed and direction of rotation. Direction can be determined by order of signals and speed can be measured by how often these signals occur.

Exercises

Servo - custom PWM

Write your own implementation of PWM that controls servo.

1. Open file **Servo_custom_PWM.ino**
2. Take a look at scheme **Servo_scheme.png** and create circuit as it shown at the image
3. Follow the instructions and implement servo control
4. Upload the code
5. Test your implementation by uploading your code and observing servo movement

Servo - Library

Write servo controller using **<Servo.h>** library.

1. Open file **Servo_library.ino**
2. Reuse scheme from previous exercise
3. Follow the instructions and implement servo control
4. Upload the code
5. Test your implementation by uploading your code and observing servo movement

Stepper - custom

Write your own implementation that controls stepper motor.

1. Open file **Stepper_custom_PWM.ino**
2. Take a look at scheme **Stepper_scheme.png** and create circuit as it shown at the image
3. Follow the instructions and implement stepper motor control
4. Upload the code
5. Test your implementation by uploading your code and observing the stepper motor movement

Stepper - library

Write stepper motor controller using **<Stepper.h>** library.

1. Open file **Stepper_library.ino**
2. Reuse scheme from previous exercise
3. Follow the instructions and implement stepper motor control
4. Upload the code
5. Test your implementation by uploading your code and observing the stepper motor movement

DC motor - Transistor

Write program that will control speed of rotation of a DC motor.

1. Open file **DC_transistor.ino**
2. Take a look at scheme **DC_transistor_scheme.png** and create circuit as it shown at the image
3. Follow the instructions and implement DC motor control
4. Upload the code
5. Test your implementation by uploading your code and observing DC motor movement

Bonus: Try and change rotation of the DC motor

DC motor - H-Bridge

Write a program that will control speed and direction of rotation of a DC motor.

1. Open file **DC_H_Bridge.ino**
2. Take a look at scheme **DC_H_Bridge_scheme.png** and create circuit as it shown at the image
3. Follow the instructions and implement DC motor control
4. Upload the code
5. Test your implementation by uploading your code and observing DC motor movement

DC motor - Driver

Write a program that will control speed and direction of rotation of a DC motor.

1. Open file **DC_L293D.ino**
2. Take a look at scheme **DC_L293D_scheme.png** and create circuit as it shown at the image
3. Follow the instructions and implement DC motor control
4. Upload the code
5. Test your implementation by uploading your code and observing DC motor movement

DC motor - Encoder

Take a look at output of an encoder with oscilloscope

1. Reuse any code and scheme that rotates the motor
2. Take a look at scheme **DC_Encoder_scheme.png** and create circuit as it shown at the image
3. Spin the motor
4. Take a look at the oscilloscope reading
5. Rotate the motor other way