

ROB - Motors laboratory

Adam Fabo

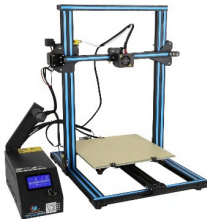
Brno University of Technology, Faculty of Information Technology
Božetěchova 1/2. 602 00 Brno - Královo Pole
xfaboa00@fit.vutbr.cz



February 6, 2022

Today's lab will be focused on **motors**.

Motors are electrical actuators that perform mechanical movement.



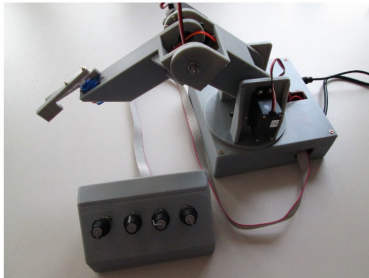
Types of motors used in this laboratory:

- Servo motor
- Stepper motor
- DC motor

Used for precise control of angular position.



Servo motors can be used in robotic arms or as a steering in RC cars.



Working principle of a servo motor:

- Contains small **DC motor** which performs the movement
- At the output shaft is a small potentiometer, which value is fed to the **controlling circuit**
- **Controlling circuit**, based on the feedback from potentiometer, controls the DC motor
- Controlling circuit is given **PWM** by user to control output angle

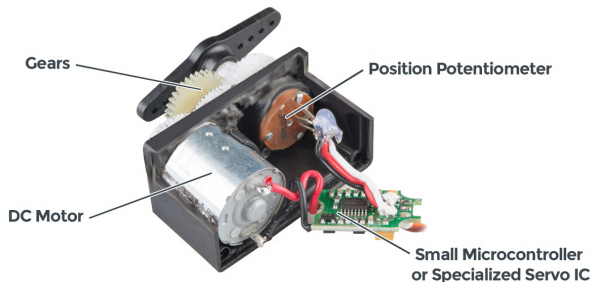
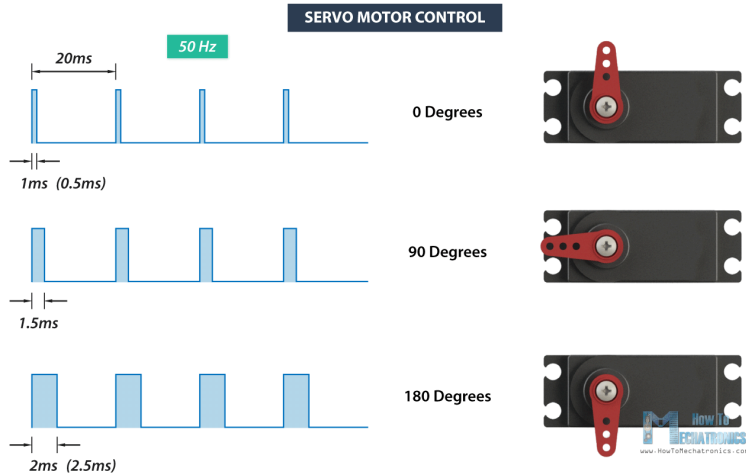
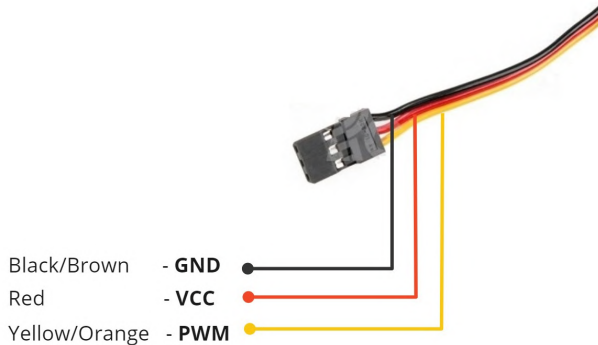


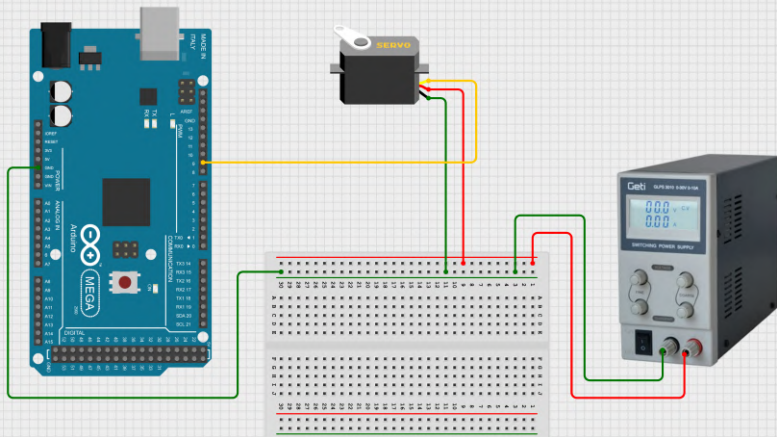
Diagram of Servo control using PWM



- Operating voltage: 5 Volts
- Range: 0 - 180 degrees
- Has 3 Pins - GND, VCC, PWM/Signal



Circuit Designer



There are 2 ways of controlling servo using Arduino:

- Writing custom **PWM** implementation
 - Great for learning
 - Might use active waiting (not so great)
 - Prone to mistakes
- Using library **<Servo.h>**
 - Higher level of abstraction (classes)
 - Implementation is all set
 - Need to read documentation

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
```

Write your own implementation of PWM that controls servo.

- Open file [Servo_custom_PWM.ino](#)
- Take a look at scheme [Servo_scheme.png](#) and create circuit as it shown at the image
- Follow the instructions and implement servo control
- Upload the code
- Test your implementation by uploading your code and observing servo movement

Write servo controller using <Servo.h> library.

- Open file [Servo_library.ino](#)
- Reuse scheme from previous exercise
- Follow the instructions and implement servo control
- Upload the code
- Test your implementation by uploading your code and observing servo movement

Used for precise control of angular position.



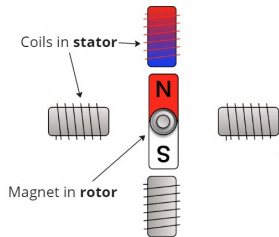
Stepper motors can be found in machines like 3D printers or CNC milling machines



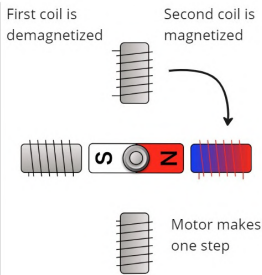
The basic concept goes as follows:

- Rotor is magnet that has south and north pole
- Stator consists of 4 coils that can be magnetized separately, one by one
- As those coils are magnetized in correct order, it "drags" rotor in circular motion

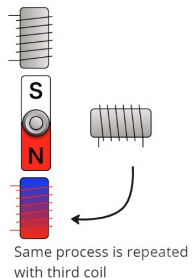
Starting condition



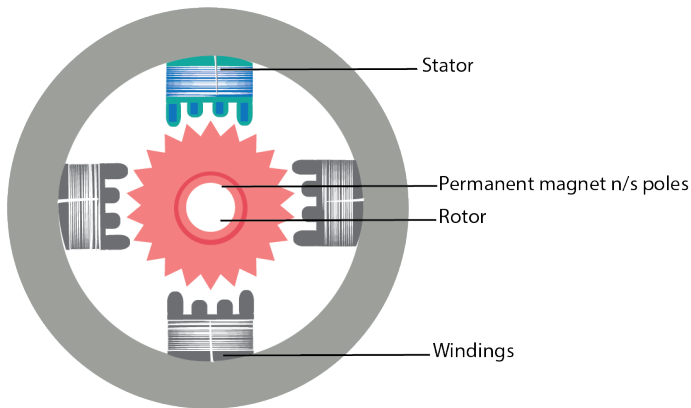
Step 1



Step 2



- Stepper motor in this lab has 2048 steps (it has 64:1 gear ratio)
- This number of steps is achieved by "toothed" electromagnets



Coils need to be magnetized in order as shown in tables

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 |

Half-step method offers more steps per rotation

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 |

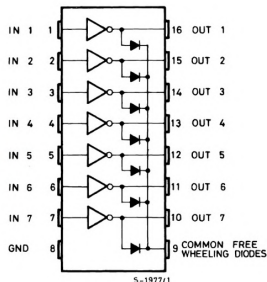
Since stepper motors can drain more current than microcontroller can provide, there is need for drivers.

- Drivers can be simple electrical circuits
- They amplify current
- Driver used in this laboratory - [ULN2003](#)
 - Consists of Darlington transistors

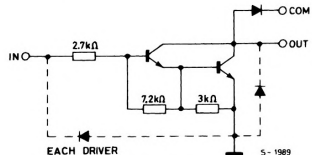
PCB with driver



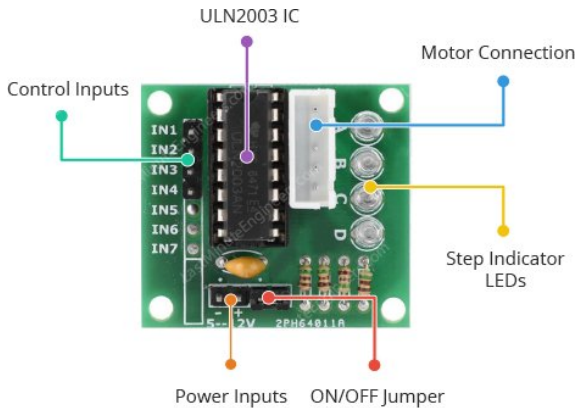
Pinout



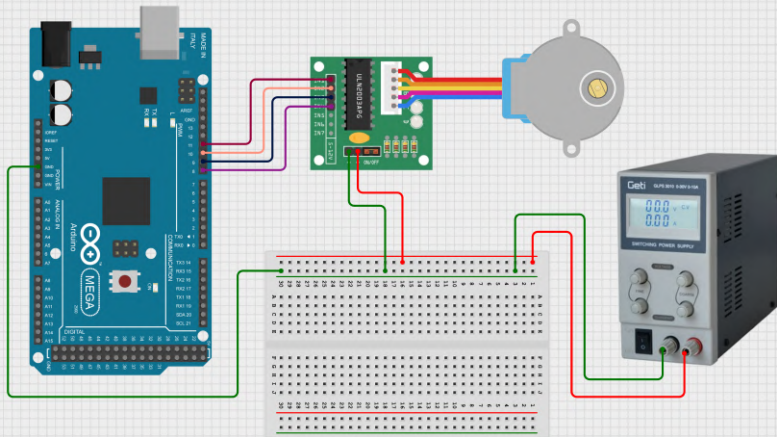
Schematic diagram



ULN2003 (each driver)



Circuit Designer



There are 2 ways of controlling stepper using Arduino:

- Writing custom implementation
 - Great for learning
 - Might use active waiting (not so great)
 - Can be modified
- Using library `<Stepper.h>`
 - Easier to use
 - Does not use active waiting
 - Cannot change step type (full/half)

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
```

Write your own implementation that controls the stepper motor.

- Open file [Stepper_custom_PWM.ino](#)
- Take a look at scheme [Stepper_scheme.png](#) and create circuit as it shown at the image
- Follow the instructions and implement stepper motor control
- Upload the code
- Test your implementation by uploading your code and observing stepper motor movement

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

- Open file `Stepper_library.ino`
- Reuse scheme from previous exercise
- Follow the instructions and implement stepper motor control
- Upload the code
- Test your implementation by uploading your code and observing stepper motor movement

Converts electrical energy into continuous rotary motion

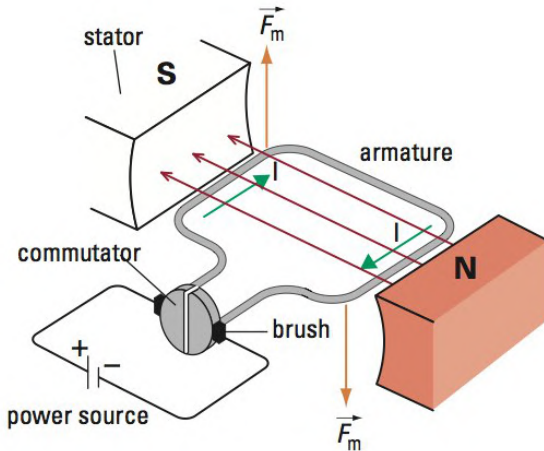


DC motors can be found in fans, RC cars, Electric cars, Trains....



Working principle:

- Works based on Lorentz Law
- The current carrying conductor placed in a magnetic and electric field experience a force

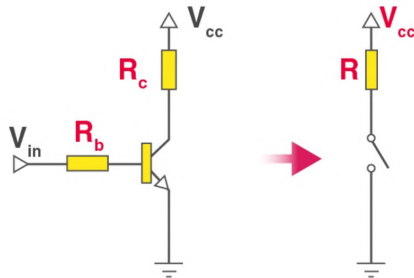


The motor itself is controlled by PWM. But this signal might be weak for power-hungry motor, therefore, it needs amplification. This lab focuses on 3 ways:

- Transistor
- H-Bridge
- Motor driver L293D

Transistor, is a semiconductor electrical part

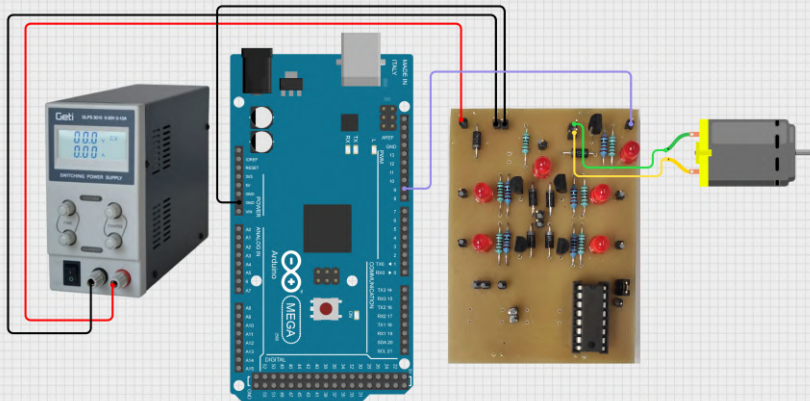
- In this application acts as a switch.
- Has 3 pins - **Collector**, **Emitter** and **Base**
- Can be switched on/off by applying or removing voltage at the **Base**
- Allows controlling motor's **speed** but **not direction**



Function for custom DC motor control using transistor:

```
// Sets PWM to given pin  
// val should be from range <0,255>  
analogWrite(pin, val);
```

Circuit Designer



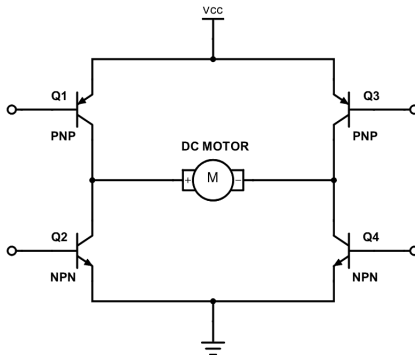
Write a program that will control speed of rotation of a DC motor.

- Open file [DC_transistor.ino](#)
- Take a look at scheme [DC_transistor_scheme.png](#) and create circuit as it shown at the image
- Follow the instructions and implement DC motor control
- Upload the code
- Test your implementation by uploading your code and observing DC motor movement

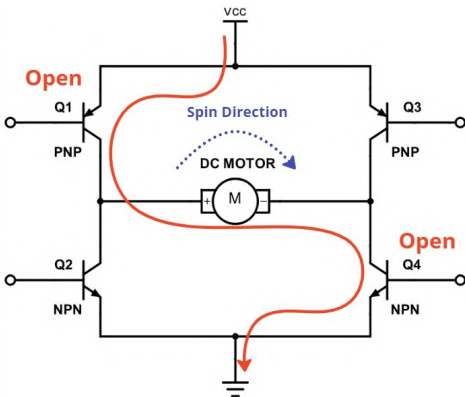
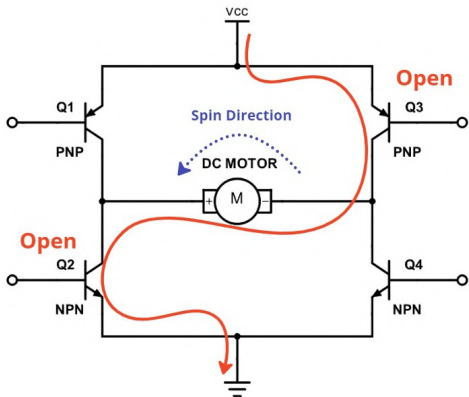
Bonus: Try and change rotation of the DC motor

A H-Bridge is an electrical circuit that consists of 4 transistors.

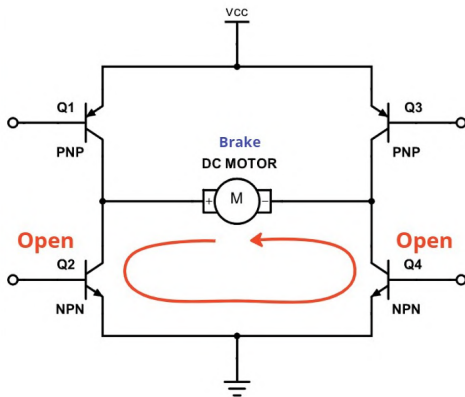
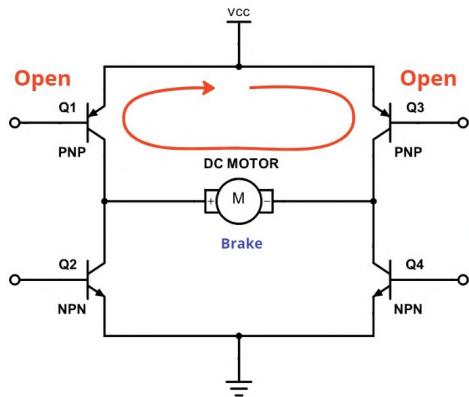
- General scheme looks like a letter H - therefore the name is H-bridge
- Allows controlling motor's **speed** as well as **direction**
- Can spin or stop the motor
- In order to spin the motor, one pair of diagonally opposite transistors need to be opened



Spinning motor



Stopping motor



Shorted power supply - BAD

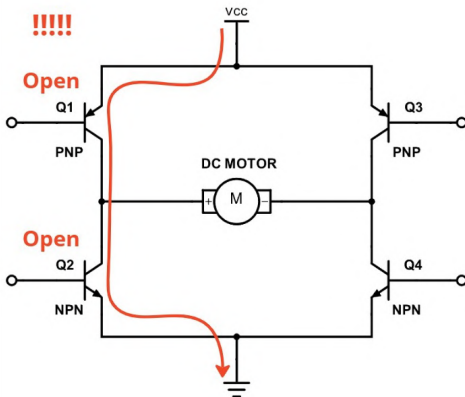
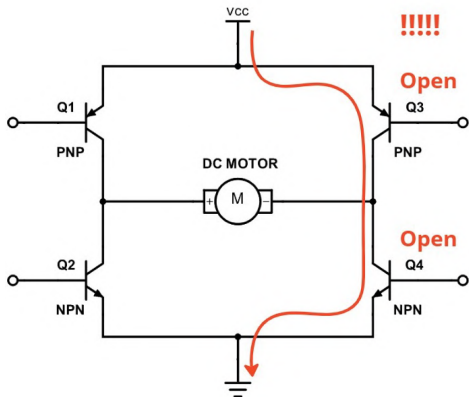


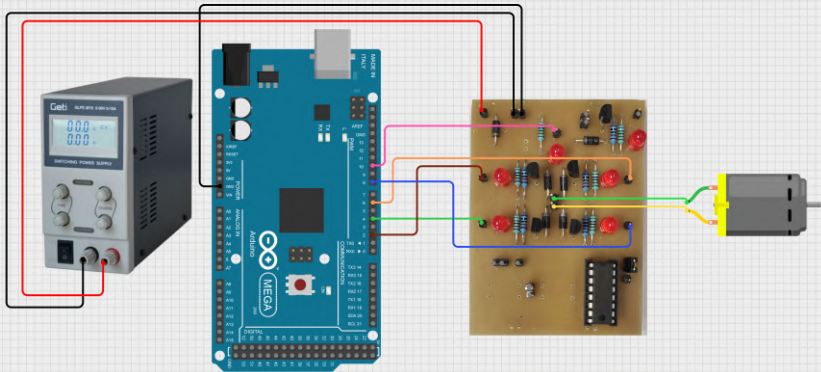
Table of H-Bridge states:

| 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 |

Function for custom DC motor control using H-Bridge:

```
// Custom function made for this exercise  
// Sets transistors T1, T2, T3, T4 to given value  
// Values: 1/true - transistor opened  
//          0/false - transistor closed  
set_H_Bridge(t1,t2,t3,t4);
```

Circuit Designer



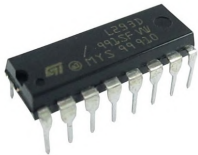
Write program that will control speed and direction of rotation of a DC motor.

- Open file [DC_H_Bridge.ino](#)
- Take a look at scheme [DC_H_Bridge_scheme.png](#) and create circuit as it shown at the image
- Follow the instructions and implement DC motor control
- Upload the code
- Test your implementation by uploading your code and observing DC motor movement

Motor driver L293D contains 4 Half-H Drivers

- Allows controlling motor's speed as well as direction
- Can support up to 2 motors
- Has 1 Enable pin and 2 Input pins
- Speed is controlled by PWM at Enable Pin
- Can not be shorted by wrong combination

L293D



L293D - diagram

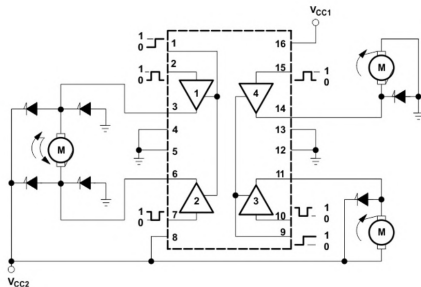


Table of possible input/output states:

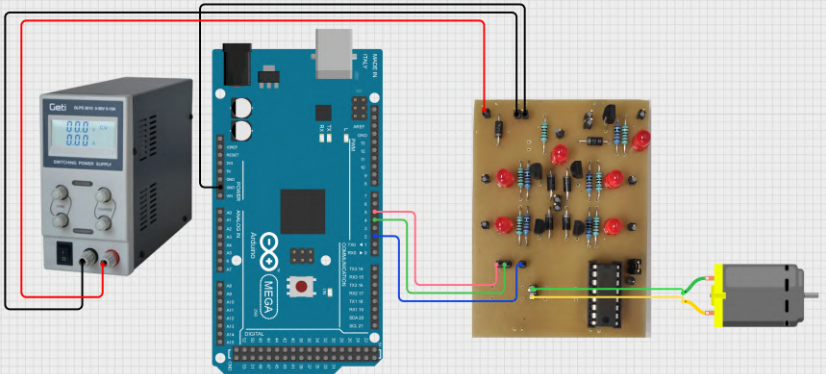
| Enable | Input 1 | Input2 | Output |
|--------|---------|--------|-------------------------|
| HIGH | HIGH | LOW | Turn right |
| HIGH | LOW | HIGH | Turn left |
| HIGH | HIGH | HIGH | Fast motor stop |
| HIGH | LOW | LOW | Fast motor stop |
| LOW | X | X | Free-running motor stop |

mira

Functions for custom DC motor control using L293D:

```
// Set pin to HIGH or LOW level  
// Used for Input pins  
digitalWrite(pin, HIGH/LOW);  
  
// Sets PWM to given pin  
// val should be from range <0,255>  
// Used for Enable pin  
analogWrite(pin, val);
```

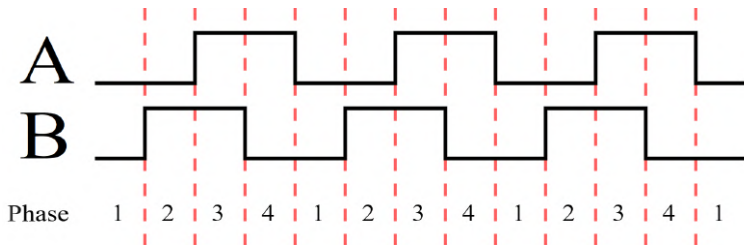
Circuit Designer



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

- Open file [DC_L293D.ino](#)
- Take a look at scheme [DC_L293D_scheme.png](#) and create circuit as it shown at the image
- Follow the instructions and implement DC motor control
- Upload the code
- Test your implementation by uploading your code and observing DC motor movement

- Encoder is an electro-mechanical device that converts the angular position or motion of a shaft to digital output signals.
- DC motors used in this laboratory have 2 channel encoder built-in.



Take a look at output from encoders at an oscilloscope

- Reuse any code that controls motor
- Take a look at scheme [DC_Encoder_scheme.png](#) and create circuit as it shown at the image
- Run the motor and look at the oscilloscope

Thank You For Your Attention !